

**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

## Оглавление

ВВЕДЕНИЕ.....	3
1. SSH СОСТАВЛЯЮЩАЯ .....	4
1.1 Основные сведения.....	4
1.2 Brute - Force.....	5
1.3 Man-In-The-Middle (MITM).....	5
2. HTTP СОСТАВЛЯЮЩАЯ .....	7
2.1 Общие сведения.....	7
2.2 SQL – инъекция .....	8
2.3 Path traversal .....	9
2.4 XSS – Cross-Site-Scripting.....	10
2.5 WebDav.....	11
2.6 Промежуточный итог .....	12
3. ПРАКТИЧЕСКАЯ ЧАСТЬ .....	13
3.1 Пояснение к дальнейшим действиям .....	13
3.2 Настройка системы.....	13
3.3 Проведение тестирования .....	17
ЗАКЛЮЧЕНИЕ .....	32
Библиографический список .....	33

## **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, которые 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 слеш), фильтрация строковых параметров, фильтрация целочисленных параметров, усечение входных параметров, а также использование параметризованных 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
```

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 уязвимостей, 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 удастся 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 скрывания 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, заблокировать 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 узкой 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 с 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.

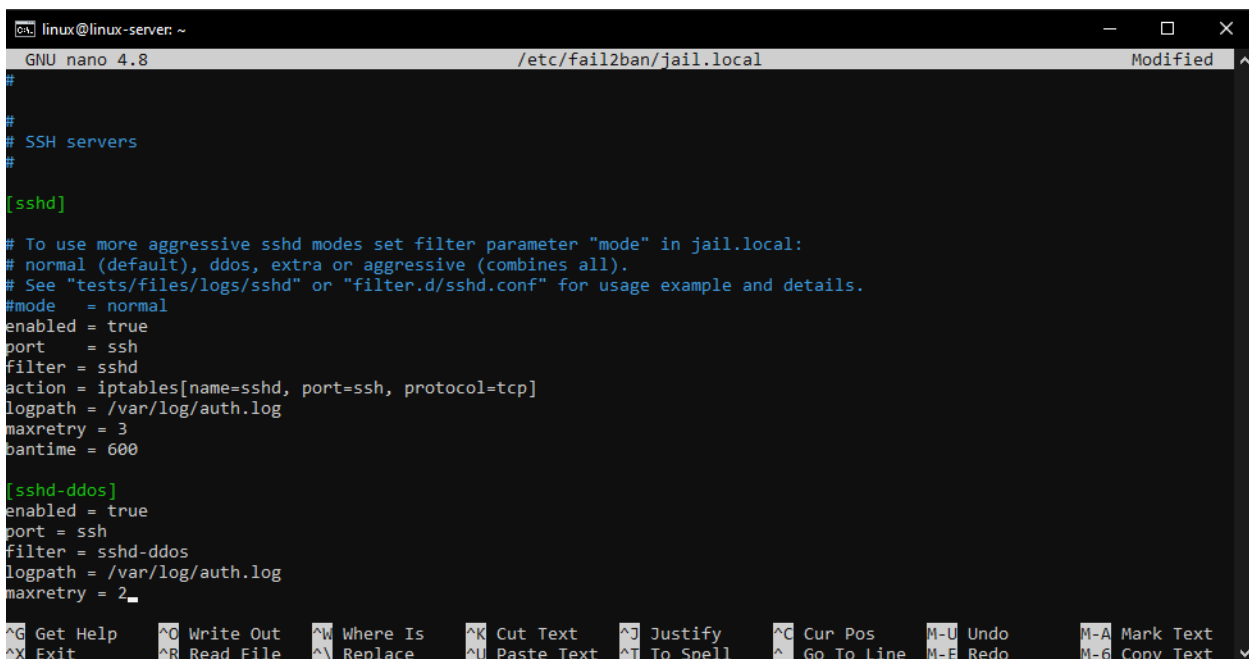
Стоит отметить, что мы решили не регистрировать доменное имя за ненадобностью, однако это всегда необходимо делать при работе с настоящим веб-сервером, в то время как возможность обращения к сервису при помощи IP стоит отключить, так как обычный рядовой пользователь в очень редких случаях будет использовать 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**

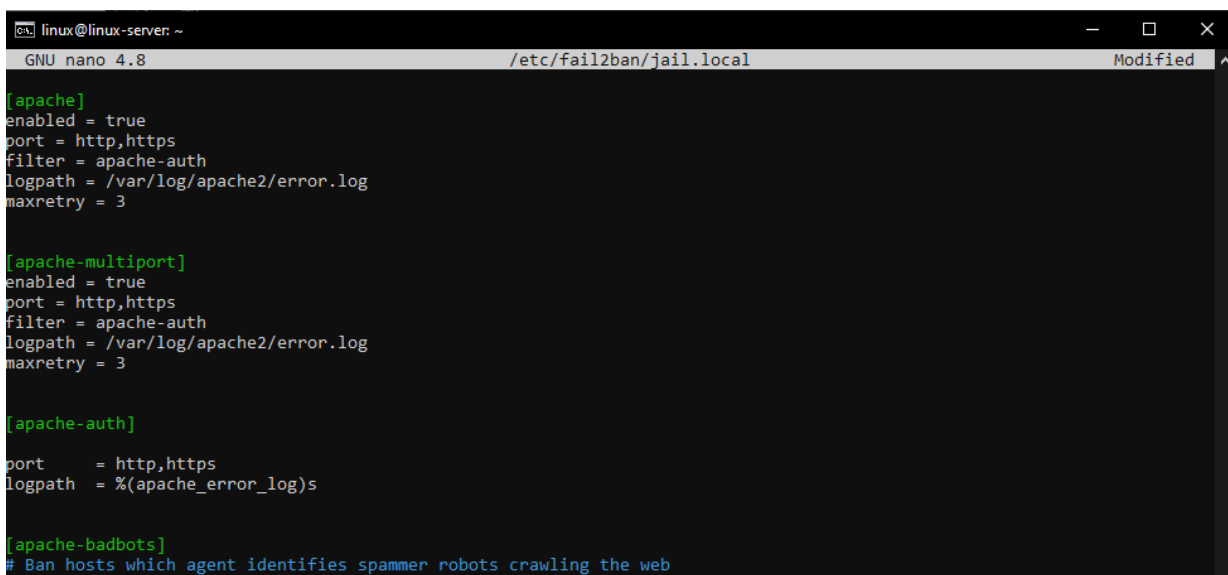
Начать было решено с сервиса fail2ban, который позволяет ограничить

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:



```
linux@linux-server: ~  
GNU nano 4.8 /etc/fail2ban/jail.local Modified  
#  
# SSH servers  
#  
[sshd]  
# To use more aggressive sshd modes set filter parameter "mode" in jail.local:  
# normal (default), ddos, extra or aggressive (combines all).  
# See "tests/files/logs/sshd" or "filter.d/sshd.conf" for usage example and details.  
#mode = normal  
enabled = true  
port = ssh  
filter = sshd  
action = iptables[name=sshd, port=ssh, protocol=tcp]  
logpath = /var/log/auth.log  
maxretry = 3  
bantime = 600  
[sshd-ddos]  
enabled = true  
port = ssh  
filter = sshd-ddos  
logpath = /var/log/auth.log  
maxretry = 2  
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos M-U Undo M-A Mark Text  
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^_ Go To Line M-E Redo M-6 Copy Text
```

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



```
linux@linux-server: ~  
GNU nano 4.8 /etc/fail2ban/jail.local Modified  
[apache]  
enabled = true  
port = http,https  
filter = apache-auth  
logpath = /var/log/apache2/error.log  
maxretry = 3  
[apache-multiport]  
enabled = true  
port = http,https  
filter = apache-auth  
logpath = /var/log/apache2/error.log  
maxretry = 3  
[apache-auth]  
port = http,https  
logpath = %(apache_error_log)s  
[apache-badbots]  
# Ban hosts which agent identifies spammer robots crawling the web  
# For valid addresses, the mail output must be fixed
```

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

```
kernel.exes-shield=1
kernel.randomize_va_space=1
net.ipv4.conf.all.rp_filter=1
net.ipv4.conf.all.accept_source_route=0
net.ipv4.icmp_echo_ignore_broadcasts=1
net.ipv4.icmp_ignore_bogus_error_messages=1
net.ipv4.conf.all.log_martians = 1
```

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):

```
linux@linux-server:~$ sudo ufw limit ssh/tcp
Rule added
Rule added (v6)
linux@linux-server:~$
```

```
linux@linux-server:~$ sudo ufw status verbose
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), deny (routed)
New profiles: skip

To Action From
--
22/tcp (OpenSSH) ALLOW IN Anywhere
500,4500/udp ALLOW IN Anywhere
500/udp ALLOW IN Anywhere
4500/udp ALLOW IN Anywhere
22/tcp LIMIT IN Anywhere
22/tcp (OpenSSH (v6)) ALLOW IN Anywhere (v6)
500,4500/udp (v6) ALLOW IN Anywhere (v6)
500/udp (v6) ALLOW IN Anywhere (v6)
4500/udp (v6) ALLOW IN Anywhere (v6)
22/tcp (v6) LIMIT IN Anywhere (v6)

linux@linux-server:~$ sudo ufw logging medium
Logging enabled
linux@linux-server:~$
```

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:

```
linux@linux-server: ~
GNU nano 4.8 /etc/default/grub Modified
# If you change this file, run 'update-grub' afterwards to update
# /boot/grub/grub.cfg.
# For full documentation of the options in this file, see:
# info -f grub -n 'Simple configuration'

GRUB_DEFAULT=0
GRUB_TIMEOUT_STYLE=hidden
GRUB_TIMEOUT=0
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT=""
GRUB_CMDLINE_LINUX="ipv6.disable = 1"

# Uncomment to enable BadRAM filtering, modify to suit your needs
# This works with Linux (no patch required) and with any kernel that obtains
# the memory map information from GRUB (GNU Mach, kernel of FreeBSD ...)
#GRUB_BADRAM="0x01234567,0xfefefefe,0x89abcdef,0xefefefef"

# Uncomment to disable graphical terminal (grub-pc only)
#GRUB_TERMINAL=console

# The resolution used on graphical terminal
# note that you can use only modes which your graphic card supports via VBE
# you can see them in real GRUB with the command 'vbeinfo'
#GRUB_GFXMODE=640x480

# Uncomment if you don't want GRUB to pass "root=UUID=xxx" parameter to Linux

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos M-U Undo M-A Mark Text
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^_ Go To Line M-E Redo M-6 Copy Text
```

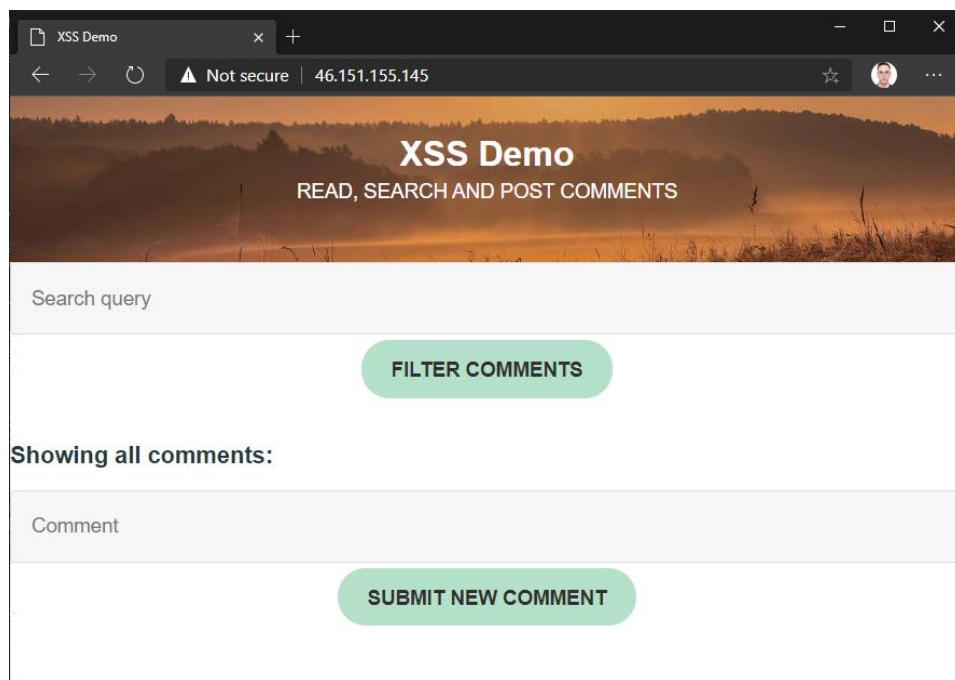


### **3.3 Проведение тестирования**

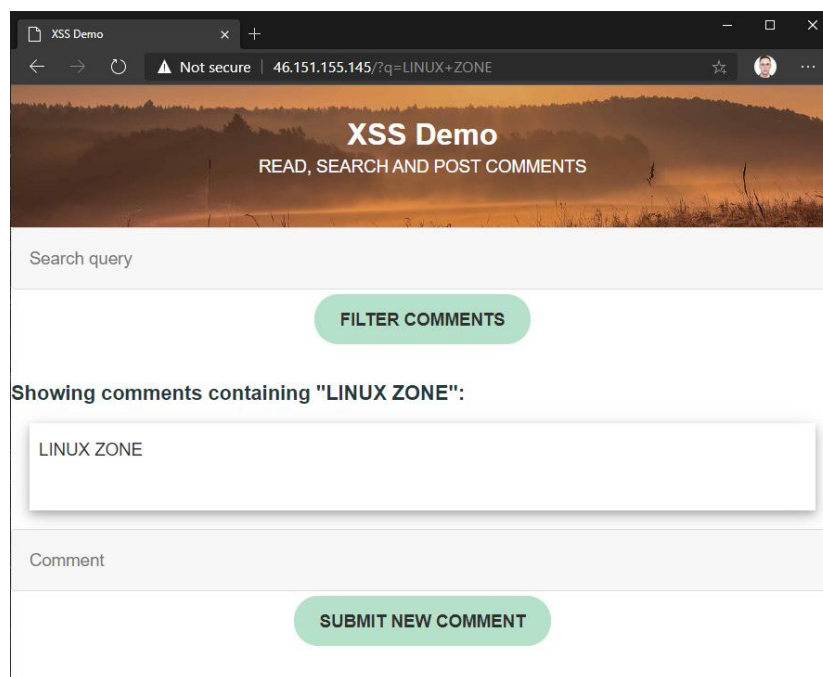
Для проведения тестов и проверки некоторых эксплойтов было решено использовать Метасплloit, который позволяет подробно изучить систему на предмет уязвимостей, а также провести некоторые типы атак. Также нам удалось застать практическую ситуацию, о которой будет рассказано после проведения тестирования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

## Проверка сайта на XSS

Сайт для проверки:



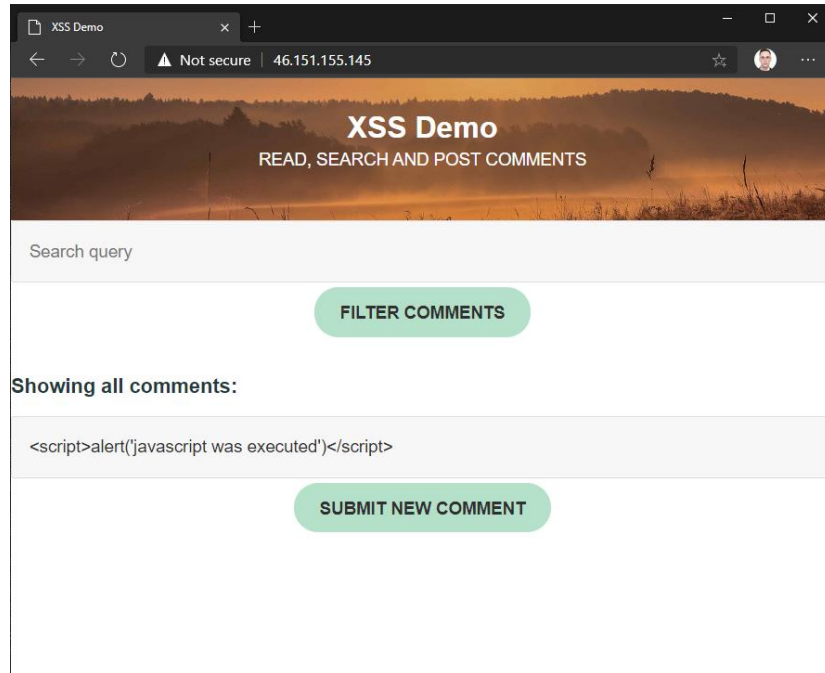
Проверка работоспособности:



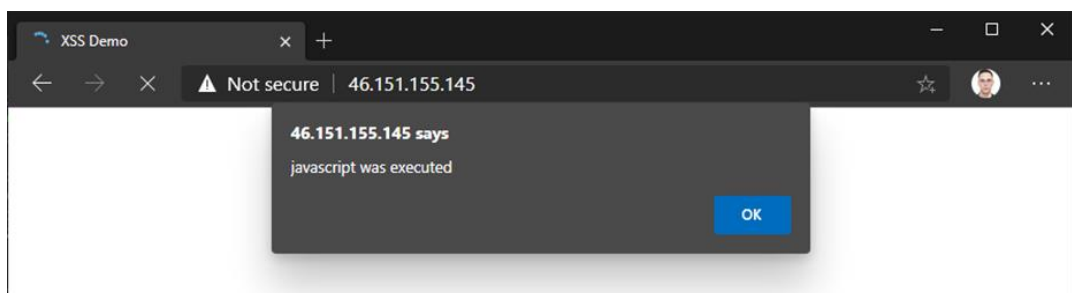
Для демонстрации недостатка XSS, изменили параметр «autoescape» на false:

```
linux@linux-server: /var/www/xss-demo/xss-demo
GNU nano 4.8
<!DOCTYPE html>
{% autoescape false %}
<html>
```

Пробуем ввести данный код:



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

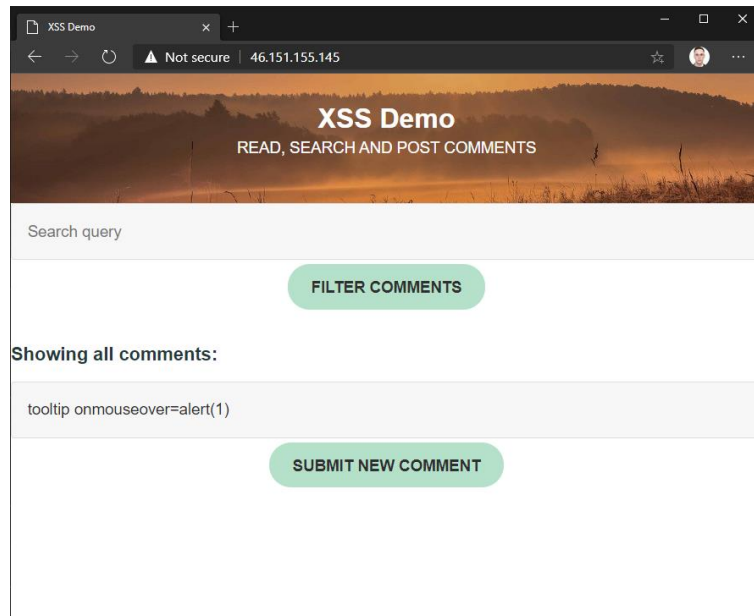


Для демонстрации необходимости контекстно-зависимой фильтрации можно изменить данные строки в том же файле, при этом вернув параметру «autoescape» значение true:

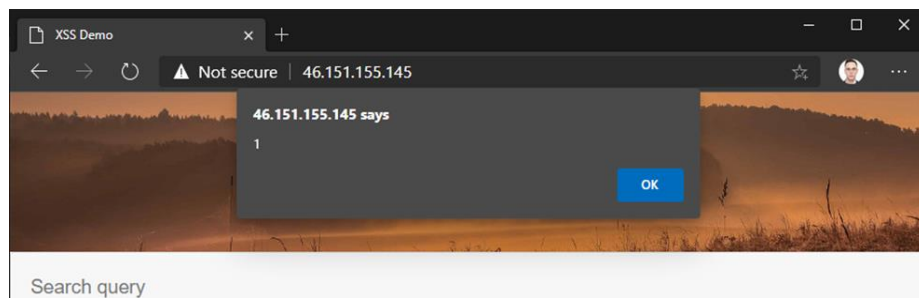
```
linux@linux-server: /var/www/xss-demo/xss-demo
GNU nano 4.8
<!DOCTYPE html>
{% autoescape true_%}
<html>

{% for comment in comments %}
  <div title={ { comment } }>
</div>
{% endfor %}
```

Пробуем ввести код:



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



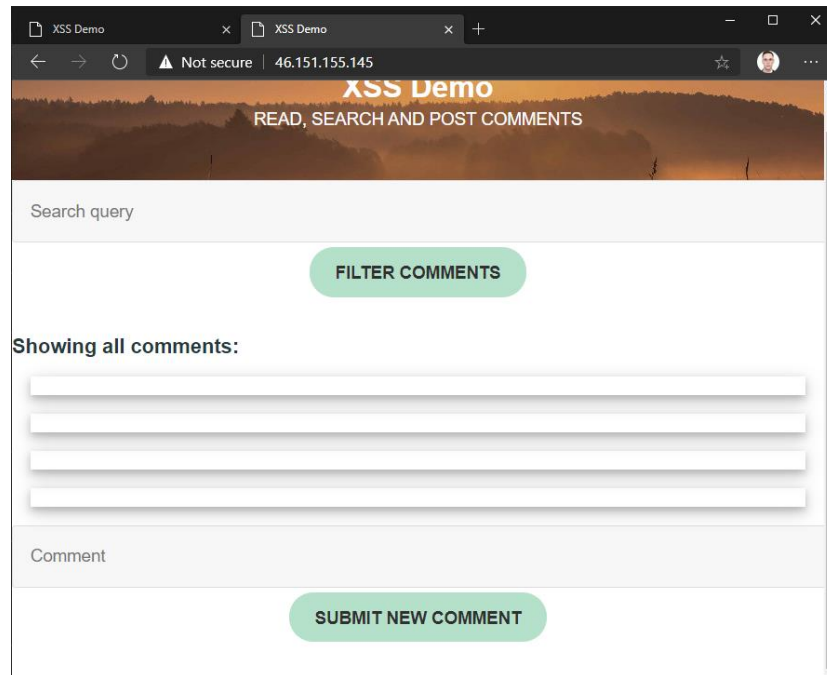
## Механизмы защиты от XSS

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 используя кавычки в нижеприведенном примере:

```
{% for comment in comments %}
<div title="{{ comment }}"></div>
{% endfor %}
```

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:

```
linux@linux-server: /var/www/xss-demo/xss-demo
GNU nano 4.8
from flask import Flask, render_template, request, make_response
import db

app = Flask(__name__)

@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        db.add_comment(request.form['comment'])

    search_query = request.args.get('q')

    comments = db.get_comments(search_query)

    r = make_response(render_template('index.html',
                                    comments=comments,
                                    search_query=search_query))
    r.headers.set('Content-Security-Policy', "script-src 'none'")
    return r

if __name__ == "__main__":
    app.run()
```

Проверка на работоспособность:

XSS Demo

READ, SEARCH AND POST COMMENTS

Search query

FILTER COMMENTS

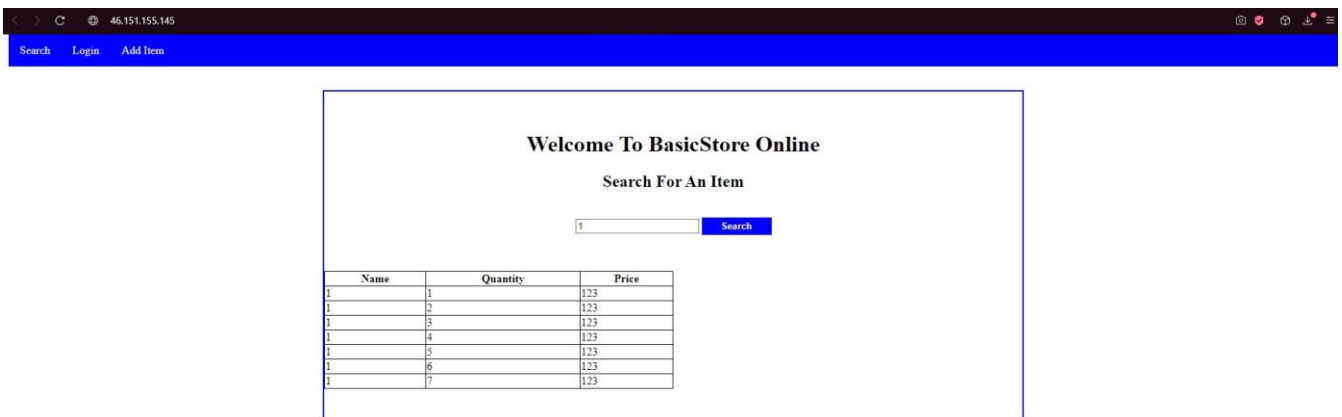
Showing all comments:

Comment

SUBMIT NEW COMMENT

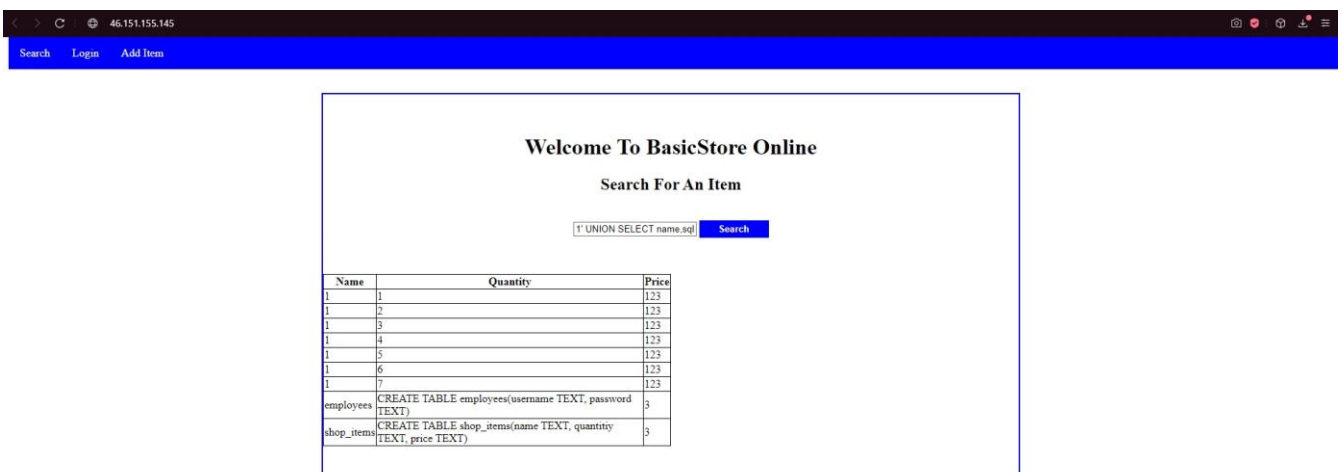
## SQL инъекции

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



Name	Quantity	Price
1	1	123
1	2	123
1	3	123
1	4	123
1	5	123
1	6	123
1	7	123

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



Name	Quantity	Price
1	1	123
1	2	123
1	3	123
1	4	123
1	5	123
1	6	123
1	7	123
employees	3	
shop_items	3	

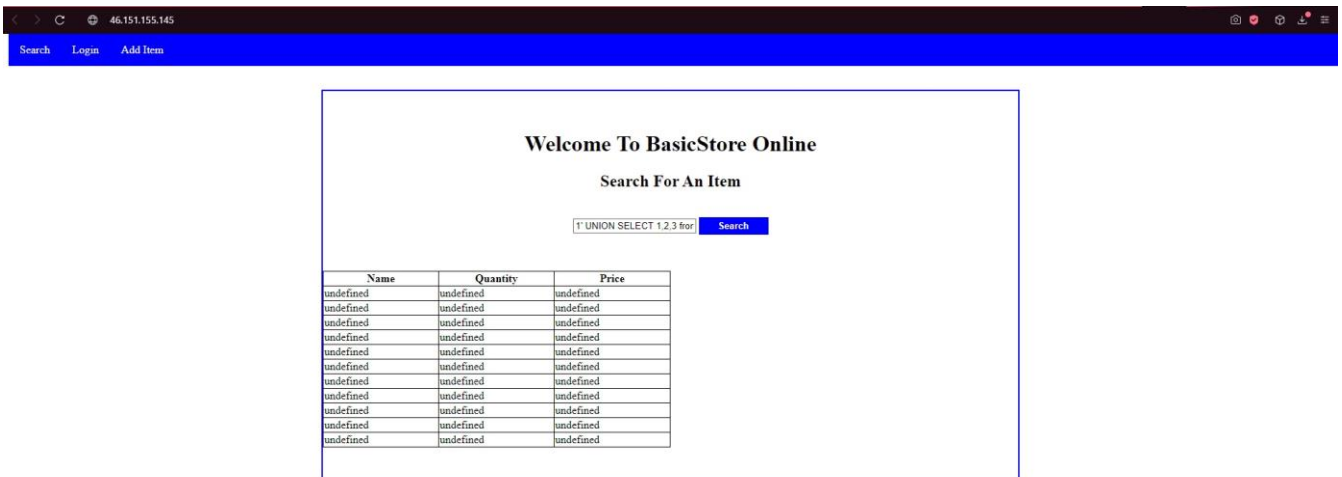
(1' UNION SELECT name,sql,3 from sqlite\_master WHERE type="table"; —)

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

```
def searchAPI(item):  
    g.db = connect_db()  
    curs = g.db.execute("SELECT * FROM shop_items WHERE name=?", item) #Безопасно  
    #curs = g.db.execute("SELECT * FROM shop_items WHERE name = '%s'" %item)  
    results = [dict(name=row[0], quantity=row[1], price=row[2]) for row in curs.fetchall()]  
    g.db.close()  
    return jsonify(results)
```

Теперь после выполнения запроса мы получили изначальную таблицу,

которая была бы при пустом поле запроса (с 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.

```
msf-pro > use auxiliary/scanner/http/cert
msf-pro auxiliary(scanner/http/cert) > show options

Module options (auxiliary/scanner/http/cert):

  Name      Current Setting  Required  Description
  ----      -
  ISSUER     .*               yes       Show a warning if the Issuer doesn't match this regex
  RHOSTS     46.151.155.145/24 yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
  RPORT      443              yes       The target port (TCP)
  SHOWALL    false            no        Show all certificates (issuer,time) regardless of match
  THREADS    16               yes       The number of concurrent threads (max one per host)

msf-pro auxiliary(scanner/http/cert) > run

[*] 46.151.155.31:443 - 46.151.155.31 - 'copyprint-s.ru' : '2021-01-02 22:40:42 UTC' - '2021-04-02 22:40:42 UTC'
[*] 46.151.155.30:443 - 46.151.155.30 - 'router.asus.com' : '2020-11-18 08:41:30 UTC' - '2030-11-19 08:41:30 UTC'
[*] 46.151.155.145/24:443 - Scanned 26 of 256 hosts (10% complete)
[*] 46.151.155.45:443 - 46.151.155.45 - 'QNAP NAS' : '2016-03-11 10:45:27 UTC' - '2026-03-09 10:45:27 UTC'
[*] 46.151.155.145/24:443 - Scanned 56 of 256 hosts (21% complete)
[*] 46.151.155.71:443 - 46.151.155.71 - 'raspberrypi3' : '2019-05-09 16:56:52 UTC' - '2029-05-06 16:56:52 UTC'
[*] 46.151.155.68:443 - 46.151.155.68 - 'UBNT Router UI' : '2015-01-01 00:00:26 UTC' - '2024-12-29 00:00:26 UTC'
[*] 46.151.155.76:443 - 46.151.155.76 - '192.168.1.1' : '2018-05-05 05:05:33 UTC' - '2028-05-05 05:05:33 UTC'
[*] 46.151.155.85:443 - 46.151.155.85 - 'ipdj.ru' : '2020-12-08 08:36:18 UTC' - '2021-03-08 08:36:18 UTC'
[*] 46.151.155.145/24:443 - Scanned 78 of 256 hosts (30% complete)
[*][*] 46.151.155.98:443 - 46.151.155.98 - 'localhost' : '2016-12-23 10:54:34 UTC' - '2031-12-20 10:54:34 UTC'
46.151.155.97:443 - 46.151.155.97 - 'angarasolaris.com' : '2020-12-09 04:14:50 UTC' - '2021-03-09 04:14:50 UTC'
[*] 46.151.155.145/24:443 - Scanned 103 of 256 hosts (40% complete)
[*] 46.151.155.138:443 - 46.151.155.138 - 'jira.vkukareko.ru' : '2020-12-17 18:35:07 UTC' - '2021-03-17 18:35:07 UTC'
[*] 46.151.155.145/24:443 - Scanned 128 of 256 hosts (50% complete)
[*] 46.151.155.146:443 - 46.151.155.146 - 'pdc.roman-sergeev.com' : '2020-07-15 00:00:00 UTC' - '2022-07-15 23:59:59 UTC'
[*] 46.151.155.145/24:443 - Scanned 154 of 256 hosts (60% complete)
[*] 46.151.155.169:443 - 46.151.155.169 - 'synology.com' : '2018-02-21 16:10:16 UTC' - '2037-11-08 16:10:16 UTC'
[*] 46.151.155.145/24:443 - Scanned 180 of 256 hosts (70% complete)
[*] 46.151.155.145/24:443 - Scanned 205 of 256 hosts (80% complete)
[*] 46.151.155.228:443 - 46.151.155.228 - 'Support' : '2012-11-30 06:44:49 UTC' - '2022-11-28 06:44:49 UTC'
[*] 46.151.155.240:443 - 46.151.155.240 - 'router.asus.com' : '2018-05-05 05:05:27 UTC' - '2028-05-05 05:05:27 UTC'
[*] 46.151.155.145/24:443 - Scanned 231 of 256 hosts (90% complete)
[*] 46.151.155.245:443 - 46.151.155.245 - '*.fssic.ru' : '2020-05-11 12:55:15 UTC' - '2020-08-09 12:55:15 UTC' (EXPIRED)
[*] 46.151.155.145/24:443 - Scanned 256 of 256 hosts (100% complete)
[*] Auxiliary module execution completed
```

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.

```
msf-pro > use auxiliary/scanner/http/dir_listing
msf-pro auxiliary(scanner/http/dir_listing) > show options

Module options (auxiliary/scanner/http/dir_listing):

  Name      Current Setting  Required  Description
  ----      -
  PATH      /                yes       The path to identify directory listing
  Proxies    no              no        A proxy chain of format type:host:port[,type:host:port][...]
  RHOSTS     46.151.155.145   yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
  RPORT      80              yes       The target port (TCP)
  SSL        false           no        Negotiate SSL/TLS for outgoing connections
  THREADS    16              yes       The number of concurrent threads (max one per host)
  VHOST      no              no        HTTP server virtual host

msf-pro auxiliary(scanner/http/dir_listing) > run

[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

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.

```
msf-pro > use auxiliary/scanner/http/dir_scanner
msf-pro auxiliary(scanner/http/dir_scanner) > show options

Module options (auxiliary/scanner/http/dir_scanner):

  Name      Current Setting  Required  Description
  ----      -
  DICTIONARY C:/metasploit/apps/pro/vendor/bundle/ruby/2.7.0/gems/metasploit-framework-6.0.20/data/urmap/urmap_dirs.txt no Path of word dictionary to use
  PATH      /                yes       The path to identify files
  Proxies    no              no        A proxy chain of format type:host:port[,type:host:port][...]
  RHOSTS     46.151.155.145   yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
  RPORT      80              yes       The target port (TCP)
  SSL        false           no        Negotiate SSL/TLS for outgoing connections
  THREADS    16              yes       The number of concurrent threads (max one per host)
  VHOST      no              no        HTTP server virtual host

msf-pro auxiliary(scanner/http/dir_scanner) > run

[*] Detecting error code
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

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.

```
msf-pro > use auxiliary/scanner/http/files_dir
msf-pro auxiliary(scanner/http/files_dir) > show options

Module options (auxiliary/scanner/http/files_dir):

  Name      Current Setting                                     Required  Description
  ----      -
  DICTIONARY C:/metasploit/apps/pro/vendor/bundle/ruby/2.7.0/gems/metasploit-framework-6.0.20/data/urmap/urmap_files.txt no       Path of word dictionary to use
  EXT        /                                                    no       Append file extension to use
  PATH       /                                                    yes      The path to identify files
  Proxies    /                                                    no       A proxy chain of format type:host:port[,type:host:port][...]
  RHOSTS     46.151.155.145                                       yes      The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
  RPORT      80                                                  yes      The target port (TCP)
  SSL        false                                               no       Negotiate SSL/TLS for outgoing connections
  THREADS    16                                                  yes      The number of concurrent threads (max one per host)
  VHOST      /                                                    no       HTTP server virtual host

msf-pro auxiliary(scanner/http/files_dir) > run
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

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.

```
msf-pro > use auxiliary/scanner/http/dir_webdav_unicode_bypass
msf-pro auxiliary(scanner/http/dir_webdav_unicode_bypass) > show options

Module options (auxiliary/scanner/http/dir_webdav_unicode_bypass):

  Name      Current Setting                                     Required  Description
  ----      -
  DICTIONARY C:/metasploit/apps/pro/vendor/bundle/ruby/2.7.0/gems/metasploit-framework-6.0.20/data/urmap/urmap_dirs.txt no       Path of word dictionary to use
  ERROR_CODE 404                                                yes      Error code for non-existent directory
  HTTP404S   C:/metasploit/apps/pro/vendor/bundle/ruby/2.7.0/gems/metasploit-framework-6.0.20/data/urmap/urmap_404s.txt no       Path of 404 signatures to use
  PATH       /                                                    yes      The path to identify files
  Proxies    /                                                    no       A proxy chain of format type:host:port[,type:host:port][...]
  RHOSTS     46.151.155.145                                       yes      The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
  RPORT      80                                                  yes      The target port (TCP)
  SSL        false                                               no       Negotiate SSL/TLS for outgoing connections
  THREADS    16                                                  yes      The number of concurrent threads (max one per host)
  VHOST      /                                                    no       HTTP server virtual host

msf-pro auxiliary(scanner/http/dir_webdav_unicode_bypass) > run
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

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.

```
msf-pro > use auxiliary/scanner/http/http_login
msf-pro auxiliary(scanner/http/http_login) > show options

Module options (auxiliary/scanner/http/http_login):

  Name           Current Setting      Required  Description
  ----           -
  AUTH_URI        /capp/               no        The URI to authenticate against (default:auto)
  BLANK_PASSWORDS false                no        Try blank passwords for all users
  BRUTEFORCE_SPEED 5                    yes       How fast to bruteforce, from 0 to 5
  DB_ALL_CREDS     false               no        Try each user/password couple stored in the current database to the list
  DB_ALL_PASS      false               no        Add all passwords in the current database to the list
  DB_ALL_USERS     false               no        Add all users in the current database to the list
  PASS_FILE        C:/metasploit/apps/pro/vendor/bundle/ruby/2.7.0/gems/metasploit-framework-6.0.20/data/wordlists/http_default_pass.txt no        File containing passwords, one per line
  Proxies          no                   no        A proxy chain of format type:host:port[,type:host:port][...]
  REQUESTTYPE      GET                  no        Use HTTP-GET or HTTP-PUT for Digest-Auth, PROPFIND for WebDAV (default:GET)
  RHOSTS           46.151.155.145       yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
  RPORT            80                  yes       The target port (TCP)
  SSL              false               no        Negotiate SSL/TLS for outgoing connections
  STOP_ON_SUCCESS  false               yes       Stop guessing when a credential works for a host
  THREADS          1                   yes       The number of concurrent threads (max one per host)
  USERPASS_FILE    C:/metasploit/apps/pro/vendor/bundle/ruby/2.7.0/gems/metasploit-framework-6.0.20/data/wordlists/http_default_userpass.txt no        File containing users and passwords separated by space, one pair per line
  USER_AS_PASS     false               no        Try the username as the password for all users
  USER_FILE        C:/metasploit/apps/pro/vendor/bundle/ruby/2.7.0/gems/metasploit-framework-6.0.20/data/wordlists/http_default_users.txt no        File containing users, one per line
  VERBOSE          false               yes       Whether to print output for all attempts
  VHOST            no                   no        HTTP server virtual host

msf-pro auxiliary(scanner/http/http_login) > run

[*] http://46.151.155.145:80 No URI found that asks for HTTP authentication
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

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.

```
msf-pro > use auxiliary/scanner/http/options
msf-pro auxiliary(scanner/http/options) > show options

Module options (auxiliary/scanner/http/options):

  Name           Current Setting      Required  Description
  ----           -
  Proxies          no                   no        A proxy chain of format type:host:port[,type:host:port][...]
  RHOSTS           46.151.155.145       yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
  RPORT            80                  yes       The target port (TCP)
  SSL              false               no        Negotiate SSL/TLS for outgoing connections
  THREADS          16                  yes       The number of concurrent threads (max one per host)
  VHOST            no                   no        HTTP server virtual host

msf-pro auxiliary(scanner/http/options) > run

[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

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.

```
msf-pro > use auxiliary/scanner/http/webdav_scanner
msf-pro auxiliary(scanner/http/webdav_scanner) > show options

Module options (auxiliary/scanner/http/webdav_scanner):

  Name      Current Setting  Required  Description
  ----      -
  PATH      /                yes       Path to use
  Proxies    /                no        A proxy chain of format type:host:port[,type:host:port][...]
  RHOSTS     46.151.155.145   yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
  RPORT      80               yes       The target port (TCP)
  SSL        false            no        Negotiate SSL/TLS for outgoing connections
  THREADS    16               yes       The number of concurrent threads (max one per host)
  VHOST      /                no        HTTP server virtual host

msf-pro auxiliary(scanner/http/webdav_scanner) > run

[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

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.

```
linux@linux-server: ~  
Microsoft Windows [Version 10.0.19042.685]  
(c) 2020 Microsoft Corporation. All rights reserved.  
  
C:\Users\efrem>ssh linux@46.151.155.145  
//  
//  
// WELCOM TO THE LINUX ZONE //  
//  
//  
//  
linux@46.151.155.145's password:  
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-59-generic x86_64)
```

```
msf-pro > use auxiliary/scanner/ssh/ssh_login  
msf-pro auxiliary(scanner/ssh/ssh_login) > show options  
  
Module options (auxiliary/scanner/ssh/ssh_login):  
  
  Name           Current Setting  Required  Description  
  ----           -  
  BLANK_PASSWORDS false           no        Try blank passwords for all users  
  BRUTEFORCE_SPEED 5               yes       How fast to bruteforce, from 0 to 5  
  DB_ALL_CREDS     false          no        Try each user/password couple stored in the current database  
  DB_ALL_PASS      false          no        Add all passwords in the current database to the list  
  DB_ALL_USERS     false          no        Add all users in the current database to the list  
  PASSWORD         false          no        A specific password to authenticate with  
  PASS_FILE        C:/pass.txt     no        File containing passwords, one per line  
  RHOSTS           46.151.155.145 yes        The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'  
  RPORT            22             yes       The target port  
  STOP_ON_SUCCESS  false          yes       Stop guessing when a credential works for a host  
  THREADS          16             yes       The number of concurrent threads (max one per host)  
  USERNAME         false          no        A specific username to authenticate as  
  USERPASS_FILE    false          no        File containing users and passwords separated by space, one pair per line  
  USER_AS_PASS     false          no        Try the username as the password for all users  
  USER_FILE        C:/users.txt    no        File containing usernames, one per line  
  VERBOSE          false          yes       Whether to print output for all attempts  
  
msf-pro auxiliary(scanner/ssh/ssh_login) > run  
[*] Scanned 1 of 1 hosts (100% complete)  
[*] Auxiliary module execution completed
```

```
C:\Users\efrem>ssh linux@46.151.155.145  
ssh: connect to host 46.151.155.145 port 22: Connection timed out
```

## 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.

```
Chain f2b-sshd (1 references)
target      prot opt source
REJECT      all  --  222.187.232.73
REJECT      all  --  221.181.185.68
```

```
% Abuse contact for '222.184.0.0 - 222.191.255.255' is 'anti-spam@
ns.chinanet.cn.net'
inetnum:      222.184.0.0 - 222.191.255.255
netname:      CHINANET-JS
descr:        CHINANET jiangsu province network
descr:        China Telecom
descr:        A12,Xin-Jie-Kou-Wai Street
descr:        Beijing 100088
country:      CN
admin-c:      CH93-AP
tech-c:       CJ186-AP
mnt-by:       APNIC-HM
mnt-lower:    MAINT-CHINANET-JS
mnt-routes:   MAINT-CHINANET-JS
mnt-irt:      IRT-CHINANET-CN
remarks:      -----
-----
```

## **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>.