Intro to Malware Sample (Simda Trojan)

Learning Objectives

- Analysis of Simda
- File identification
- Custom packer detection using PEStudio
- Identifying abnormal function epilogue
- Using Ghidra and xdbg to analyze abnormal epilogues
- Unpacking and dumping embedded code
- Alternative to VirtualAlloc method

Normal Function

- Each function maintains the frame
- A dedicated register EBP is used to keep the frame pointer
- Each function uses prologue code (blue), and epilogue (yellow) to maintain the frame

```
my_function:
    push ebp          ; save original EBP value on stack
    mov ebp, esp      ; new EBP = ESP
    ....              ; function body
    pop ebp           ; restore original EBP value
    ret
```

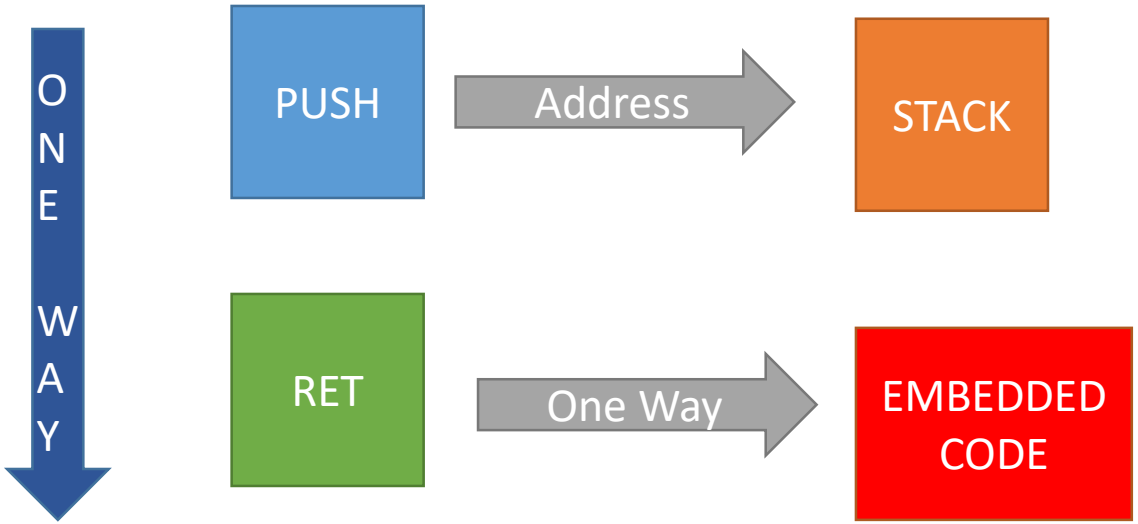
Abnormal Function Epilogue

Unpacking code is often “one-way”, look for code with abnormal transfer control

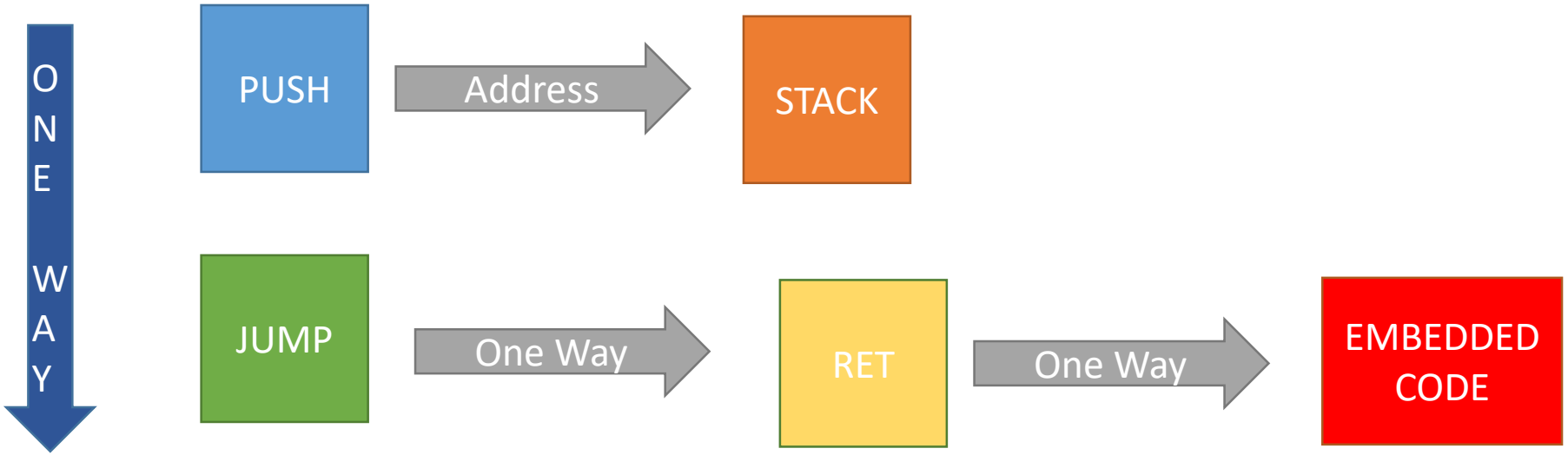
Lack of standard epilogue, JMP instead of RET, PUSH-RET and other deviations are good indicators

Often occur at the end of a function, don’t get caught up in all the details!

FAKE RETURNS



UNEXPECTED JUMPS

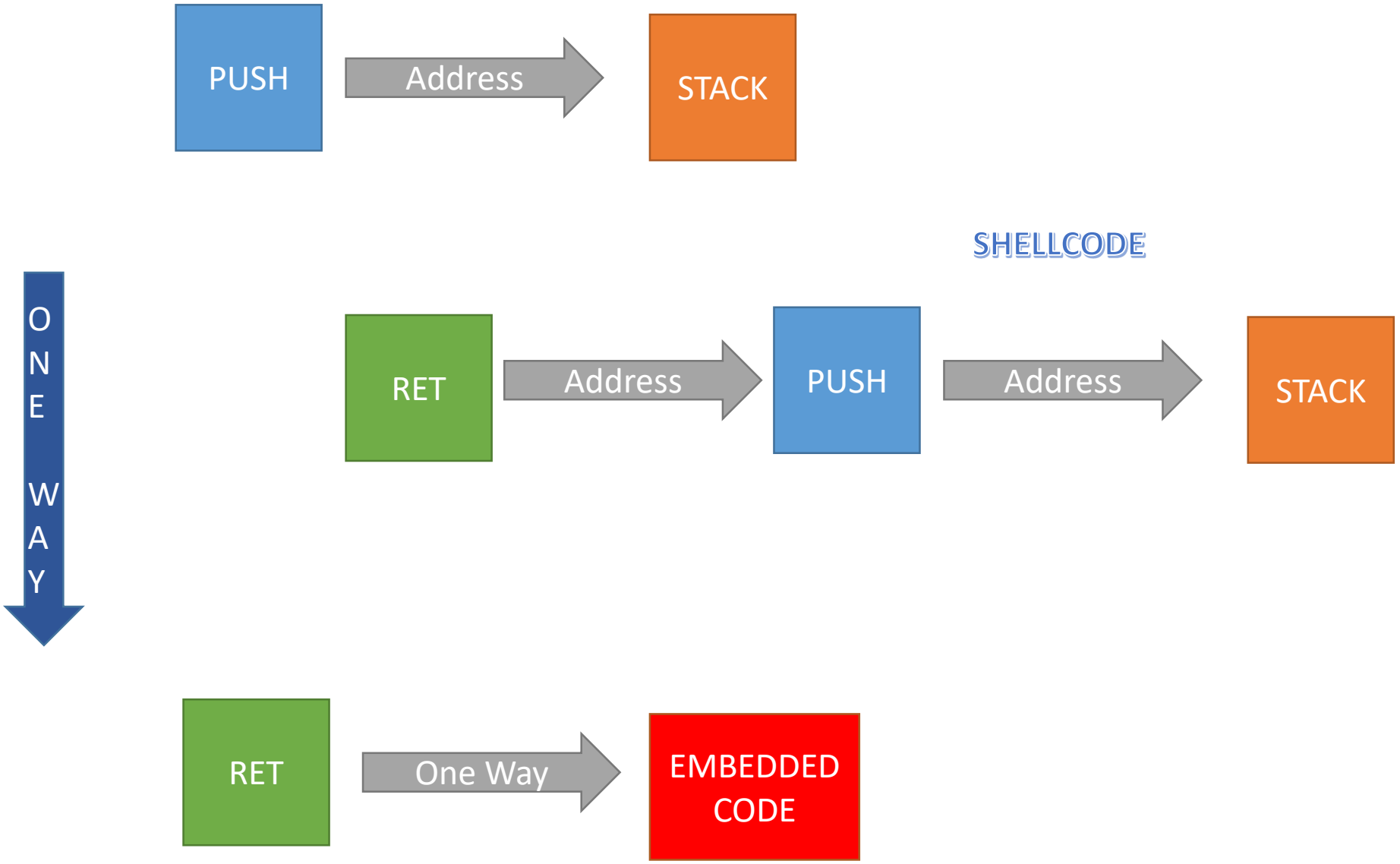


What are shellcodes?

- Historically shellcodes are machine code that spawns a command shell (eg, cmd or bash)
- Injected into vulnerable programs
- Used in the above way = exploits
- In Malware, shellcodes can do anything, eg, unpacking malicious instructions, or inserting fake rets or unexpected jumps

How to write shellcodes:

https://www.sentinelone.com/blog/malicious-input-how-hackers-use-shellcode/


PUSH
Address
STACK
SHELLCODE
ONE WAY
RET
Address
PUSH
Address
STACK
RET
One Way
EMBEDDED CODE

Thank you