

Запись о встрече

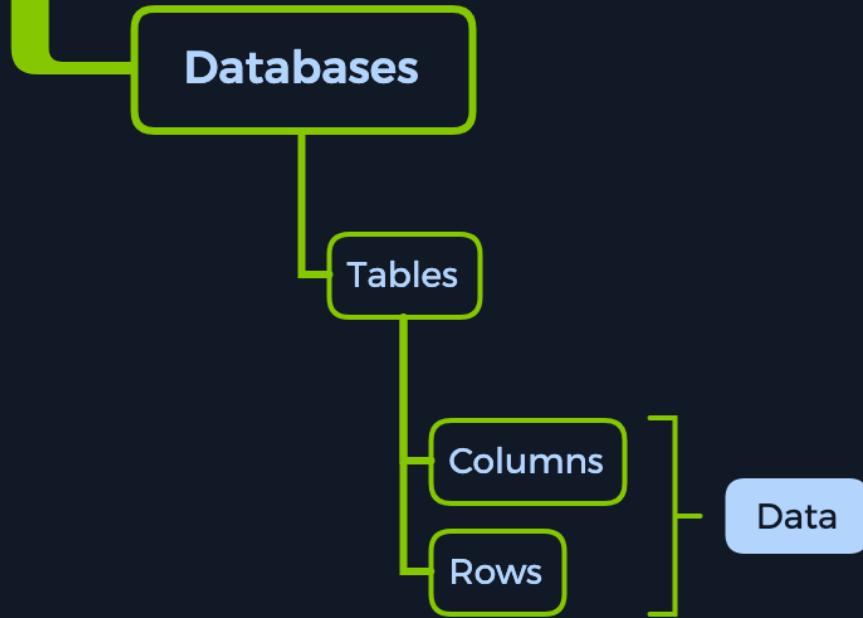
Подготовил: One-nine9, ilinor

Вступление

Назначение — это окно, которое в основном ориентировано на веб-приложение. Более конкретно, мы узнаем, как выполнить SQL-инъекцию против Базы данных SQL включенное веб-приложение. Наша цель запускает веб-сайт с возможностями поиска по серверной базе данных, содержащей доступные для поиска элементы, уязвимые для этого типа атак. Не все элементы в этой базе данных должны быть видны любому пользователю, поэтому разные привилегии на веб-сайте предоставят вам разные результаты поиска.

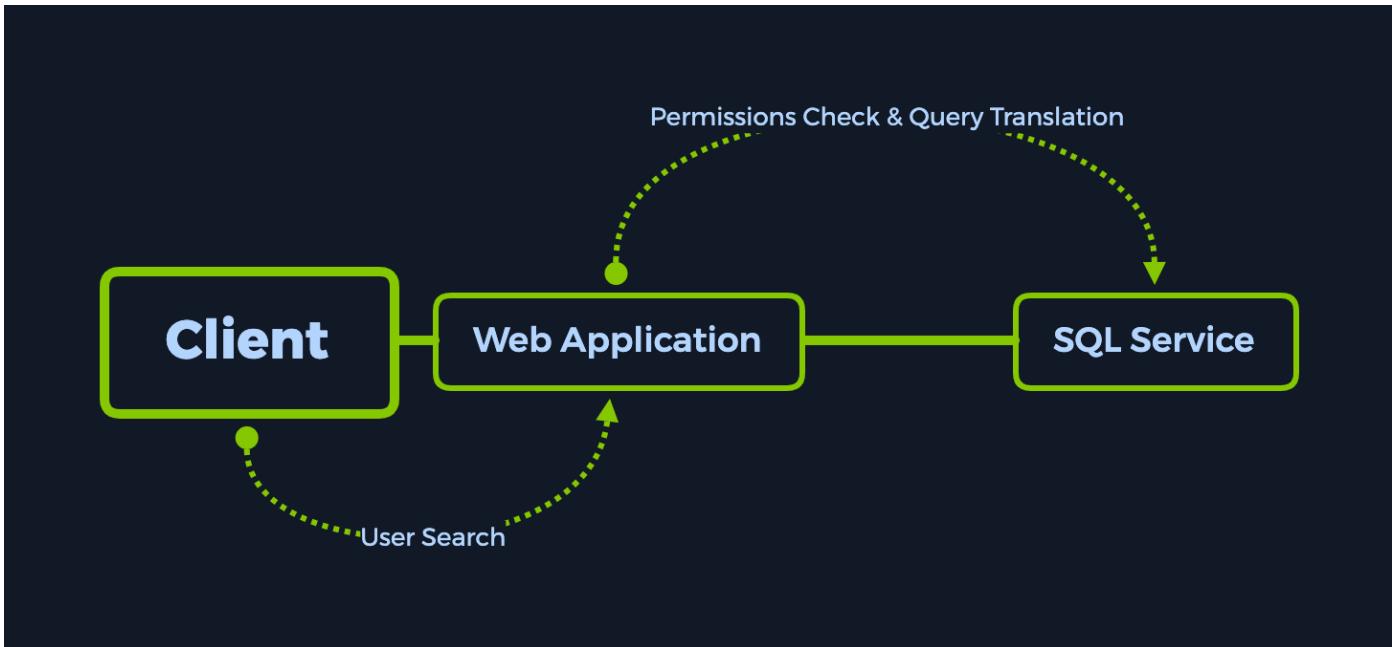
Гипотетически администратор веб-сайта будет искать пользователей, их электронные письма, платежную информацию, адрес доставки и другие данные. Напротив, простой пользователь или посетитель, не прошедший проверку подлинности, может иметь разрешение только на поиск товаров в продаже. Эти столы информации будут отдельно. Однако для злоумышленника, обладающего знаниями об уязвимостях веб-приложений — в данном случае, в частности, о SQL-инъекциях — разделение между этими таблицами ничего не будет значить, поскольку он сможет использовать веб-приложение для прямого запроса любой таблицы, найденной в базе данных SQL веб-сервера.

SQL Service



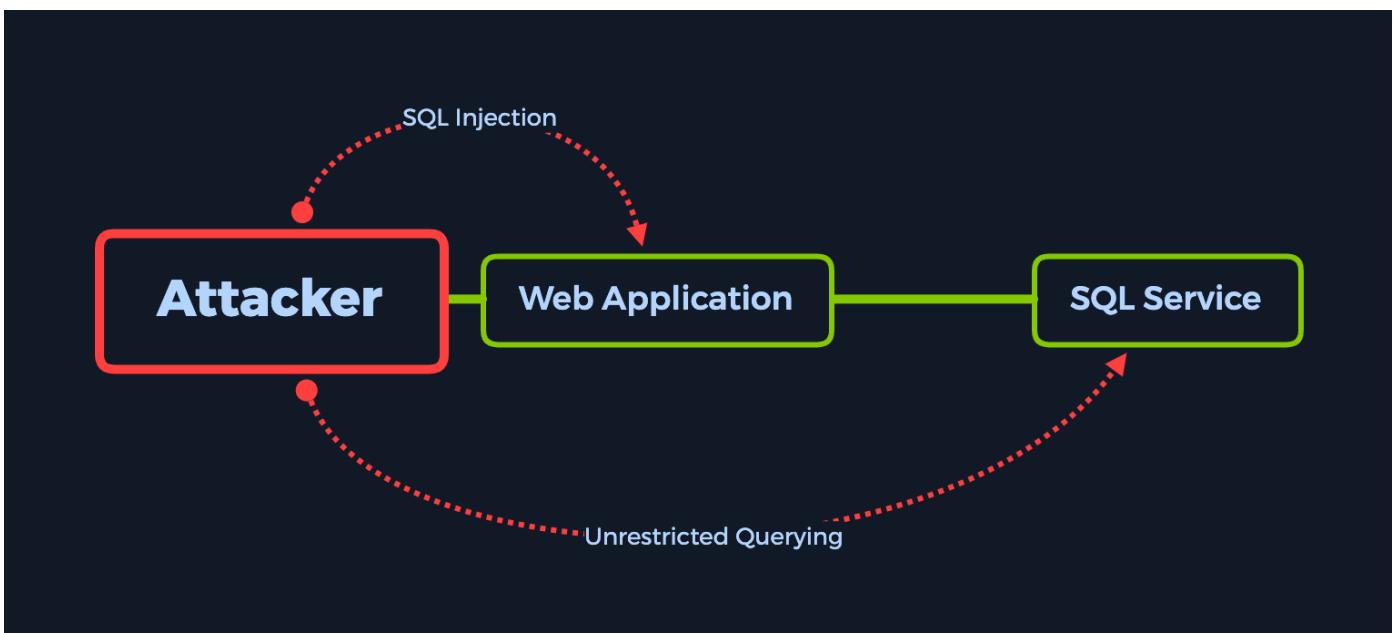
Отличным примером того, как обычно работает служба SQL, является процесс входа в систему, используемый для любого пользователя. Каждый раз, когда пользователь хочет войти в систему, веб-приложение отправляет ввод страницы входа (комбинацию имени пользователя и пароля) в службу SQL, сравнивая ее с сохраненными записями базы данных для этого конкретного пользователя. Предположим, что указанные имя пользователя и пароль совпадают с любой записью в базе данных. В этом случае служба SQL сообщит об этом веб-приложению, которое, в свою очередь, войдет в систему пользователя, предоставив ему доступ к закрытым частям веб-сайта. После входа в систему веб-приложение установит пользователю специальное разрешение в виде файла cookie или токена аутентификации, которое связывает его присутствие в сети с его аутентифицированным присутствием на веб-сайте. Этот файл cookie хранится как локально, в хранилище браузера пользователя, так и на веб-сервере.

После этого, если пользователь хочет выполнить поиск в элементах списка, перечисленных на странице, для чего-то конкретного, он введет имя объекта в строку поиска, что запустит ту же службу SQL для выполнения SQL-запроса от имени пользователя. Предположим, что запись для искомого элемента существует в базе данных, обычно в другой таблице. В этом случае связанная информация извлекается и отправляется в веб-приложение для представления пользователю в виде изображений, текста, ссылок и других типов, таких как комментарии и обзоры.



Причина, по которой веб-сайты используют базы данных, такие как MySQL, MariaDB или другие, заключается в том, что данные, которые они собирают или обслуживают, должны где-то храниться. Данные могут быть именами пользователей, паролями, публикациями, сообщениями или более конфиденциальными наборами, такими как [РП \(персональные идентифицируемые информации\)](#), которая защищена международными законами о конфиденциальности данных. Любое предприятие, которое пренебрегает защитой РП своих пользователей, приветствуется очень большими штрафами от международных регуляторов и агентств по обеспечению конфиденциальности данных.

Внедрение SQL — распространенный способ эксплуатации веб-страниц, использующих вводимые пользователем данные. При неправильной настройке можно использовать эту атаку для эксплуатации известной уязвимости, что очень опасно. Существует множество различных методов защиты от SQL-инъекций, некоторые из которых включают проверку ввода, параметризованные запросы, хранимые процедуры и реализацию WAF (брандмауэра веб-приложений) на периметре сети сервера. Однако могут быть обнаружены случаи, когда ни одно из этих исправлений не установлено, поэтому этот тип атаки распространен, согласно [Top-10 OWASP](#) список веб-уязвимостей.



Давайте посмотрим на нашу цель и посмотрим, соответствует ли она требованиям для этого сценария.

перечисление

Сначала мы выполняем сканирование nmap, чтобы найти открытые и доступные порты и их службы. Если в синтаксисе команды не указан альтернативный флаг, nmap просканирует наиболее распространенные 1000 TCP-портов на наличие активных служб. В нашем случае это подойдет.

Кроме того, нам потребуются привилегии суперпользователя для запуска приведенной ниже команды с **-sC ИЛИ -sV** флагами. Этот факт связано с тем, что сканирование сценария (-sC) и определение версии (-sV) считаются более интразивными методами сканирования цели. Это приводит к более высокой вероятности быть пойманым устройством безопасности периметра в целевой сети.

- sC: выполняет сканирование сценария с использованием набора сценариев по умолчанию. Это эквивалентно -- script=default. Некоторые скрипты в этой категории считаются навязчивыми и не должны запускаться в целевой сети без разрешения.

- sV: включает определение версии, которое будет определять, какие версии работают на каком порту.

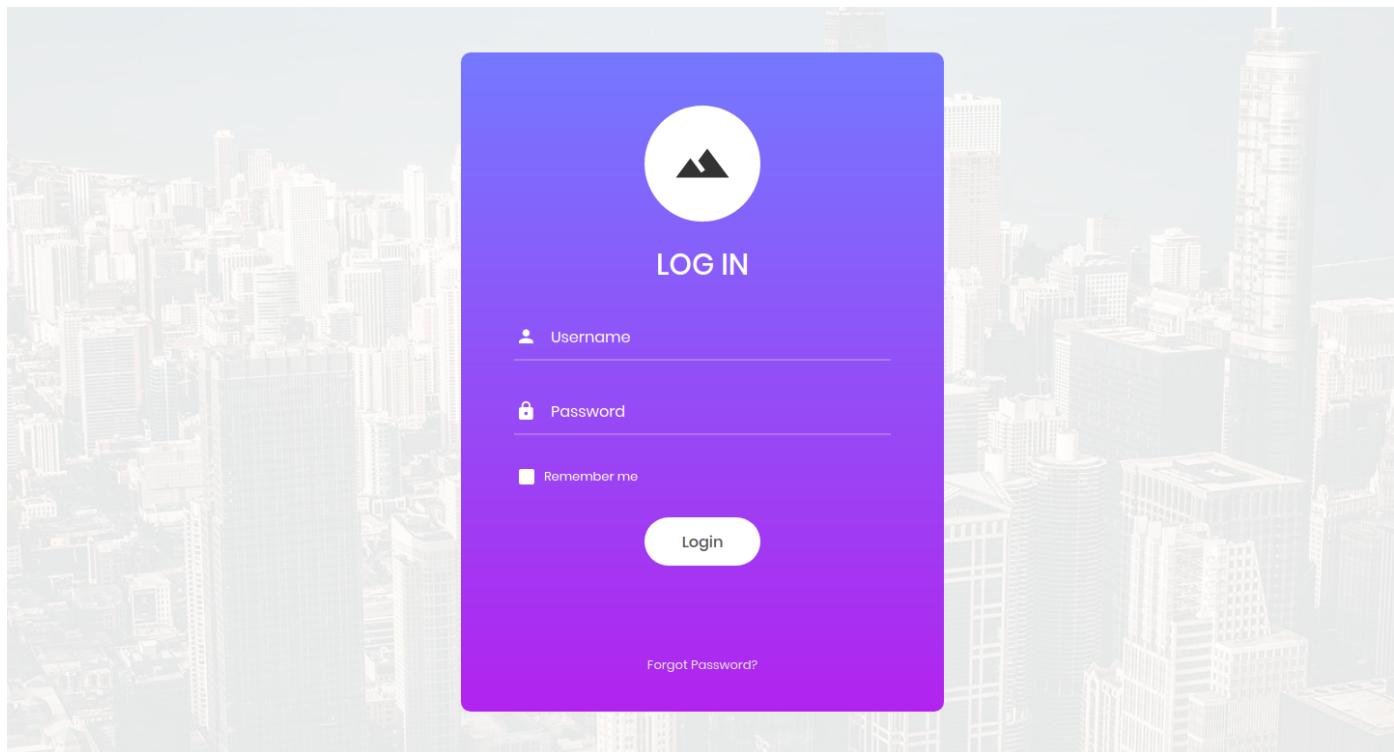
```
$ sudo nmap -sC -sV {target_IP}
PORT      STATE SERVICE VERSION
80/tcp    open  http    Apache httpd 2.4.38 ((Debian))
|_http-server-header: Apache/2.4.38 (Debian)
|_http-title: Login
```

Единственный открытый порт, который мы обнаруживаем, — это порт 80 TCP, на котором работает Апач httpd версия сервера 2.4.38.

Apache HTTP Server — это бесплатное приложение с открытым исходным кодом, которое запускает веб-страницы на физических или виртуальных веб-серверах. Это один из самых популярных HTTP-серверов, который обычно работает на стандартных HTTP-портах, таких как порты 80 TCP, 443 TCP, и альтернативно на HTTP-портах, таких как 8080 TCP или 8000 TCP. HTTP означает протокол передачи гипертекста, и это протокол прикладного уровня, используемый для передачи гипермедиа-документов, таких как HTML (язык гипертекстовой разметки).

Сканирование nmap предоставило нам точную версию службы Apache httpd — 2.4.38. Обычно хорошей идеей будет поиск версии службы в популярных онлайн-базах уязвимостей, чтобы узнать, существует ли какая-либо уязвимость для указанной версии. Однако в нашем случае эта версия не содержит ни одной известной уязвимости, которую мы могли бы потенциально использовать.

Для дальнейшего перечисления службы, работающей на порту 80, мы можем перейти непосредственно к IP-адресу цели из нашего браузера, внутри нашего экземпляра Pwnbox или виртуальной машины.



Примечание: Начиная с приведенного ниже, мы будем изучать концепцию перебора различных каталогов для включения в нашу фазу перечисления, что не поможет нам использовать цель и считается необязательным, но полезным шагом. Этот раздел считается необязательным для уровня 0, но представляет собой ценный шаг в процессе оценки. Если вы хотите пропустить это, вы можете прокрутить вниз до [плацдарм](#) раздел.

Введя IP-адрес цели в поле URL нашего браузера, мы сталкиваемся с веб-сайтом, содержащим форму входа. Формы входа используются для аутентификации пользователей и предоставления им доступа к закрытым частям веб-сайта в зависимости от уровня привилегий, связанного с введенным именем пользователя. Поскольку нам неизвестны какие-либо конкретные учетные данные, которые мы могли бы использовать для входа в систему, мы проверим, есть ли какие-либо другие каталоги или страницы, полезные для нас в процессе перечисления. Всегда считается хорошей практикой полностью перечислить цель, прежде чем мы нацелимся на конкретную уязвимость, о которой мы знаем, например, в данном случае уязвимость SQL Injection. Нам нужна полная картина, чтобы убедиться, что мы ничего не упустили и не попали в кроличью нору, что может быстро разочаровать.

Думайте о веб-каталогах как о «веб-папках», в которых хранятся и организованы другие ресурсы и соответствующие файлы, такие как другие страницы, формы входа в систему, административные формы входа в систему, изображения и хранилище файлов конфигурации, таких как CSS, JavaScript, PHP, и более. Некоторые из этих ресурсов связаны непосредственно с целевой страницей веб-сайта. Страницы, к которым мы все привыкли, такие как

[Дом](#), [О](#), [Контакт](#), [регистр](#), И [Авторизоваться](#) страницы,

считываются отдельными веб-каталогами. При переходе на эти страницы URL-адрес в верхней части окна нашего браузера будет меняться в зависимости от нашего текущего местоположения. Например, если мы перейдем со страницы на Дом Контакт страницы веб-сайта, URL-адрес изменится следующим образом:

Домашняя страница:

`https://www.example.com/home`

Контактная страница:

`https://www.example.com/contact`

Некоторые страницы могут быть вложенный в других, что означает, что каталог для одной страницы может быть найден в больший каталог, содержащий предыдущую страницу. Забыли пароль страница для этого примера. Эти Возьмем обычно находятся под авторизоваться каталог, потому что вы можете быть перенаправлены на него из формы входа, если вы забыл пароль пользователя.

Страница авторизации:

`https://www.example.com/логин`

Страница «Забыли пароль»:

`https://www.example.com/логин/забыл`

Однако, допустим, кнопки и ссылки на нужные каталоги не предусмотрены. В этом случае, поскольку каталоги, которые мы ищем, либо содержат конфиденциальные материалы, либо просто ресурсы для веб-сайта для загрузки изображений и видео, мы можем указать имена этих каталогов или веб-страниц в том же поле URL-адреса браузера, чтобы увидеть, загрузится ли он. что либо. Ваш браузер сам по себе не заблокирует доступ к этим каталогам просто потому, что на странице для них нет ссылки или кнопки. Администраторы веб-сайтов должны будут убедиться, что каталоги, содержащие конфиденциальную информацию, должным образом защищены, чтобы пользователи не могли просто перейти к ним вручную.

При навигации по веб-каталогам HTTP-клиент, которым является ваш браузер, связывается с HTTP-сервером (в данном случае Apache 2.4.38) с использованием протокола HTTP, отправляя HTTP-запрос (GET или POST-сообщение), который сервер затем обрабатывать и возвращать с HTTP-ответом.

Ответы HTTP содержат коды состояния, в которых подробно описывается статус взаимодействия между запросом клиента и то, как сервер обработал его. Некоторые из наиболее распространенных кодов состояния для протокола HTTP:

HTTP1/1 200 OK: страница/ресурс существует, продолжается отправка вам данных.

HTTP1/1 404 Not Found: страница/ресурс не существует.

HTTP1/1 302 Found: страница/ресурс найдены, но перенаправлены в другой каталог (временно перемещены). Это приглашение пользовательскому агенту (веб-браузеру) сделать второй идентичный запрос на новый URL-адрес, указанный в поле местоположения. Вы будете воспринимать весь процесс как плавное перенаправление на новый URL указанного ресурса.

Если вы хотите узнать больше о том, как работают веб-запросы, вы можете проверить наш модуль Академии под названием [«Веб-запросы»](#)!



Давайте посмотрим на полный процесс поиска и доступа к скрытому каталогу. Указав IP-адрес цели, на которой работает HTTP-сервер, в поле URL-адреса браузера, за которым следует косая черта (/) и имя искомого каталога или файла, произойдут следующие события:

- Пользовательский агент (браузер / HTTP-клиент) отправит запрос GET на HTTP-сервер с URL-адресом запрошенного нами ресурса.
- HTTP-сервер будет искать ресурс в указанном месте (данний URL-адрес).
- Если ресурс или каталог существует, мы получим ответ HTTP-сервера, содержащий данные, которые мы

запрошенный (будь то веб-страница, изображение, аудиофайл, скрипт и т. д.) и **200 OK**, так как код ответа — ресурс найден и запрос успешно выполнен.

- Если ресурс или директория не могут быть найдены по указанному адресу и для них не реализовано перенаправление администратором сервера, ответ HTTP-сервера будет содержать типичный

404

Страница с кодом ответа **404 не найдено** прикрепил.

Эти два приведенных выше случая — это то, на чем мы сосредоточимся при попытке перечислить скрытые каталоги или ресурсы. Однако вместо того, чтобы вручную перемещаться по строке поиска URL-адресов, чтобы найти эти скрытые данные, мы будем использовать инструмент, который автоматизирует поиск для нас. Именно здесь вступают в игру такие инструменты, как Gobuster, Dirbuster, Dirb и другие.

Они известны как инструменты грубой силы. Перебор — это метод отправки данных, предоставленных через специально составленный список переменных, известный как список слов, в попытке угадать правильный ввод для его проверки и получения доступа. Применение метода грубой силы к веб-каталогам и ресурсам с помощью такого инструмента будет вводить переменные из списка слов одну за другой, непоследовательные запросы к HTTP-серверу, а затем читать код ответа HTTP для каждого запроса, чтобы увидеть, принят ли он как существующий ресурс или нет. В нашем случае списки слов, которые мы будем использовать, будут содержать стандартные имена каталогов и файлов (например,

картинки, **сценарии**, **логин.php**, **admin.php**), в сочетании с более необычными.

Тот же тип атаки используется для перебора паролей — отправка паролей из списка слов до тех пор, пока мы не найдем правильный пароль для указанного имени пользователя. Этот метод распространен среди малоквалифицированных злоумышленников из-за его низкой сложности, а недостатком является «шумность», что означает отправку большого количества запросов каждую секунду, настолько много, что его легко обнаруживают устройства безопасности периметра. настроен для прослушивания нечеловеческих взаимодействий с формами входа.

В нашем случае мы будем запускать инструмент под названием **Гобастер**, который будет перебирать каталоги и файлы представленный в списке слов по нашему выбору. Gobuster поставляется с предустановленной ОС Parrot. Однако, если вы используете другую операционную систему, вы можете установить ее, выполнив следующие действия.

Гобастер

Gobuster написан на Golang, что является сокращением от языка программирования Go. В соответствии [сопределение Википедии](#) Голанг:

Go — это статически типизированный компилируемый язык программирования, разработанный в Google [...].

Go синтаксически похож на C, но с безопасностью памяти, сборкой мусора, структурной типизацией и параллелизмом в стиле CSP. Этот язык часто называют Golang из-за его доменного имени golang.org, но правильное имя — Go.

Go находится под влиянием C, но с упором на большую простоту и безопасность. Язык состоит из:

- Синтаксис и среда, использующие шаблоны, более распространенные в динамических языках:
- Необязательное краткое объявление переменных и инициализация посредством вывода типа [...] .

- Быстрая компиляция.
 - Удаленное управление пакетами (go get) и онлайн-документация по пакетам.
- Отличительные подходы к конкретным проблемам:
- Встроенные примитивы параллелизма: облегченные процессы (горутины), каналы и оператор выбора.
 - Система интерфейса вместо виртуального наследования и встраивания типов вместо невиртуальное наследование.
 - Цепочка инструментов, которая по умолчанию создает статически связанные бинарные файлы без внешние зависимости.
- Желание сохранить спецификацию языка достаточно простой, чтобы программист мог держать ее в голове, частично за счет исключения функций, общих для подобных языков.

Установка

Поскольку этот инструмент написан на Go, вам необходимо установить язык/компилятор Go/и т. д. Полную информацию об установке и настройке можно найти [на Go Lan граммуга граммэлектронный сайт](#). Вам нужна как минимум версия Go 1.16.0 для компиляции Gobuster.

После установки у вас есть два варианта:

A. Использование [иди устанавливай](#) Команда

Если у тебя есть [Идтфеда](#) готова к работе (по крайней мере, 1.16), это так же просто, как:

```
установить github.com/OJ/gobuster/ v3@latest
```

После завершения установки вы можете перейти к разделу «Использование Gobuster».

B. Сборка из исходного кода и компиляция

Во-первых, вам нужно будет клонировать репозиторий:

```
git-клон https://github.com/OJ/gobuster.git
```

После завершения клонирования появится `клеветник` папку в каталоге, в котором вы сейчас находитесь. Перейдите к папка, в которой находятся файлы. `клеветник` имеет внешние зависимости, поэтому они должны быть Втянулись первыми:

иди возьми && иди строй

Это создаст `клеветник` бинарный для вас. Если вы хотите установить его в `$GOPATH/бин` папку, вы можете запустить:

иди устанавливай

Если у вас уже есть все зависимости, вы можете использовать скрипты сборки:

- `делать` - строит для текущей конфигурации Go (т.е. запускает `иди строй`).
- `сделать окна` - создает 32- и 64-битные двоичные файлы для Windows и записывает их в `строить` папка.
- `сделать линукс` - создает 32- и 64-битные бинарные файлы для Linux и записывает их в `строить` папка.
- `сделать Дарвина` - создает 32- и 64-битные двоичные файлы для Darwin и записывает их в `строить` папка.
- `сделать все` - строит для всех платформ и архитектур и записывает полученные бинарники в `строить` папка.
- `сделать чистым` - очищает от `строить` папка.
- `сделать тест` - запускает тесты.

Использование Гобастер

Чтобы увидеть, как мы можем использовать Gobuster, мы можем использовать следующий синтаксис.



```
$ gobuster --help

Usage:
  gobuster [command]

Available Commands:
  dir      Uses directory/file enumeration mode
  dns      Uses DNS subdomain enumeration mode
  fuzz     Uses fuzzing mode
  help     Help about any command
  s3       Uses aws bucket enumeration mode
  version  shows the current version
  vhost    Uses VHOST enumeration mode

Flags:
  --delay duration  Time each thread waits between requests (e.g. 1500ms)
  -h, --help         help for gobuster
  --no-error        Don't display errors
  -z, --no-progress  Don't display progress
  -o, --output string Output file to write results to (defaults to stdout)
  -p, --pattern string File containing replacement patterns
  -q, --quiet        Don't print the banner and other noise
  -t, --threads int  Number of concurrent threads (default 10)
  -v, --verbose      Verbose output (errors)
  -w, --wordlist string Path to the wordlist

Use "gobuster [command] --help" for more information about a command.
```

После проверки [помощь](#) вариант, теперь мы можем использовать программу для поиска интересных страниц и каталогов. Тем не менее, нам все еще не хватает списка слов. Существует специальная папка с множеством списков слов, словарей и радужных таблиц, которая предустановлена вместе с ОС Parrot и находится по пути </usr/share/wordlists>. Однако вы также можете загрузить коллекцию SecLists, которая является одной из самых известных коллекций списков слов, используемых сегодня. SecList можно скачать [следовать граммэта ссылка](#).

Чтобы загрузить его, введите следующую команду:

```
клон git https://github.com/danielmiessler/SecLists.git
```

Теперь мы можем начать правильно использовать Gobuster в полной мере. Флаги, которые мы будем использовать с этой командой, следующие:

```
dir : Укажите, что мы хотим выполнить перечисление веб-каталога.  
-- url : укажите веб-адрес целевой машины, на которой работает HTTP-сервер.  
--wordlist : Укажите список слов, который мы хотим использовать.
```

```
$ gobuster dir --url http://{target_IP}/ --wordlist {wordlist_location}/directory-list-2.3-small.txt  
=====  
Gobuster v3.1.0  
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)  
=====  
[+] Url:          http://{target_IP}/  
[+] Method:       GET  
[+] Threads:      10  
[+] Wordlist:     {wordlist_location}/directory-list-2.3-small.txt  
[+] Negative Status codes: 404  
[+] User Agent:   gobuster/3.1.0  
[+] Timeout:      10s  
=====  
2021/07/08 14:19:03 Starting gobuster in directory enumeration mode  
=====  
/images           (Status: 301) [Size: 311] [--> http://{target_IP}/images/]  
/css              (Status: 301) [Size: 308] [--> http://{target_IP}/css/]  
/js               (Status: 301) [Size: 307] [--> http://{target_IP}/js/]  
/vendor            (Status: 301) [Size: 311] [--> http://{target_IP}/vendor/]  
/fonts             (Status: 301) [Size: 310] [--> http://{target_IP}/fonts/]  
=====  
2021/07/08 14:27:18 Finished  
=====
```

Проверив веб-каталоги, мы не нашли никакой полезной информации. Результаты, представленные в наших выходных данных, представляют собой каталоги по умолчанию для большинства веб-сайтов, и в большинстве случаев они не содержат файлов, которые могут быть использованы злоумышленником или каким-либо образом полезны для него. Тем не менее, проверить их все же стоит, потому что иногда они могут содержать нестандартные файлы, помещенные туда по ошибке.

плацдарм

Поскольку Gobuster не нашел ничего полезного, нам нужно проверить учетные данные по умолчанию или как-то обойти страницу входа. Чтобы проверить учетные данные по умолчанию, мы могли бы ввести наиболее распространенные комбинации в полях имени пользователя и пароля, например:

```
администратор:админ  
ГОСТЬ: ГОСТЬ  
пользователь: пользователь  
корень: корень  
администратор:пароль
```

После всех этих комбинаций нам так и не удалось войти в систему. Гипотетически мы могли бы использовать инструмент для попытки перебора страницы входа. Однако это займет много времени и может привести к срабатыванию мер безопасности.

Следующей разумной тактикой будет проверка формы входа на предмет возможной уязвимости SQL Injection. Этот вектор подробно описан в [Вступление](#) раздел сочинения:

Внедрение SQL — это распространенный способ использования веб-страниц, использующих «выражения SQL», которые извлекают и сохраняют вводимые пользователем данные. При неправильной настройке можно использовать эту атаку для эксплуатации известной уязвимости SQL Injection, что очень опасно. Существует множество различных методов защиты от SQL-инъекций, некоторые из которых включают проверку ввода, параметризованные запросы, хранимые процедуры и реализацию WAF (брандмауэра веб-приложений) на периметре сети сервера. Однако могут быть обнаружены случаи, когда ни одно из этих исправлений не установлено, поэтому этот тип атаки распространен, согласно [OWASP Top 10] (<https://owasp.org/www-project-top-ten/>). список веб-уязвимостей.

Вот пример того, как работает аутентификация с использованием PHP и SQL:

```
<?php

mysql_connect("локальный хост","db_имя_пользователя","БД_пароль");# Подключение к базе данных
SQL.

mysql_select_db("пользователи");# Таблица базы данных, в которой хранится информация о пользователях.

$имя_пользователязнак равно$_POST['имя пользователя'];# Пользовательское имя
пользователя. $парользнак равно$_POST['пароль'];#Указанный пользователем пароль.

$sqlзнак равно"ВЫБЕРИТЕ * ОТ пользователей, ГДЕ имя пользователя ="$имя_пользователя'И пароль='$пароль'";
# Запрос на получение пользователя/пароля из БД.

$результатзнак равномysql_query($sql);
# Выполняет запрос, хранящийся в $sql, и сохраняет его в $result.

$количествознак равномysql_num_rows($результат);
# Задает для переменной $count количество строк, хранящихся в $result.

если($количество==1){
    # Проверяет, есть ли хотя бы 1 результат, и если да:
    $_СЕССИЯ['имя пользователя']знак равно$имя_пользователя;# Создает сеанс с указанным $username.
    $_СЕССИЯ['пароль']знак равно$пароль;# Создает сеанс с указанным $password. заголовок(
    "местоположение:home.php");# Перенаправление на домашнюю страницу.
```

```
}

еще{# Если нет единственного результата комбинации пользователя/пароля:
заголовок("местоположение: логин.php");
# Без перенаправления, т.к. вход в систему завершился неудачно, если переменная $count не равна
1, код ответа HTTP 200 OK. }

?>
```

Обратите внимание, как после `#` СИМВОЛ, все превращается в комментарий? Так работает язык PHP.

Имейте это в виду на потом».

Приведенный выше код уязвим для атак с внедрением SQL-кода, когда вы можете изменить `$sql` переменную (через форму входа на веб-странице, чтобы запрос делал то, что не должен делать — вообще обойти вход в систему!

Обратите внимание, что мы можем указать имя пользователя и пароль через форму входа на веб-странице. Однако он будет непосредственно встроен в переменную `$sql`, которая выполняет SQL-запрос без проверки ввода. Обратите внимание, что никакие регулярные выражения или функции не запрещают нам вставлять специальные символы, такие как одинарная кавычка или хэштег. Это опасная практика, поскольку эти специальные символы могут использоваться для изменения запросов. Пара одинарных кавычек используется для указания точных данных, которые необходимо извлечь из базы данных SQL, а символ хэштега используется для комментариев. Следовательно, мы могли бы манипулировать командой запроса, введя следующее:

```
Имя пользователя: admin'#
```

Мы закроем запрос этой одинарной кавычкой, позволяя скрипту искать добавив хэштег, мы закомментируем остальную часть запроса, что сделает поиск подходящего пароля для указанного имени пользователя устаревшим. Если мы посмотрим дальше в приведенном выше PHP-коде, мы увидим, что код подтвердит вход в систему только в том случае, если есть ровно один результат нашей комбинации имени пользователя и пароля. Однако, поскольку мы пропустили часть поиска пароля в нашем запросе, скрипт теперь будет искать только в том случае, если существует какая-либо запись с именем пользователя. `администратор`. В данном случае нам повезло. Действительно существует аккаунт называется `администратор`, который подтвердит нашу SQL-инъекцию и вернет `1` значение переменной `$count`, который будет поставлен через `если оператор`, что позволяет нам войти в систему, не зная пароля. Если там не было `администратор` учетную запись, мы могли бы попробовать любые другие учетные записи, пока не найдем существующую. (`администратор`, корень, Джон Доу и т. д.) Любое действительное существующее имя пользователя заставит нашу SQL-инъекцию работать.

В этом случае, поскольку часть запроса, связанная с поиском пароля, была пропущена, мы можем добавить в поле пароля все, что захотим, и это не будет иметь значения.

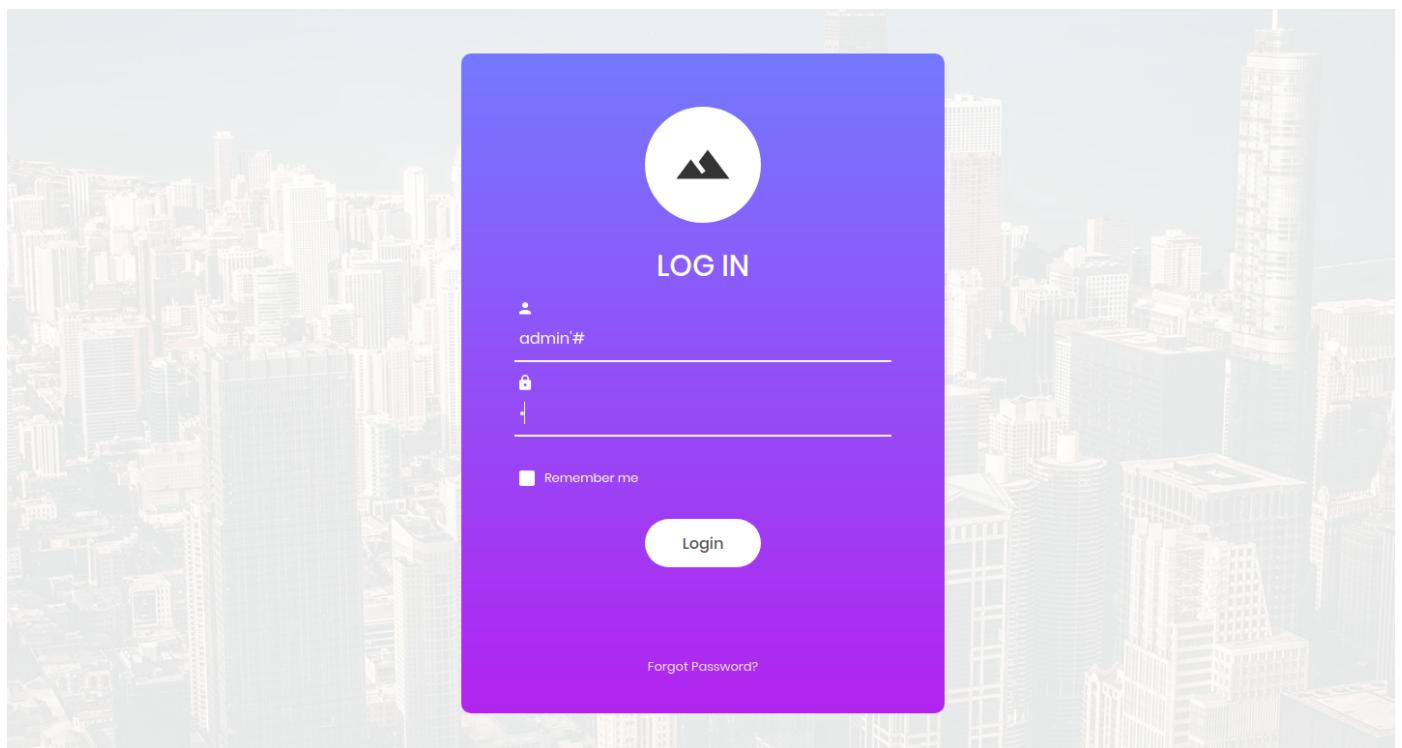
Пароль: abc123

Чтобы быть более точным, вот как наш ввод влияет на часть запроса PHP-кода.

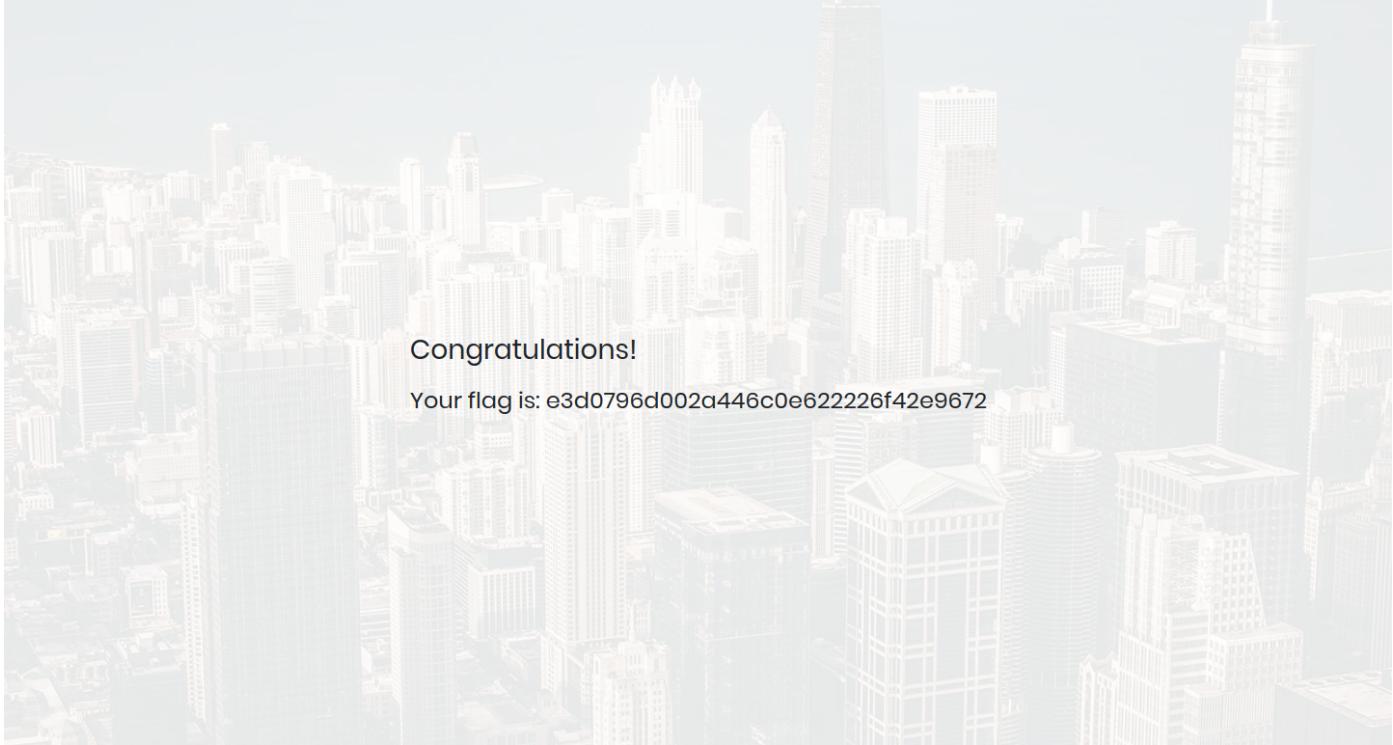


```
SELECT * FROM users WHERE username='admin'#' AND password='a'
```

Обратите внимание, как после нашего ввода мы закомментировали раздел проверки пароля в запросе? Это приведет к тому, что PHP-скрипт вернет значение 1 (найдена 1 строка) для **имя пользователя = 'админ'** без проверки **пароль** поле для соответствия этой записи. Это связано с отсутствием проверки ввода в коде PHP, показанном выше.



После нажатия кнопки входа отправляется код эксплойта, и, как и предполагалось, нам открывается следующая страница:



Congratulations!

Your flag is: e3d0796d002a446c0e622226f42e9672

Мы успешно выполнили первичную SQL-инъекцию и получили флаг.

Поздравляем!