Chapter 3
Malware Analysis

Mohd Zaki Mas’ud

1 Nov 2020

Topic

- Introduction
- Creating a safe environment
- Static & Dynamic analysis
  - Static
  - Dynamic
- Automated analysis
- Malware Sample Source
- Armored malware

Introduction

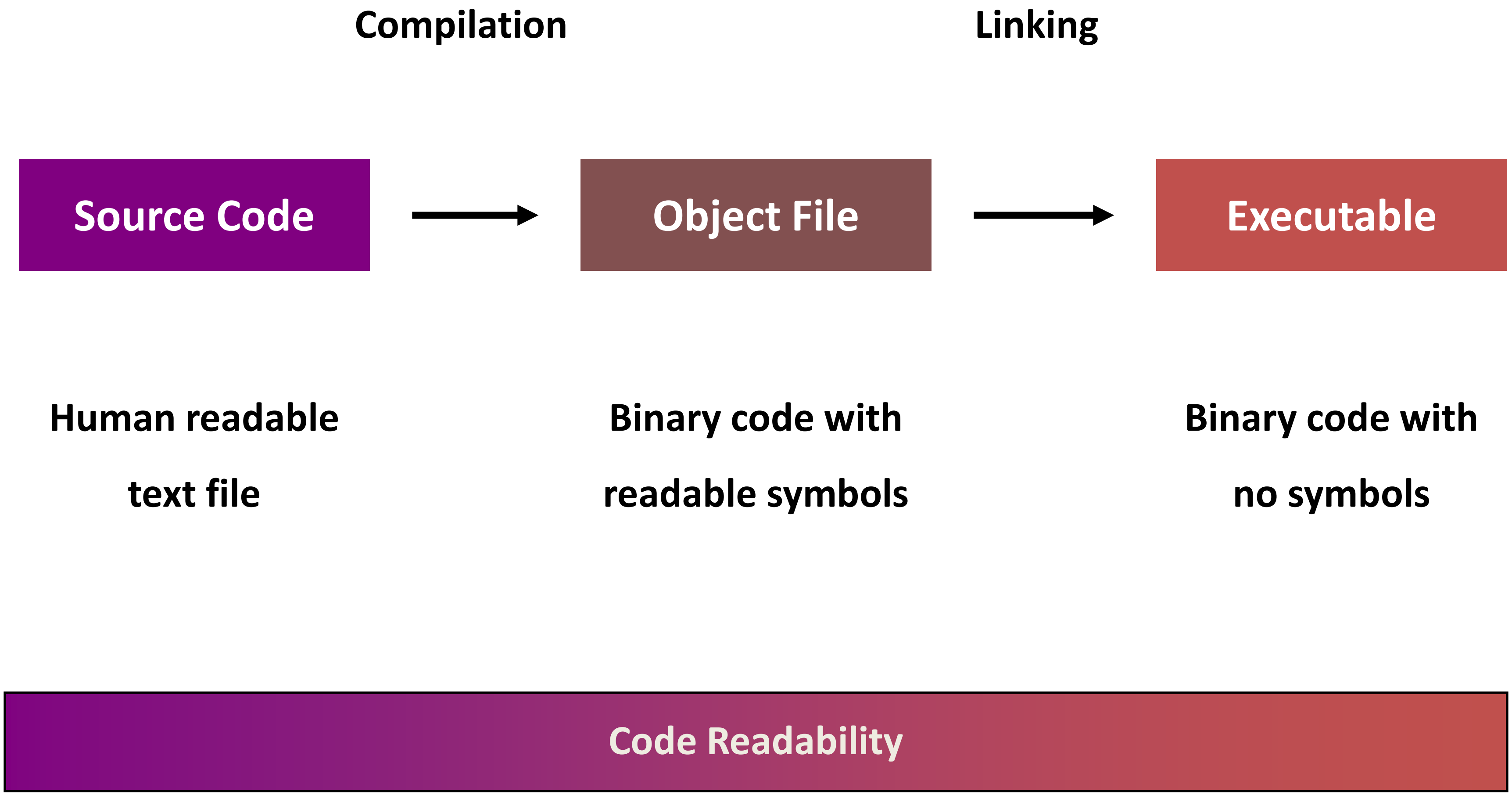
Introduction

- Malware analysis is the art of dissecting malware to understand how it works, how to identify it, and how to defeat or eliminate it
- It goals are
  - to determine exactly what happened, and to ensure the location of all the infected machines and files
  - to determine exactly what a particular suspect binary can do, how to detect it on the network, and how to measure and contain its damage
- Once identify which files require full analysis, it’s time to develop signatures to detect malware infections on the network
- It also Known as Reverse Engineering (RE)

Signatures

- Host-based signatures
  - Identify files or registry keys on a victim computer that indicate an infection
  - Focus on what the malware did to the system, not the malware itself
    - Signature can be
      - Hash Signatures
      - Byte-Signatures
      - Binary Diffing
      - Heuristics
- Network signatures
  - Detect malware by analyzing network traffic
  - More effective when made using malware analysis

Forward Engineering



Compilation

Linking

Source Code

Object File

Executable

Human readable text file

Binary code with readable symbols

Binary code with no symbols

Code Readability

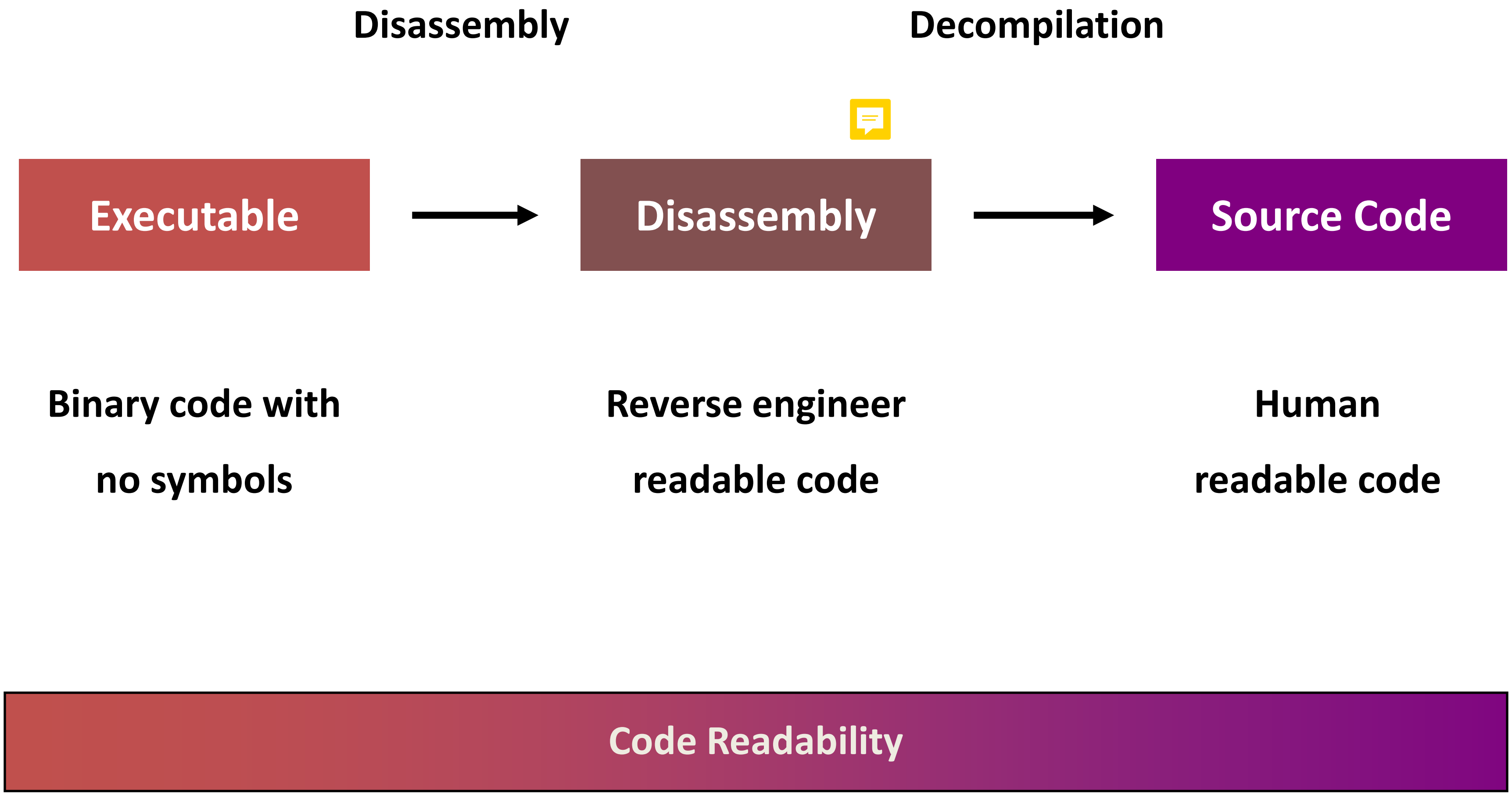
```
int ExecFile(char *FileName)
{
    PyObject* PyFi

    if (!PyFileObj
    {
        return 0;
    }

    if (PyRun_Simp
    {
        Py_DECREF(
        return 1;
    }
    else
    {
        Py_DECREF(
        return 0;
    }
}
```

```
.text:00401250  E8 BB DA 0E 00 89 44 24  04 A1 2C A3 57 00 8B 40  F++.ëD$í,úW.ï@
.text:00401260  10 89 04 24 E8 27 D5 0E  00 8B 15 2C A3 57 00 E9  ë$F'+.ï§,úW.T
.text:00401270  4B FF FF FF 8D B6 00 00  00 00 8D BF 00 00 00 00  K   ì¦....ì+....
.text:00401280  55 89 E5 83 EC 08 C7 04  24 01 00 00 00 FF 15 18  Uësâ8¦$... §
.text:00401290  A3 57 00 E8 B8 FE FF FF  90 8D B4 26 00 00 00 00  úW.F+¦  Éì¦&....
.text:004012A0  55 89 E5 83 EC 08 C7 04  24 02 00 00 00 FF 15 18  Uësâ8¦$... §
.text:004012B0  A3 57 00 E8 98 FE FF FF  90 8D B4 26 00 00 00 00  úW.Fÿ¦  Éì¦&....
.text:004012C0  55 8B 0D 54 A3 57 00 89  E5 5D FF E1 8D 74 26 00  UïTúW.ës] ßìt&.
.text:004012D0  55 8B 0D 34 A3 57 00 89  E5 5D FF E1 90 90 90 90  Uï4úW.ës] ßÉÉÉÉ
.text:004012E0  83 EC 7C B8 70 B5 4E 00  89 44 24 34 B8 74 30 4F  â8|+p¦N.ëD$4+tOO
.text:004012F0  00 89 44 24 38 8D 44 24  60 89 44 24 3C B8 90 13  .ëD$8ìD$`ëD$<+É
.text:00401300  40 00 89 44 24 40 8D 44  24 1C 89 7C 24 74 89 5C  @.ëD$@ìD$ë|$të\
.text:00401310  24 6C 89 74 24 70 89 6C  24 78 89 64 24 44 89 04  $lët$pël$xëd$Dë
.text:00401320  24 E8 3A BE 0E 00 8B BC  24 80 00 00 00 85 FF 0F  $F:+.ï+$Ç...à ¤
.text:00401330  84 8B 00 00 00 C7 04 24  10 20 57 00 8B 94 24 80  äï...¦$ W.ïö$Ç
.text:00401340  00 00 00 8D 44 24 50 89  44 24 04 BE 88 E1 56 00  ...ìD$PëD$+êßV.
.text:00401350  31 DB 89 74 24 50 B9 01  00 00 00 89 54 24 54 89  1¦ët$P¦...ëT$Të
.text:00401360  5C 24 58 89 4C 24 20 E8  D4 59 00 00 89 44 24 04  \$XëL$ F+Y..ëD$
.text:00401370  C7 04 24 10 20 57 00 E8  B4 5A 00 00 85 C0 74 2E  ¦$ W.F¦Z..à+t.
.text:00401380  8B 40 08 BA E8 EC 56 00  89 54 24 50 EB 34 66 90  ï@¦F8V.ëT$Pd4fÉ
.text:00401390  B8 E8 EC 56 00 89 44 24  50 8B 44 24 24 89 04 24  +F8V.ëD$PïD$$ë$
.text:004013A0  B8 FF FF FF FF 89 44 24  20 E8 72 C4 0E 00 B8 E8  +    ëD$ Fr-.+F
.text:004013B0  EC 56 00 89 44 24 50 89  F6 8D BC 27 00 00 00 00  8V.ëD$Pë÷ì+'....
.text:004013C0  31 C0 89 44 24 18 8D 44  24 1C 89 04 24 E8 6E BE  1+ëD$ìD$ë$Fn+
.text:004013D0  0E 00 8B 44 24 18 8B 5C  24 6C 8B 74 24 70 8B 7C  .ïD$ï\$lït$pï|
.text:004013E0  24 74 8B 6C 24 78 83 C4  7C C3 8D B6 00 00 00 00  $tïl$xâ-|+ì¦....
```

Reverse Engineering



Disassembly

Decompilation

Executable → Disassembly → Source Code

Binary code with no symbols

Reverse engineer readable code

Human readable code

Code Readability

```
.text:004013F0 sub_4013F0      proc near               ; CODE XREF: sub_406AB0+6Fp
.text:004013F0                                         ; sub_4601D0+5Dp
.text:004013F0
.text:004013F0 var_1C          = dword ptr -1Ch
.text:004013F0 var_18          = dword ptr -18h
.text:004013F0 arg_0           = dword ptr  4
.text:004013F0
.text:004013F0                 push    edi
.text:004013F1                 push    esi
.text:004013F2                 push    ebx
.text:004013F3                 sub     esp, 10h
.text:004013F6                 mov     edi, [esp+1Ch+arg_0]
.text:004013FA                 test    edi, edi
.text:004013FC                 jz      short loc_40143D
.text:004013FE                 mov     [esp+1Ch+var_1C], offset dword_572010
.text:00401405                 call    sub_406F80
.text:0040140A                 mov     ebx, eax
.text:0040140C                 jmp     short loc_401439
.text:0040140C ; ---------------------------------------------------------------------------
.text:0040140E                 align 10h
.text:00401410
.text:00401410 loc_401410:                             ; CODE XREF: sub_4013F0+4Bj
.text:00401410                 mov     [esp+1Ch+var_18], ebx
.text:00401414                 mov     [esp+1Ch+var_1C], offset dword_572010
.text:0040141B                 call    sub_406E30
.text:00401420                 mov     [esp+1Ch+var_18], ebx
```

Why Malware Analysis?

- To access damage from a violation
- To discover and sort out indicators of compromise that will reveal other machines that have been affected by the same virus or intruders
- To determine the sophistication level of the virus writer
- To identify the vulnerability that was exploited to allow the virus to get there in the first place
- To identify the intruder or insider that is responsible for putting in the virus
- To learn and have fun
- To answer the following Q ….

- Business Question
  - What is the purpose of the malware?
  - How did it get here?
  - Who is targeting us and how good are they?
  - How can I get rid of it?
  - What did they steal?
  - How long has it been here?
  - Does it spread on its own?
  - How can I find it on other machines?
  - How do I prevent this from happening in the future?

- Technical Question
  - Network Indicators?
  - Host-based Indicator?
  - Persistence Mechanism?
  - Date of Compilation?
  - Date of Installation?
  - What language was it written in?
  - Is it packed?
  - Was it designed to thwart analysis?
  - Does it have any rootkit functionality?

Transformer - Malware In Disguise

- Most of malware (especially backdoors) originally given/ renamed themselves to other common names to the OS

- UNIX/ Linux OSes
  - initd, init, inet, cron, network, httpd, httpb

- MS Windows OSes
  - svchost, win, iexplore
  - Prior to Vista & Windows 2008, Task Manager and taskkill.exe cannot kill:
  - *csrss.exe, services.exe, smss.exe, system, system idle process, winlogon.exe*

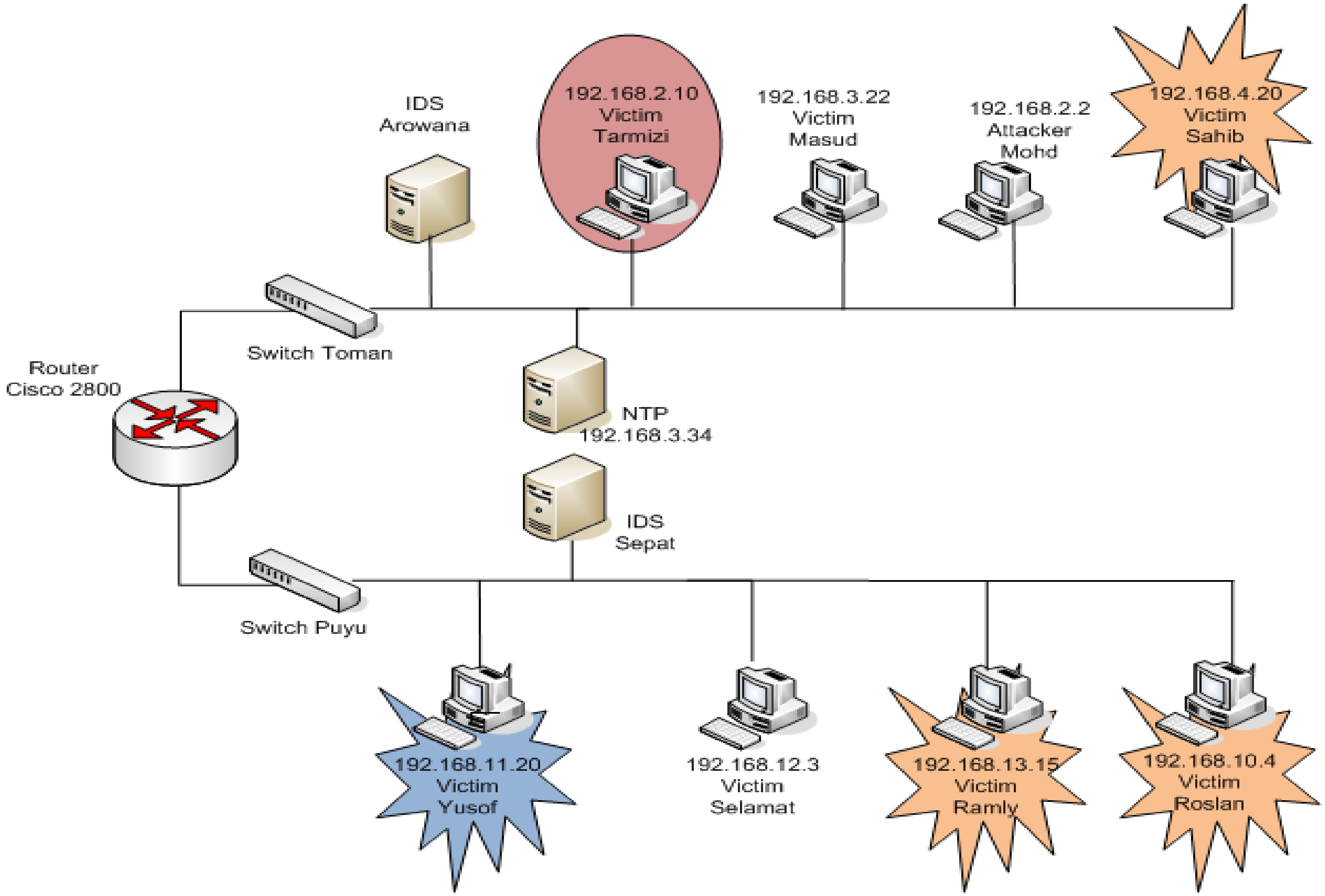
Creating A safe Environment

Creating a Safe Environment

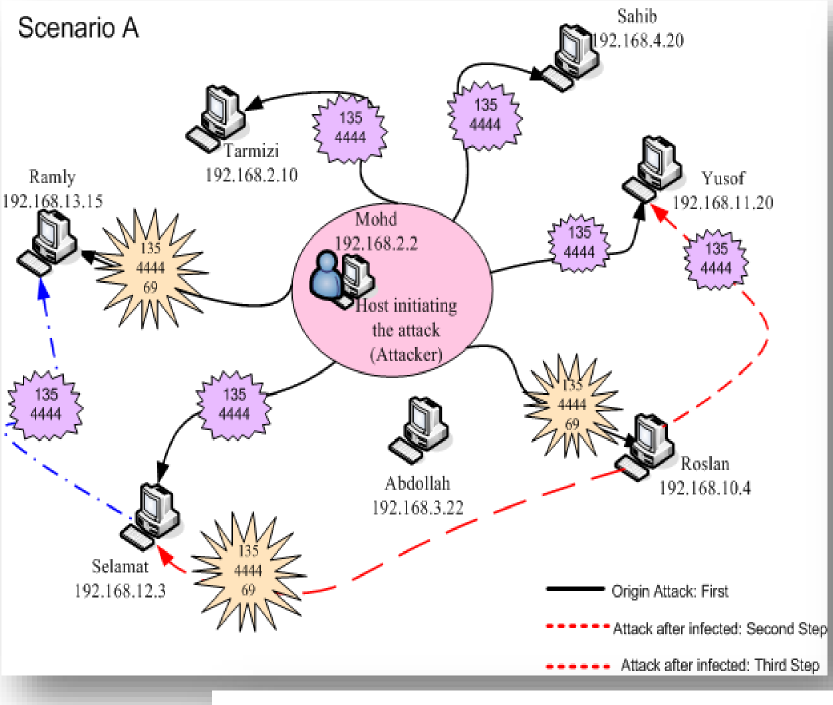
- Do not run malware on the personal computer
- Create an isolated environment
  - Use virtualization machine (Vmware, Virtualbox)
  - Create a network testbed environment (GNS3)
- Perform analysis (Static) on a different OS than the malware target’s OS.
- However !!!! ….some malware has it own defend mechanism.

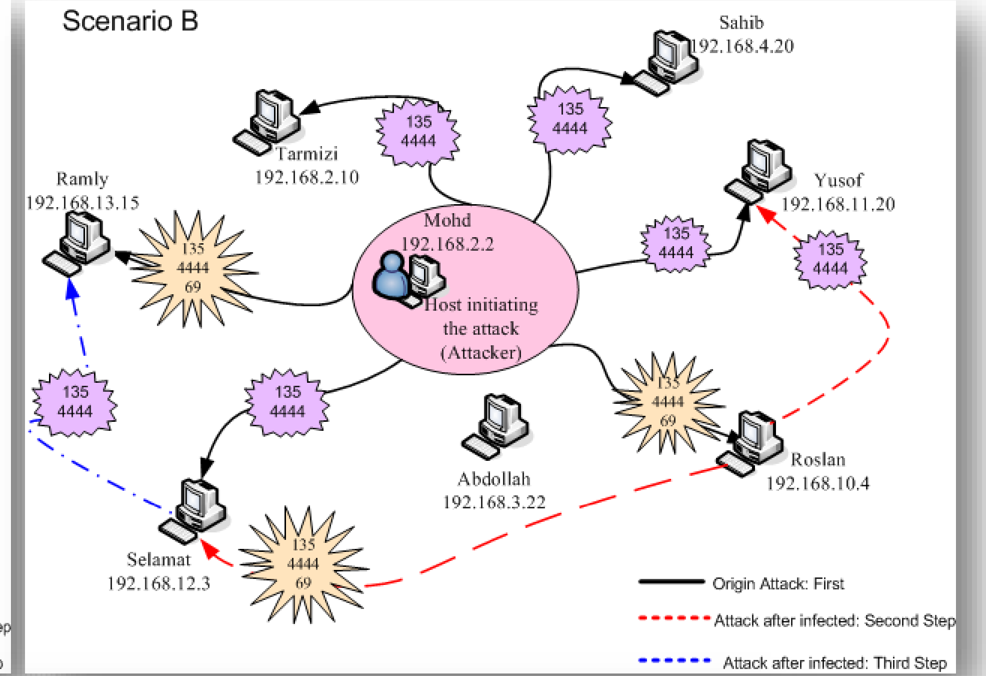
- Malware change it behavior
- Allowing malware to connect to a controlling server might bring the analysis tp areal time battle with an actual human for control of your analysis
- The IP address use in the analysis might become the target for additional attacks
- Experiment become a real threat to user around the experiment.

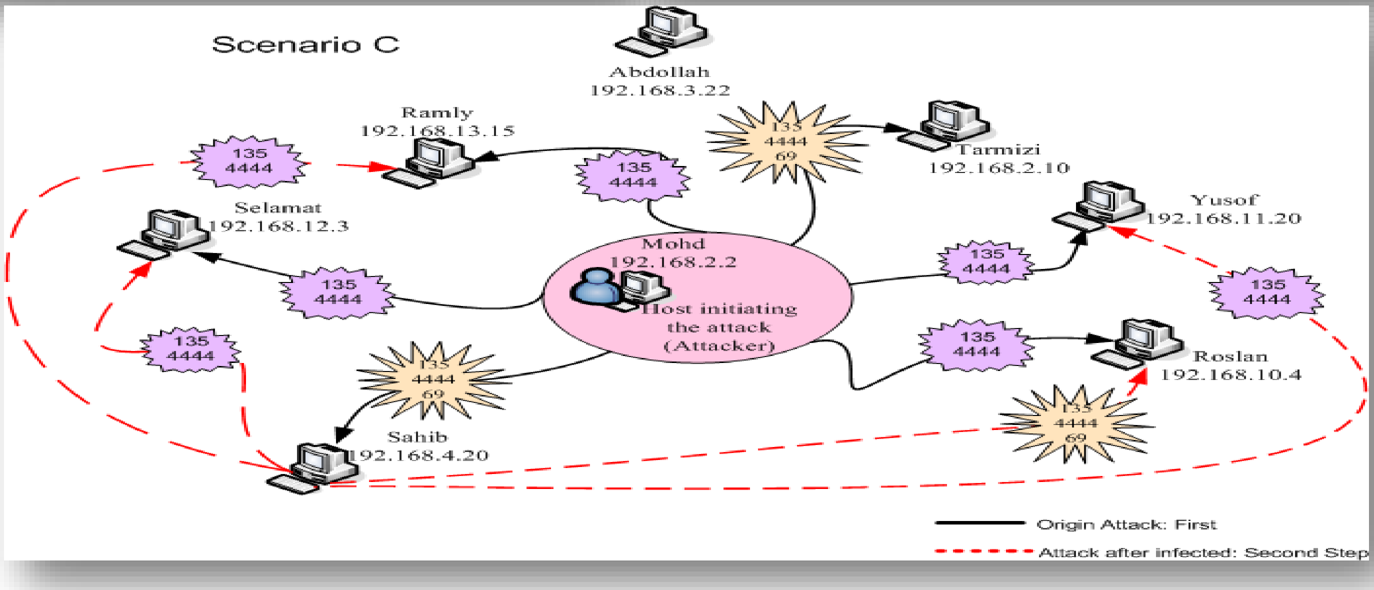
- To overcome the issue
  - Use the host-only networking features of the virtualization platform
  - Use network simulation tools such as GNS3
  - Establish a real services(DNS, Web, FTP and etc…) on the host OS


IDS
Arowana
192.168.2.10
Victim
Tarmizi
192.168.3.22
Victim
Masud
192.168.2.2
Attacker
Mohd
192.168.4.20
Victim
Sahib
Switch Toman
Router
Cisco 2800
NTP
192.168.3.34
IDS
Sepat
Switch Puyu
192.168.11.20
Victim
Yusof
192.168.12.3
Victim
Selamat
192.168.13.15
Victim
Ramly
192.168.10.4
Victim
Roslan

RO2 -Blaster Incident Scenario







Required Skill

- General knowledge about computer architectures
- Assembly language of target processor
- Operating systems
- File formats
- High level programming languages
- Logical thinking, ability to solve puzzles and think outside the box
- Google skills ☺
- Persistence

Static & Dynamic Analysis

Static and Dynamic Analysis

- Static
  - Analysis of file structure and contents
  - Code is not executed
  - Autopsy or Dissection of “Dead” Code
- Dynamic
  - Target of analysis is executing
    - Behaviour monitoring on real system
    - Emulation
- In most real cases you use both static and dynamic analysis

Static Analysis

- Static analysis: looking into the program without actually executing it
- Advantage
  - can reveal how a program would behave under unusual conditions, because we can examine parts of a program that normally do not execute.
- Disadvantage
  - Tedious process and it is impossible to fully predict the behaviour.
- Example of tool use in static analysis
  - Decompilers
  - Disassemblers (e.g. IDA)
  - Hex editors (e.g. HT, Hiew)
  - strings, BinText, etc.

Other Benefit

- No need to run the code
  - You may not want to run the code if it’s malicious
  - You may not have the environment to run the code
  - The code may have checks that prevents it from running (e.g. anti-virtualization)

- Some of the tools give quick answers (e.g. strings)

- Great for browsing and documenting the code (e.g. naming variables and functions)

Static Analysis step

- To make sure the file are not changing during analysis the hash or md5sum function is use
- Always Scan new malware with an up to date virus scanner. If the file is not sensitive the binary can be submitted to www.virustotal.com
- Identify the binary using PEiD that is able to identifies over 600 different packers and compilers
- Using String, Bintext, Hex editor or IDA Pro to look for obvious string.

Dynamic Analysis

- Dynamic analysis: analyzing a program while it executes
- Advantage – it can be fast and accurate.
- Disadvantage – it is “what you see is what you get”.
- Some of the analysis process:
  - Process monitoring
  - Registry monitoring
  - File monitoring
  - Network sniffing using Wireshark

- Tools:
  - File/process/registry monitors (e.g. Procmon, Process explorer)
  - Network monitors (e.g. Wireshark, Fiddler)
  - Debuggers (e.g. OllyDbg, WinDbg)

Other Benefit

- May give quick answers to the question “What does the program do?”
  - e.g. Procmon, Wireshark
- Debugging the program helps you understand it
  - You can execute it instructions at a time and see how the values change
- Needed with packed/protected/encrypted code

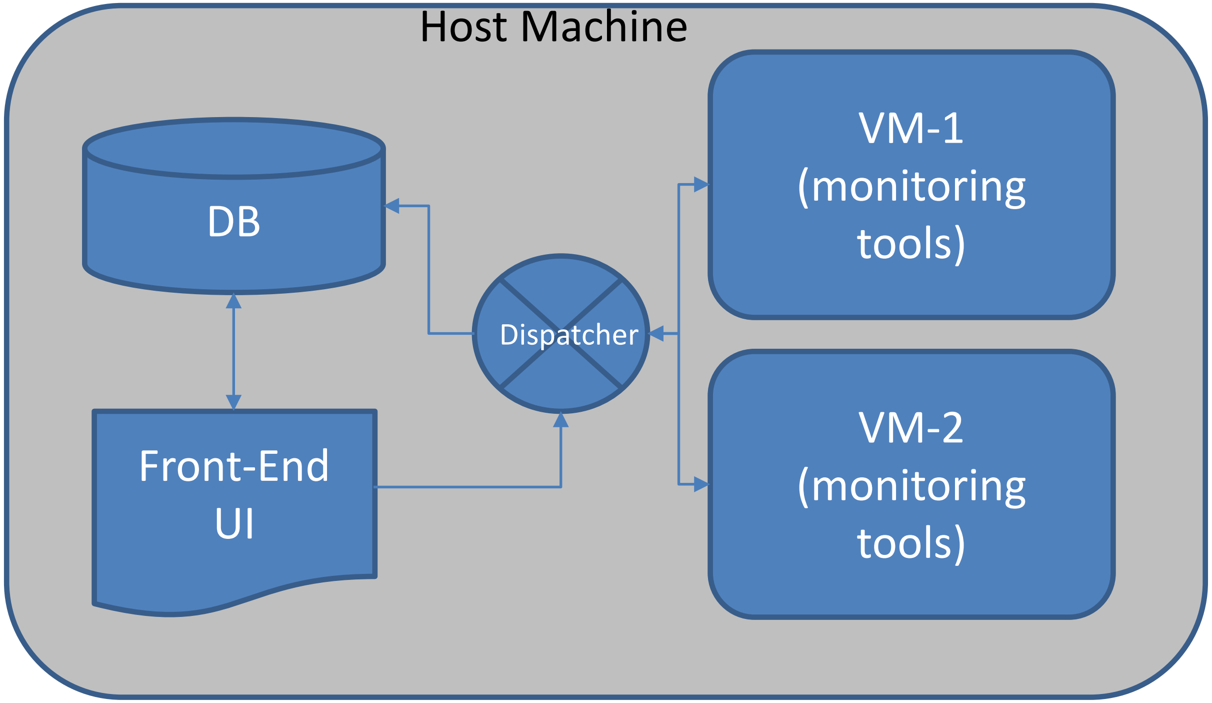
Automated Analysis

Automated Analysis

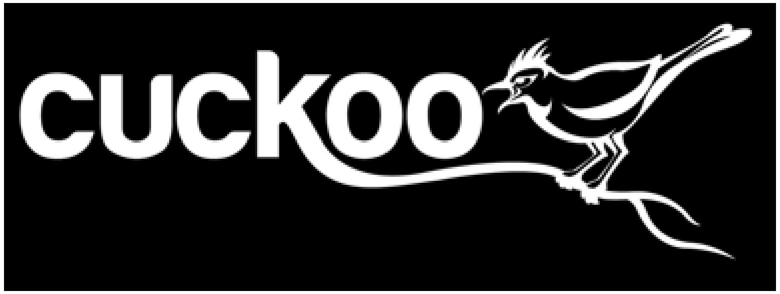
- Utilize various free resources to perform quick analysis of malware
- Online Scanner
  - Virus Total - http://www.virustotal.com
  - Comodo - http://camas.comodo.com/
  - Jotti - http://virusscan.jotti.org/
- Sandboxing
  - Cuckoo Sanbox
  - Anubis
  - Copperdroid (android)
  - Malwr(online Cuckoo)
  - FireEye

Sandbox

- security mechanism for running untrusted programs in a safe environment without fear of harming “real” systems. Sandboxes comprise virtualized environments that often simulate network services in some fashion to ensure that the software or malware being tested will function normally
- A controlled environment where you can run/execute programs to test and analyze its behavior.
- Tit is a controlled environment where you can run/execute programs and test its behavior.
- You can have a sandbox on a Physical Machine or you can have it on Virtual Machine for easy manageability.
- Most of Sandboxes are on Virtual Machine so it can be easily automated.


Host Machine
DB
Dispatcher
VM-1 (monitoring tools)
VM-2 (monitoring tools)
Front-End UI

Type of sandbox



MALWARE SAMPLE SOURCE

Where can I get Malware sample

- Hybrid Analysis (https://www.hybrid-analysis.com/)
- VirusBay (https://beta.virusbay.io/)
- Contagio malware dump (http://contagiodump.blogspot.com/)
- Malwr (?)
- VirusShare (https://virusshare.com/)
- theZoo (https://thezoo.morirt.com/)
- https://zeltser.com/malware-sample-sources/

Armored Malware

Armored Malware

Malware that have the ability to thwart malware analysis strategies.

- Encryption
- Compression
- Obfuscation
- Anti-Patching
- CRC Checking
- Anti-Tracing
  - Detection code
  - Crashes OS if they are found in memory
- Anti-Unpacking
- Anti-Vmware
- Polymorphic/ Self Mutating
- Restrictive Dates
- Password Protected
- Configuration Files

Packers

- Packers are used on executables for two main reasons:
  - to shrink programs
  - To thwart detection or analysis.
- Even though there are a wide variety of packers, they all follow a similar pattern:
  - transform an executable to create a new executable that stores the transformed executable as data
  - contains an unpacking stub that is called by the OS.

Summary

- Malware analysis help security personnel to understand how malware behave and help to developed a mitigation process to fight the malware.
- Also known as Reverse Engineering
- Must be done on safe and isolated environment
- There are 2 strategies to analyze the malware which are Static and Dynamic.
- Automated Analysis can be done using a sandbox
- Armored Malware have the ability to defense themselves using several techniques such as packers.