ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«Московский политехнический университет»**

**КУРСОВАЯ РАБОТА**

**по дисциплине «Администрирование операционных систем Linux»**

на тему:

«Анализ уязвимостей веб-серверов на базе ОС Linux»

Выполнили:

студенты гр. 191-351 и 191-352, Ефремов Никита Алексеевич (1999), Балдуев Никита Александрович, Лунин Иван Алексеевич Проверили:

Гневшев Александр Юрьевич

Москва, 2021

**Оглавление**

[ВВЕДЕНИЕ 3](#_Toc60869658)

[1. SSH СОСТАВЛЯЮЩАЯ 4](#_Toc60869659)

[1.1 Основные сведения 4](#_Toc60869660)

[1.2 Brute - Force 5](#_Toc60869661)

[1.3 Man-In-The-Middle (MITM) 5](#_Toc60869662)

[2. HTTP СОСТАВЛЯЮЩАЯ 7](#_Toc60869663)

[2.1 Общие сведения 7](#_Toc60869664)

[2.2 SQL – инъекция 8](#_Toc60869665)

[2.3 Path traversal 9](#_Toc60869666)

[2.4 XSS – Cross-Site-Scripting 10](#_Toc60869667)

[2.5 WebDav 11](#_Toc60869668)

[2.6 Промежуточный итог 12](#_Toc60869669)

[3. ПРАКТИЧЕСКАЯ ЧАСТЬ 13](#_Toc60869670)

[3.1 Пояснение к дальнейшим действиям 13](#_Toc60869671)

[3.2 Настройка системы 13](#_Toc60869672)

[3.3 Проведение тестирования 17](#_Toc60869673)

[ЗАКЛЮЧЕНИЕ 32](#_Toc60869674)

[Библиографический список 33](#_Toc60869675)

# ВВЕДЕНИЕ

При помощи интернета и его различных составляющих каждый день передаётся, хранится, создаётся и обрабатывается колоссальное количество различной информации. Социальные сети, рабочие зоны, веб-приложения, инфраструктуры различных организаций и общественных зон, а также многие другие аспекты информационного поля нуждаются в качественном обслуживании и обеспечении работоспособности различных сервисов, сайтов и приложений. Для централизации подобных действий создаются серверы — мощные вычислительные единицы, способные содержать, передавать и обрабатывать базы данных, данные приложений, сайты — в общем, являются центром многих инфраструктур.

Именно по той причине, что сервер содержит в себе большой объём информации, большая часть из которой является конфиденциальной (например, информация о пользователях систем и приложений), серверы на протяжении всего своего существования являлись объектом для нападения различного рода хакеров, а за всё время был разработан огромный список различных методов обхода систем защиты, который постоянно пополняется. Сервер нуждается в серьёзной защите, структура которой должна зависеть от того, что на данном сервере хранится (базы данных, веб-сайты/веб-приложения, среды разработки и т. д.).

В данной исследовательской работе будут описаны основные (самые частые и самые опасные) уязвимости веб-серверов, рассказано об основных методах борьбы с ними, а также будет представлена практическая часть, в которой будет проделана работа по настройке соединения для команды разработки (SSH), а также проделаны основные пункты, необходимые для защиты практически любого веб-сервера.

# 1. SSH СОСТАВЛЯЮЩАЯ

## 1.1 Основные сведения

SSH - один из самых распространенных протоколов, обеспечивающий безопасное зашифрованное соединение, например для подключения к удаленным машинам, обмена файлами, организации защищенных туннелей, удаленного управления без ручной авторизации и так далее. Он был создан для замены многих нешифрованных протоколов, таких как telnet, FTP, RSH и подобных. Ведь известно, что одна из проблем этих старых протоколов (помимо того, что они все данные передают в открытом виде) — возможность организации man-in-the-middle атак. Хакер с доступом во внутреннюю сеть может перехватывать пакеты, записывать их и лишь потом передавать по назначению. Более того, нападающий может изменять информацию в пакетах - например написать rm -r вместо ls -l, или передать трояна в ходе FTP скачки.

Подключение к удалённому серверу при помощи SSH является основным и одним из самых часто используемых методов работы на сервере. Практически для любого веб-сервера будут открыты именно два порта: 22 — SSH, и 80 (443)

– HTTP (HTTPS). Именно поэтому важно уделить особое внимание защите данного соединения, так как если злоумышленнику удастся получить доступ к системе изнутри, то ничто не мешает заменить/испортить/украсть абсолютно любые данные и файлы с сервера, а, значит, прервать работоспособность веб- приложений и подвергнуть опасности конфиденциальную информацию пользователей.

Далее будут рассмотрены основные способы проникновения в систему по SSH, а также методы борьбы с попытками несанкционированного доступа к веб-серверу при помощи SSH и препятствования изменению файлов, если

попытку проникновения всё же не удалось вовремя предупредить и ликвидировать.

## 1.2 Brute - Force

Метод проникновения, предусматривающий полный перебор вариантов паролей в надежде получить доступ. Данный способ часто используют для проникновения на личные страницы пользователей различных веб-приложений, но, так как в данном случае многое зависит именно от особенностей веб- приложения (возможно, что «Брутфорс» вообще не имеет смысла, так как сайт или сервис не предусматривает хранение личных данных пользователя в каком- либо виде), то данный метод было решено отнести к угрозам соединения по SSH, так как в данном случае также используется система Логин/Пароль для доступа к серверу.

Способы борьбы с данной угрозой подразумевают тонкую настройку системы пользователей сервера, ведение парольной политики, использование связанных систем Логин/Пароль — Логин/Ключ, которые вместе практически исключают возможность несанкционированного проникновения, настройка сервиса fail2ban или Pam-Shield, котораые не позволяют свободно проводить манипуляции «Брутфорсом», просто блокируя IP адрес после нескольких неудачных попыток доступа (Некоторые из данных пунктов будут представлены в практической части)

## 1.3 Man-In-The-Middle (MITM)

Данный тип атаки предусматривает перехват данных при первом установлении соединения SSH с пользователем и обменом ключей и перенаправление данной информации на собственный сервер, что позволяет

подменить ключи на собственные и таким образом установить соединение с сервером. Очень долгое время данной угрозе был подвержен протокол SSH первой версии, но на данный момент атаку «человек посередине» можно провести и с подключением по SSH второй версии.

Важной информацией при борьбе с данным типом атаки является тот факт, что атакующий при любых обстоятельствах оставляет информацию о собственном подключении и IP адресе в системе, что позволяет администратору отследить и заблокировать данное подключение. Естественно, это означает, что должна быть настроена система логирования, которая будет записывать все подключения и информацию о данных подключениях.

Также полную безопасность при подключении гарантирует использование системы ключей, а не парольной защиты, что полностью исключает возможность MITM атаки.

# 2. HTTP СОСТАВЛЯЮЩАЯ

## 2.1 Общие сведения

HTTP – протокол передачи гипертекста (зачастую вместе с HTML страницей или другой произвольной информацией).

Веб-сервер (web-server) – это сервер, отвечающий за прием и обработку запросов (HTTP-запросов) от клиентов к веб-сайту. В качестве клиентов обычно выступают различные веб-браузеры. В ответ веб-сервер выдает клиентам HTTP-ответы, в большинстве случаев – вместе с HTML-страницей, которая может содержать: всевозможные файлы, изображения, медиа-поток или любые другие данные.

Так как Веб-сервер предусматривает использование на стороне клиента именно HTTP запросов и ответов, уязвимости данного протокола могут напрямую повлиять как на безопасность конфиденциальных данных отдельного пользователя, так и на безопасность всей информации, содержащейся на сервере. Также стоит учитывать, что зачастую веб-сервер представляет собой систему «Клиент — сервер — база данных», так что к уязвимостям HTTP составляющей будут отнесены некоторые уязвимости работы с базами данных и HTML страницами, содержащимися или обрабатываемыми на сервере.

Далее будут перечислены основные уязвимости веб-сервера на данной стороне, а также возможные решения данных проблем, реализуемые как при помощи системных возможностей сервисов, установленных на сервере, и баз данных, так и при помощи простой корректировки информации, отображаемой и содержащейся на HTML странице.

## 2.2 SQL – инъекция

Данная атака завязана на использовании уязвимости базы данных и веб- приложения к переходам по страницам при помощи SQL запросов в поисковом поле (например, по страницам пользователей). Если в результате запроса для отображения определённой информации на странице используются индивидуальные идентификаторы пользователя, то, манипулируя данными параметрами, можно попасть на страницы тех пользователей, доступа к которым у злоумышленника быть не должно, но при этом будет возможность просматривать или даже изменять информацию на странице.

В некоторых случаях злоумышленнику может открыться возможность полностью управлять базой данных, что может нести непоправимые для работы сервиса последствия. Данный тип атаки является одним из самых распространённых и вместе с тем одним из самых простых для исполнения, чем активно пользуются и, не смотря на широкую осведомлённость, многие веб- серверы до сих пор имеют минимальную защищённость от подобной угрозы.

Способы защиты могут быть различными, но зачастую они завязаны на модификациях процесса обработки запросов, таких как: экранирование специальных символов (кавычки двойные и одиночные, а также слеши), фильтрация строковых параметров, фильтрация целочисленных параметров, усечение входных параметров, а также использование параметризированных запросов (в таком случае к выполнению будут допускаться только определённые запросы. Все же остальные, не подходящие по виду, будут перенаправляться или автоматически экранироваться), конвертирование строчных значений, если данная возможность имеется.

## 2.3 Path traversal

Обход директорий (Path traversal или Directory traversal) заключается в том, что хакер получает доступ к директориям или файлам на сервере с помощью манипуляций переменных, ссылающиеся на эти файлы. Например, для скачивания файла с сервера указывается его имя:

[www.site.ru/download?file=file.pdf](http://www.site.ru/download?file=file.pdf)

С помощью символа, обозначающего директории (../), можно получить доступ к другим файлам, просто добавив его в строку:

www.site.ru/download?file=../../../etc/passwd

Если имя файла никак не валидируется, то злоумышленник сможет увидеть все файлы системы. И это очень опасно, так как важная информация (файлы конфигурации, логи и прочее) находится в заранее известных местах. Кроме того, можно читать исходный код приложения.

Уязвимость становится ещё опаснее, если помимо чтения файлов есть ещё и возможность загружать их в произвольные директории. Соответственно, это одна из тех узвимостей, которая может не просто предоставить доступ к отдельной информации, но и позволить полноценно управлять содержимым сервера: добавлять, изменять, перемещать, удалять различные файлы, даже не авторизуясь для этого в системе.

Методы борьбы с данной уязвимостью предусматривают попытки максимального ограничения количества обращений к файловой системе сервера при пользовательском вводе, а также, если без данного вызова обойтись не получится, необходимо проверять вводимые пользователем данные перед их обработкой.

## 2.4 XSS – Cross-Site-Scripting

XSS (англ. Cross-Site Scripting — «межсайтовый скриптинг») — довольно распространенная уязвимость, которую можно обнаружить на множестве веб- приложений. Ее суть довольно проста, злоумышленнику удается внедрить на страницу JavaScript-код, который не был предусмотрен разработчиками. Этот код будет выполняться каждый раз, когда жертвы (обычные пользователи) будут заходить на страницу приложения, куда этот код был добавлен. А дальше существует несколько сценариев развития:

Первый: злоумышленнику удастся заполучить авторизационные данные пользователя и войти в его аккаунт.

Второй: злоумышленник может незаметно для жертвы перенаправить его на другую страницу-клон. Эта страница может выглядеть совершенно идентично той, на которой пользователь рассчитывал оказаться. Но вот принадлежать она будет злоумышленнику. Если пользователь не заметит подмены и на этой странице введет какие-то sensitive data, то есть личные данные, они окажутся у злоумышленника.

Методы борьбы с данной уязвимостью предусматривают предварительную обработку полей ввода, а также экранирование текста и значений перед их выводом на странице. Также есть некоторые способы для ограничения возможности вывода информации (вроде скрытия в JavaScript возможности вывода данных cookie – HTTP only), которые будут полезны в том случае, если XSS событие предотвратить не удалось.

## 2.5 WebDav

WebDAV (Web Distributed Authoring and Versioning) — это протокол для передачи данных и работы с ними, построенный поверх HTTP 1.1. Здесь следует заметить, что передача может быть как защищенной, так и незащищенной. В самом протоколе защищенность отсутствует, но она может быть добавлена через реализацию аутентификации на веб-сервере и шифрование посредством SSL, следовательно, в таком случае будет использоваться не HTTP, а HTTPS.

Изначально DAV разрабатывался для совместного создания и редактирования веб-страниц, но в процессе использования он нашел применение в качестве сетевой распределенной файловой системы, эффективной для работы в высоконагруженной среде и поддерживающей неустойчивое соединение. Таким образом, DAV подходит для управления файлами на веб-серверах, иными словами, реализации облачных хранилищ информации, где и был применен. С его помощью можно выполнять основные операции над файлами, содержащимися на сервере, проводить расширенные операции, как то: блокировка, получение метаданных, контроль версий и другие. Этот протокол стал заменой для старого доброго FTP, чье время подошло к концу.

WebDAV предоставляет семь команд:

* PROPFIND — получение свойств объекта на сервере в формате XML;
* PROPPATCH — изменение свойств объекта;
* MKCOL — создать папку на сервере;
* COPY — копирование на стороне сервера;
* MOVE — перемещение на стороне сервера;
* LOCK — заблокировать объект;
* UNLOCK — снять блокировку с объекта.

Таким образом, WebDAV позволяет изменять свойства хранящихся на сервере объектов, выполнять поиск с учетом свойств, блокировать объект (в нашем случае — файл) для организации возможности его редактирования только одним пользователем в распределенной среде, в которой доступ могут иметь много юзеров, управлять версиями файлов (посредством унаследованных команд check -in, -out), а также производить расширенный контроль доступа к файлам на основе списков.

## 2.6 Промежуточный итог

Как можно видеть из предложенного выше списка, уязвимостей у веб сервера предостаточно. Конечно, данный список не является полным или исчерпывающим, однако это основные уязвимости, которые следует перекрывать разработчикам и администраторам сервера и веб-приложений. Также стоит упомянуть некоторые атаки, которые применяются реже просто по той причине, что обладают меньшей эффективностью, более требовательны в реализации или же нецелесообразны ввиду их узкй направленности, что просто не позволяет учитывать их в общей статистике:

* JSON Hijacking - атака, в некотором смысле похожая на подделку межсайтовых запросов (CSRF), при которой злоумышленник старается перехватить данные JSON, отправленные веб- приложению с веб-сервера.
* XML External Entity - это тип атаки, в котором используется широко доступная, но редко используемая функция синтаксических анализаторов XML. Используя XXE, злоумышленник может вызвать отказ в обслуживании (DoS), а также получить доступ к локальному и удаленному контенту и службам. XXE может использоваться для выполнения подделки запросов на стороне сервера (SSRF), заставляя веб-приложение выполнять запросы к другим приложениям. В некоторых случаях c помощью XXE может даже выполнить сканирование портов и удаленное выполнение кода.

# 3. ПРАКТИЧЕСКАЯ ЧАСТЬ

IP сервера: 46.151.155.145

Версии установленных и тестируемых сервисов:

Сервер - Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-59-generic x86\_64);

Metasploit – v6.0.20-dev;

Apache –2.4.41 (Ubuntu);

Flask – 1.1.2;

OpenSSH – 8.2p1 Ubuntu-4ubuntu0.1, OpenSSL 1.1.1f 31 Mar 2020;

SQLite – version 3.31.1 2020-01-27.

## 

## 3.1 Пояснение к дальнейшим действиям

В практической части работы мы провели начальную настройку собственного сервера. Мы опустим установку системы (в качестве системы использоваться будет Ubuntu Server), покупку и регистрацию статического IP адреса, проброс портов на роутере владельца сервера, так как это промежуточные пункты в настройке сервера.

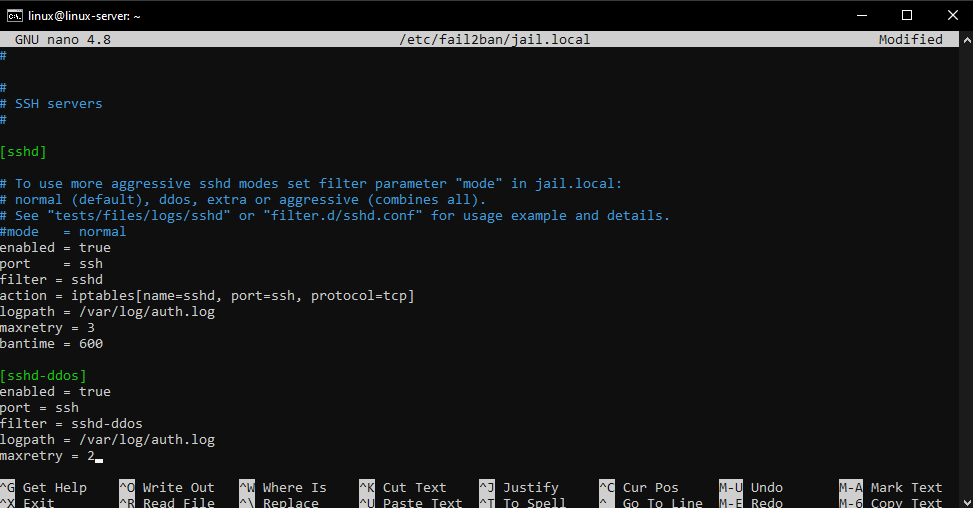
Стоит отметить, что мы решили не регистрировать доменное имя за ненадобностью, однако это всегда необходимо делать при работе с настоящим веб-сервером, в то время как возможность обращения к сервису при помощи IP стоит отключить, так как обычный рядовой пользователь в очень редких случаях будет использовать вход не по доменному имени, а по адресу сервера, в то время как злоумышленники часто проверяют именно адрес.

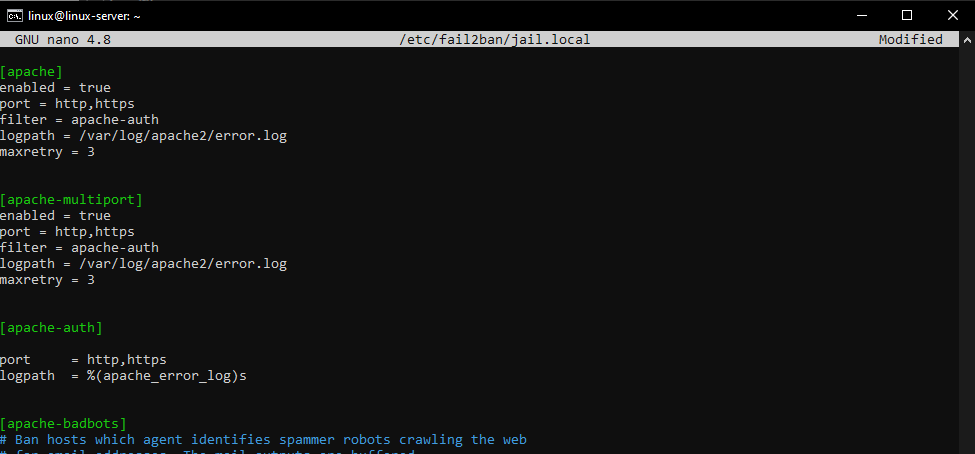
Особое же внимание мы уделили защите SSH соединения, как одному из главных каналов разработки и связи с сервером, а также тем пунктам защиты HTTP составляющей сервера, которые можно затронуть при помощи внутренних настроек системы.

## 

## 3.2 Настройка системы

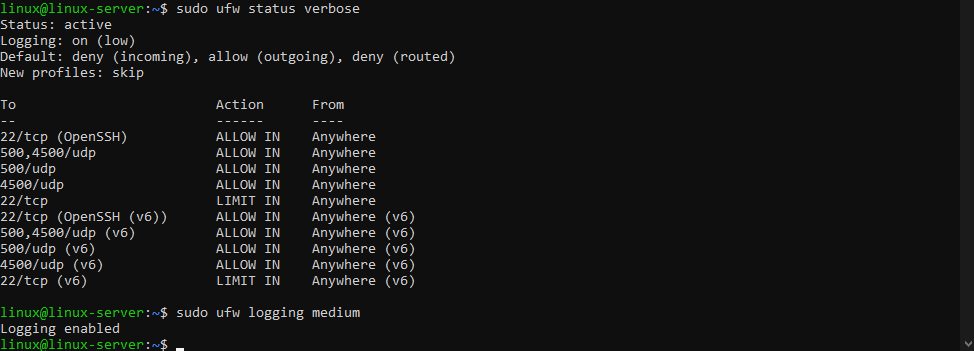
Начать было решено с сервиса fail2ban, который позволяет ограничить количество попыток подключения с одного IP адреса и заблокировать данный IP адрес, если было превышено количество попыток. Для этого необходимо зайти в конфигурационный файл fail2ban и выставить настройки соединения, количество попыток на подключение, а также время, на которое данный адрес

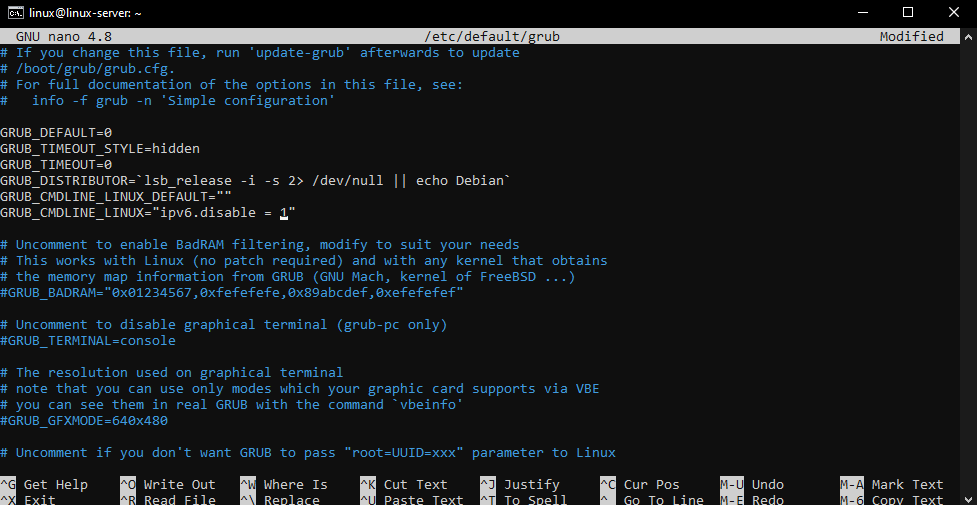
будет заблокирован. В качестве соединения был выбран 22 порт (SSH), количество попыток было ограничено тремя, то есть стандартное количество попыток при одном подключении через OpenSSH. Также все логи входов будут записываться в файл auth.log, что позволит отслеживать подключения:

После этого аналогичные настройки были поставлены на аутентификацию через HTTP. В данном случае логи будут записываться в файл логов ошибок Apache:

Далее были изменены настройки ядра linux, перекрывающие некоторые возможные уязвимости. В частности, была включена защита от переполнения буфера execShield, защита от подделывания IP адресов, была отключена возможность перенаправления IP адресов, включено игнорирование широковещательных запросов, а также включено логгировние всех подделанных пакетов:

Далее мы уделили внимание брандмауэру, так как это одна из самых важных частей защиты сервера. Для этого используем сервис UFW, где настраиваем открытые порты, количество возможных установленных сессий. Соответственно, установили правила для 80 и для 22 порта. На данном моменте стоит отметить, что также дополнительным методом защиты в нашем случае является то, что сервер не напрямую присоединён к сети, а входящие пакеты на него перенаправляются при помощи роутера, на котором были настроены соответствующие правила только для 22 и 80 портов (SSH и HTTP соответственно):



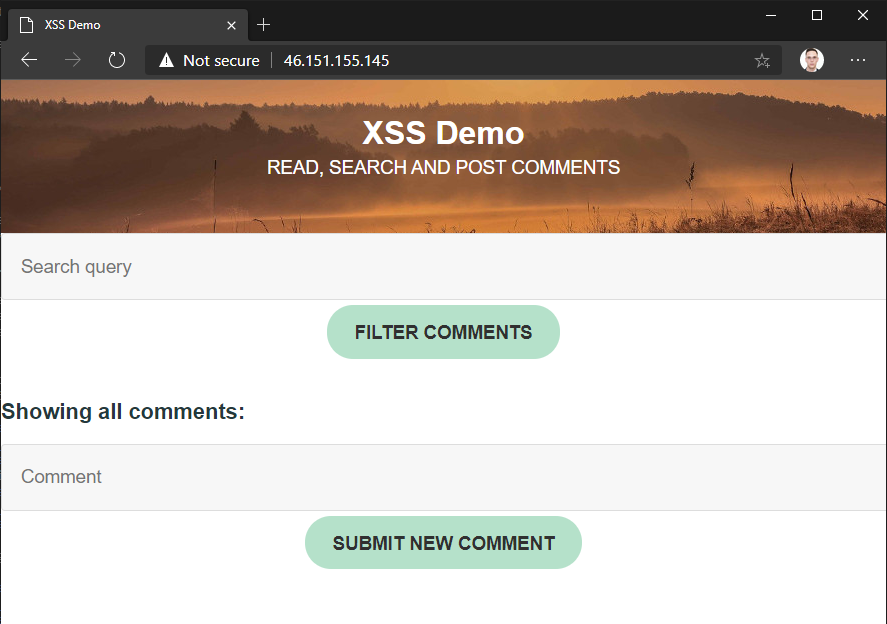
Далее было принято решение об отключении IPv6. Нам известно, что часть трафика уже сейчас активно передаётся при помощи IPv6, но в условиях небольшого сервера с одним сайтом и небольшой базой данных в использовании IPv6 нет совершенно никакого смысла:

## 3.3 Проведение тестирования

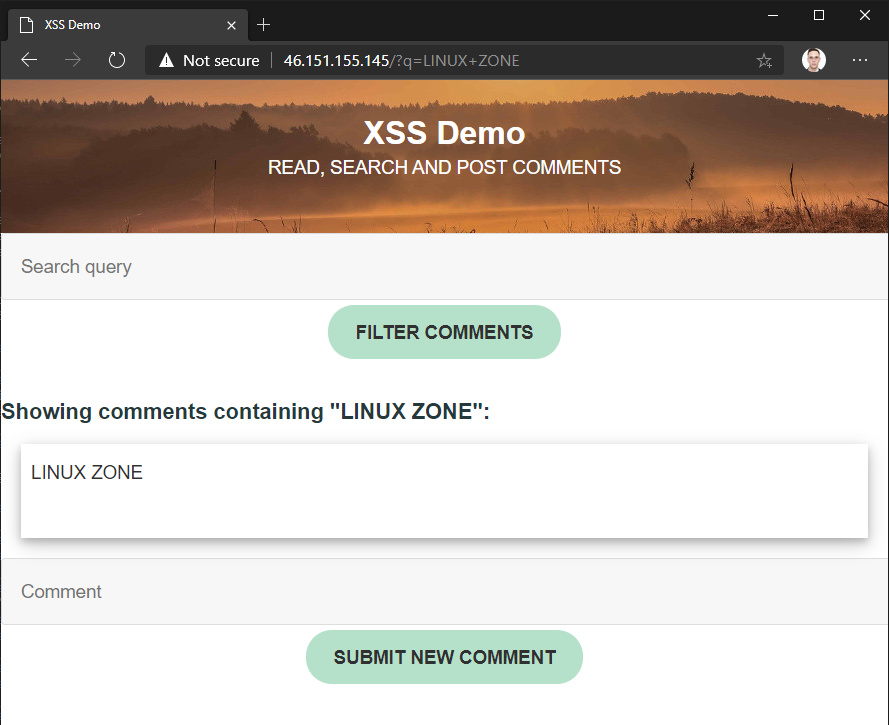
Для проведения тестов и проверки некоторых эксплойтов было решено использовать Метасплойт, который позволяет подробно изучить систему на предмет уязвимостей, а также провести некоторые типы атак. Также нам удалось застать практическую ситуацию, о которой будет рассказано после проведения тестирования различными модулями. Далее будет представлен список использованных модулей, скриншоты их работы, а также результаты работы модулей.

**Проверка сайта на XSS**

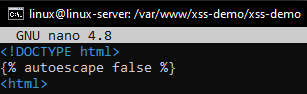
Сайт для проверки:



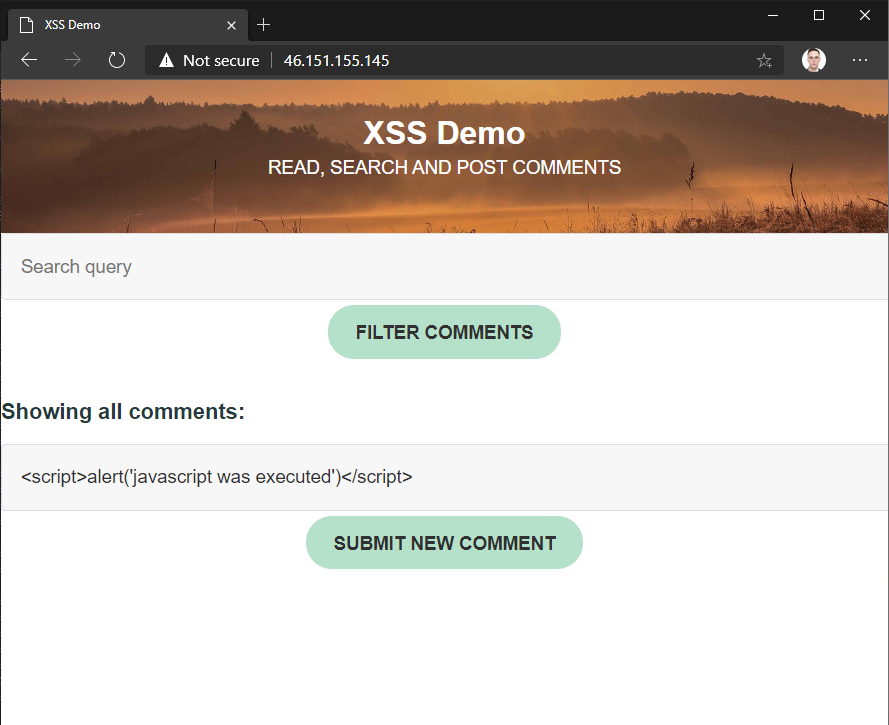
Проверка работоспособности:



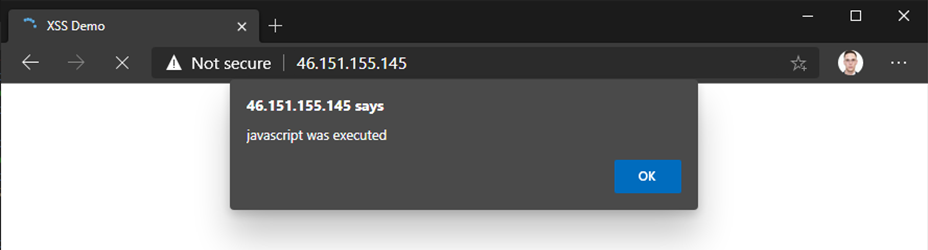
Для демонстрации недостатка XSS, изменили параметр «autoescape» на false:



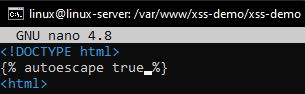
Пробуем ввести данный код:

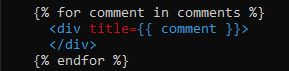


В результате, после его отправки, сайт начинает постоянно выдавать баннер с данными из кода:

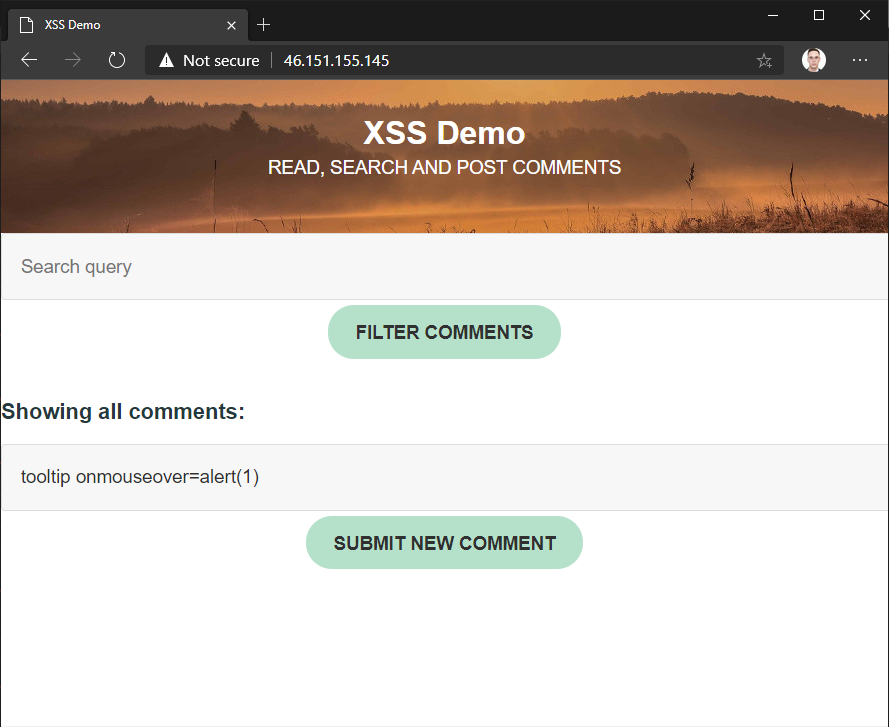


Для демонстрации необходимости контекстно-зависимой фильтрации можно изменить данные строки в том же файле, при этом вернув параметру «autoescape» значение true:

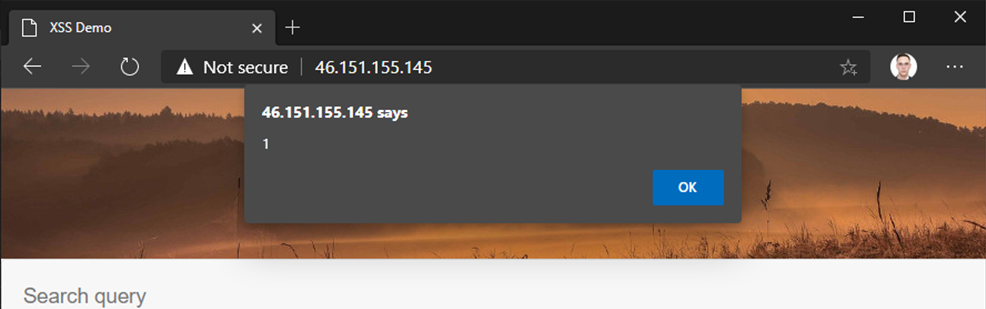




Пробуем ввести код:



Получаем похожий результат, как и в прошлом примере, баннер с «ошибкой»:



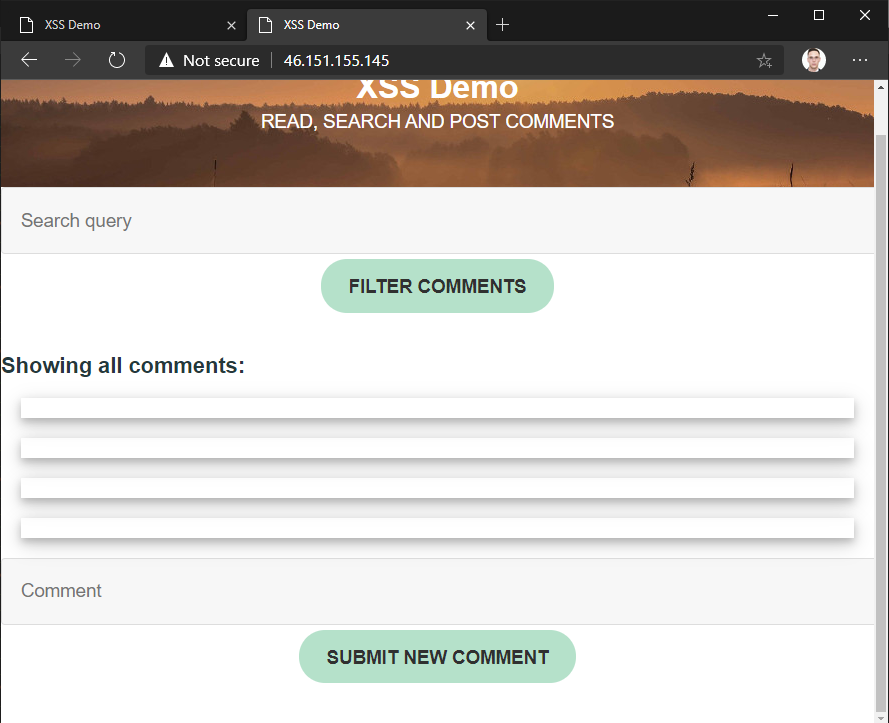
Механизмы защиты от XSS

В качестве примера мы ставили параметру «autoescape» значение false, так делать нельзя, иначе мы не сможем получать стандартную фильтрацию контекста HTML для переменных из нашего шаблона.

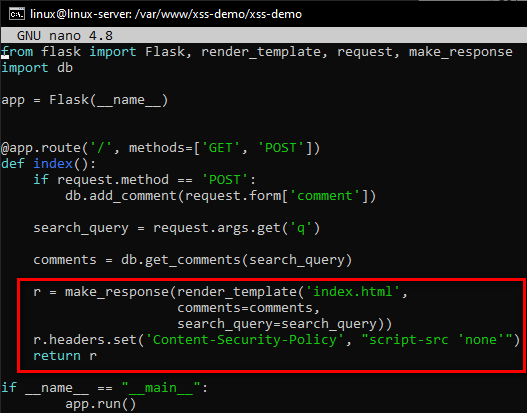
Помимо этого, постоянно всплывающие баннеры можно избежать, правильно использую кавычки в нижеприведенном примере:



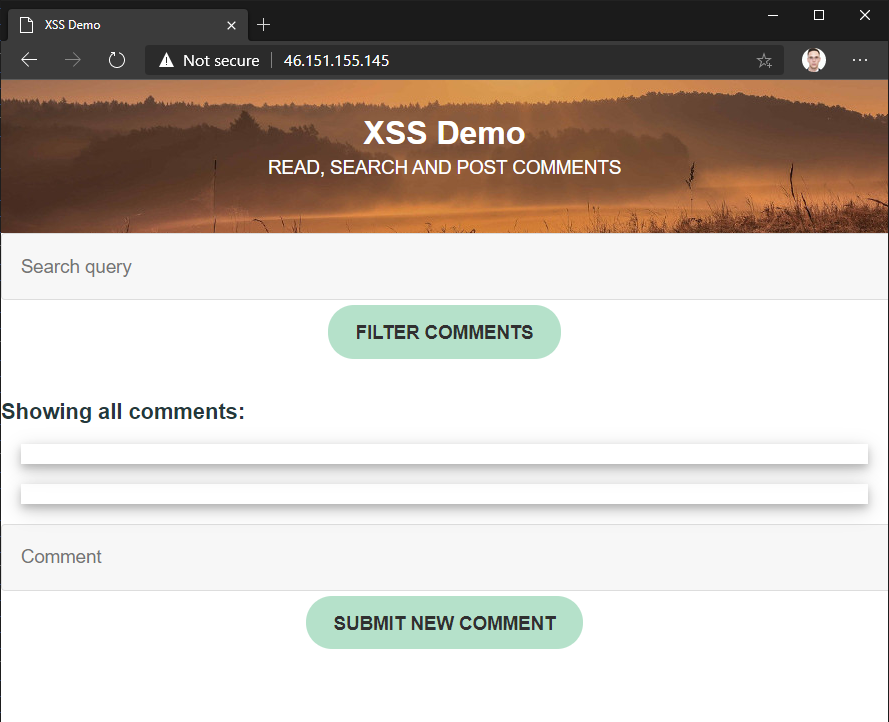
В результате чего, вводя оба наших кода, подсказки исчезают, остаются лишь пустые значения в переменных:



Также можно протестировать использование «Заголовков политики безопасности», чтобы запретить небезопасный встроенный javascript, заменив несколько строк:

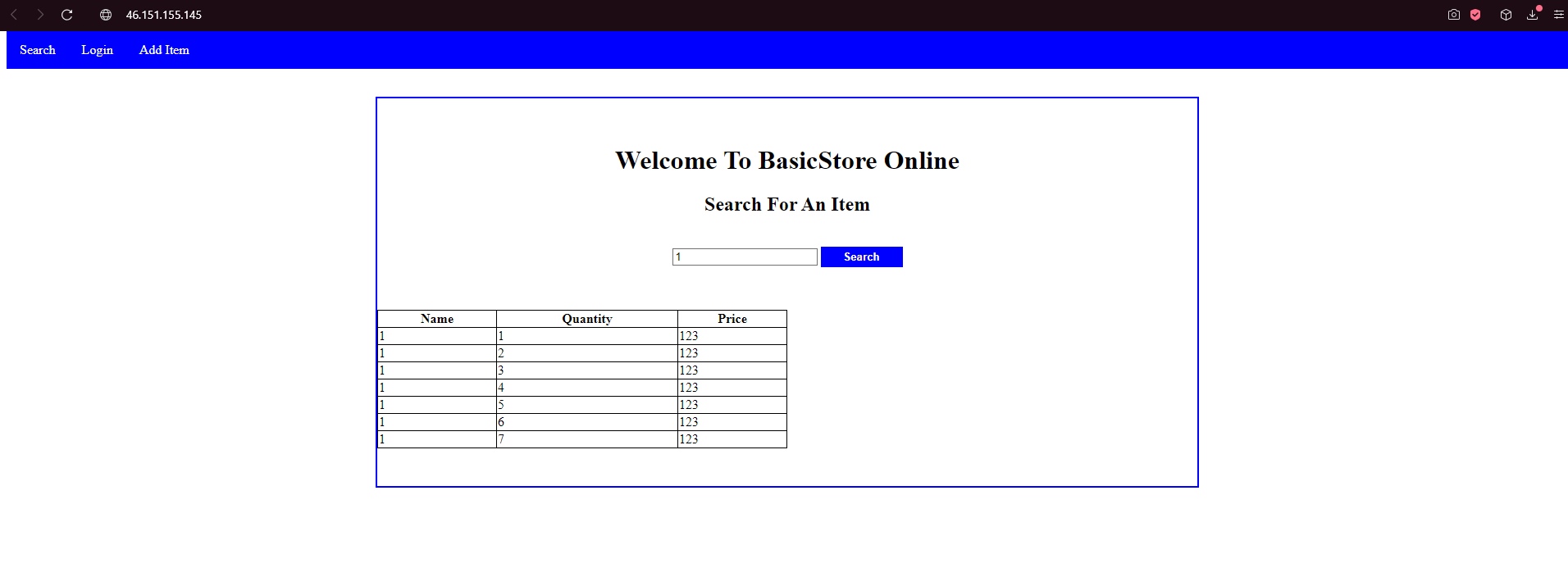


Проверка на работоспособность:

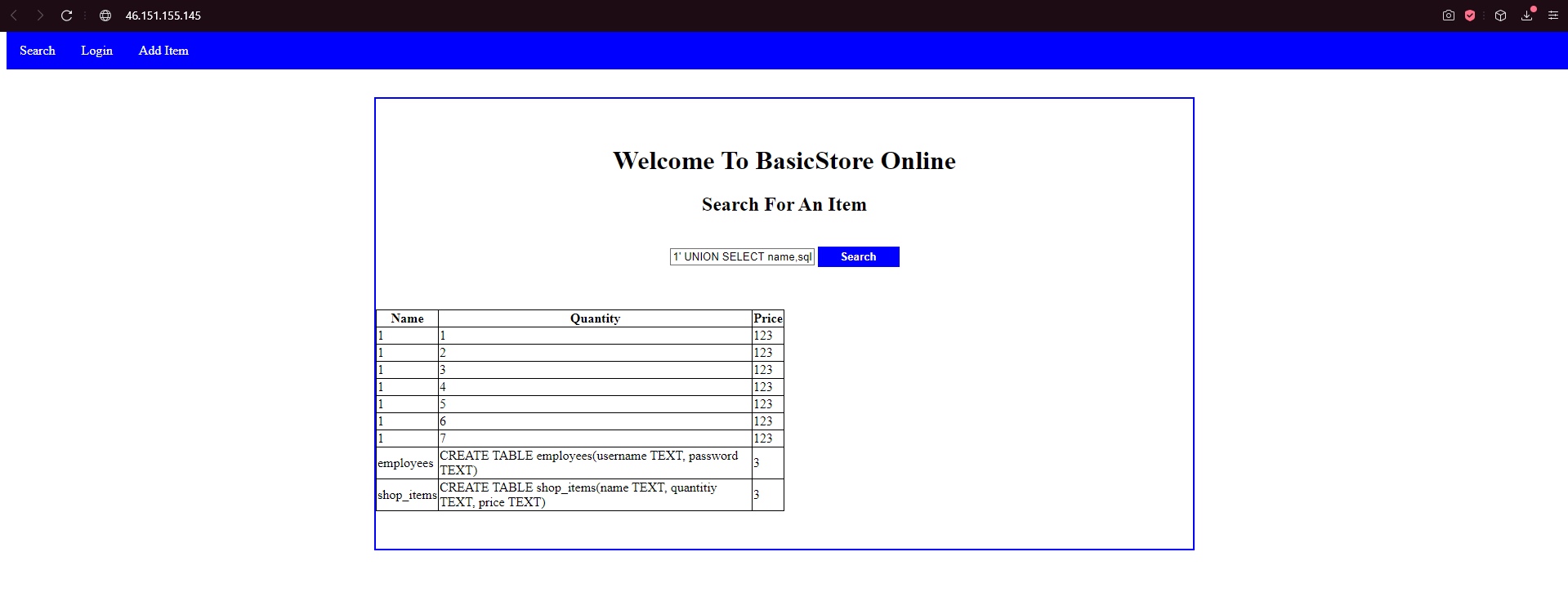


**SQL инъекции**

Сайт с простой базой данных, состоящий из одной таблицы и трёх атрибутов, поиск осуществляется по имени:

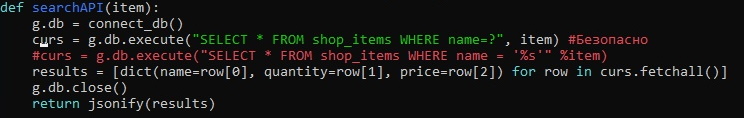


Пример SQL инъекции. При помощи UNION ALL и небольших манипуляций с запросом, вывели информацию о таблицах:

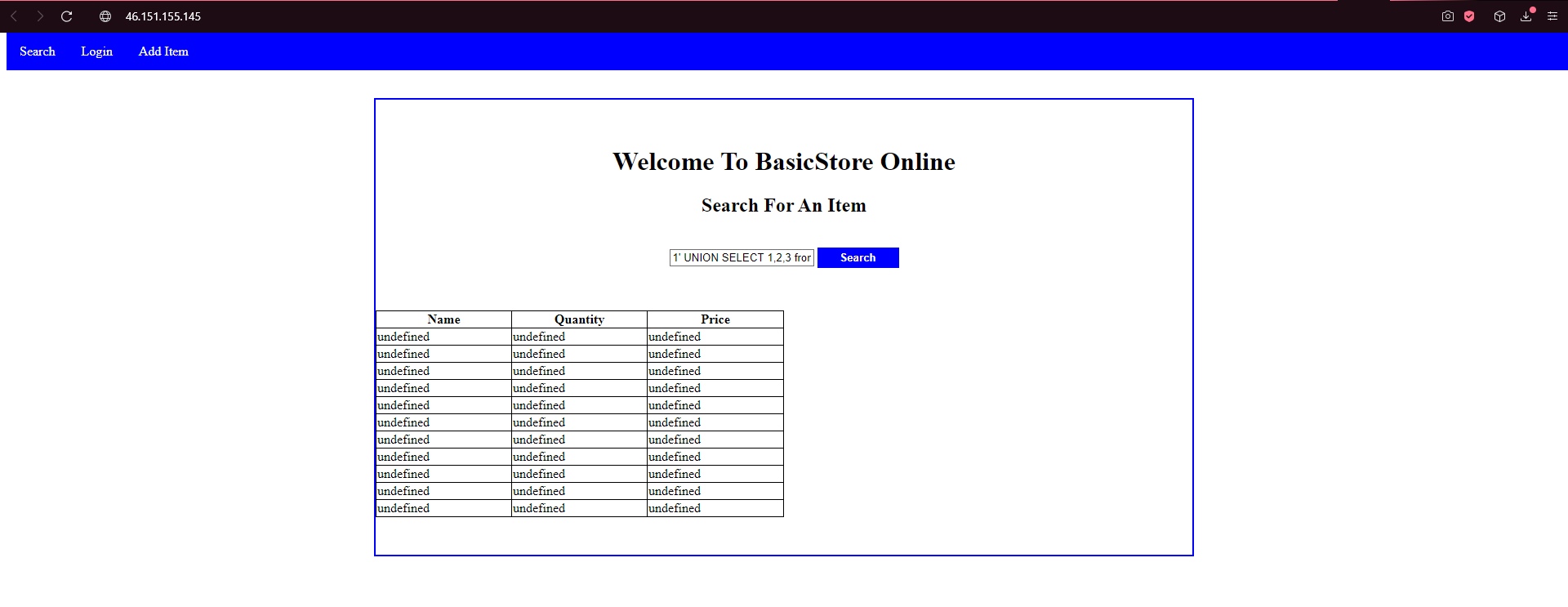


(1' UNION SELECT name,sql,3 from sqlite\_master WHERE type="table"; —)

Для предотвращения обработки строкой запросов следует экранировать спец-символы, а также не передавать напрямую запрос в код, а перед этим обработать его:

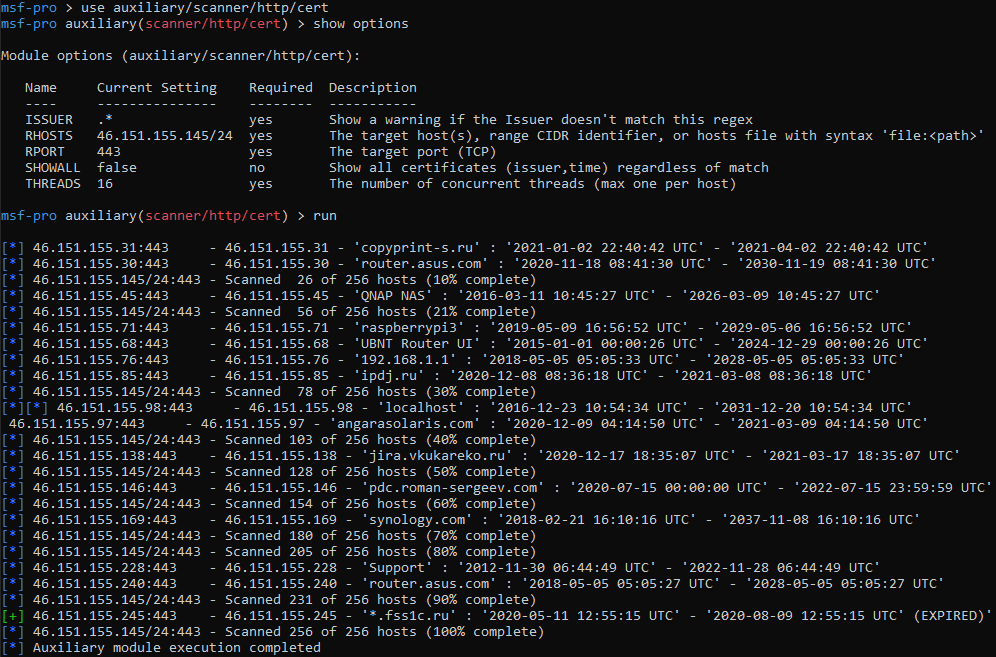


Теперь после выполнения запроса мы получили изначальную таблицу, которая была бы при пустом поле запроса (c выводом некоторых значений полей были проблемы, данный баг в виде undefined не имеет ничего общего с инъекцией, как минимум потому что не совпадают поля вывода, которые были при том же запросе в прошлый раз):



**CERT**

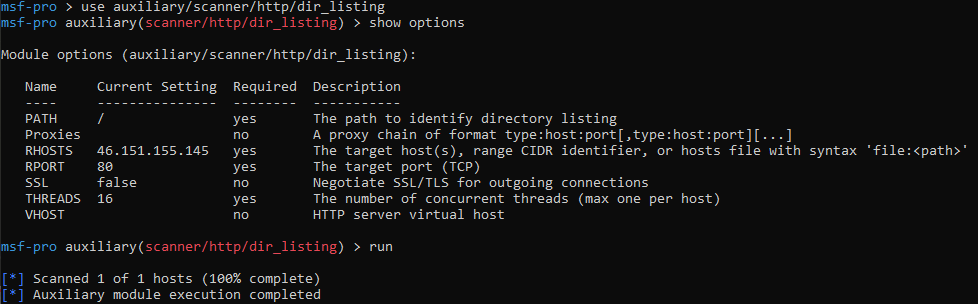
Модуль сканера сертификатов - это административный сканер, который позволяет покрыть подсеть, чтобы проверить, истек ли срок действия сертификатов серверов.



Выходные данные модуля показывают издателя сертификата, дату выпуска и срок действия.

**DIR\_LISTING**

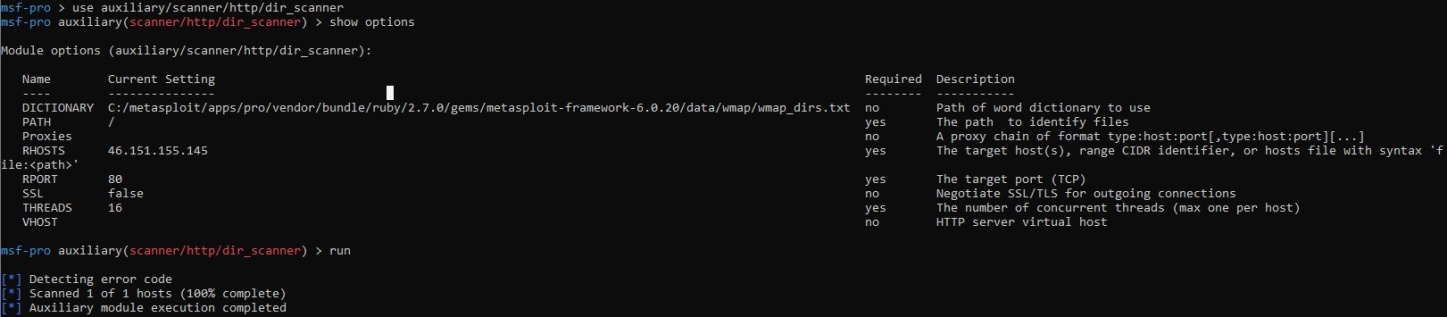
Модуль dir\_listing будет подключаться к указанному диапазону веб- серверов и определять, разрешены ли на них списки каталогов.



Как видно из вышеприведенного скрина, Metasploit не смог обнаружить ни один каталог в указанном диапазоне серверов.

**DIR\_SCANNER**

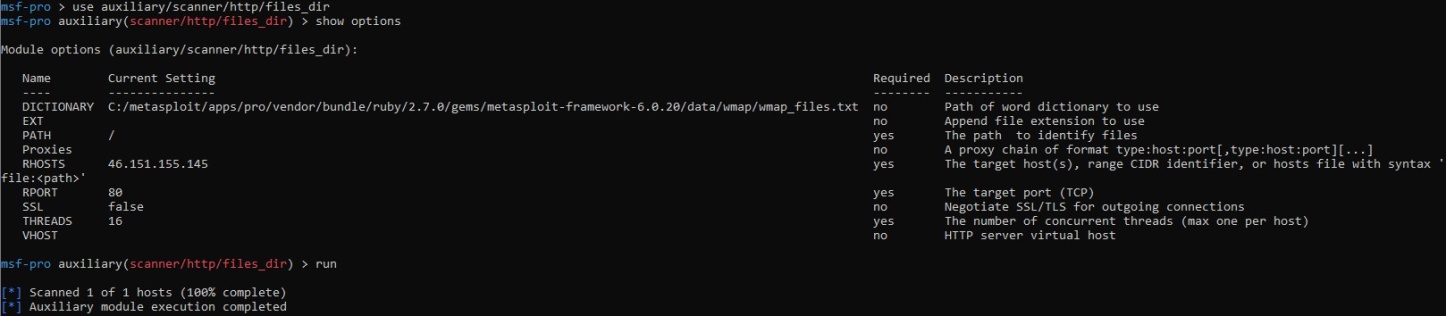
Модуль dir\_scanner сканирует один или несколько веб-серверов в поисках интересных каталогов, которые можно изучить в дальнейшем.



Быстрое сканирование не выявило каталогов на нашем сервере.

**FILES\_DIR**

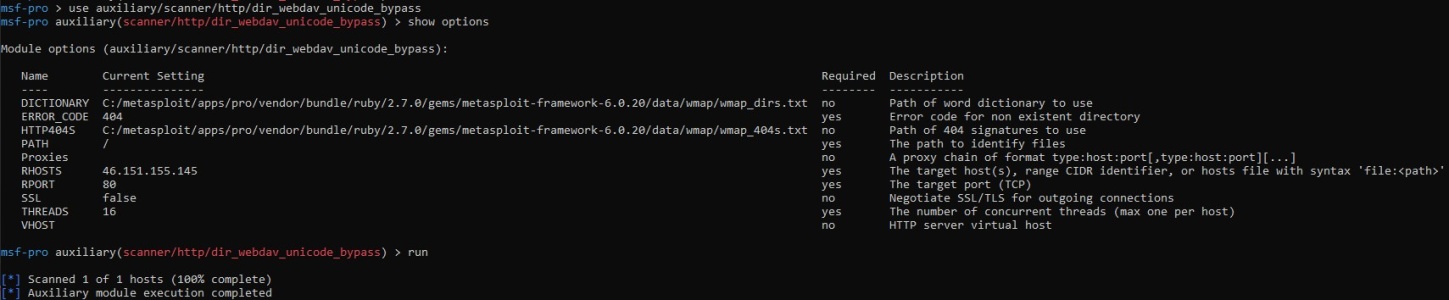
Files\_dir принимает список слов в качестве входных данных и запрашивает у хоста или диапазона хостов наличие подобных файлов на сервере.



Сканер не обнаружил файлы, которые подходили бы под наш заданный список поиска

**DIR\_WEBDAV\_UNICODE\_BYPASS**

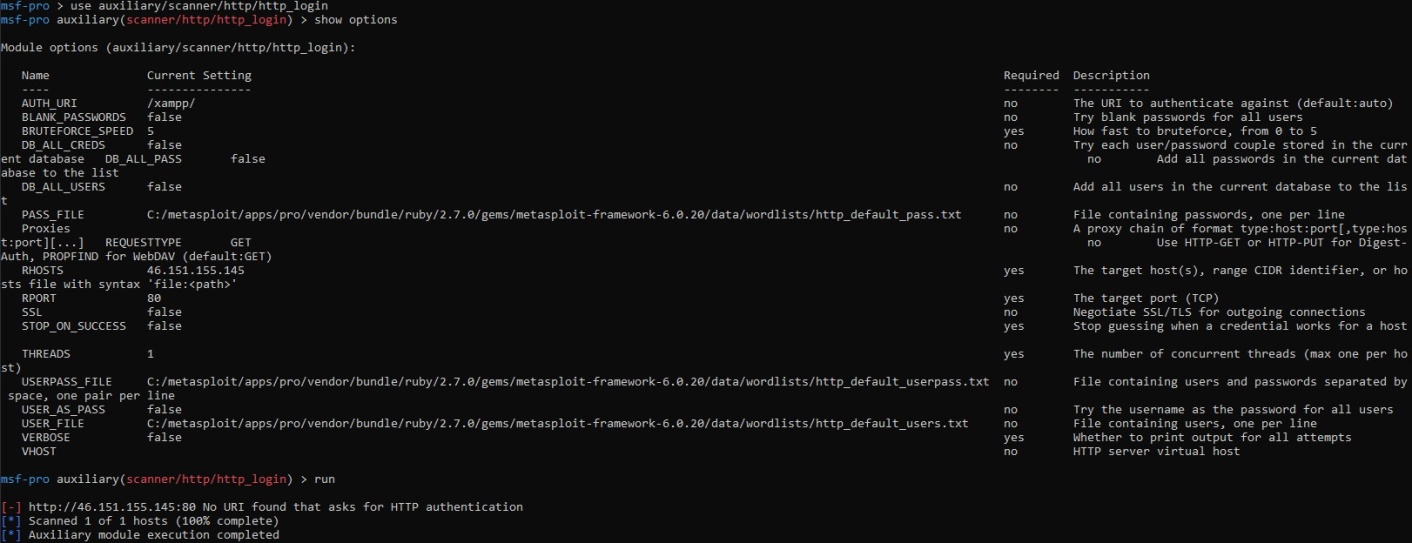
Модуль dir\_webdav\_unicode\_bypass сканирует заданный диапазон веб- серверов и пытается обойти аутентификацию, используя уязвимость WebDAV IIS6 Unicode.



Сканирование не обнаружило данную уязвимость в сервере.

**HTTP\_LOGIN**

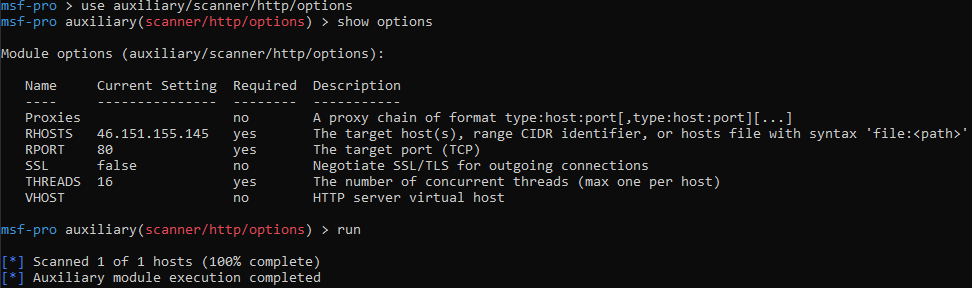
Модуль http\_login - это сканер входа в систему методом грубой силы, который пытается пройти аутентификацию в системе с использованием HTTP- аутентификации.



Данный эксплойт ничего не обнаружил, т.к. на нашем сервере не предусмотрена аутентификация по HTTP.

**OPTIONS**

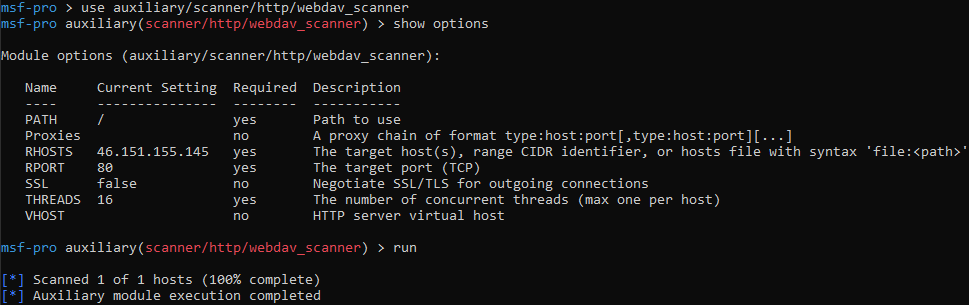
Модуль сканера параметров подключается к заданному диапазону IP- адресов и запрашивает у любых веб-серверов доступные для них параметры. Некоторые из этих опций можно использовать для проникновения в систему.



Модуль не обнаружил параметры, которые позволили бы проникнуть в систему.

**WEBDAV\_SCANNER**

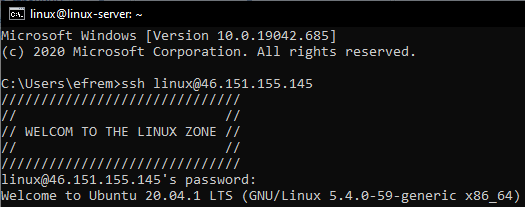
Модуль webdav\_scanner сканирует сервер или ряд серверов и пытается определить, включен ли WebDav. Это позволяет нам лучше настраивать наши атаки.

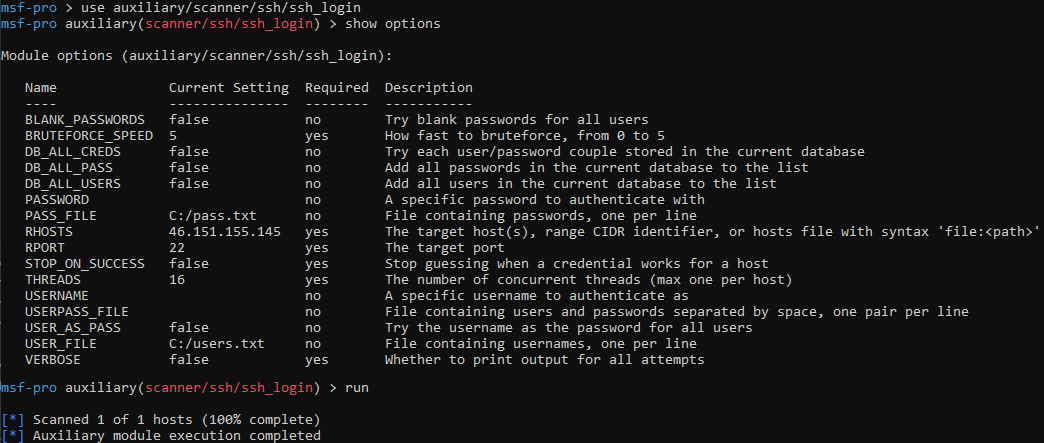


Из скриншота видно, что WebDav на нашем сервере не включен.

**BRUTEFORCE**

Для доступа по SSH было решено использовать стандартный модуль доступа «грубой силой». Был запущен модуль перебора паролей.

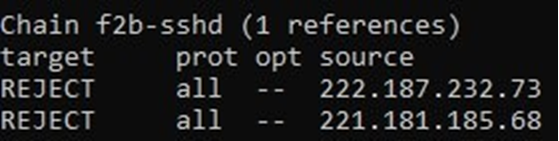


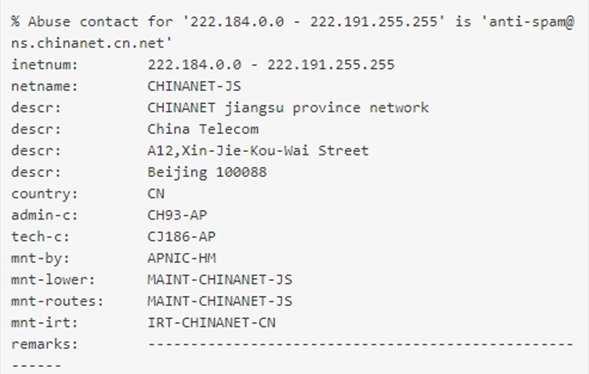




**ПРАКТИЧЕСКИЙ ПРИМЕР**

Через какое-то время после настройки Fail2Ban было решено проверить заблокированные подключения, так как у одного из нашей группы не получалось получить доступ к серверу. При просмотре списка заблокированных адресов были обнаружены два адреса, не принадлежащие никому из нашей команды. Решили проверить данные адреса при помощи сервиса WHOIS, оказалось, что эти адреса принадлежат китайской компании. Не станем делать выводы по поводу того, при каких обстоятельствах и в ходе каких действий был найден наш небольшой тестовый сервер, однако раз данные адреса оказались заблокированными, значит, что была попытка соединения, которая оказалась неудачной или специально прерванной.





# ЗАКЛЮЧЕНИЕ

В данной работе были рассмотрены основные уязвимости веб-сервера, а также сервисов и приложений, на данном сервере расположенных. Кроме того, была проделана небольшая практическая работа, позволившая оценить объём работы при создании хорошо-защищённого сервера, а также возможности различных систем тестирования безопасности. В ходе работы были получены практические навыки работы и поднятия сервера, его начальной настройки, удалённой работы на сервере, и установления различных видов соединения.

Стоит отметить, что уязвимости, представленные и описанные в данной работе, являются далеко не единственными, а реализация каждой уязвимости может отличаться в зависимости от сайта, веб-приложения, типа базы данных и каналов обмена данными с сервером.

# Библиографический список

* + 1. «Внутреннее устройство Linux», Дмитрий Кетов, 2017 г.;
    2. «Linux карманный справочник», Скотт Граннеман, 2016 г.;
    3. Виртуальная энциклопедия Linux: <http://rus-linux.net/MyLDP/sec/cyber-> attacks-web-exploitation.html;
    4. Информационная безопасность: https://habr.com/ru/post/176693/;
    5. Блог компании Издательский дом «Питер»: https://habr.com/ru/company/piter/blog/425571;
    6. Тестирование веб-сервисов: https://habr.com/ru/post/511318;
    7. Все о свободном программном обеспечении и операционной системе Linux: https://losst.ru/bezopasnost-servera-linux;
    8. Все о свободном программном обеспечении и операционной системе Linux: https://losst.ru/nastrojka-ufw-ubuntu;
    9. Metasploit Documentation: https://docs.rapid7.com/metasploit;
    10. Веб-сервис для хостинга IT-проектов и их совместной разработки: https://github.com/bgres/xss-demo;
    11. Веб-сервис для хостинга IT-проектов и их совместной разработки: https://github.com/JasonHinds13/hackable;
    12. https://xakep.ru/2014/09/09/webdav.