MALWARE DEVELOPMENT BASICS

SJU ACM STUDENT CHAPTER

SJU ACM

Student Chapter

SIGN IN FORM:

DISCLAIMER

BEFORE WE BEGIN

- THIS LAB IS FOR EDUCATIONAL PURPOSES ONLY
- YOU’RE GOING TO LEARN HOW BASIC MALWARE IS WRITTEN, AS WELL AS CONCEPTS EMPLOYED.
- DO NOT ATTEMPT TO USE THE KNOWLEDGE YOU LEARN HERE TO PERFORM ANY MALICIOUS ACTIONS

LOGGING IN TO THE LAB MACHINES

1. REBOOT THE MACHINE
2. WHEN GIVEN THE OPTION BETWEEN “LAB/CLASSROOM” AND “CLOSED NETWORK” CHOOSE “CLOSED NETWORK”
3. LOGIN TO THE MACHINE:
   A. USERNAME: student
   B. PASSWORD: Security2021
4. OPEN VIRTUALBOX, THERE SHOULD BE A MACHINE CALLED “SJU ACM MAL DEV”
5. DO NOT START THE MACHINE YET
6. Passw0rd!

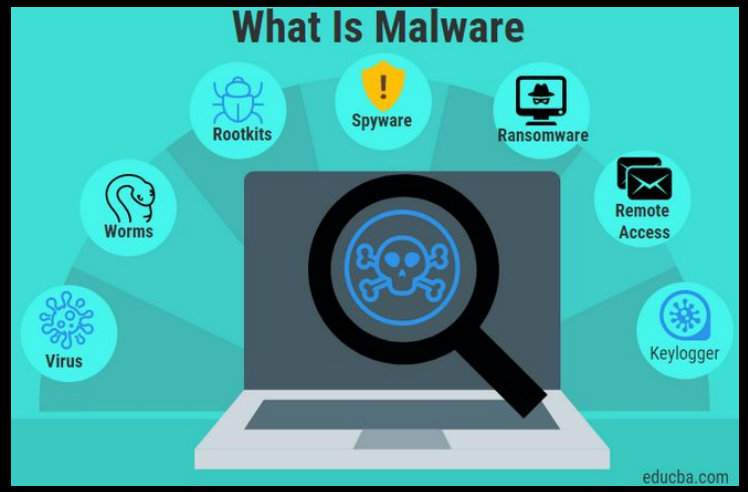
THE BASICS

WHAT IS MALWARE?

- CISCO DEFINES MALWARE AS:
  - INTRUSIVE SOFTWARE THAT IS DESIGNED TO DAMAGE AND DESTROY COMPUTERS, SYSTEMS AND NETWORKS.
- MALWARE = MALICIOUS SOFTWARE
- SOMETIMES THE GOAL IS TO DAMAGE SYSTEMS, BUT VERY OFTEN THE GOAL IS ACTUALLY TO GAIN ACCESS TO A SYSTEM OR NETWORK.

TYPES OF MALWARE

- DROPPERS
  - THE MOST BASIC FORM OF MALWARE. USED SOLELY TO “DROP” ANOTHER PIECE OF SOFTWARE/MALWARE ONTO THE TARGET MACHINE
- TROJANS
  - MALWARE THAT DISGUISES ITSELF AS A LEGIT PROGRAM
- SPYWARE
  - COLLECTS INFO ABOUT THE USER WITHOUT THEIR KNOWLEDGE
- RANSOMWARE
  - HOLDS FILES OR EVEN AN ENTIRE SYSTEM HOSTAGE UNTIL A RANSOM IS PAID



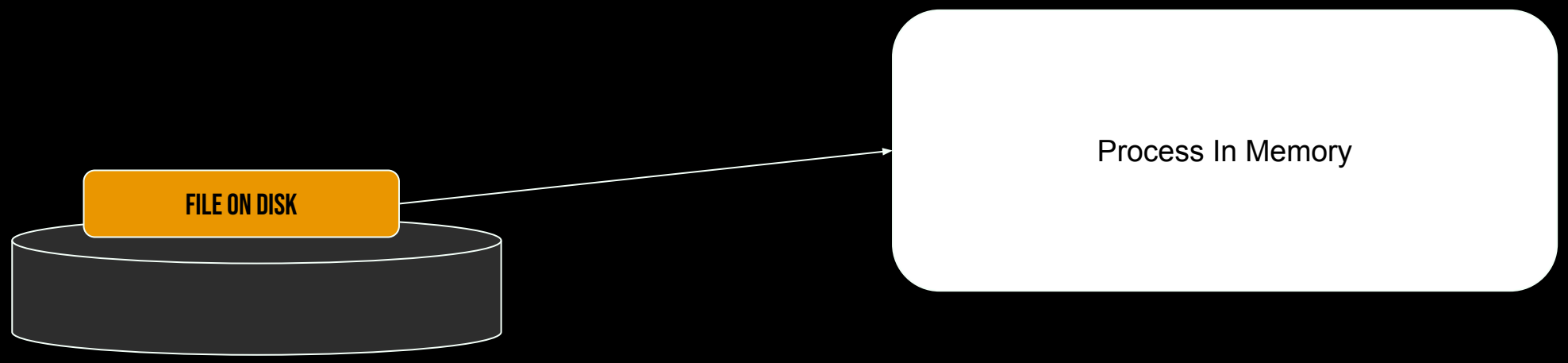
WHY LEARN TO WRITE MALWARE?

- HOW COULD YOU POSSIBLY HAVE AN ETHICAL REASON FOR LEARNING TO WRITE MALWARE?
- PENTESTING / RED TEAMING
  - MALWARE EXTENDS BEYOND WHAT YOU MAY THINK. PAYLOADS AND IMPLANTS THAT ARE COMMONLY USED DURING PENTESTS AND RED TEAM ENGAGEMENTS ARE ALSO COMMONLY CONSIDERED FORMS OF MALWARE
- BETTER UNDERSTANDING FOR DEFENSE
  - ITS VERY HARD TO DEFEND AGAINST SOMETHING IF YOU DON'T UNDERSTAND HOW IT WORKS

PE FILES

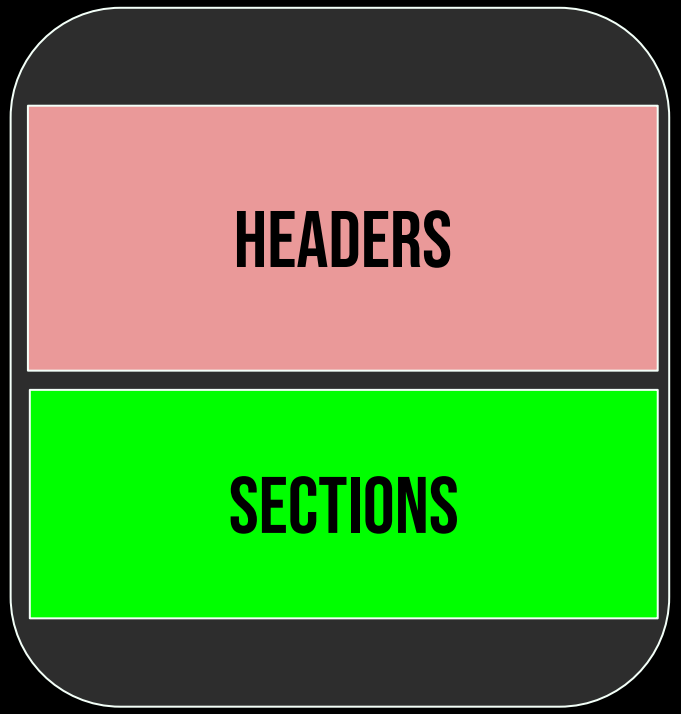
WHAT IS A PE?

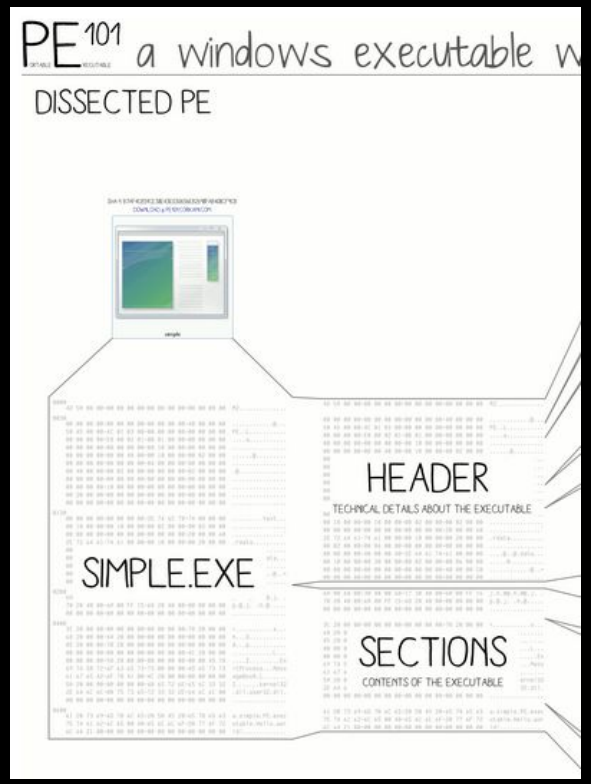
- PORTABLE EXECUTABLE
- OS LOADER CAN READ THIS FILE FORMAT AND LOAD IT INTO MEMORY AS A PROCESS



PE FORMAT (CONT.)

- COMPLICATED
- ESSENTIALLY, A PE IS A “BOOK” THAT CONTAINS “DATA” AND “METADATA”

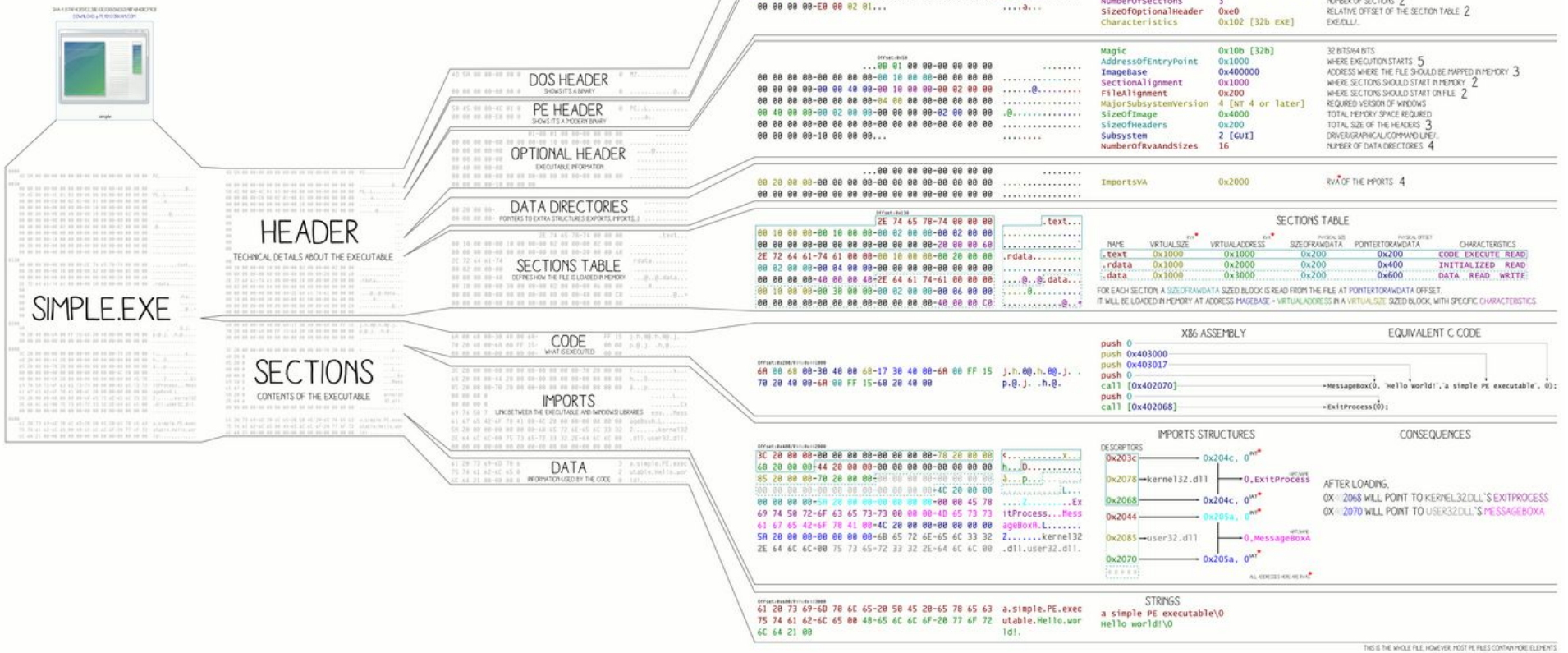




PE 101 a windows executable walkthrough

ANGE ALBERTINI
CORKAMI.COM

DISSECTED PE



LOADING PROCESS

1 HEADERS

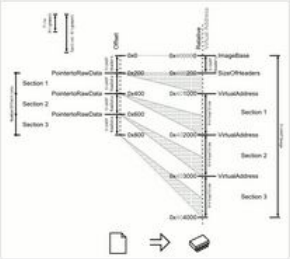
THE DOS HEADER IS PARSED
THE PE HEADER IS PARSED
ITS OFFSET IS DOS HEADER'S E_LFANEW
THE OPTIONAL HEADER IS PARSED
IT FOLLOWS THE PE HEADER

2 SECTIONS TABLE

SECTIONS TABLE IS PARSED
IT IS LOCATED AT OFFSET (OPTIONALHEADER + SIZEOFOPTIONALHEADER)
IT CONTAINS NUMBEROFSECTIONS ELEMENTS
IT IS CHECKED FOR VALIDITY WITH ALIGNMENTS:
FILEALIGNMENTS AND SECTIONALIGNMENTS

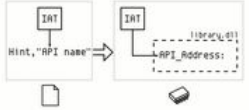
3 MAPPING

THE FILE IS MAPPED IN MEMORY ACCORDING TO:
THE IMAGEBASE
THE SIZEOFHEADERS
THE SECTIONS TABLE



4 IMPORTS

DATADIRECTORIES ARE PARSED
THEY FOLLOW THE OPTIONALHEADER
THEIR NUMBER IS NUMOFRVAANDSIZES
IMPORTS ARE ALWAYS #2
IMPORTS ARE PARSED
EACH DESCRIPTOR SPECIFIES A DLL/NAME
THIS DLL IS LOADED IN MEMORY
IAT AND INT ARE PARSED SIMULTANEOUSLY
FOR EACH API IN INT
ITS ADDRESS IS WRITTEN IN THE IAT ENTRY



5 EXECUTION

CODE IS CALLED AT THE ENTRYPOINT
THE CALLS OF THE CODE GO VIA THE IAT TO THE APIS

NOTES

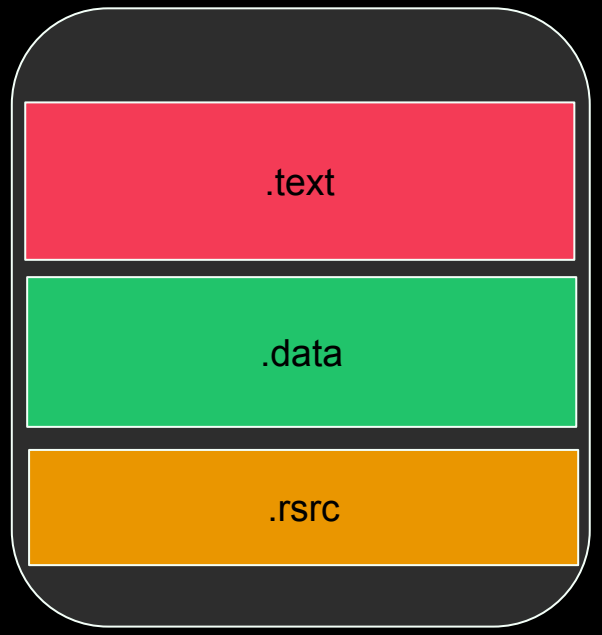
MZ HEADER AKA DOS_HEADER
STARTS WITH 'MZ' (INITIALS OF MARK ZBIKOWSKI MS-DOS DEVELOPER)
PE HEADER AKA IMAGE_FILE_HEADERS / COFF FILE HEADER
STARTS WITH 'PE' (PORTABLE EXECUTABLE)
OPTIONAL HEADER AKA IMAGE_OPTIONAL_HEADER
OPTIONAL ONLY FOR NON-STANDARD PES BUT REQUIRED FOR EXECUTABLES
RVA RELATIVE VIRTUAL ADDRESS
ADDRESS RELATIVE TO IMAGEBASE (AT IMAGEBASE, RVA = 0)
ALMOST ALL ADDRESSES OF THE HEADERS ARE RVAS
IN CODE, ADDRESSES ARE NOT RELATIVE.
INT IMPORT NAME TABLE
NULL-TERMINATED LIST OF POINTERS TO HINT /NAME STRUCTURES
IAT IMPORT ADDRESS TABLE
NULL-TERMINATED LIST OF POINTERS
ON FILE IT IS A COPY OF THE INT
AFTER LOADING IT POINTS TO THE IMPORTED APIS
HINT
INDEX IN THE EXPORTS TABLE OF A DLL TO BE IMPORTED
NOT REQUIRED BUT PROVIDES A SPEED-UP BY REDUCING LOOK-UP

WHAT IS A DROPPER?

- “DROPS” SOME SORT OF PAYLOAD ONTO THE TARGET MACHINE
- SHELLCODE

PAYLOAD STORAGE

- WHERE TO STORE PAYLOADS?
- 3 IMPORTANT SECTIONS
- EACH HAS ITS OWN BENEFITS AND DRAWBACKS
  - .TEXT = WITHIN A FUNCTION
  - .DATA = WITHIN A GLOBAL VARIABLE
  - .RSRC = AS A SEPARATE FILE STORED WITHIN THE PE

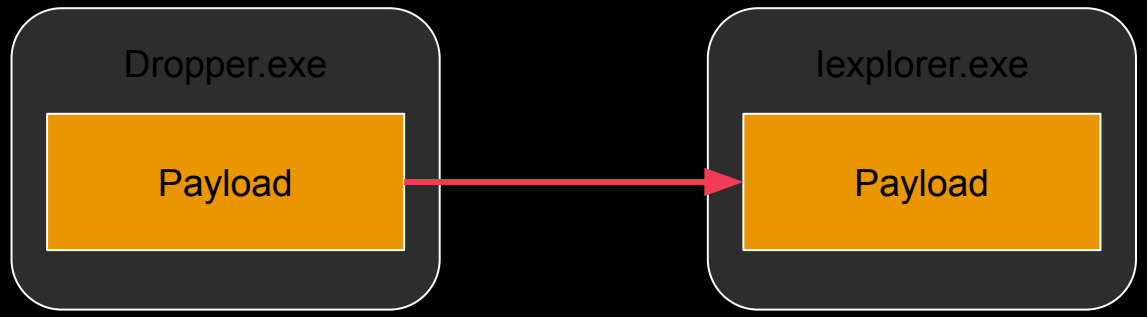


FUNCTION CALL OBFUSCATION

- CALLING EXTERNAL FUNCTIONS
- DETECTION BASED ON IMPORTED DLLS AND FUNCTIONS
- GETMODULEHANDLE AND GETPROCADDRESS
- EX:
  - HANDLE = GETMODULEHANDLE("SOUND.DLL")
  - GETPROCADDRESS(HANDLE, "PLAYSOUND")

CODE INJECTION

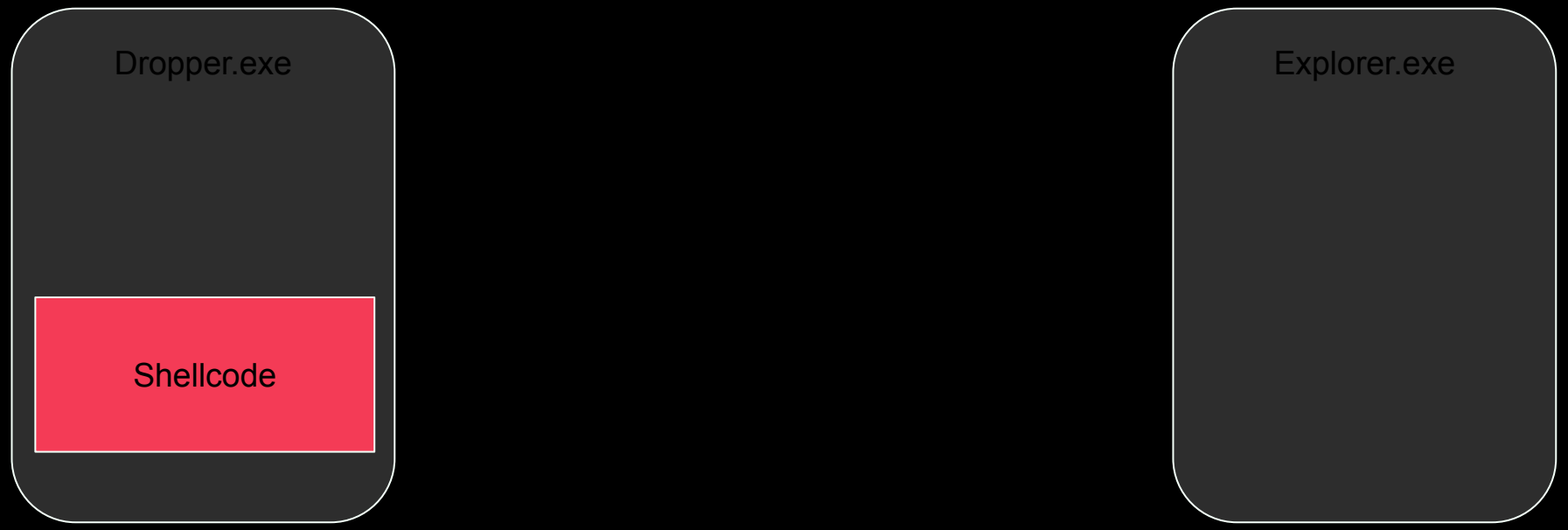
- A METHOD OF TRANSFERRING YOUR PAYLOAD FROM ONE PROCESS TO ANOTHER
- ESCAPE FROM A SHORT LIVE PROCESS
- ESTABLISH A BACKUP C2 CHANNEL (TOON "TWO IS ONE, ONE IS NONE")



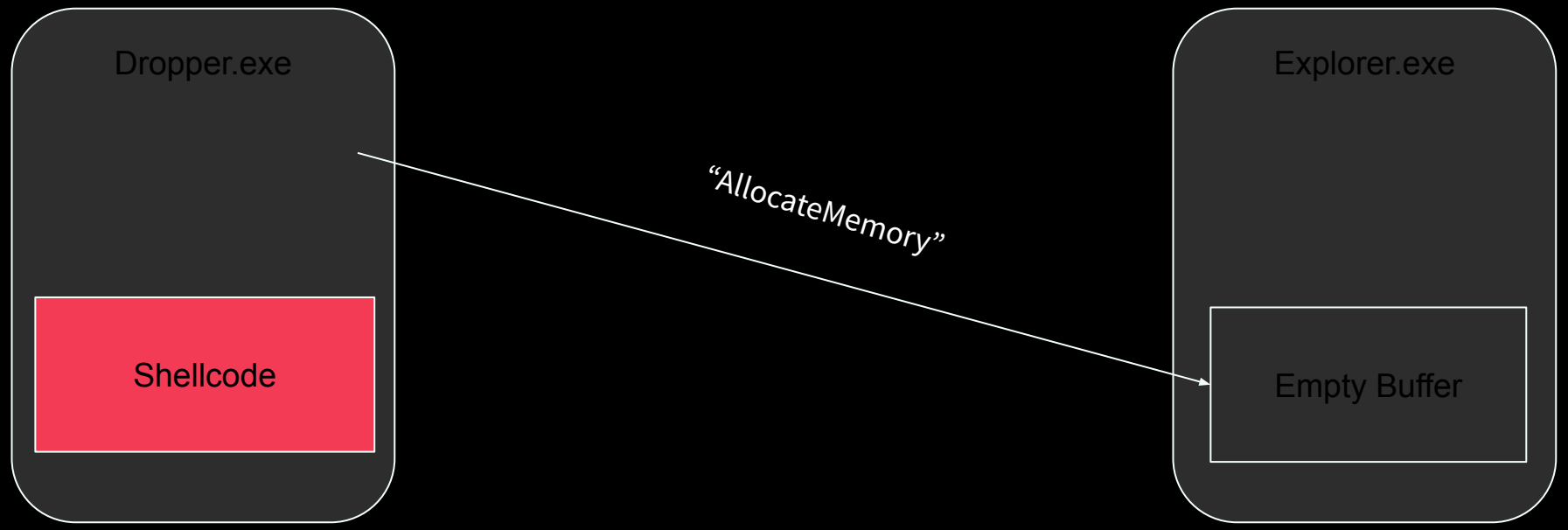
CLASSIC METHOD:

- SHELLCODE INJECTION
  - PHASE 1: COPY SHELLCODE TO TARGET PROCESS (YOU MUST HAVE CORRECT ACCESS)
  - PHASE2: MAKE THE TARGET PROCESS EXECUTE THE SHELLCODE

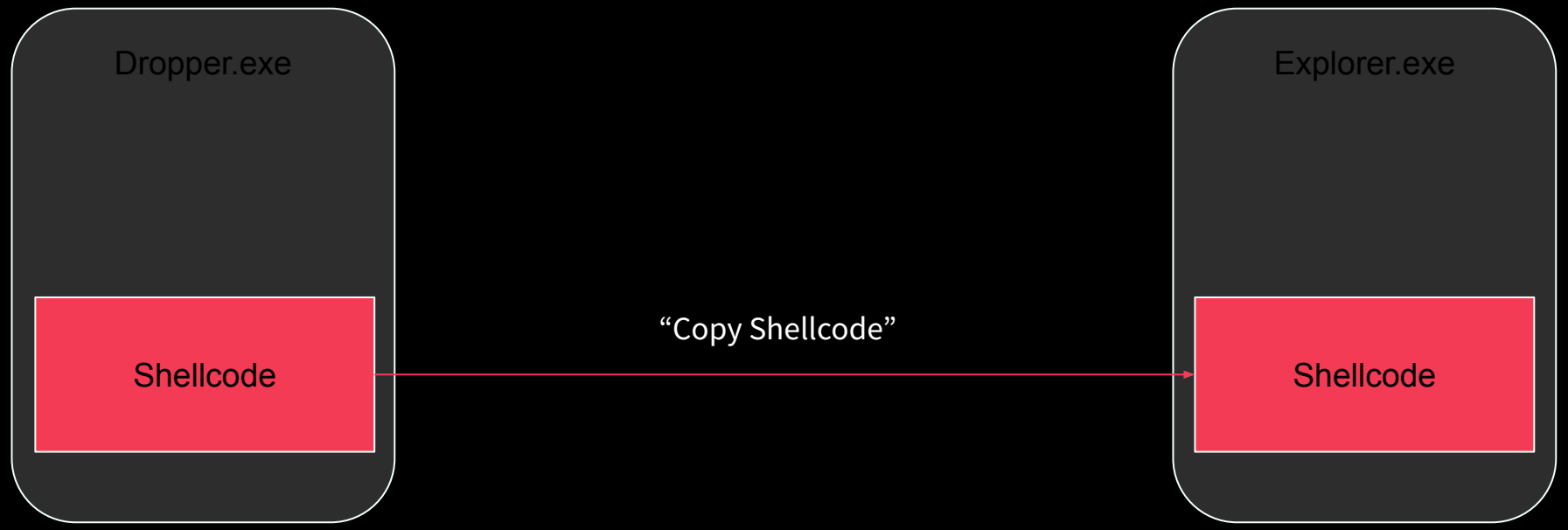
CODE INJECTION (CONT.)



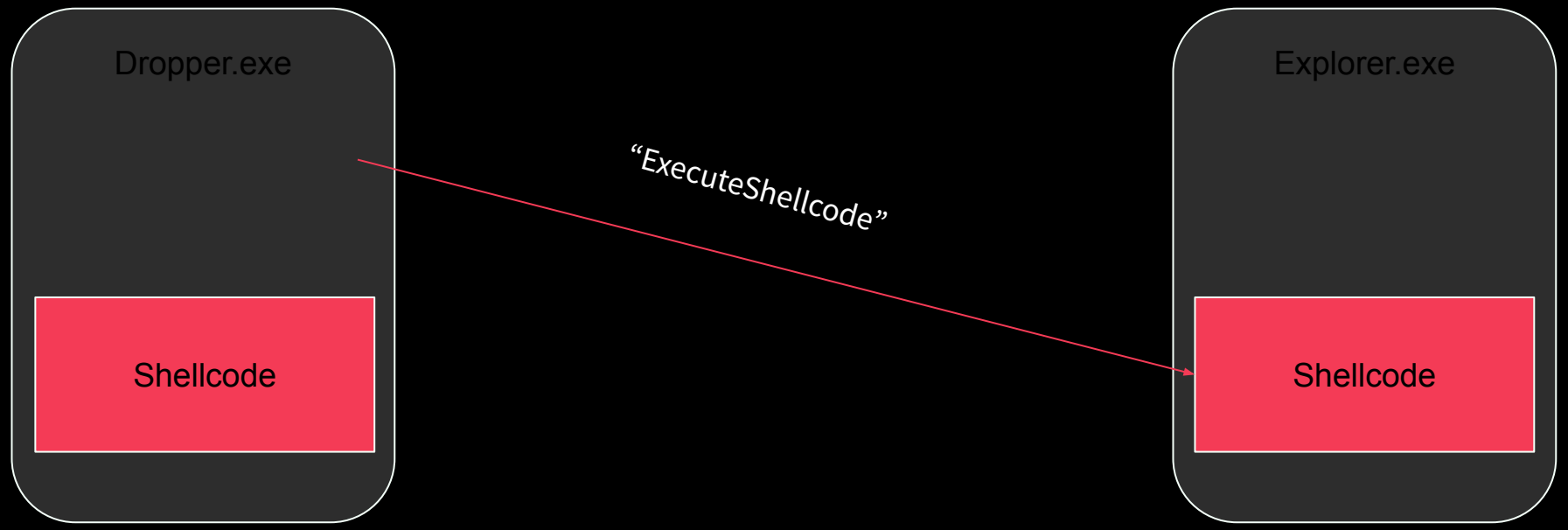
CODE INJECTION (CONT.)



CODE INJECTION (CONT.)



CODE INJECTION (CONT.)



CODE INJECTION (CONT.)

- MOST POPULAR METHOD:
  - VIRTUALALLOCEX
  - WRITEPROCESSMEMORY
  - CREATEREMOTETHREAD

THANK YOU!