# II. MALWARE ANALYSIS

## Malware

* any software/code designed & built to exploit, dmg & compromise networks, comsys or digital assets
* typically performs harmful or unauthorized activity: stealing sensitive data, disruption of systems, unauthorized access & more
* operate in stealth to evade detection

## Malware Analysis: Goals

- study malware fx & behavior
  - malicious or benign (effects)
  - indicators of compromise (IoCs)
- analyze threat actor(s)'
  - intent & purpose
  - crimeware/nation state?
- facilitate incident detection & response
  - build detection, threat intel, threat hunting
  - incident response engagements

## Malware Types

### Exploit Attachments

### Viruses
infects devices & replicates itself across systems
- ↳ require human intervention to propagate
- ↳ can
  - modify computer fxs & apps
  - copy, delete & exfiltrate data
  - encrypt data to perform ransomware atks
  - carry out DDoS atks

### Worms
self-replicates & infects other computers w/out human intervention
- ↳ inserts itself in devices via security vulnerabilities or malicious links or files

### Ransomware
locks or encrypts files or devices, forces victims to pay ransom for recovery
- ↳ common types:
  - → locker ransomware — completely locks users out of device
  - → crypto ransomware — encrypts files on device
  - → extortionware — involves attkers stealing data & threatening to publish it unless ransom is paid
  - → double extortion ransomware (2x ransom) = crypto ransomware + extortionware
  - → triple extortion ransomware (3x ransom) = double extortion ransomware + DDoS
  - → ransomware as a service (RaaS): subscription
    * enables affiliates or customers to rent ransomware; ransomware dev gets a cut of each paid ransom

### Trojans
malicious software that appears legit to users
- ↳ relies on social engineering techniques to invade devices
- ↳ gives attkers backdoor access to a device, perform keylogging, install viruses or worms, or steal data
- ↳ deployed as a malicious payload to facilitate the exploit
  - → typically as a seemingly legit program or email attachment
  - → using exploit toolkits (ex., Metasploit)

### Remote Access Trojans (RATs)
- ↳ enable attkers to take ctrl of an infected device
- ↳ attkers can use infected device to infect other devices w/ the RAT & create a botnet

### Spyware
malware that downloads onto device w/out user's knowledge
- ↳ steals users' data to sell to advertisers & external users
- ↳ types:
  - → adware — software app that displays ads while program is running
  - → keyboard loggers — used to monitor & log keystrokes
  - → trojans
  - → mobile spyware — typically enters via SMS or MMS communication channels

### Rootkit
malicious software that enables threat actors to remotely access & ctrl a device
- ↳ facilitate spreading of other kinds of malware
- ↳ often go undetected
  - ∴ once in device, can deactivate antimalware & antivirus software
  - → infects boot loader (outside of OS); resides in boot section
    - ⇒ v. persistent, but v. difficult to create a good one

## Malware Analysis: Key Elements

1. static analysis
2. dynamic/behavioral analysis
3. reverse engineer/code analysis
4. extraction of IoCs
5. documentation

## Malware Analysis Process (categories)

### Fully-Automated Analysis (easiest)
run ("detonate") sus file in sandbox (to get report on its activities)

### Static Properties Analysis
examine metadata & other details embedded in file (ex., strings) w/out running it (spot areas to examine more deeply in subsequent steps)

### Interactive Behavior Analysis
run file in isolated lab environment (which you fully ctrl), tweaking lab's config in series of iterative experiments to study specimen's behavior

### Manual Code Reversing (most challenging)
examine code that compromises file w/ disassembler & debugger (to understand key capabilities & fill gaps for earlier analysis steps)

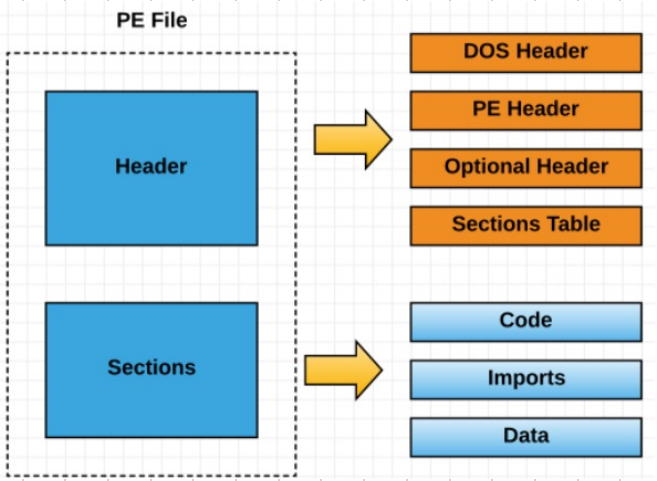
# Static Malware Analysis

art of collecting all available info abt the malicious binary

## Portable Executable (PE) File

* file formats for executables, DLLs, & obj. codes used in 32-bit & 64-bit Windows vers.
* contain useful info for malware analysts:
  - ↳ imports, exports, time-date stamps, subsys, sections, resources

DLL: dynamic-link library
↳ contains fxs & data that can be used by another app\DLL




Imptance of finding API calls:
* to understand feats. & capabilities of malware

DOS Header\DOS Stub
* incl. DOS executable header, found at beginning of file to indicate a DOS executable.
* MZ magic number 0x4d5a found here

PE Header: contains info, incl. location & size of code

PE Sections: contains main contents of file

## Tools

### Linux Cmds

- file — determines file type — `file [filename]`
- strings — used to extract readable strings fm a binary file — `strings [option] [filename]`
  - * can be used to find IoCs (ex., IP calls to DLL, understand services like ftp)
- shasum — returns file hash — `shasum -a 256 [filename]` // SHA-256 file hash

### manalyze

`manalyze [filename]`

* static analysis tool for PE files
* collect weak signals that could indicate malicious behavior; displays info that can help subsequent manual analysis

### peframe

`peframe [filename]`

* open source tool to perform static analysis on PE malware & generic sus. files
* detect packer, xor, digital signatures, mutex, anti-debug, anti-VM, sus. sections & fxs, etc.

### pedump

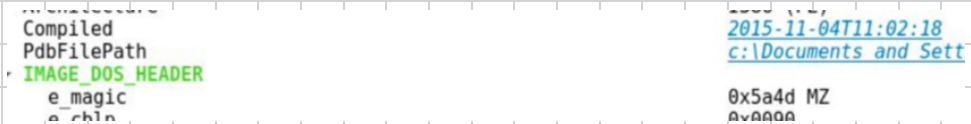
`pedump [filename]`

* contains script to dump headers, sections, extract resources of win32 PE exe, dll, etc.
* provides PE header view

### PE Tree

`petree [filename]`

* Python module for viewing PE files in a tree-view (graphically) using pefile & PyQt5
* note: MZ Header "Magic Value": 5A4D (in hex)




### oleid

`oleid [filename]`

analyzes OLE files (ex., MS office docs) for specific chars usually found in malicious files (ex., VBA macros, embedded flash objects)

### olevba

`olevba [filename]`

parses OLE or OpenXML files (ex., MS office docs) to
↳ detect VBA macros, extract src code & detect security-related patterns
- → auto-executable macros
- → anti-sandboxing & anti-virtualization techniques
- → sus. VBA keywords used by malware
- → potential IoCs (ex., IP addrs., URLs, executable filenames, etc.)

↳ detects & decodes several common obfuscation methods (ex., Hex encoding, StrReverse, Base64, Dridex, VBA expressions) & extracts IoCs fm decoded strings

### exiftool

`exiftool [filename]`

allow reading, writing, & editing of a file's meta info.

### PE Detective

Windows-based tool for analyzing & inspecting PE files

### dnSpy

.NET debugger & assembly editor; used in reverse engineering .NET apps.

### PE Bear

freeware, multi-platform reversing tool for PE files

*importance of filters in malware analysis

## Dynamic Malware Analysis

involves running malware in a ctrled environment (ex., VM, sandbox) to observe behavior in real-time

## Wireshark filters

tls.handshake.value == 1 → client_hello
2 → server_hello

http.request → packets w/ HTTP requests (ex., GET, POST, PUT, DELETE)

http.request == 1 → GET
2 → POST
4 → PUT
5 → DELETE

dns && ip.addr → DNS packets related to any IP addr (src & dest)
→ useful for monitoring DNS traffic to & frm a particular host or for analyzing DNS-related security issues (ex., DNS tunneling, sus. domain lookups)

tcp.flags.syn == 1 → SYN; TCP segment is initiating connection request

tcp.flags.ack == 1 → ACK; acknowledgment num-field valid

ftp.request.command → packets w/ FTP requests

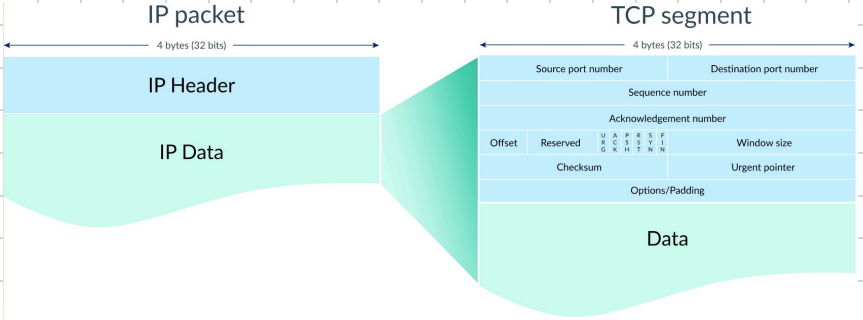
ftp.request.command == USER → specifies username for logging into FTP server
PASS → specifies password for logging into FTP server
LIST → request list of files in current dir on FTP server
STOR → initiates file upload (client → server)
RETR → initiates file download (server → client)
QUIT → terminate FTP session & close connection

ftp.data → indicates data transfers over FTP connection (ex., upload, download)

smtp → email traffic packets

smtp.req.command → SMTP request cmds

smtp.data.fragment → SMTP data fragments, ie., segments of email msg payload that are transmitted over network in separate pieces (due to packet fragmentation/segmentation)




(dns && ip.addr) || http.request
↳ allow simultaneous monitoring of DNS & HTTP traffic
↳ comprehensive traffic analysis: helps in understanding interplay btwn DNS & web traffic and detecting anomalies/threats which use both

(http.request || tls.handshake.type == 1) && !(ssdp)
↳ packets related to HTTP requests or TLS client hello msgs, except frm Simple Service Discovery Protocol (SSDP)

TCP 3-way handshake:
tcp.flags.syn == 1 && tcp.flags.ack == 0 — SYN packet
tcp.flags.syn == 1 && tcp.flags.ack == 1 — SYN-ACK packet
tcp.flags.syn == 0 && tcp.flags.ack == 1 — ACK packet

ftp.data && tcp.seq == 1
↳ identifies initial data packet in FTP transfer

smtp || dns
↳ identifies CC servers using SMTP or DNS servers; can show domain names and/or emails involved
↳ detect data exfiltration via email or DNS tunneling

## Importance of Wireshark filters

* comparing fields — with protocol against a specific val.
  └ against fields
* check existence of specified fields or protocols
* used by other feats. (ex., statistics generation, packet list colorization)