Blue Team Fundamentals Workbooks

- Many important steps of an attack are endpoint centric. Whilst data collection from each and every source is important, endpoints have a great vantage point.

**Service side exploits**: - an open, listening port required, repeated exploitation available, works if some software isn't patched, port isn't blocked and firewall allows traffic  
**Client side exploits**: - a client side exploit basically tricks the end user into either downloading a file or visiting a malicious website(phishing)

**Post-Exploitation: (ATT&CK framework) - Tactics:**

1. Execution: establishes code execution, happens after successful exploitation, used as a foothold for running additional programs.

2. Persistence: exploits may be patched, persistence itself is highly detectable, it does however give, albeit dependable repeated access over time. The ATT&CK lists 59 different techniques. Most common include: Autorun items, malicious services, scheduled tasks, browser extensions, account creds.

3. Discovery: attackers must explore the environment to proceed with attacks. Things to look for as an attacker are: account names and groups, user permissions and privileges, folders and files on the local system and network, checking for running local and network services, other hosts on the network and installed applications.

4. Privilege escalation: escalation of permissions starting at (hopefully) the lowest level. Attackers accomplish this via exploiting something that has a higher privilege such as admin users, programs ran by root, kernel, software and operating system features.

5. Credential dumps: privilege escalation via credential dumping, usually referenced for Windows machines, admin can read memory or registry keys, windows stores passwords in these locations. A requirement is to have admin privileges in order to perform a credential dump. A great example of a tool that allows this is the infamous Mimikatz tool, which can be recompiled into many different programming languages or compiled from source which can throw off some AVs.

6. Lateral movement: since attackers rarely reach their goal/goals from the first compromised host they must pivot further into the environment to gather further access. Remote connection protocols commonly used by admninistrators to do work from afar so to speak, provides a great way for an attack to pivot deeper into an organization.

7. Collection: basically collecting compromised data and pillaging each host for juicy information, whilst preparing it for exfil.

8. Exfiltration: unless the attack had other objectives (such as forking a system) it usually ends with exfiltrating compromised data. The attacker could have to (or rather would want to) potentially move gigabytes of data, probably not from the source itself, has to find the means(such as an open port) and break the data down and send it continuously to be stealthier.

\*Finding a way out: If a network is properly set-up and it has a default deny outbound policy, then an attacker might not have much choice, if at all. He or she will have to identify what ports and destinations are left open (and to which host) and then decide on, preferably, the stealthiest protocol to use over the open port. Once the attacker figures something out, provided they use the correct service to use over that protocol a SOC analyst might only have a chance of detection upon examination of the destination URL a volume and or timing of the upload to tip them off.

**Preventing exploitation:**

Software Inventory/ Scanning - network scanning checks the network for software running on endpoints(nmap f.e.) If we don't know what we are protecting we can't protect it. Since new zero days are discovered all the time and threats change and adapt extremely quickly, continuous scanning is required, as it keeps track of the running software versions on hosts. Patching OS, server software and client-side software is the best way to prevent exploitation.

Some tools such as EMET(Enhanced mitigation experience toolkit) can stop certain exploits before they had the chance to actually infect the system. They can also log an exploitation attempt and retroactively apply protections to legacy systems. They can also prevent ransomware attacks by using controlled folder access (defining a "protected folder" that only certain applications can access).

**AV's:**

Prevent installation phase, identify files before they had the chance to be run, but struggle with in-memory only malware and aren't fully reliable because they're relatively easy to fool by advanced attackers. Traditional AV's were even worse since they were signature-based, newer ones use machine learning but they're still far from perfect.

**Application Control**:

Since many incidents involve executables (at some point) this is one of the best prevention and detection tools as they stop unknown executables from running. Can be implemented via: name, path, signature or hash. Reduces noise but also imperfect. This can however, be bypassed by injecting code into trusted processes, use OS exe's or finding a permission error etc.

**Host Intrusion Prevention and Detection systems:**

Register and detect changes to registry or files. They also collect logs, traffic and process monitoring. File integrity monitoring(FIM) can be another useful feature usually implemented through the HIDS/HIPS which periodically verifies integrity of files, folders and enables a fairly rapid detection of unwanted applications that way. One example of FIM usage could be to detect intrusions in the root folder of a webserver (webshell/backdoor).

**PAWs(Privileged access workstations):**

These are separate computers, virtual machines or jump boxes used for admin accounts.

**EDR:**

Logs processes, services, DLLs, files etc. Enhances visibility and can have immediate response actions for remediation.

**DLP:**

Focuses on discovery, collection and exfil tactics. Prevents sensitive data leakage, either from employee mistakes of malicious insiders.

**Audit Policies and Logging:**

Centralized logging is a good idea. Event logs can audit trail of events that occurred, tells us what was made to the system and who made it. Audit policies control what is and is not logged.

**Windows and Linux logging**

**Windows**stores its logs as xml and a more user-friendly description. One can view these logs via Event Viewer, they contain useful (and less useful) info and have their own special IDs.

Another important thing to look for in Windows are Kerberos service logs. If possible, Windows always authenticates using Kerberos. The client (device that wants to connect to a service) authenticates first by exchaning information with the KDC(key distribution center). Windows Domain Controllers will be creating logs along the way which an analyst might later view to interpret information about logons.

**Linux** OS written logs are in the syslog format, which is traditionally just plaintext. The journald daemon replaces plain text files on some versions in a more structured format. Logs are categorized by facility and severity, can be sent over UDP or TCP and can be encrypted.

**Syslog** can be confused with different terms. Basically the**syslog format logs** are picked by the **syslog daemon** and sent to a SIEM using the **syslog protocol(udp 514)**. Traditionally the linux log path is in **/var/log.**Linux has an auditing tool built in called**autidt.**One can also install a tool called Snoopy which is easier to interpret than auditd. There are numerous additional auditing tools for Linux such Auditbeat which can also output to a SIEM.

 From an endpoint, log collecting mainly consists of the usage of agents - which is either a built-in OS service or a third party. A lot of SIEM vendors have agents designed to work with their product. Scripts using push and pull or other tools might be used instead. To understand collection abilities well, one should know which log channels and sources are being picked up, which ones are excluded, what filters are applied and what fields have been picked up. Also a log format is important (xml, json, csv, syslog).

***Log enrichment is an important process which gives additional context and information that helps analysts understand a log better.***This additional info (such as a source, destination or file information) is useful in any case. SIEM field normalization fixes the issue of different log sources using different field names.

**Identifying Files:**

Using a Linux command "file" should automatically attempt to identify the type of file. One can also use **hexdump** and investigate for "magic bytes", look at strings or other nested content for clues. Magic bytes a.k.a file signature is a standard that almost all file types have in order to govern how their bytes must be structured for the program that reads them.

|  |  |  |
| --- | --- | --- |
| Executable Binaries | Mnemonic | Signature |
| DOS Executable | "MZ" | 0x4D 0x5A |
| PE32 Executable | "MZ"...."PE.." | 0x4D 0x5A ... 0x50 0x45 0x00 0x00 |

|  |  |  |
| --- | --- | --- |
| Compressed Archives | Mnemonic | Signature |
| Zip Archive | "PK.." | 0x50 0x4B 0x03 0x04 |
| Rar Archive | "Rar!...." | 0x52 0x61 0x72 0x21 0x1A 0x07 0x01 0x00 |

|  |  |  |
| --- | --- | --- |
| Image File Formats | Mnemonic | Signature |
| PNG Image | ".PNG...." | 0x89 0x50 0x4E 0x47 0x0D 0x0A 0x1A 0x0A |
| BMP Image | "BM" | 0x42 0x4D |
| GIF Image | "GIF87a" | 0x47 0x49 0x46 0x38 0x37 0x61 |

**What makes a file valid**? It's not the filename or the extension, it's purely based on the bytes of that file. If the magic bytes indicate a file is a PDF, content follows the specifications and opens in a PDF Reader, then for all intents and purposes the file is a PDF. The extensions are simply for the OS (mostly windows, since Linux doesn't really have an extension on every file) so that the OS knows what to open the file with.

**What are strings**? Since some bytes in a program encode to characters(ASCII, UTF-8 etc.) Strings are a continuous run of "printable" characters and a string terminates at the first unprintable character. Linux program conveniently called "strings" can be used to display printable strings from a file. A plaintext file's contents are fully printable.

**Exploit vs Payload:** An exploit can be thought of as a flaw in the programming that was misused(exploited), basically an unexpected input to a program. **Payload** is what happens next, after the exploit was successful(crypto mining, user creation, etc.).

*Analyzing malicious docs cheat sheet*: <https://zeltser.com/analyzing-malicious-documents/>