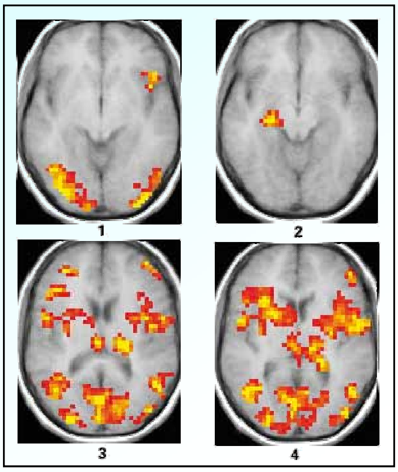


Applying

User-mode Memory Scanning

On

Windows NT based systems

VB 2008

Eric Uday Kumar
Authentium Inc.

Overview

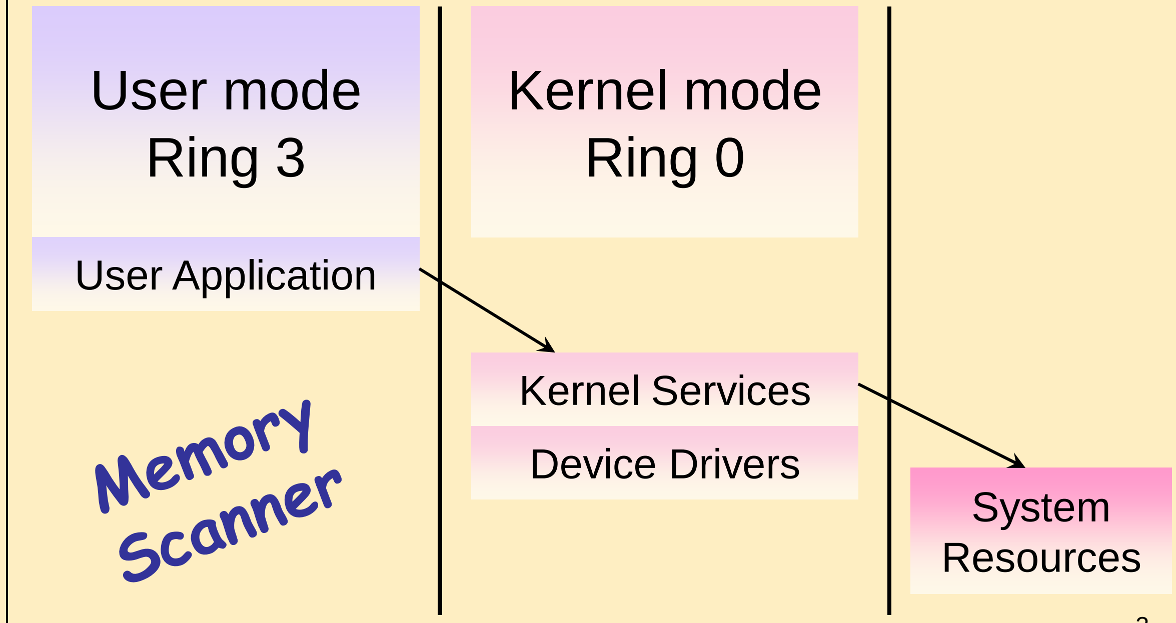
Introducing user-mode memory scan on Windows

Enumerating objects in memory

Demos – real world malware samples

Pros & cons of user-mode memory scanning

Windows NT architecture



User mode
Ring 3

User Application

Kernel mode
Ring 0

Kernel Services

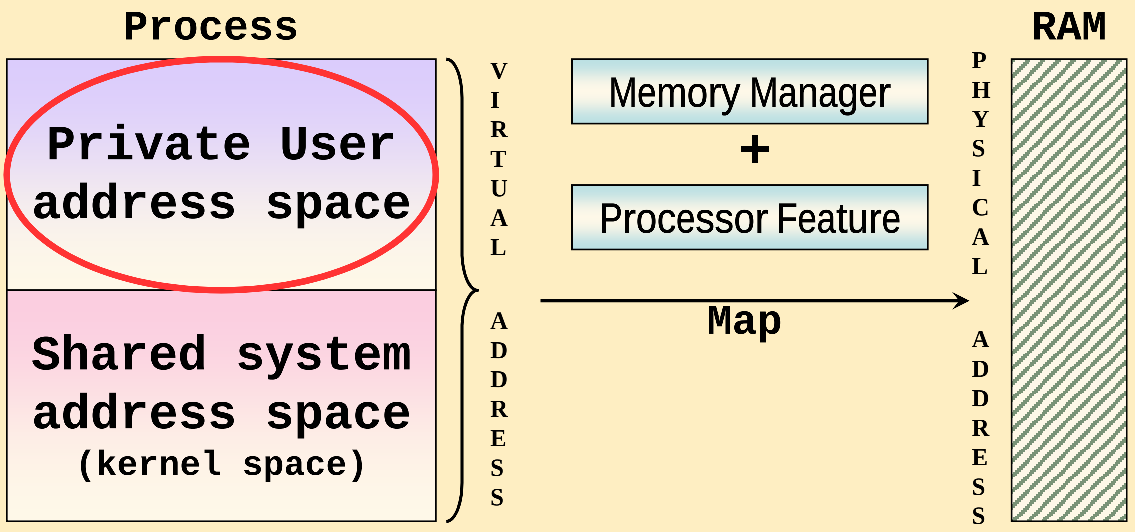
Device Drivers

System
Resources

Memory
Scanner

Virtual Memory

Logical view of Physical Memory



Process

Private User address space

Shared system address space (kernel space)

VIRTUAL ADDRESS

Memory Manager

+

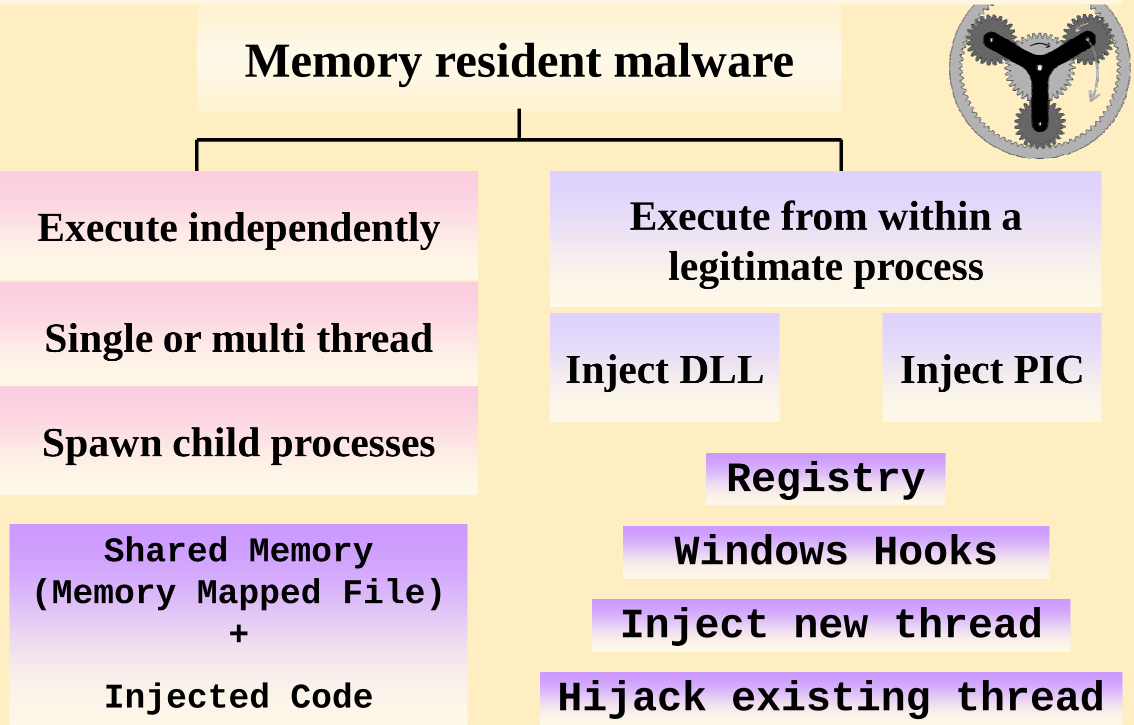
Processor Feature

Map

PHYSICAL ADDRESS

RAM

Everything revolves around a PROCESS



Memory resident malware

Execute independently

Single or multi thread

Spawn child processes

```
Shared Memory
(Memory Mapped File)
+
Injected Code
```

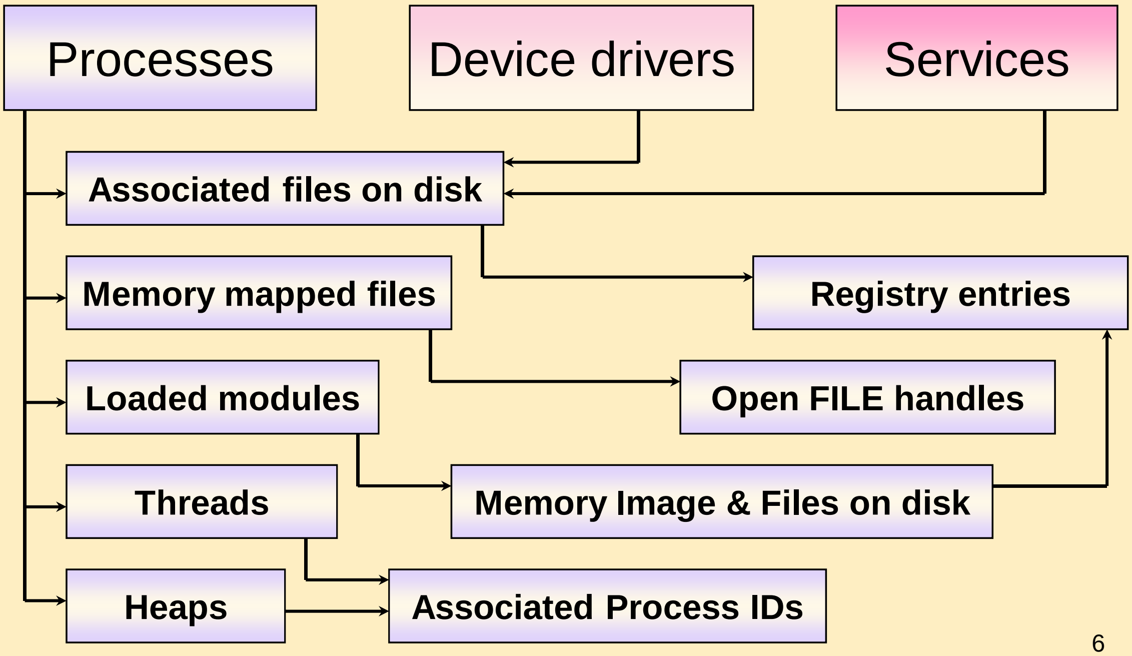
Execute from within a legitimate process

Inject DLL

Inject PIC

```
Registry
Windows Hooks
Inject new thread
Hijack existing thread
```

Objects of interest



Processes

Device drivers

Services

Associated files on disk

Memory mapped files

Registry entries

Loaded modules

Open FILE handles

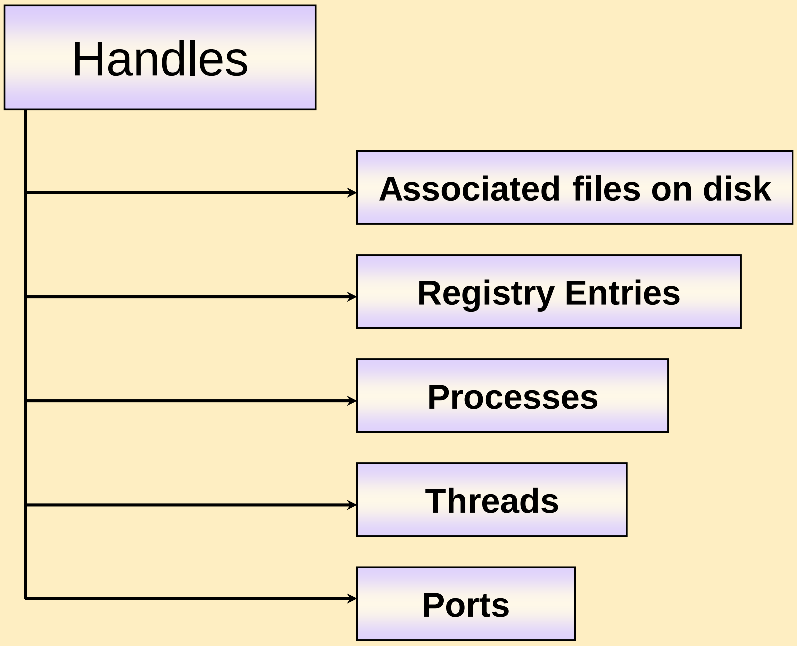
Threads

Memory Image & Files on disk

Heaps

Associated Process IDs

Objects of interest



Handles

Associated files on disk

Registry Entries

Processes

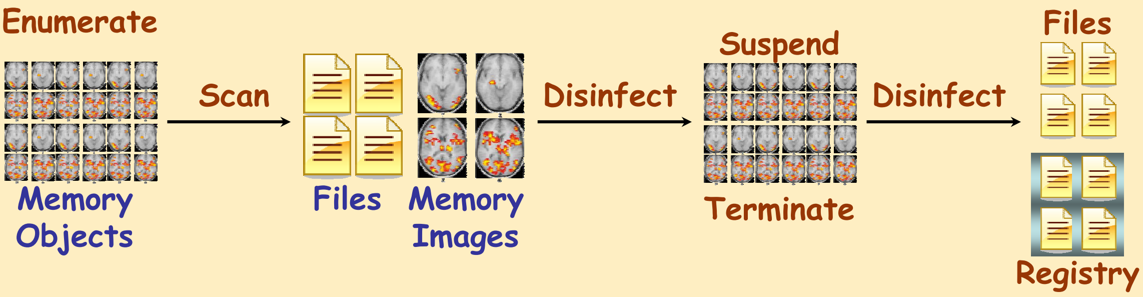
Threads

Ports

An approach – the more the merrier

Enumerate as many objects in memory as possible

Redundant info can reveal hidden components



Enumerate

Memory Objects

Scan

Files

Memory Images

Disinfect

Suspend

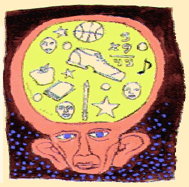
Terminate

Disinfect

Files

Registry

Enumerating Objects in Memory



KERNEL32.DLL

PSAPI.DLL

PDH.DLL

ADVAPI.DLL

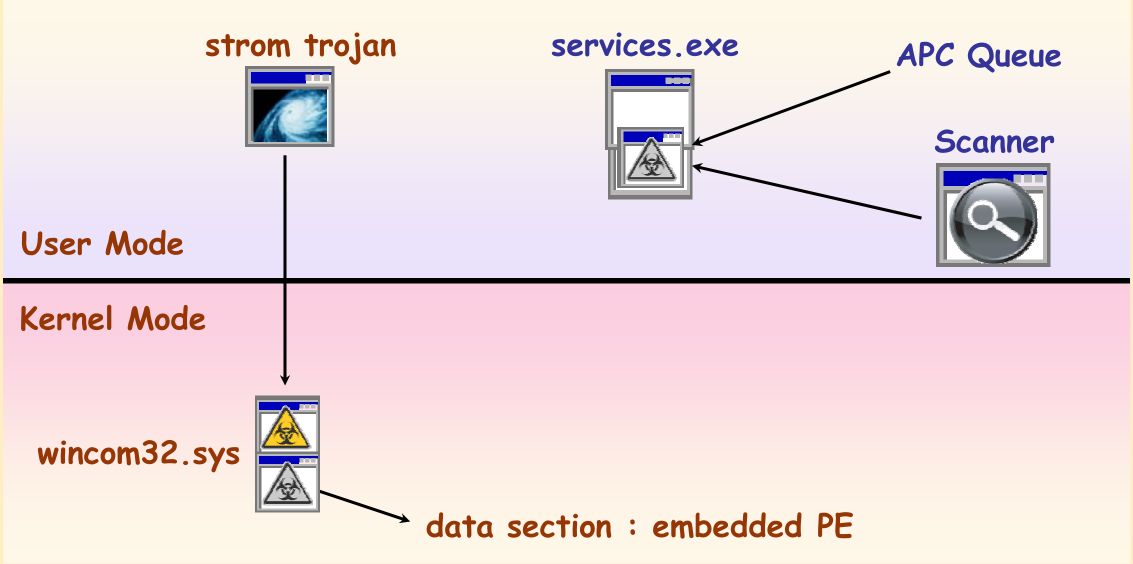
WMI (COM)

NTDLL.DLL

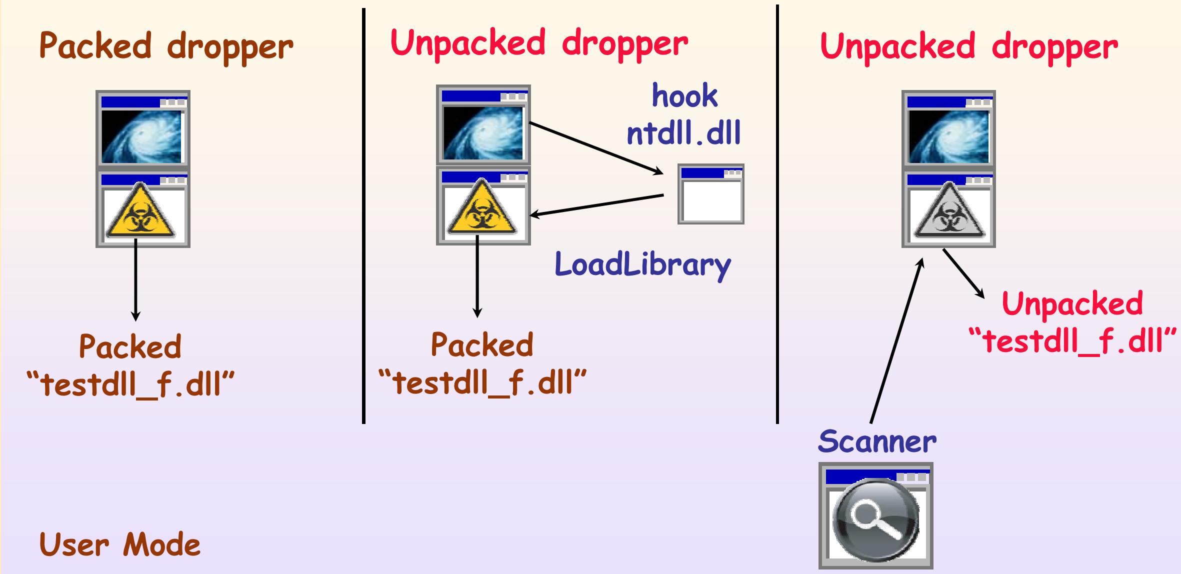
DEMO – 1 (un)

Detecting Storm Trojan's Injected Code

Storm Trojan *(01/22/2007)*



Storm Trojan *(06/25/2008)*



Packed dropper

Packed "testdll_f.dll"

Unpacked dropper

hook ntdll.dll

LoadLibrary

Packed "testdll_f.dll"

Unpacked dropper

Unpacked "testdll_f.dll"

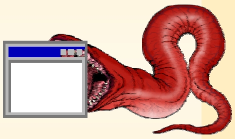
Scanner

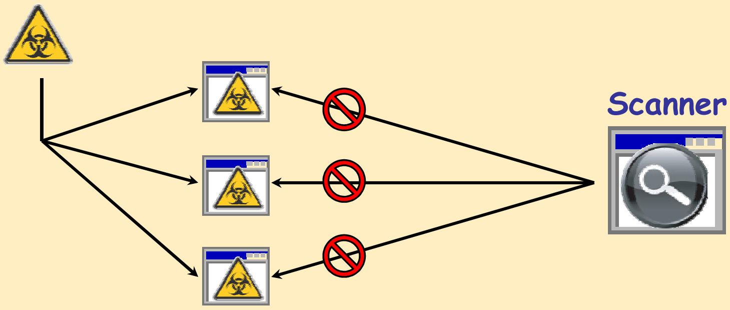
User Mode

DEMO – 2 (deux)

Accessing protected files

Sober mass-mailing worm





Obtain handle to self

Createfile() SharedMode = 0

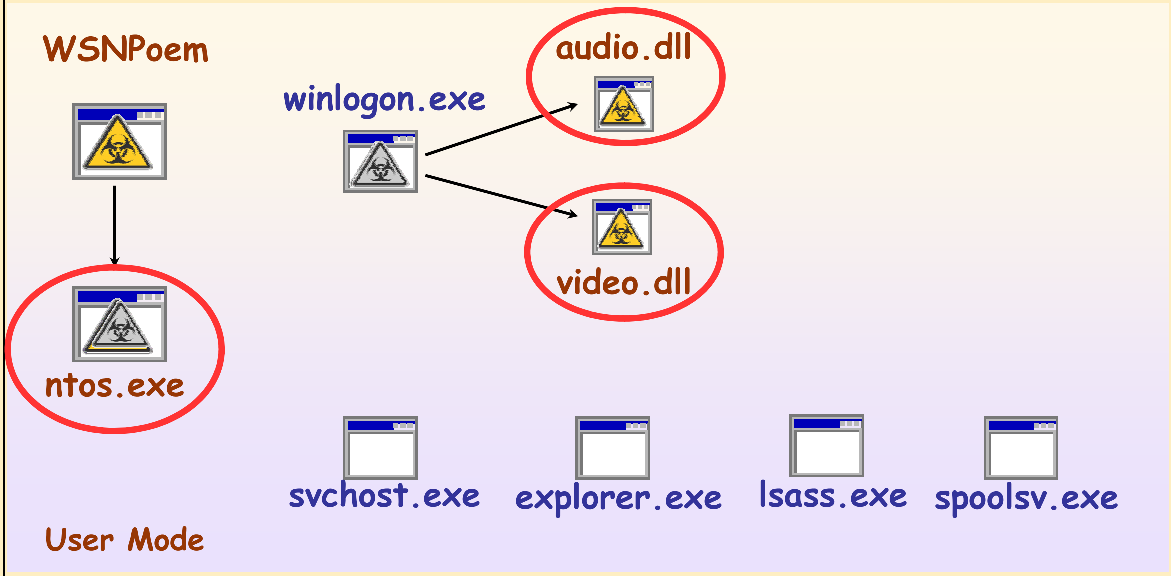
DEMO – 3 (trois)

Detecting Injected Code

&

Hidden Files

WSNPoem (03/25/2008)

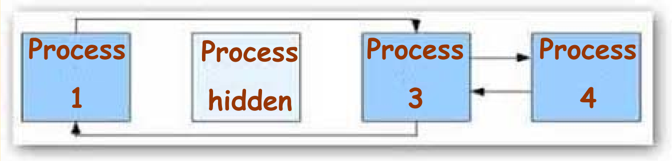


DEMO – 4 (quatre)

Detecting Hidden Processes

Fu and *FuTo* rootkits (DKOM)

EPROCESS



OpenProcess() brute-force-Process-ID method

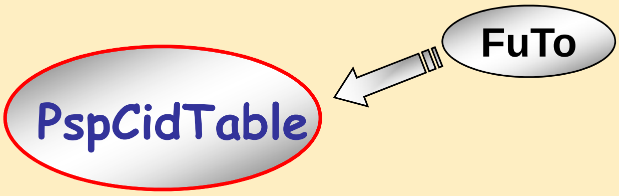
PID = 0x0000

...

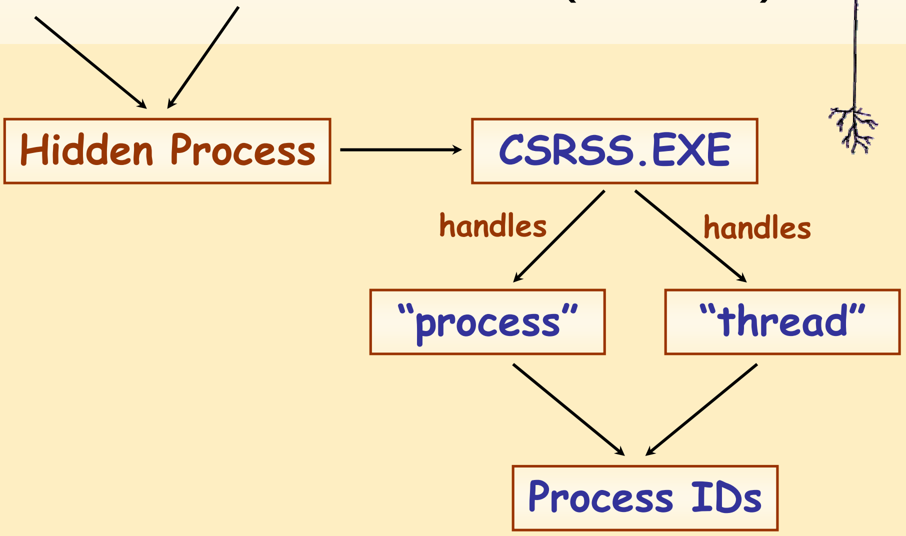
...

...

PID = 0x4E1C



Fu and *FuTo* rootkits (DKOM)



Cross-view based diff

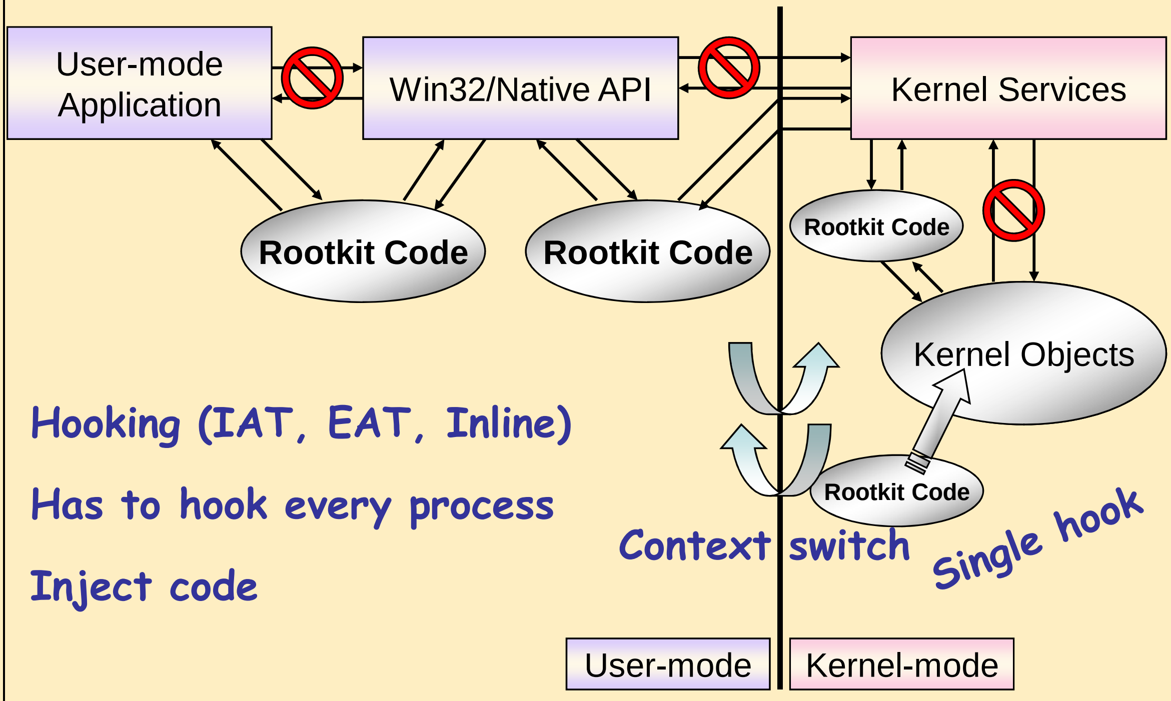
Possible false positives

Pros and Cons

Cons:

- Intercepting control transfers and modifying results

Interception & Modification



Hooking (IAT, EAT, Inline)

Has to hook every process

Inject code

Pros and Cons

Cons:

- Intercepting control transfers and modifying results
- Limited privileges if user logged in as a limited user

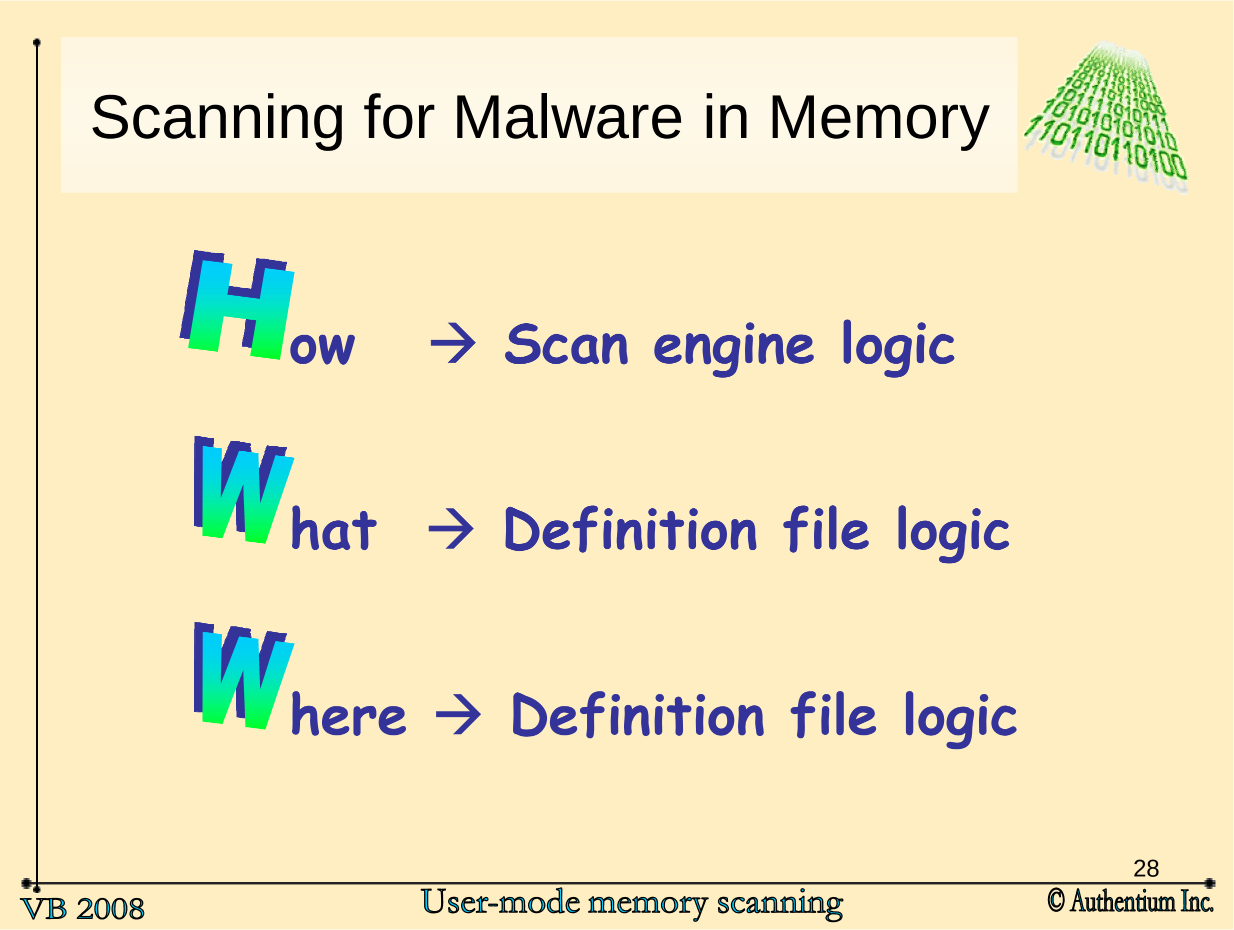
Pros:

- Easy to implement, debug and deploy
- No risk of causing system wide crash
- Compatibility issues are easy to overcome

Best Practice

Implement memory scanner in both User mode & Kernel mode

Reveal hidden memory objects by comparing results (cross-view diff)



Scanning for Malware in Memory

How → Scan engine logic

What → Definition file logic

Where → Definition file logic