Malicious Software II

Mobile Code

- Mobile code refers to “programs (e.g. script, macro or other portable instruction that can be shipped unchanged to a heterogeneous collection of platforms and executed with identical semantics”
- Transmitted from a remote system to a local system and then executed on the local system
- It often acts as a mechanism for a virus, worm, or Trojan horse to be transmitted to the user’s workstation.
- Takes advantage of vulnerabilities to perform its own exploits such as unauthorized data access or root compromise.
- Popular vehicles include:
    - ActiveX, PDF, Postscript, Flash Animations
    - JavaScript
    - VBScript
- Most common ways of using mobile code for malicious operations on local system are:
    - Cross-site scripting
    - Interactive and dynamic Web sites
    - E-mail attachments
    - Downloads from untrusted sites or of untrusted software

Mobile Phone Worms

- First discovery was Cabir worm in 2004 then Lasco and CommWarrior in 2005
- Communicate through Bluetooth wireless connections or MMS
- Target is the smartphone which is a mobile phone that permits users to install software applications from sources other than the cellular network operator.
- Can completely disable the phone, delete data on the phone, or force the device to send costly messages
- CommWarrior replicates by means of Bluetooth to other phones, sends itself as an MMS file to contacts and as an auto reply to incoming text messages.
- Vast majority of mobile phone malware use Trojan apps to install themselves.

Cabir Worm

Classification

Category : Malware **Type :** Bluetooth-Worm

Platform : SymbOS **Aliases :** Bluetooth-Worm:SymbOS/Cabir

- Cabir can only reach mobile phones that support Bluetooth, and are in discoverable mode.
- Setting your phone into non-discoverable (hidden) Bluetooth mode will protect your phone from the Cabir worm.
- Once the phone is infected however it will try to infect other systems even if user tries to disable Bluetooth from system settings.

Cabir Worm: Propagation

- Cabir replicates over Bluetooth with a file named caribe.sis that contains the worm's main executable caribe.app, system recognizer flo.mdl and resource file caribe.rsc.
- The caribe.sis file will not arrive automatically to the target device, so the user needs to answer yes to the transfer question while the infected device is still in range.
- The question will be repeated to the user if they select no.
- The user must click the SIS file in order to install it.

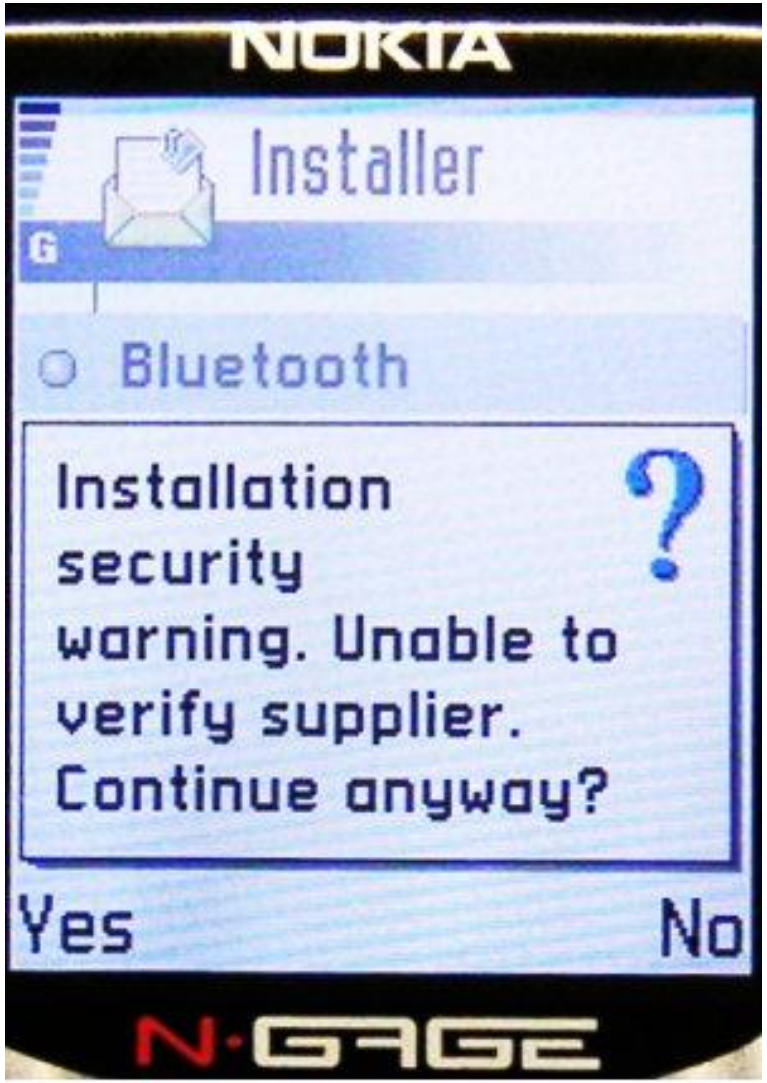
Cabir Worm: Propagation

- The SIS file contains autostart settings that will automatically execute caribe.app after the SIS file is installed.
- The worm then activates and starts looking for new devices to infect via Bluetooth.
- The Cabir worm starts sending infected caribe.sis files to the first device it finds.
- The replication routine in Cabir contains a bug that causes it to lock onto the first device found; it won't continue look for other devices.
- Cabir is capable of sending infected files to only one other device per activation.

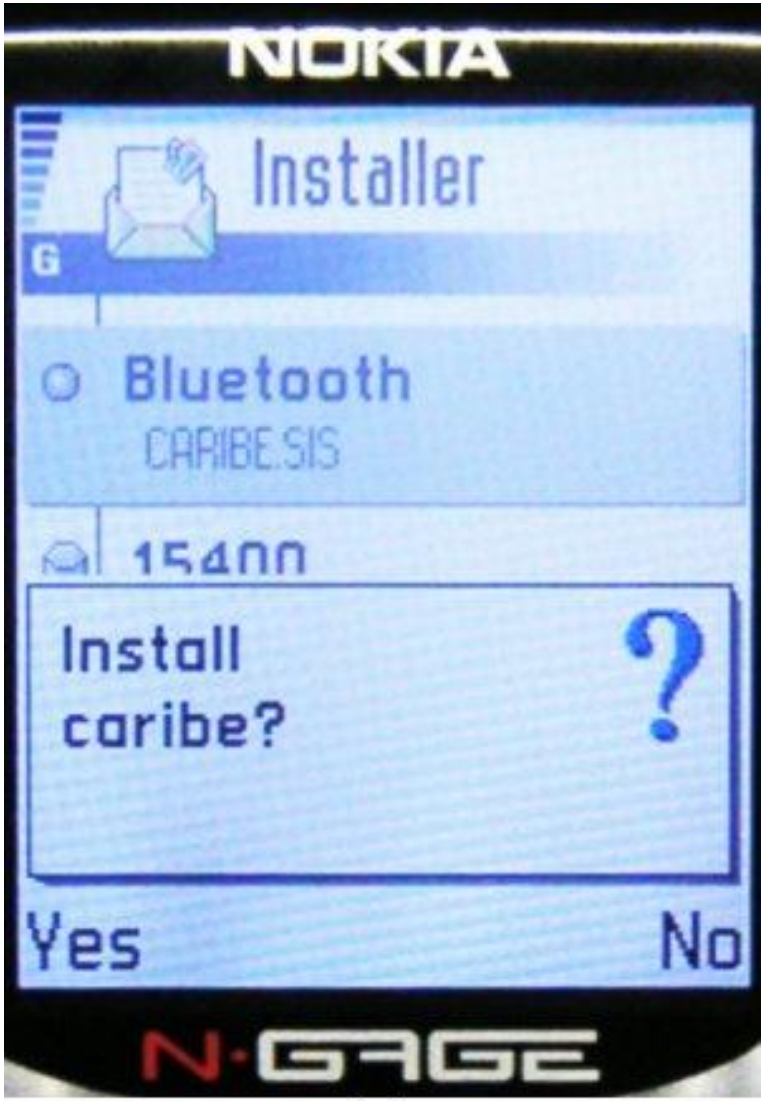
Cabir Worm: Propagation

- It will try to infect one other device when it is activated the first time, and then one more each time when the phone is rebooted.
- It is also found that the newly infected phone will first look for the phone that sent it the infected file.
- This means Cabir is capable of spreading widely only in cases where the phone that sent the infected file is out of range before the user activates Cabir on the new phone.
- Cabir is capable of spreading in the wild, it would spread quite slowly and would not cause a large epidemic.

Cabir Worm: Propagation

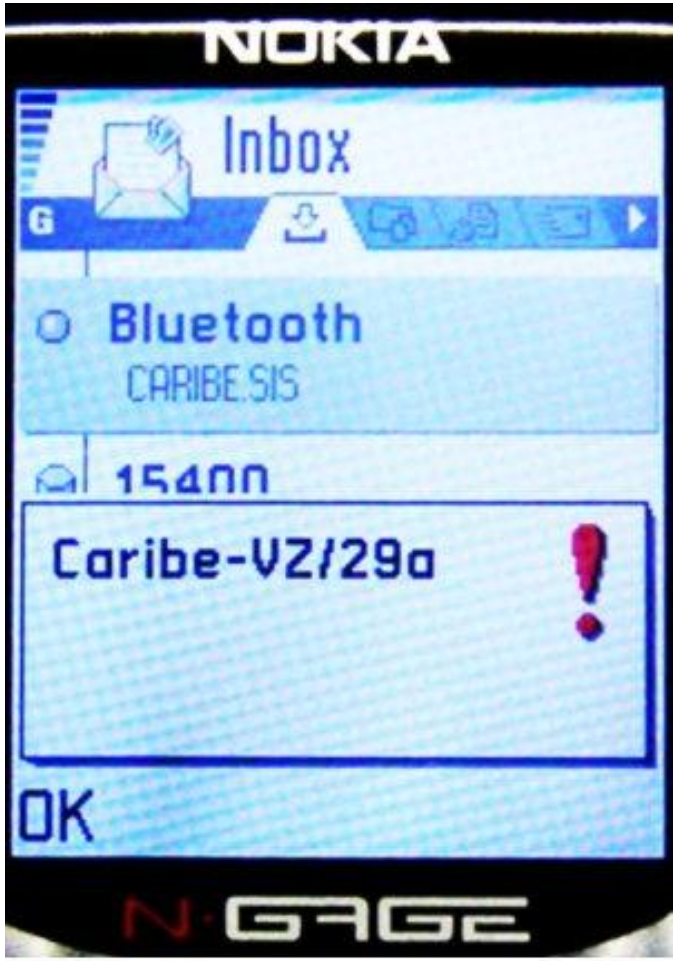


Copyright F-Secure Corp. 2004



Copyright F-Secure Corp. 2004

Cabir Worm: Propagation



- When the caribe.sis file is installed the installer will copy the worm's executables into following locations:

c:\system\apps\caribe\caribe.app
c:\system\apps\caribe\caribe.rsc
c:\system\apps\caribe\flo.mdl

- When the caribe.app is executed it copies the following files:

caribe.app to
c:\system\symbiansecuredata\caribesecuritymanager\
caribe.rsc to
c:\system\symbiansecuredata\caribesecuritymanager\
flo.mdl to c:\system\recogs

Drive-By-Downloads

Exploits browser and plugin vulnerabilities so when the user views a web page controlled by the attacker, it contains code that exploits the bug to download and install malware on the system without the user's knowledge or consent

In most cases the malware does not actively propagate as a worm does

Spreads when users visit the malicious Web page

Watering-Hole Attacks

- A variant of drive-by-download used in highly targeted attacks
- The attacker researches their intended victims to identify websites they are likely to visit, then scans these sites to identify those with vulnerabilities that allow their compromise
- They then wait for one of their intended victims to visit one of the compromised sites
- Attack code may even be written so that it will only infect systems belonging to the target organization and take no action for other visitors to the site
- This greatly increases the likelihood of the site compromise remaining undetected

Malvertising

Places malware on websites without actually compromising them

The attacker pays for advertisements that are highly likely to be placed on their intended target websites and incorporate malware in them

Using these malicious ads, attackers can infect visitors to sites displaying them

The malware code may be dynamically generated to either reduce the chance of detection or to only infect specific systems

Has grown rapidly in recent years because they are easy to place on desired websites with few questions asked and are hard to track

Attackers can place these ads for as little as a few hours, when they expect their intended victims could be browsing the targeted websites, greatly reducing their visibility

Social Engineering

- "Tricking" users to assist in the compromise of their own systems

- Spam: Unsolicited bulk e-mail.
  - significant carrier of malware
  - used for phishing attacks
- Trojan Horse
  - Program or utility containing harmful hidden code
  - Used to accomplish functions that the attacker could not accomplish directly.

Payload- System Corruption

- Once malware is active on the target system, the next concern is what actions it will take on this system.

- That is, what payload does it carry.

Payload- System Corruption

Ransomware

- Encrypts the user's data and demands payment in order to access the key needed to recover the information

- Mid-2006 a number of worms and Trojans appeared that used public-key cryptography with increasingly larger key sizes to encrypt data

- The user needed to pay a ransom, or to make a purchase from certain sites, in order to receive the key to decrypt this data

Ransomware: WannaCry

- Infected a large number of systems across world in May 2017.
- When got installed on infected systems, it encrypts a large number of files and then demanded a ransom payment in Bitcoins to recover them.
- Recovery of this information was generally only possible if the organization had good backups and an appropriate incident response and disaster recovery plan.
- Targets widened beyond personal computer systems to include mobile devices and Linux servers.

WannaCry: Analysis

- The malware makes use of an exploit named EternalBlue developed by **NSA analysts** which was patched by **Microsoft** on 14 March 2017, although there are many unpatched systems still vulnerable.

- Applying this patch mitigates the spread of WannaCry, but will not prevent infection.

WannaCry: Analysis

- **EternalBlue**, exploits a vulnerability in the **Server Message Block (SMB) protocol** which allows the malware to spread to all unpatched Windows systems from XP to 2016 on a network that have this protocol enabled.

- The vulnerability, tracked as **CVE-2017-0144**, allows remote attackers to execute arbitrary code on vulnerable systems without authentication.

- By sending specially crafted packets to the SMBv1 server, attackers can exploit this vulnerability to gain unauthorized access to systems running unpatched versions of Windows.

- **SMB protocol** is a **network file sharing protocol** that provides shared access to files, printers, serial ports, and other resources on a network.

WannaCry: Analysis

- When the dropper is executed, it first attempts to make a connection to **http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com** and exits if the connection is successful.

- The threat actors may have added this HTTP connection test to prevent automated sandboxes from running and analyzing the malware.

- This domain was previously unregistered, causing this connection to fail.

- On the afternoon of May 12; however, this domain was registered and sinkholed by researcher **MalwareTech**, effectively acting as a "killswitch" for many systems, and thereby slowing the rate of infection.

- However, the method by which the malware opens the connection does not affect systems connecting through a proxy server, leaving those systems still vulnerable.

WannaCry: Analysis

- If the connection fails, the dropper attempts to create a service named “mssecsvc2.0” with the DisplayName “Microsoft Security Center (2.0) Service”.

- The dropper then extracts the encrypter binary from its resource R/1831, writes it to the hardcoded filename %WinDir%\tasksche.exe, and then executes it.

- When executed, the encrypter checks to see if the mutex “MsWinZonesCacheCounterMutexA0” exists, and will not proceed if present.

WannaCry: Analysis

```
push    offset aGlobalMswinzon ; "Global\\MsWinZonesCacheCounterMutexA"
lea     eax, [ebp+Dest]
push    offset aSD      ; The sprintf format "%s%d" appends a "0" to the end of the mutex name
push    eax             ; Dest
call    ds:sprintf      ; Global\\MsWinZonesCacheCounterMutexA0
xor     esi, esi
add     esp, 10h
cmp     [ebp+arg_0], esi
jle     short loc_401F4C

                        ; CODE XREF: check_mutex+4B↓j
lea     eax, [ebp+Dest]
push    eax             ; lpName
push    1               ; bInheritHandle
push    100000h         ; dwDesiredAccess
call    ds:OpenMutexA   ; Check for existence of mutex
test    eax, eax
jnz     short loc_401F51 ; If this mutex exists, the malware exits
push    1000            ; dwMilliseconds
call    ds:Sleep
inc     esi             ; Increment the counter
cmp     esi, [ebp+arg_0] ; Compares the incrementer to the value 60, effectively
                        ; performing this mutex check each second for one minute
jl      short loc_401F26
```

WannaCry: Analysis

- The encrypter binary also contains a password-protected zip file (password: WNcry@2ol7) containing the following files:
  - A directory named “msg” containing Rich Text Format files with the extension .wnry
  - These files are the “Readme” file used by the @WanaDecryptor@.exe decrypter program in different languages:

WannaCry: Analysis

- b.wnry, a bitmap file displaying instructions for decryption
- c.wnry, containing the following addresses:
  - gx7ekbenv2riucmf.onion
  - 57g7spgrzlojinas.onion
  - xxlvbrloxvriy2c5.onion
  - 76jdd2ir2embyv47.onion
  - cwwnhwhlz52maqm7.onion
  - https://dist.torproject.org/torbrowser/6.5.1/tor-win32-0.2.9.10.zip
- r.wnry, additional decryption instructions used by the decrypter tool, in English
- s.wnry, a zip file containing the Tor software executable
- t.wnry, encrypted using the WANACRY! encryption format, where "WANACRY!" is the file header
- taskdl.exe, (hash 4a468603fdcb7a2eb5770705898cf9ef37aade532a7964642ecd705a74794b79), file deletion tool
- taskse.exe, (hash 2ca2d550e603d74dedda03156023135b38da3630cb014e3d00b1263358c5f00d), enumerates Remote Desktop Protocol (RDP) sessions and executes the malware on each session
- u.wnry (hash b9c5d4339809e0ad9a00d4d3dd26fdf44a32819a54abf846bb9b560d81391c25), "@WanaDecryptor@.exe" decrypter file

WannaCry: Analysis

- After dropping these files to its working directory, the malware attempts to change the attributes of all the files to “hidden” and grant full access to all files in the current directory and any directories below.

- It does this by executing “attrib +h .”,

  followed by “icacls . /grant Everyone:F /T /C /Q”.

```
push    ebx             ; lpExitCode
push    ebx             ; dwMilliseconds
push    offset CommandLine ; "attrib +h ."
call    sub_401064
push    ebx             ; lpExitCode
push    ebx             ; dwMilliseconds
push    offset aIcacls_GrantEv ; "icacls . /grant Everyone:F /T /C /Q"
call    sub_401064
add     esp, 20h
```

WannaCry: Analysis

- WannaCry then proceeds to encrypt files on the system, searching for the following file extensions, which are hard-coded in the binary:

.docx	.ppam	.sti	.vcd	.3gp	.sch	.myd	.wb2
.docb	.potx	.sldx	.jpeg	.mp4	.dch	.frm	.slk
.docm	.potm	.sldm	.jpg	.mov	.dip	.odb	.dif
.dot	.pst	.sldm	.bmp	.avi	.pl	.dbf	.stc
.dotm	.ost	.vdi	.png	.asf	.vb	.db	.sxc
.dotx	.msg	.vmdk	.gif	.mpeg	.vbs	.mdb	.ots
.xls	.eml	.vmx	.raw	.vob	.ps1	.accdb	.ods
.xlsm	.vsd	.aes	.tif	.wmv	.cmd	.sqlitedb	.max
.xlsb	.vsdx	.ARC	.tiff	.fla	.js	.sqlite3	.3ds
.xlw	.txt	.PAQ	.nef	.swf	.asm	.asc	.uot
.xlt	.csv	.bz2	.psd	.wav	.h	.lay6	.stw
.xlm	.rtf	.tbk	.ai	.mp3	.pas	.lay	.sxw
.xlc	.123	.bak	.svg	.sh	.cpp	.mml	.ott
.xltx	.wks	.tar	.djvu	.class	.c	.sxm	.odt

WannaCry: Analysis

- The WannaCry encrypter launches the embedded decrypter binary "@WanaDecryptor@.exe," which displays two timers and instructions for sending the ransom in the configured language of the infected system.

```
push    ebp
mov     ebp, esp
sub     esp, 318h
lea     eax, [ebp+var_318]
push    1               ; int
push    eax             ; void *
mov     [ebp+var_C], offset a13am4vw2dhxygx ; "13AM4VW2dhxYgXeQepoHkHSQuy6NgaEb94"
mov     [ebp+var_8], offset a12t9ydpgwuez9n ; "12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw"
mov     [ebp+var_4], offset a115p7ummngoj1p ; "115p7UMMngoj1pMvkpHijcRdfJNXj6LrLn"
call    sub_401000
pop     ecx
```

WannaCry: Analysis

WannaCry: Analysis

- After the files are encrypted, the decrypter program attempts to delete any Windows Shadow Copies via this command:

- **cmd.exe /c vssadmin delete shadows /all /quiet & wmic shadowcopy delete & bcdedit /set {default} bootstatuspolicy ignoreallfailures & bcdedit /set {default} recoveryenabled no & wbadmin delete catalog –quiet**

Payload – Attack Agents Bots

- Takes over another Internet attached computer and uses that computer to launch or manage attacks
- ***Botnet*** - collection of bots capable of acting in a coordinated manner
- Uses:
  - Distributed denial-of-service (DDoS) attacks
  - Spamming
  - Sniffing traffic
  - Spreading new malware
  - Manipulating online polls/games

Remote Control Facility of BOT

- Distinguishing a bot from a worm
  - Worm propagates itself and activates itself
  - Bot is initially controlled from some central facility
- Typical means of implementing the remote control facility is on an IRC (internet relay chat) server
  - Bots join a specific channel on this server and treat incoming messages as commands
  - More recent botnets use covert communication channels via protocols such as HTTP
  - Distributed control mechanisms use peer-to-peer protocols to avoid a single point of failure

Payload – Information Theft: Keyloggers and Spyware

Keylogger

- Captures keystrokes to allow attacker to monitor sensitive information
- Typically uses some form of filtering mechanism that only returns information close to keywords (“login”, “password”)

Spyware

- Subverts the compromised machine to allow monitoring of a wide range of activity on the system
  - Monitoring history and content of browsing activity
  - Redirecting certain web page requests to fake sites
  - Dynamically modifying data exchanged between the browser and certain Web sites of interest

Payload – Information Theft Phishing

- Exploits social engineering to leverage the user's trust by masquerading as communication from a trusted source
  - Include a URL in a spam email that links to a fake website that mimics the login page of a banking, gaming, or similar site
  - Suggests that urgent action is required by the user to authenticate their account
  - Attacker exploits the account using the captured credentials

- **Spear-phishing**
  - Recipients are carefully researched by the attacker
  - E-mail is crafted to specifically suit its recipient, often quoting a range of information to convince them of its authenticity

Payload – Stealthing Backdoor

- Also known as a *trapdoor*
- Secret entry point into a program allowing the attacker to gain access and bypass the security access procedures
- *Maintenance hook* is a backdoor used by Programmers to debug and test programs.
- This usually is done when the programmer is developing an application that has an authentication procedure, or a long setup, requiring the user to enter many different values to run the application.

Payload – Stealthing Backdoor

- The programmer may also want to ensure that there is a method of activating the program should something be wrong with the authentication procedure that is being built into the application.
- The backdoor is code that recognizes some special sequence of input or is triggered by being run from a certain user ID or by an unlikely sequence of events.
- Difficult to implement operating system controls for backdoors in applications

Payload – Stealthing- Rootkit

- Malware that hide (from spyware blockers, antivirus, system management tools) on your system.

- Rootkits have two primary functions: remote command/control (back door) and software eavesdropping.

- Rootkits allow someone, **legitimate or otherwise**, to administratively control a computer.

Rootkit Classification Characteristics

Persistent (Registry/File System)	Memory based (Cannot survive reboot)	User mode (intercepts API calls)
Kernel mode (Remove itself from kernel active processes)	Virtual machine based	External mode (exists in BIOS Mode)

Rootkit Classification

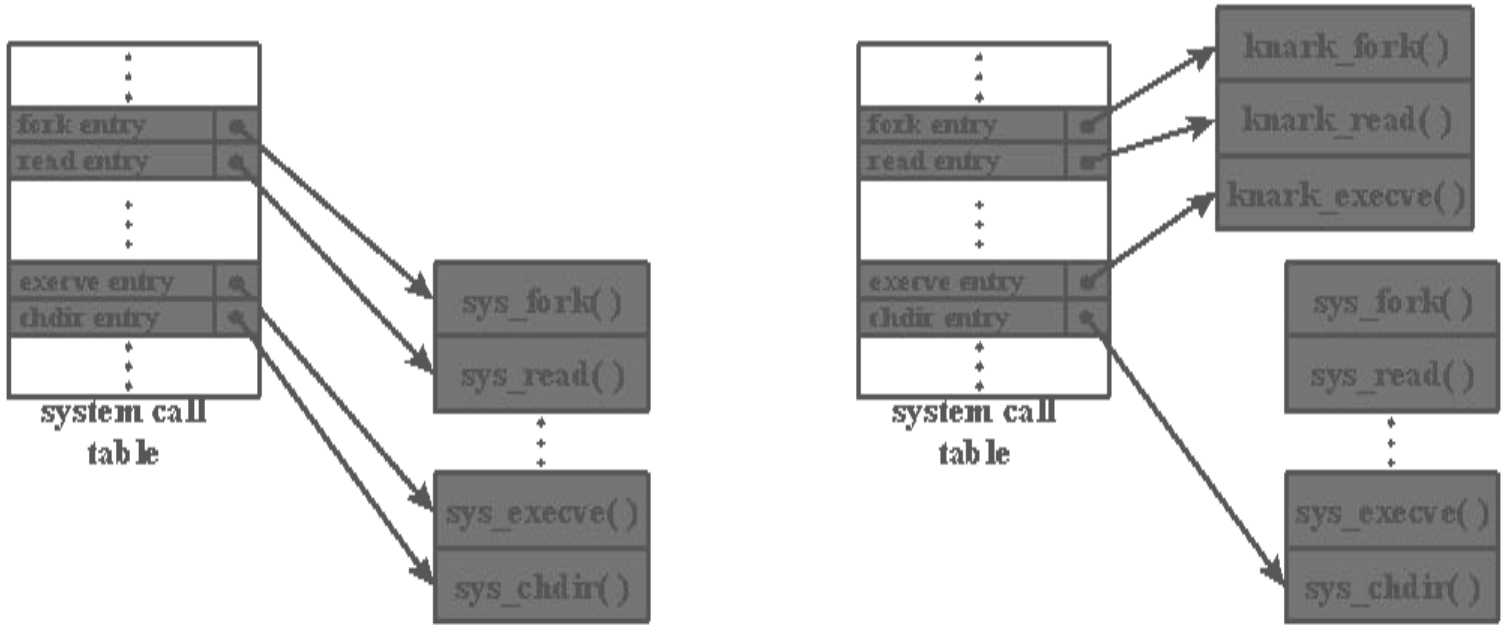
- **Persistent:**
  - Activates each time the system boots.
  - The rootkit must store code in a persistent store, such as the registry or file system, and configure a method by which the code executes without user intervention.
  - This means it is easier to detect, as the copy in persistent storage can potentially be scanned.
- Memory based:
  - Has no persistent code and therefore cannot survive a reboot.
  - However, because it is only in memory, it can be harder to detect.
- User mode:
  - Intercepts calls to APIs (application program interfaces) and modifies returned results.
  - For example, when an application performs a directory listing, the return results do not include entries identifying the files associated with the rootkit.

Rootkit Classification

- **Kernel mode:**
  - Can intercept calls to native APIs in kernel mode.
  - The rootkit can also hide the presence of a malware process by removing it from the kernel's list of active processes.

- **Virtual machine based:**
  - This type of rootkit installs a lightweight virtual machine monitor, and then runs the operating system in a virtual machine above it.
  - The rootkit can then transparently intercept and modify states and events occurring in the virtualized system.

- **External mode:**
  - The malware is located outside the normal operation mode of the targeted system, in BIOS or system management mode, where it can directly access hardware.



(a) Normal kernel memory layout

(b) After nkark install

Figure 6.3 System Call Table Modification by Rootkit

Techniques to change system calls

- **Modify the system call table:** The attacker modifies selected syscall addresses stored in the system call table. This enables the rootkit to direct a system call away from the legitimate routine to the rootkit's replacement.

- **Modify system call table targets:** The attacker overwrites selected legitimate system call routines with malicious code. The system call table is not changed.

- **Redirect the system call table:** The attacker redirects references to the entire system call table to a new table in a new kernel memory location.

Malware Countermeasure Approaches

- Ideal solution to the threat of malware is prevention

Four main elements of prevention:

- Policy
- Awareness
- Vulnerability mitigation
- Threat mitigation

- Ensure systems are up to date to prevent vulnerability exploit.
- Appropriate access controls, to reduce number of files that can be accessed by an user.
- These measures directly target the key propagation mechanisms used by worms, viruses, and some Trojans.

Malware Countermeasure Approaches

- Ideal solution to the threat of malware is prevention

Four main elements of prevention:

- Policy
- Awareness
- Vulnerability mitigation
- Threat mitigation

- If prevention fails, technical mechanisms can be used to support the following threat mitigation options:
  - Detection
  - Identification
  - Removal

Generations of Anti-Virus Software

- First generation: simple scanners
  - Requires a malware signature to identify the malware
  - Limited to the detection of known malware
- Second generation: heuristic scanners
  - Uses heuristic rules to search for probable malware instances
  - Another approach is integrity checking
- Third generation: activity traps
  - Memory-resident programs that identify malware by its actions rather than its structure in an infected program
- Fourth generation: full-featured protection
  - Packages consisting of a variety of anti-virus techniques used in conjunction
  - Include scanning and activity trap components and access control capability

First generation: Simple Scanners

- A first-generation scanner requires a malware signature to identify the malware.
- The signature essentially has same structure and bit pattern in all copies of the malware.
- Such signature-specific scanners are limited to the detection of known malware.
- Another type of first-generation scanner maintains a record of the length of programs and looks for changes in length as a result of virus infection.

First generation: Simple Scanners

- Many security products rely on file signatures in order to detect malware and other malicious files.
- The technique involves reading or scanning a file and testing to see if the file matches a set of predetermined attributes.
- These attributes are known as the malware's 'signature'.
- Malware signatures, which can occur in many different formats, are created by vendors and security researchers.
- Sets of signatures are collected in databases, some of which may be public and shared while others are contained in proprietary databases exclusive to a particular vendor.

Second Generation Scanners

- A second-generation scanner does not rely on a specific signature. Rather, the scanner uses heuristic rules to search for probable malware instances.

- One class of such scanners looks for fragments of code that are often associated with malware.

- Another second-generation approach is integrity checking. A checksum can be appended to each program. If malware alters or replaces some program without changing the checksum, then an integrity check will catch this change.

- Hash functions can also be used by which the malware is prevented from adjusting the program to produce the same hash code as before.

Third Generation Scanners

- Third-generation programs are **memory-resident programs** that identify malware by its actions rather than its structure in an infected program.

- Such programs have the advantage that it is not necessary to develop signatures and heuristics for a wide array of malware.

- Rather, it is necessary only to identify the **small set of actions** that indicate malicious activity is being attempted and then to intervene.

Fourth Generation Scanners

- Fourth-generation products are packages consisting of a variety of anti-virus techniques used in conjunction.
- These include scanning and activity trap components.
- In addition, such a package includes access control capability, which limits the ability of malware to penetrate a system and then limits the ability of a malware to update files in order to propagate.

Sandbox Analysis

- Running potentially malicious code in an emulated sandbox or on a virtual machine
- Allows the code to execute in a controlled environment where its behavior can be closely monitored without threatening the security of a real system
- Running potentially malicious software in such environments enables the detection of complex encrypted, polymorphic, or metamorphic malware
- The most difficult design issue with sandbox analysis is to determine how long to run each interpretation

Host-Based Behavior-Blocking Software

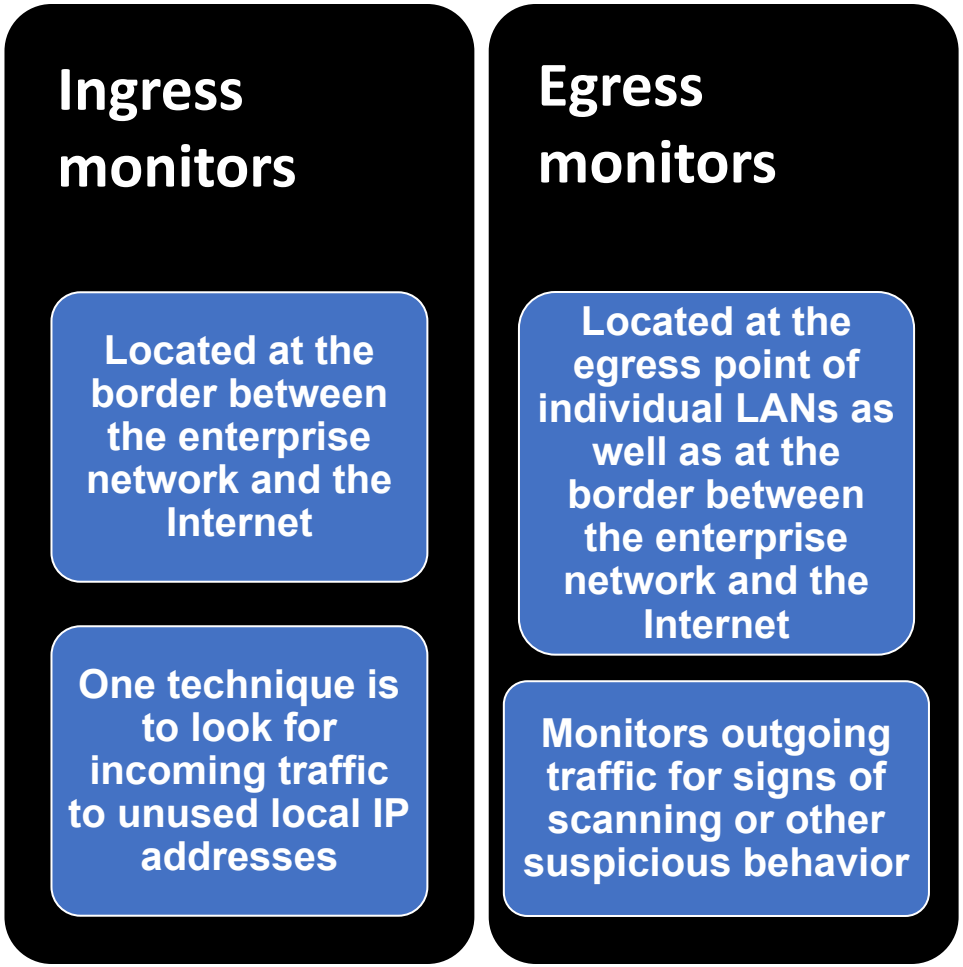
- Integrates with the operating system of a host computer and monitors program behavior in real time for malicious action
  - Blocks potentially malicious actions before they have a chance to affect the system.
  - Blocks software in real time so it has an advantage over anti-virus detection techniques such as fingerprinting or heuristics.

Limitations

- Because malicious code must run on the target machine before all its behaviors can be identified, it can cause harm before it has been detected and blocked

Perimeter Scanning Approaches

- Anti-virus software typically included in e-mail and web proxy services running on an organization's firewall and IDS
- May also be included in the traffic analysis component of an IDS
- May include intrusion prevention measures, blocking the flow of any suspicious traffic
- Approach is limited to scanning malware



Two types of monitoring software