Analyzing Vulnerabilities

- Analyze Vulnerability Scan Results
- Leverage Information to Prepare for Exploitation

Asset Categorization (Slide 1 of 2)

The process of placing business assets with similar characteristics into the same group.

- Helps a business shape how it works with each asset.
  - Example: How it prioritizes what assets receive what protections.
- Useful to pen tester for determining how to approach exploitation.
  - Assets in one category might be treated as less relevant or less applicable.
  - Assets in another category might be more important or more applicable.
- Categories vary from circumstance to circumstance.
- Example categories:
  - Public
  - Private
  - Restricted
  - Confidential

Asset Categorization (Slide 2 of 2)

- Previous categories describe sensitivity of data in terms of needed protections.
- Help you make decisions about what assets are:
  - The most challenging to compromise.
  - The most likely to be targeted.
  - The most valuable to an attacker.
  - The most devastating to the business.
- There are other ways to approach categorization.
- Categories that describe assets in terms of business roles: People, hardware, software, data, physical environment, processes, and third parties.
- Exploiting roles might have a greater impact on the business.

False Positives (Slides 1 of 2)

- Reasons for false positives:
  - Vendor may be trying to make their product look better.
  - Scanner can't recognize compensating controls.
  - Scanner is using outdated vulnerability database.
  - Scanner scores a vulnerability higher than it should be.
  - Target environment customizations trip up the scanner.
  - Scanner is improperly configured.
- Identify false positives to avoid wasting time on dead ends.

False Positives (Slides 2 of 2)

- Results validation is effective.
  - Compare what you've learned to individual results.
  - Are results truly applicable and accurate?
  - Scanner might be in error.
  - Example: Vulnerability identified in SMBv1, but you already know server is running SMBv3.
- Easier for defensive teams to identify false positives.
- Pen testers may have gaps in knowledge.
  - You may not be able to avoid all false positives.

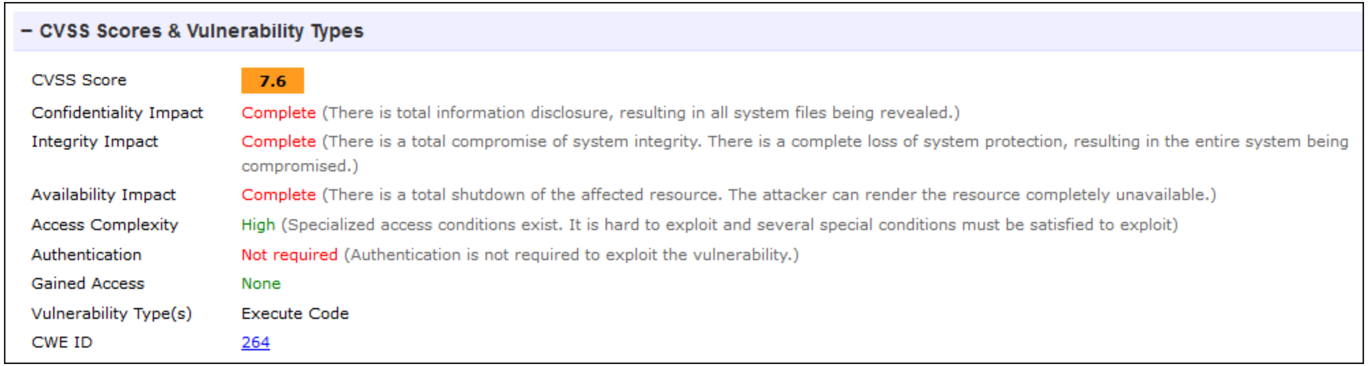
Adjudication (Slide 1 of 2)

The process of evaluating and ranking vulnerabilities in terms of the potential threat they may pose to the organization.

- Also implies action can/will be taken to minimize threat.
- Important factor in prioritizing exploit efforts.
  - Goal is to maximize test's efficiency.
- Consider using an established system.
- CVSS is an open standard for ranking vulnerabilities.
  - Quantifies vulnerability data through three metric groups.
  - Scoring is numerical with associated ratings.
- CVSS is leveraged by recognized vulnerability databases.
  - U.S. government's NVD.
  - Paired with vulnerabilities in CVE.

Rating	Score Range
None	0.0
Low	0.1–3.9
Medium	4.0–6.9
High	7.0–8.9
Critical	9.0–10.0

Adjudication (Slide 2 of 2)



– CVSS Scores & Vulnerability Types

CVSS Score	7.6
Confidentiality Impact	Complete (There is total information disclosure, resulting in all system files being revealed.)
Integrity Impact	Complete (There is a total compromise of system integrity. There is a complete loss of system protection, resulting in the entire system being compromised.)
Availability Impact	Complete (There is a total shutdown of the affected resource. The attacker can render the resource completely unavailable.)
Access Complexity	High (Specialized access conditions exist. It is hard to exploit and several special conditions must be satisfied to exploit)
Authentication	Not required (Authentication is not required to exploit the vulnerability.)
Gained Access	None
Vulnerability Type(s)	Execute Code
CWE ID	264

Vulnerability Prioritization (Slide 1 of 2)

- Need to decide what to dedicate time and money on.
  - You have a deadline and a limited budget.
- You determine what vulnerabilities get the most attention.
  - Time and money used effectively.
- Adjudication will influence prioritization.
- "Critical" vulnerabilities may have highest priority.
  - Easiest to exploit and/or have the most impact.

Vulnerability Prioritization (Slide 2 of 2)

- Not always best to prioritize by threat rating.
  - Strike a balance between ease of exploitation and impact.
  - Informed by client's environment.
  - Example: Domain controller compromise has severe consequences, but may be difficult.
  - You might demote "critical" vulnerabilities and promote "high" or "medium."
- Prioritization also influenced by mitigation cost.
  - Difficult for organization to fix some vulnerabilities.
  - More likely they'll accept the risk.
  - Your chances of exploitation might be better.

Common Themes (Slide 1 of 2)

- Examples:
  - Lax physical security.
  - Employees not following policies/best practices.
  - Lack of adequate training.
  - Lack of software patching.
  - Lack of OS hardening.
  - Poor software development practices.
  - Use of outdated networking protocols.
  - Use of obsolete cryptographic protocols.

Common Themes (Slide 2 of 2)

- You might stumble on a pattern of behavior.
  - Pattern can extend to assets you haven't yet tested.
- You can make educated guesses about how to test other assets.
  - Can make your job easier.
  - Can lead you down useful paths you wouldn't have taken otherwise.
- Identifying common themes provides you with a more complete picture.

Guidelines for Analyzing Vulnerability Scan Results (Slide 1 of 2)

- Determine an approach to categorizing client assets.
- Categorize assets according to chosen approach.
- Identify reasons why a scanner may produce false positives.
- Conduct results validation.
- Acknowledge you may not be able to eliminate false positives entirely.
- Rank vulnerabilities in terms of the potential threat they pose.
- Consider using an established ranking system like the CVSS.
- Prioritize vulnerabilities to use your time and money effectively.

Guidelines for Analyzing Vulnerability Scan Results (Slide 2 of 2)

- Use threat rankings to influence how you prioritize vulnerabilities.
- Strike a balance between a vulnerability's impact and ease of exploitation.
- Consider mitigation costs as an effect on your vulnerability prioritization.
- Identify common themes in your vulnerability results.
- Leverage a pattern of behavior on future testing efforts.
- Use common themes to develop a more complete picture.

Vulnerability Mapping

The act of recognizing the connection between a vulnerability and its associated target.

- Target can be person, process, device, etc.
- Gives you a reference for choosing attack techniques/exploits.
- More comprehensive than single scan results.
  - Informed by all scans.
- Update mapping with newly discovered vulnerabilities.
- Can also contain non-technical info like phishing targets/weaknesses.
- Can be a separate document or part of larger tactical planning document.

Activity Priorities

- Give priority to activities that are most likely to achieve objectives.
- Day-to-day activities may have shifting priorities.
  - Some investigations turn up promising leads.
  - Some reach dead ends.
- May need to shift priorities based on time constraints or target availability.
- Best practices:
  - Project manage pen test team.
  - Give early priority to time-consuming or opportunistic activities.
  - Give priority to activities most likely to reveal new targets/vectors.
  - Consider going after quick wins or low-hanging fruit to demonstrate success when asked.

Common Attack Techniques (Slide 1 of 2)

- Common types of attacks:
  - Social engineering
  - Code/command injection
  - DoS
  - Session hijacking/man-in-the-middle
  - Credential reuse
  - Brute forcing/password cracking
- Many attack types leverage common techniques.
- Most involve at least some level of social engineering.

Common Attack Techniques (Slide 2 of 2)

- Private networks are difficult to access.
- Access techniques:
  - Installing socially engineered malware on internal hosts.
  - Breaking into a WAP or remote access server.
  - Physically planting a malicious device on network.
  - Colluding with an insider.

Exploits and Payloads (Slide 1 of 3)

- Work together, but not the same thing.
- Exploit is a mechanism that delivers the payload.
  - Sequence of commands that takes advantage of a vulnerability.
- Example types of exploits:
  - Buffer overflows
  - Code injection
  - Web app exploits
- Some tools rank exploits based on reliability/effectiveness.
  - Example: Metasploit modules ranked from Manual to Excellent.
- Payload is code that runs on the target.
  - Performs a task like giving attacker control.

Exploits and Payloads (Slide 2 of 3)

- Example payloads:
  - Meterpreter session
  - VNC or other remote access
  - Backdoors/Trojans
  - Malicious DLLs
  - Worms/viruses
- Payloads can perform task on their own or wait for commands.
  - Can open a listening port and wait for attacker to connect.
  - Can make a connection back to attacker.
    - Useful when victim is behind a firewall.
- Most exploits/payloads are platform-specific.

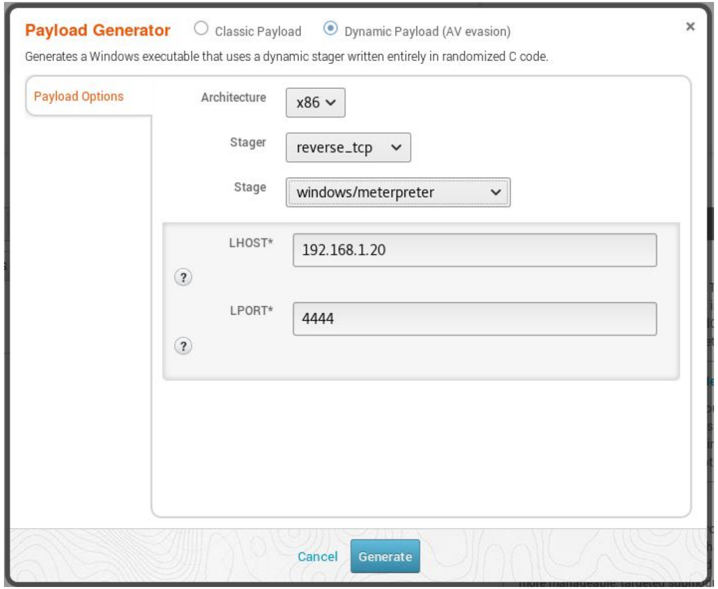
Exploits and Payloads (Slide 3 of 3)

- You usually have the choice of which payload the exploit delivers.
- Exploit sometimes delivers small payload called a stager.
  - Lightweight and reliable.
  - Gains foothold on victim.
  - Downloads larger payload (the stage) from attacker.
- Some payloads are self-contained (no staging necessary).
  - Called singles in Metasploit.

Cross-Compiled Code

Code that has been compiled into an executable on one platform, but is designed to run on a different platform.

- Common approach when crafting your own exploits.
- Example: Use Metasploit to craft a payload on Kali Linux.
  - You want to send payload to Windows.
  - Connects back to listener on Kali Linux.
- Using same tool on multiple targets is convenient and time-saving.



Payload Generator ○ Classic Payload ● Dynamic Payload (AV evasion) ×

Generates a Windows executable that uses a dynamic stager written entirely in randomized C code.

Payload Options

Architecture: x86

Stager: reverse_tcp

Stage: windows/meterpreter

LHOST*: 192.168.1.20

LPORT*: 4444

Exploit Modification

The process of changing an exploit that works against a particular vulnerability, but does not work under certain conditions.

- Example: Buffer overflow works against a Windows service.
  - Doesn't work if service packs have been applied.
  - Service packs patch vulnerability.
  - Author of service pack may have only worked with a variant of the vulnerability.
  - Pen tester can modify exploit to account for difference.
- Debugger can demonstrate how target responds to modified code.
- Example tools for modifying exploit code:
  - Metasploit
  - Immunity Debugger
  - Android Debug Bridge (ADB)
  - Java Debugger (jdb)
  - Mona.py

Exploit Chaining

The act of using multiple exploits to form a larger attack.

- Success may depend on all exploits doing their part.
- Distributed nature makes them complex and difficult to defend against.
- Some chained exploits must run consecutively.
- Some run in parallel.
- Examples:
  - Metasploit exploit that gives user-level shell, then privilege escalation to give system shell.
  - Module runs SQL injection, authentication bypass, and other exploits to give root shell.
  - Physically planting a device in an intrusion, then using that device to attack systems.
  - Distracting a guard so colleague can tamper with alarm system while another breaks into an office to steal documents.

Proof of Concept Development

Proof of concept: A benign exploit developed to highlight vulnerabilities.

- Usually created by security researchers.
- Demonstrates security issue to target organization or general public.
- Technical aspects might be published in great detail.
- Or, researcher may not include specifics.
  - Discourages malicious actors from using this exploit in the wild.

Deception Tactics (Slide 1 of 2)

- Primary mechanism in social engineering.
- Create trust, fear, etc., to induce victim to reveal info or do something they shouldn't.
- Common impersonation tactics:
  - Beleaguered fellow employee who needs help.
  - Authority figure from law enforcement threatening arrest or legal penalties.
  - New employee asking for help.
  - IT personnel asking someone to re-enter credentials.
  - Vendor or manufacturer warning about security vulnerability and offering a fix.
  - Customer trying to reset their password.
  - Co-worker using insider lingo to gain trust, asking for something to be done.
  - Friend or relative asking for help.
  - Vendor or creditor demanding overdue payment.

Deception Tactics (Slide 2 of 2)

- Common tactics to offer something not needed:
  - Distributing malware disguised as free media.
  - Offering help if a problem occurs.
  - Sending false pop-up windows asking for credentials.
  - Sending email with infected attachment.
  - Posting link to malicious site on social media.
  - Planting infected physical media in a workspace.

Task Completion Through Social Engineering

- Attacker may need to persuade victim to do something for them:
  - Disabling or bypassing security controls.
  - Granting physical or network access.
  - Creating or resetting credentials.
  - Delivering messages.
  - Installing software.
  - Authorizing payments.
  - Connecting or disconnecting devices.
  - Reconfiguring systems.

Dictionary Attacks (Slide 1 of 3)

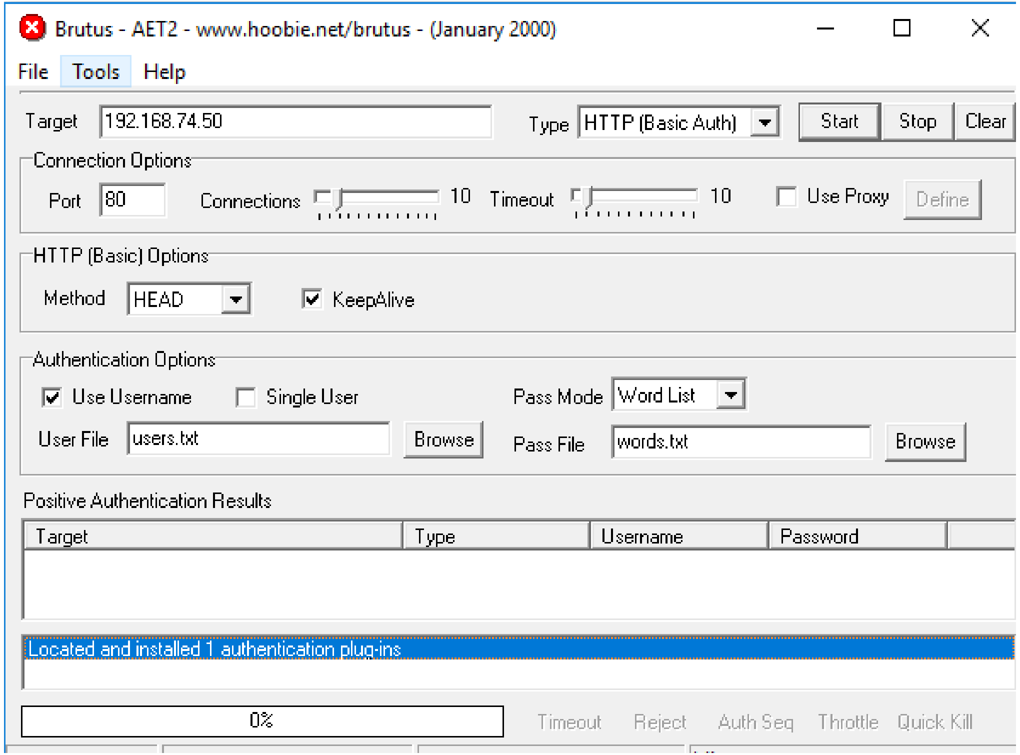
An attack in which a password cracking tool goes through a list of words until it either finds the password or exhausts the list.

- Hope is that a large enough list contains the password.
  - Most users choose simple, easy-to-remember passwords.
- Researchers have spent years collating wordlists.
  - Some websites collect passwords under the guise of testing their strength.
- Practical limitations:
  - Must know user name, though user names can also be in wordlists.
  - Lists can become unwieldy in their size (1.5 billion words ≈ 15 GB uncompressed).
  - Lockout policies on authentication systems.

Dictionary Attacks (Slide 2 of 3)

- Bypassing techniques:
  - Steal copy of file or database containing credentials (offline cracking).
  - Induce system to dump hashed passwords.
  - Intercept authentication and send to a password cracker.
  - Run cracker against network service without lockout.
  - Run cracker against accounts exempt from lockout (e.g., admin/root).

Dictionary Attacks (Slide 3 of 3)



Brutus - AET2 - www.hoobie.net/brutus - (January 2000)

Target 192.168.74.50

Type HTTP (Basic Auth)

Connection Options

Port 80

Connections 10

Timeout 10

□ Use Proxy

HTTP (Basic) Options

Method HEAD

☑ KeepAlive

Authentication Options

☑ Use Username

□ Single User

Pass Mode Word List

User File users.txt

Pass File words.txt

Positive Authentication Results

Target	Type	Username	Password

Located and installed 1 authentication plug-ins

0%

Rainbow Table Attacks

An attack in which the passwords in the wordlist have been pre-computed into their corresponding hashes, then compressed in a highly efficient manner.

- Makes offline cracking much faster.
  - No need to compute hashes of every password tried.
- Works with stolen file of password hashes.
- Reduction function reduces size of table.
  - Example: 2.5 million hashes could be stored in a text file of 25 entries.
  - 64 GB of a rainbow table can contain around 70 trillion hashes.
  - 64 GB of a wordlist can only contain around 6.5 billion passwords.
- Requires less computational power than plaintext dictionary.
- Password crackers that can use rainbow tables include Ophcrack, RainbowCrack, and CAPEC.

Credential Brute Force Attacks

An attack in which the attacker tries many passwords in the hope of eventually guessing the right one.

- If wordlist is exhausted, tool can try variations.
    - Substitute numbers or special characters for letters.
    - Combinations of characters.
- Can guess an encryption key created by a password.
    - Example: Wi-Fi password used to create hex-based key.
    - Attacker can extract key rather than discover actual password.
- Short passwords (e.g., 4-digit PIN) can be brute forced in minutes or even seconds.
- As length and complexity of password increases, brute forcing becomes harder.
- If brute forcing isn't feasible, attacker might steal hash and use that to authenticate.

Guidelines for Leveraging Information to Prepare for Exploitation

- Record vulnerability-to-target mappings in a reference document.
- Prioritize activities based on value to objectives, timing, and probability of success.
- Choose exploits based on platform and ranking.
- Choose payloads based on platform, connection type, effect, and level of control.
- Cross-compile exploits/payloads on single system.
- Use modified exploits against systems with different patch levels.
- Chain exploits for greater success.
- Use PoC exploits as basis to develop your own exploit code.
- Use deception tactics in social engineering to obtain desired information.
- Choose the password cracking technique that best suits your need.

Reflective Questions

1. Do you have any experience modifying exploit and/or payload code? Do you think you might do so in the future? Why or why not?

2. Discuss one or more instances in which a false positive has led you to a dead end, and how you dealt with it.