

Лекция 2

ПРО КОНФИДЕНЦИАЛЬНЫЕ ФАЙЛЫ
А ТАКЖЕ ПРО SQL ИНЪЕКЦИИ

Конфиденциальные файлы

Каков кошмар web-программиста?

- Утечка дампов баз данных клиентов с персональными данными?
- Утечка учетной записи для доступа к админке сервера?
- Или может банальное появление кода его сайта в открытом доступе?

Security through obscurity

- Перевод: Безопасность через неясность
- Основополагающий принцип любых систем сомнительной безопасности: в исходном коде твоего продукта не найдут багов, если ты его никому не предоставишь
 - В мире криптографии такой принцип очень не любят: существует даже Принцип Керкгоффа, утверждающий, что в криптосистеме должны быть неизвестны только секретные параметры, а сам алгоритм должен быть открыт
- В мире проприетарного ПО такой принцип наоборот пользуется огромной популярностью
 - Код DroidGuard, ранее использовавшегося Android для проверки SafetyNet, был намеренно запутан
 - Сканеры Web-уязвимостей считают возможность получения исходников проблемой безопасности

Security through obscurity и Web

- В случае с Web-приложениями данный подход особенно актуален, поскольку код backend-части приложения выполняется на сервере и обычно недоступен для загрузки
 - Это позволяет добавить в код дополнительных параметров для отладки, о которых пользователь будет не в курсе
 - Например, можно включать отладочный вывод при наличии параметра: `index.php?debug=True`
 - Также можно разместить в коде различные встроенные учетные записи с максимальными правами для администраторов
 - В конфигурационные файлы можно добавить различных ключей API для интеграции с другими ресурсами, или же пароли от базы данных
 - И это не считая настоящих уязвимостей, которые становятся не так заметны
- В общем, получение пользователем доступа к коду сайта это зачастую катастрофа для его владельца

Итак, что секретного может быть на сайте?

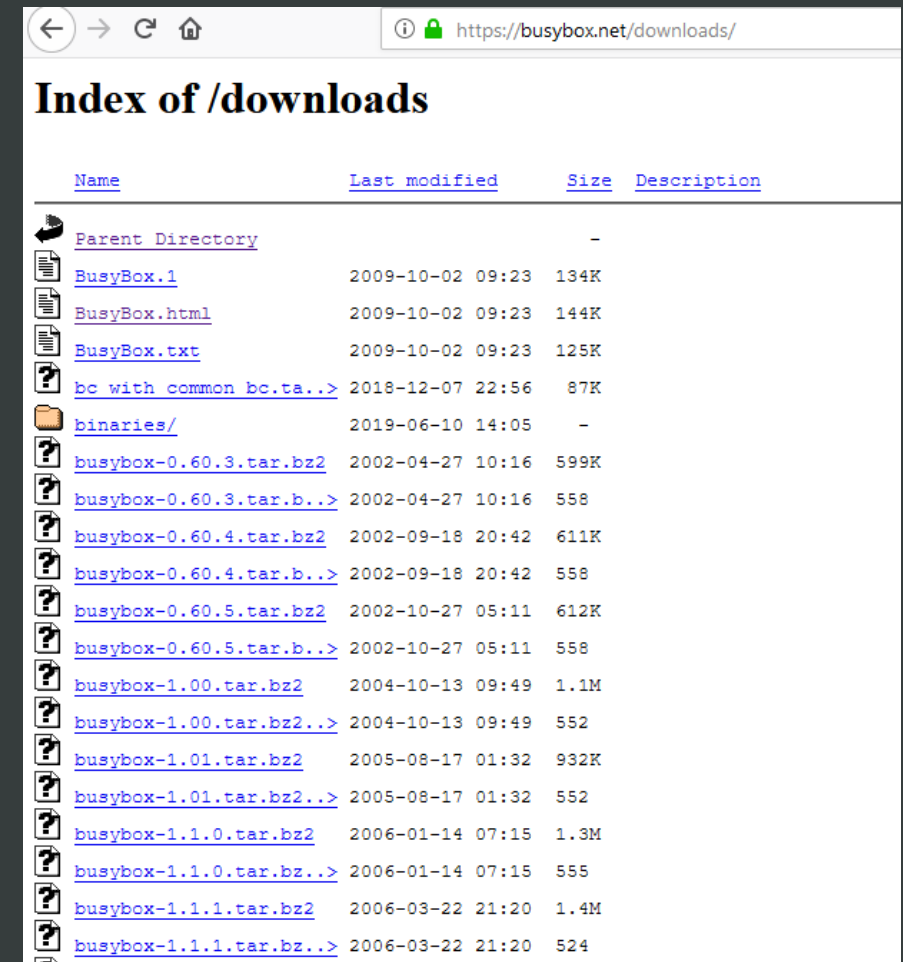
- Секретные подпути, например `/s3cr3tny_admin`
- Секретные параметры, например `/index.php?mode=admin`
- Секретные учетные записи
- Ключи для шифрования Cookies
 - Некоторые сайты хранят всю информацию о пользовательской сессии в Cookies пользователя в зашифрованном и подписанном виде (например с использованием JWT)
- Пароли для доступа к базам данных
- Многое другое















Как раздобыть что-нибудь секретное?

- При помощи банального листинга директории
- При помощи robots.txt
- При помощи исходного кода страницы
- При помощи перебора путей
- При помощи различных бэкапов, оставленных администратором сервера
- При помощи файлов систем контроля версий
- При помощи .DS_Store

Листинг директории

- Стандартная возможность многих веб-серверов, позволяет просматривать список файлов в директории
 - Вы могли встречаться с таким листингом, качая, например, какой-нибудь GPL-софт
- Работает только если в папке нет файла `index.html` / `index.htm` / `index.php` и т.д. и возможность листинга не заблокирована в глобальных настройках или `.htaccess`
 - То есть обычно не на главной странице, а в папках в духе `/images/`
- Зачастую отсутствует в фреймворках современных языков программирования (Node.js, Python)



<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 Parent Directory		-	
 BusyBox.1	2009-10-02 09:23	134K	
 BusyBox.html	2009-10-02 09:23	144K	
 BusyBox.txt	2009-10-02 09:23	125K	
 bc with common bc.ta..>	2018-12-07 22:56	87K	
 binaries/	2019-06-10 14:05	-	
 busybox-0.60.3.tar.bz2	2002-04-27 10:16	599K	
 busybox-0.60.3.tar.b..>	2002-04-27 10:16	558	
 busybox-0.60.4.tar.bz2	2002-09-18 20:42	611K	
 busybox-0.60.4.tar.b..>	2002-09-18 20:42	558	
 busybox-0.60.5.tar.bz2	2002-10-27 05:11	612K	
 busybox-0.60.5.tar.b..>	2002-10-27 05:11	558	
 busybox-1.00.tar.bz2	2004-10-13 09:49	1.1M	
 busybox-1.00.tar.bz2..>	2004-10-13 09:49	552	
 busybox-1.01.tar.bz2	2005-08-17 01:32	932K	
 busybox-1.01.tar.bz2..>	2005-08-17 01:32	552	
 busybox-1.1.0.tar.bz2	2006-01-14 07:15	1.3M	
 busybox-1.1.0.tar.bz..>	2006-01-14 07:15	555	
 busybox-1.1.1.tar.bz2	2006-03-22 21:20	1.4M	
 busybox-1.1.1.tar.bz..>	2006-03-22 21:20	524	

Robots.txt

- Данный файл расположен в корневом каталоге сайта: `example.com/robots.txt`
- Текстовый файл, который содержит параметры индексирования сайта для роботов поисковых систем
- Используется для исключения страниц, появление которых в поисковиках нежелательно
 - Неправильная настройка может привести к утечке пользовательских данных, если их браузер отправляет все посещенные URL поисковому боту (вроде бы так делает браузер от Яндекс)
 - Также администратор может записать туда секретные URL, например от админки, чтобы они не гуглились
 - Наивно полагая, что этот файл могут читать только поисковые боты :)

Robots.txt

- Пример такого файла:

```
User-agent: Googlebot  
Disallow: /private/
```

- Директива Disallow запрещает доступ к данной директории
 - Disallow: / запрещает всё
- Остальные директивы специалистам по ИБ не слишком нужны
 - Ну может быть кроме директивы Sitemap, которая указывает на XML-файл sitemap (обычно sitemap.xml) со списком страниц, которые владелец сайта рекомендует проиндексировать поисковику

Исходный код страницы

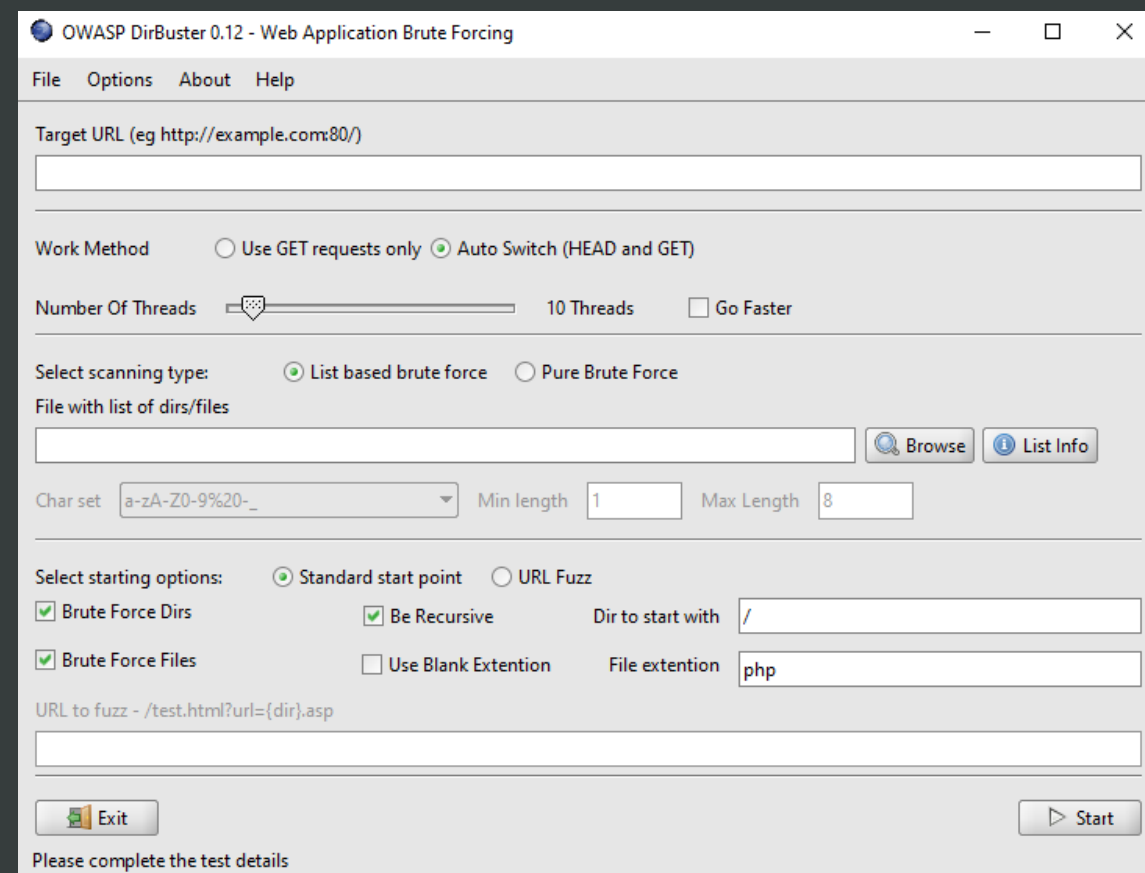
- В случае, если программисты пишут код HTML-страницы вручную, они вполне могут пользоваться HTML-комментариями, чтобы прятать различные ссылки, кнопки и формы, например, если функциональность была удалена или наоборот еще не была реализована
- Посмотреть исходный код можно при помощи ПКМ - Исходный код страницы или Ctrl-U
- Обычно ничего совсем секретного там не бывает, но все равно стоит проверить

```
...
<div class="col-lg-3 col-md-3 col-sm-4 col-xs-7">
<!-- Logo -->
    <!-- 'start_frame_cache_SEk85C' -->
<!-- 'end_frame_cache_SEk85C' -->
    <!---->
    <!---->
    <!---->
    </div>
<div class="hidden-lg hidden-md hidden-sm col-xs-3">
    <div class="change_lang_wrap">
...

```

Перебор путей

- Что если листинга директории нет, но очень хочется?
- Почему бы просто не перебрать все имена файлов?
- OWASP DirBuster предназначен как раз для этого
 - Умеет перебирать файлы и директории по словарю
 - Умеет и без словаря, но лучше нет
- На нормальных CTF он не нужен
 - Организаторы могут забанить вас за флуд



Бэкапы

- Бэкапы можно разделить на два вида:
 - Бэкапы, созданные администратором (или какими-то инструментами специально для бэкапа)
 - Бэкапы, которые создались сами собой

Бэкапы, созданные администратором

- Обычно это файлы, содержащие все страницы и данные сайта в формате архива
 - В Windows это обычно .zip
 - В Linux это обычно .tar или .tar.gz, хотя возможны варианты
- Также это могут быть бэкапы баз данных:
 - .sql, например
- Поскольку такие файлы особенно часто встречаются в корневой директории, их вряд ли можно найти при помощи листинга директории
 - Поэтому основным способом поиска таких файлов является перебор имен
- Пример пути к такому файлу: `example.com/backup.tar.gz`

Бэкапы, создавшиеся сами собой

- Возникают при использовании текстовых редакторов прямо в папке сайта
- Могут иметь следующие имена (для оригинального файла `file.txt`):
 - `file.txt~`, самый популярный вариант (используется, например, VIM, gedit и emacs, но по умолчанию скорее всего выключено)
 - `.file.txt.swp` и `#file.txt#`, создаются VIM и Emacs на время редактирования файла (или если из них выйти по `kill -9 / kill`)
 - `file.txt.bak`, `file.txt.old`
 - `.txt.orig` (иногда остаются при применении патчей к сайту при помощи GNU patch)
- Замечательное свойство таких бэкапов:
 - При доступе к файлам `.php` на сервере, поддерживающем PHP, они обычно запускаются как программы и пользователь видит только результат их исполнения
 - Однако файлы бэкапов можно скачать «как есть» и посмотреть исходный код сайта, потому что их расширение уже не `.php` (или другое исполняемое)

Системы контроля версий

- Данная уязвимость касается любителей деплоить сайты напрямую из системы контроля версий
- Конструкция вида «`git clone git@github.com:user/website.git .`» может сделать ваш сайт небезопасным буквально в одну команду
 - Других систем контроля версий, например, SVN, это тоже касается в различной степени

Системы контроля версий. Git

- Эта система контроля версий замечательна тем, что при каждом клонировании репозитория по умолчанию скачивает его полную копию со всей историей
 - Все эти данные хранятся в папке `.git`
 - Папка `.git` при неосторожном деплое может оказаться в корне вашего веб-сервера
- Для получения копии репозитория при наличии папки `.git` даже не нужны права на листинг директорий
 - Имея 403 при попытке зайти в папку `.git` вы все равно можете скачать репозиторий до тех пор пока у вас есть доступ к файлам
- Из данных репозитория можно достаточно тривиально восстановить исходный код сайта
 - Причем можно еще и перемещаться в истории, находя, например, данные, которые были оттуда стерты по разным причинам (в том числе по причинам секретности)

Системы контроля версий. Инструменты

- Несмотря на то что с форматом файлов системы Git можно работать и вручную, это не очень удобно
- Существует инструмент `dvcs-ripper`, позволяющий загружать файлы из следующих систем контроля версий:
 - GIT
 - Mercurial / HG
 - Bazaar / bzt
 - SVN
 - CVS
- Скачать его можно со страницы <https://github.com/kost/dvcs-ripper>
 - Не забывайте следовать инструкциям по установке, иначе получите ошибку Perl

.DS_Store

«Think Different»

- Файл, используемый в Finder / Spotlight
- Создается автоматически для хранения различных атрибутов папки
 - К числу «атрибутов» также относится и список файлов в папке
- При наличии на сервере позволяет также установить список файлов в папке без листинга директории
 - Существует даже написанная для этого программа,
https://github.com/lijiejie/ds_store_exp
- Thumbs.db из Windows можно считать блеклой тенью этого файла, т.к. от него не так много толку (не считая того, что на Windows 10 он по умолчанию не создается)

SQL-инъекции

Уязвимости типа «инъекции»

- Крайне важный и широкий тип уязвимостей
- Топ-3 в списке OWASP (в 2017 году – Топ-1)
- Возникает, когда ввод пользователя при определенных обстоятельствах может быть воспринят как код (будь то код запроса, шаблона, командной строки или даже веб-страницы)
- Примеры:
 - SQL-инъекции (SQLi)
 - NoSQL-инъекции (NoSQLi)
 - Инъекции шаблонов (SSTI)
 - Инъекции команд
 - CRLF инъекции

Уязвимости типа «инъекции»

Ввод нормального пользователя

Могу выполнять запросы вида
ДАЙ ФАЙЛ С ИМЕНЕМ X

X = "Cat.jpg"

Отдаю ФАЙЛ С ИМЕНЕМ
Cat.jpg

Ввод хакера

Могу выполнять запросы вида
ДАЙ ФАЙЛ С ИМЕНЕМ X

X = "Cat.jpg И ВСЕ СВОИ
СЕКРЕТНЫЕ ДАННЫЕ"

Отдаю ФАЙЛ С ИМЕНЕМ
Cat.jpg И ВСЕ СВОИ СЕКРЕТНЫЕ
ДАННЫЕ

SQL

- Язык работы с базами данных и в целом тип баз данных
- SQL базы данных являются реляционными, т.е. хранят данные в таблицах
 - Таблицы, в свою очередь, содержат колонки разных типов
- И таблицы и колонки имеют имена
- В некоторых базах данных есть также больший термин «база данных», которых может быть несколько (а они, в свою очередь, хранят в себе таблицы, колонки и данные)
- SQL является псевдоестественным языком, запросы на нем больше похожи на осмысленные предложения, чем на выражения языков программирования

SQL

ID	login	password
0	Admin	Admin123
1	Vasya	qwerty
2	Petya	asdasddf
3	Hacker1337	leethaxxor1337

Типичная таблица Users

SQL. Select

ID	login	password
0	Admin	Admin123
1	Vasya	qwerty
2	Petya	asdasddf
3	Hacker1337	leethaxxor1337

```
SELECT ID FROM Users
```

Данный запрос вернет все ID из базы данных (по одному на строку)

SQL. Select

ID	login	password
0	Admin	Admin123
1	Vasya	qwerty
2	Petya	asdasddf
3	Hacker1337	leethaxxor1337

```
SELECT ID FROM Users WHERE login='Admin' AND password='Admin123'
```

Данный запрос вернет единственную строку, содержащую поле ID равное 0

SQL. Select

ID	login	password
0	Admin	Admin123
1	Vasya	qwerty
2	Petya	asdasddf
3	Hacker1337	leethaxxor1337

```
SELECT ID, login FROM Users
```

Данный запрос вернет все ID и логины из базы данных

SQL. Select *

ID	login	password
0	Admin	Admin123
1	Vasya	qwerty
2	Petya	asdasddf
3	Hacker1337	leethaxxor1337

```
SELECT * FROM Users WHERE login='Admin' AND password='Admin123'
```

Данный запрос вернет строку целиком (со всеми столбцами), в данном случае состоящую из ID, login и password

SQL. Union

ID	login	password
0	Admin	Admin
1	Vasya	qwerty
2	Petya	asdasddf
3	Hacker1337	leethaxxor1337

ID	login	password
101	Admin	Admin2
102	Dima	sgnjif
103	Lena	rReiujioeiogr
104	Hacker2007	Leethaxxor1337

```
SELECT ID FROM Users WHERE login='Admin' AND password='Admin' UNION  
SELECT ID FROM OldUsers WHERE login='Admin' AND password='Admin'
```

Данный запрос вернет строку, состоящую из ID=0

```
SELECT ID FROM Users WHERE login='Dima' AND password='sgnjif' UNION  
SELECT ID FROM OldUsers WHERE login='Dima' AND password='sgnjif'
```

Вернет, в свою очередь, ID=102

Типы запросов SQL

- SELECT – выбирает данные из таблицы
 - Пример: `SELECT password FROM users WHERE login='user'`
 - Выбирать можно без таблицы: `SELECT 123` (или `SELECT 123 from DUAL`)
- INSERT – добавляет новые данные
 - Пример: `INSERT INTO users (login, password) VALUES ('newuser', 'somepassword')`
- UPDATE – изменяет существующие данные
 - Пример: `UPDATE users SET password='newpassword' WHERE login='newuser'`
- DELETE – удаляет данные
 - Пример: `DELETE FROM users WHERE login='newuser'`

Важные конструкции SQL

- NULL – пустое значение в поле, принципиально отличается от нуля или пустой строки, конвертируется к любому типу
- " --", #, /* – начало комментариев в различных диалектах SQL
 - Комментарии очень важны чтобы сделать оставшуюся после SQL-инъекции часть запроса не влияющей на результат / корректность запроса
- /**/ – комментарий, который может использоваться вместо пробела, если пробелы по какой-то причине недоступны
- &&, || – альтернативы AND и OR

SQL

ID	login	password
0	Admin	Admin123
1	Vasya	qwerty
2	Petya	asdasddf
3	Hacker1337	leethaxxor1337

```
SELECT * FROM Users WHERE login='Admin' AND password='Admin123'
```

А что если вместо Admin была бы строка, в которой самой есть апостроф?

```
SELECT * FROM Users WHERE login='Admin'' AND password='Admin123'
```

(конкретно в данном случае была бы ошибка)

SQL-инъекции

- Возникают в результате неправильной вставки контролируемых пользователем данных в SQL запросы.
- Пример уязвимой части кода (PHP / MySQL):

```
mysql_query("SELECT * FROM Users WHERE login = '${_GET['login']}'  
AND password = '${_GET['password']}'");
```

- Если выполнить запрос /login?login='or'1'='1'or'1'='1&password= то получим следующий SQL запрос:

```
SELECT * FROM Users WHERE login = ' 'or'1'='1'or'1'='1' AND  
password = ' '
```

- В результате пользователь залогинится под первым попавшимся пользователем без пароля

SQL-инъекции

- Чтобы протестировать сайт на SQL-инъекции, можно просто ввести во все поля сайта подозрительных символов и конструкций:
 - '
 - "
 - #
 - ;
 -)
 - \
 - 'or'1'='1'or'1'='1
- Данные символы по одиночке при наличии SQL-инъекции скорее всего вызовут ошибку, которую можно будет отличить от нормального поведения сайта

SQL-инъекции

- Виды SQL-инъекций:
 - UNION-based SQL инъекция
 - Boolean-based SQL инъекция (также называемая слепой SQL-инъекцией)
 - Time-based SQL инъекция (подвид слепой SQL-инъекции)
 - Error-based SQL инъекция
 - Stacked queries SQL инъекция

UNION-based SQL инъекция

- Самый простой сценарий, рассмотрен на первом слайде темы
- Основан на ключевом слове UNION
- Работает, если данные, получаемые контролируемым запросом куда-нибудь выводятся (например на экран, если это поиск по сайту)
- Пример кода

```
SELECT id, name FROM products LIMIT 10 OFFSET 1 UNION SELECT NULL,  
topsecret FROM secret;
```

- Типы данных объединяемых столбцов и их количество должны совпадать
 - Столбцы, которые вам не интересны, можно забить NULL

Определение числа столбцов

- Достаточно жизненная тема при проведении атак, поскольку при несовпадении числа столбцов UNION приведет к ошибке
- В большинстве эту проблему можно решить добавив нужное количество колонок NULL в запрос
 - Поскольку NULL конвертируется к любому типу, это сработает
- Также можно применить ORDER BY или GROUP BY к оригинальной части запроса, в качестве аргумента они могут принимать индекс столбца, а не имя
 - Это позволяет перейти к бинарному поиску числа столбцов вместо линейного
 - Пример использования:

```
SELECT id, name FROM products WHERE name= ' ' ORDER BY 10 -- asd'
```



А вот и комментарий пригодился

Boolean-based SQL инъекция

- Менее простой, но более популярный сценарий
- Работает, если от данных, получаемых контролируемым запросом, зависит поведение сайта
 - Например, где-нибудь на странице логина
- Пример кода

```
SELECT * FROM users WHERE login='admin' AND password='1' OR  
(SELECT SUBSTRING(password, 1, 1) FROM users LIMIT 1)='1'
```



Подзапрос, истинность этого условия мы проверяем

- Данные из таблицы в данном случае получаются побитово
 - Делать это руками достаточно утомительно, поэтому существуют автоматические инструменты и фреймворки для таких атак

Time-based SQL инъекция

- В принципе, очень похож на предыдущий сценарий
- Работает, если от данных, получаемых контролируемым запросом, зависит время загрузки страницы
 - Например, если какая-нибудь аналитика сайта чего-нибудь записывает в фоне, но пользователь этого не видит
- Пример кода

```
SELECT * FROM analytics WHERE ip='1.3.3.7' AND  
useragent='SomeFakeBrowser' OR 1=(1-SLEEP(15)) AND '1'='1'
```

- Считается самым неудобным в эксплуатации способом, поскольку никто не гарантирует, что сайт тормознул именно из-за запроса, а не из-за проблем с сетью, например

Error-based SQL инъекция

- Особый вид SQL-инъекции, доступен если на странице есть вывод ошибок (например включен `display_errors` языка PHP)
- Подразумевает вывод нужных данных в сообщении об ошибке
- Пример кода

```
SELECT * FROM analytics WHERE ip='1.3.3.7' AND  
useragent='SomeFakeBrowser' OR 1=(select exp(~(select*from(select  
user()))x))) AND '1'='1'
```

- Очень хороший способ для ручного осуществления атак, если вы нашли error-based SQL инъекцию, дальше в принципе можно не искать
- В случае если вывод ошибок недоступен, можно считать аналогом слепой SQL-инъекции

Stacked queries SQL инъекция

- Особый вид SQL-инъекции, доступен если база данных позволяет осуществление последовательных запросов в рамках одного вызова
 - Запросы в этом случае разделяются точкой с запятой
- Позволяет выполнять отличные от текущего запроса команды
 - Т.е. вместо SELECT можно выполнить DROP
- Пример кода

```
SELECT id, name FROM products LIMIT 10 OFFSET 1; DROP TABLE products;
```

- Довольно редко встречается, в большинстве библиотек для работы с базами данных возможности выполнять несколько запросов за раз нет

SQL-инъекции в INSERT

- Важно помнить, что SQL-инъекции могут встречаться не только в запросах SELECT, но и в других, например, INSERT
- Результат такого запроса, соответственно, можно получить, повторно запросив из базы поле, которое было вставлено (если это, конечно, возможно)
- Пример:

```
INSERT INTO notes (login, note) VALUES ('haxxor',''),  
('haxxor',(SELECT secret FROM topsecret)) -- '
```

Как узнать имена таблиц и колонок?

- Подобрать ``_(`\`)_`/``
- Достать из специальных информационных таблиц, которые всегда существуют
 - В случае MySQL / MariaDB это `information_schema`
 - В случае PostgreSQL это `pg_catalog`
 - В случае SQLite3 это `sqlite_master`

Как защититься от SQL-инъекций?

- Использовать ORM (Object-Relational Mapping)
- Использовать параметризованные запросы (PDO)
- В конце концов, использовать официальные функции фильтрации пользовательских данных, например `mysql_real_escape_string`
- Не использовать SQL...

Как не защититься от SQL-инъекций

- Пытаться изобретать велосипеды для фильтрации пользовательских данных
- Примером крайне опасного велосипеда является удаление всех апострофов из пользовательского кода, поскольку это отнюдь не обязательно защитит вас от SQL-инъекций!
 - Следующий запрос обходит авторизацию без использования апострофов:

```
SELECT ID FROM users WHERE login='hacker\' AND password=' OR 1=1 -  
- commented';
```

- Существует целая статья, посвященная борьбе с велосипедостроителями:
<https://websec.wordpress.com/2010/03/19/exploiting-hard-filtered-sql-injections/>



Обратите внимание на год :)

Полезные программы. sqlmap

- sqlmap – ведущий инструмент для автоматического проведения атак
 - Умеет определять вид базы данных, используемой сервером
 - Умеет перечислять таблицы и столбцы в базах, где это возможно
 - Умеет проводить атаки всех видов
 - Умеет при наличии прав даже исполнять на хосте команды или загружать с него файлы
 - Установить можно через пакетный менеджер Python pip или через apt-get
- Для проведения POST запросов используется флаг --data
- В адрес или POST можно поставить звездочку (*) в то место, где, как вы уверены, находится SQL-инъекция
- Остальные флаги можно посмотреть через --help
 - А чтобы посмотреть вообще все флаги, используйте -hh

Полезные программы. sqlmap

- Иногда sqlmap тупит при проведении слепых SQL-инъекций, для того чтобы ему помочь рекомендуется использовать флаг `--string=` позволяющий указать, какая подстрока содержится в ответе, характеризующем положительный результат
 - Или `--not-string`, чтобы указать ответ, характеризующий отрицательный результат (сюда хорошо подойдет значение вроде "Wrong password")
- sqlmap по умолчанию не осуществляет тесты, которые он считает излишними или опасными, чтобы он делал и их, ему можно передать параметры `--risk` и `--level`
 - `--level` — число от 1 до 5 (по умолчанию 1), отвечает за тщательность проверок
 - `--risk` — число от 1 до 3 (по умолчанию 1), отвечает за то, как сильно может пострадать сервер от ваших проверок (например SQLi в UPDATE с risk, равным 3, приведет к обновлению всех полей базы)
 - Для CTF и похожих задач вполне подойдет `--level 5 --risk 3` :)

Полезные программы. Ваш любимый браузер

- Вообще такие атаки можно зачастую проводить прямо из браузера, поэтому он может оказаться полезен без всяких дополнительных условий
 - Ввод 'or'1'='1 во все поля не требует каких-либо особых инструментов или дополнений
- Чтобы узнать, какие именно запросы и с какими параметрами осуществляла страница (а затем, например, передать их как параметры -u и --data инструменту sqlmap), можно воспользоваться инструментами разработчика в браузере
 - Их можно открыть при помощи Ctrl+Shift+I
- После этого можно перейти на вкладку "Сеть" и выбрать интересующий вас запрос и посмотреть список параметров
 - Или воспользоваться ПКМ - Копировать - Копировать как Curl (POSIX), там вы сможете найти и URL (без параметра) и POST-параметры, если они есть (в параметре data-raw)

Полезные программы. Ваш любимый браузер

Команда curl, скопированная из браузера:

```
curl 'http://example.com/login' -H 'User-Agent: Mozilla/5.0 (Windows NT 10.0; x64; rv:81.0) Gecko/20100101 Firefox/81.0' -H 'Accept:text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8' -H 'Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3' --compressed -H 'Content-Type: application/x-www-form-urlencoded' -H 'Origin: http://example.com' -H 'Connection: keep-alive' -H 'Referer: http://example.com/login' -H 'Upgrade-Insecure-Requests: 1' -H 'Pragma: no-cache' -H 'Cache-Control: no-cache' --data-raw 'login=puti&password=plz'
```

Вызов sqlmap:

```
sqlmap -u 'http://example.com/login' --data 'login=puti&password=plz'
```

Не забывайте кавычки в bash

PayloadsAllTheThings

- Удобный GitHub-репозиторий, где можно найти информацию полезную для эксплуатации самых различных уязвимостей
 - <https://github.com/swisskyrepo/PayloadsAllTheThings/>
- В случае SQL-инъекций там можно найти информацию об особенностях различных баз данных
 - Например, какие там информационные таблицы
 - Или вектора для error-based SQLi (но они не везде есть)

Спасибо за внимание!
Задачи доступны на

nsuctf.ru

- Пожалуйста, используйте имя пользователя формата “Фамилия Имя”
 - e-mail можно забить любой, сервером он не проверяется
- Для вопросов по задачам рекомендую присоединиться к @NSUCTF в Telegram
 - Только, пожалуйста, без спойлеров