Malware

CS-576 Systems Security

Instructor: Georgios Portokalidis

Fall 2018

Malware

Sample definition

"a program that is inserted into a system, usually covertly, with the intent of compromising the confidentiality, integrity, or availability of the victim's data, applications, or operating system or otherwise annoying or disrupting the victim."

Evil Takes Many Forms

Viruses

Dialers

Worms

Droppers

Rootkits

Spyware

Keyloggers

Adware

Trojan Horses

Backdoors

Ransomware

Main Classification

Infection vector

The type of vulnerability the malware exploits to infect a host

- Software vulnerability, download, design flaw, social engineering, …

The method the malware uses to propagate

- Disks, USB, network, website, ad, …

Payload

The actions the malware takes after infecting

- DDoS, encrypting disk, stealing data, backdoor, …

Also Classified By

Code of malware

- Binary, interpreted (JS, VB), macros, …

Infection point

- File, boot sector, firmware, memory-only, BIOS, …

Activation

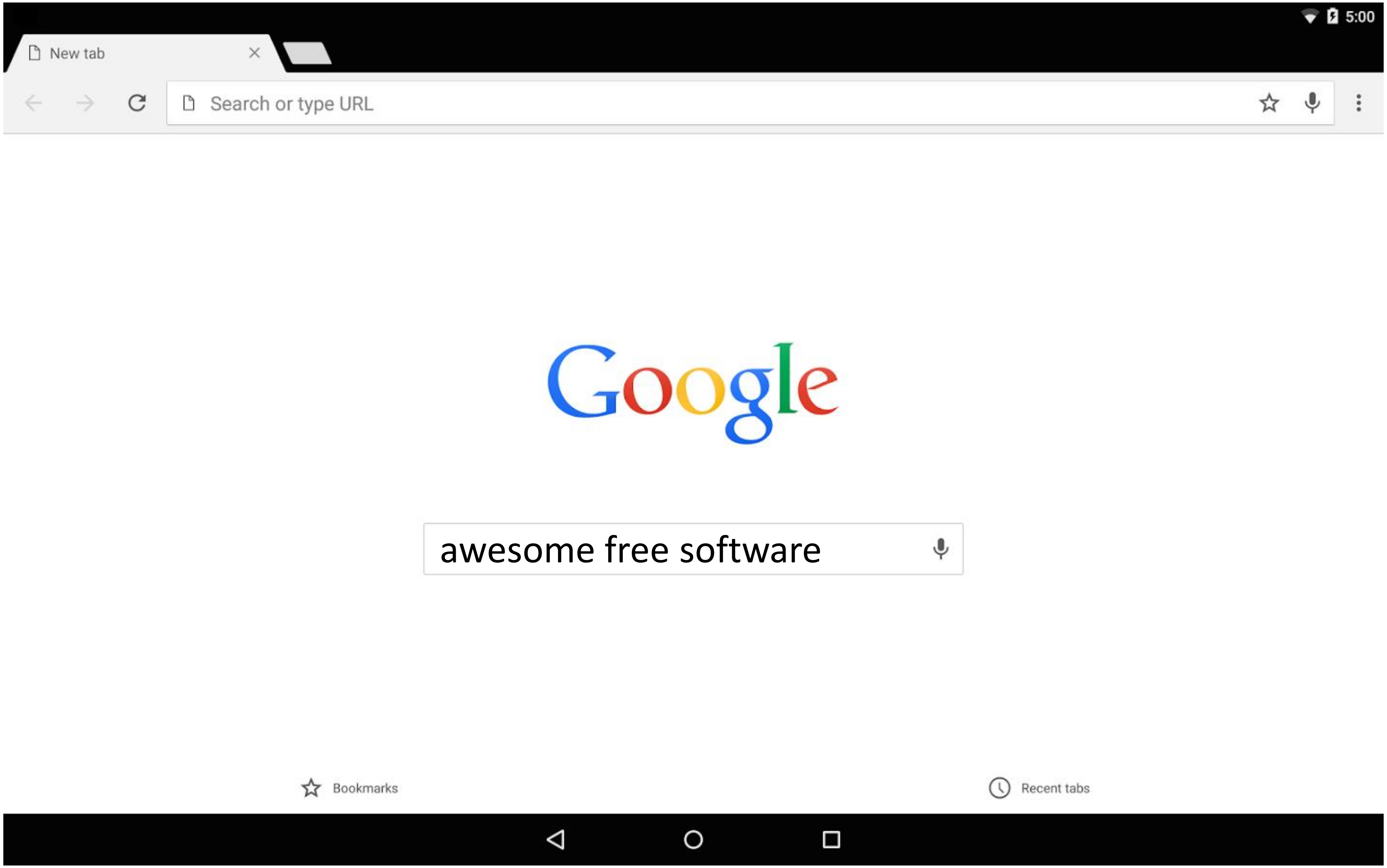
- Through user interaction, automatic, mixed

Evasion strategies

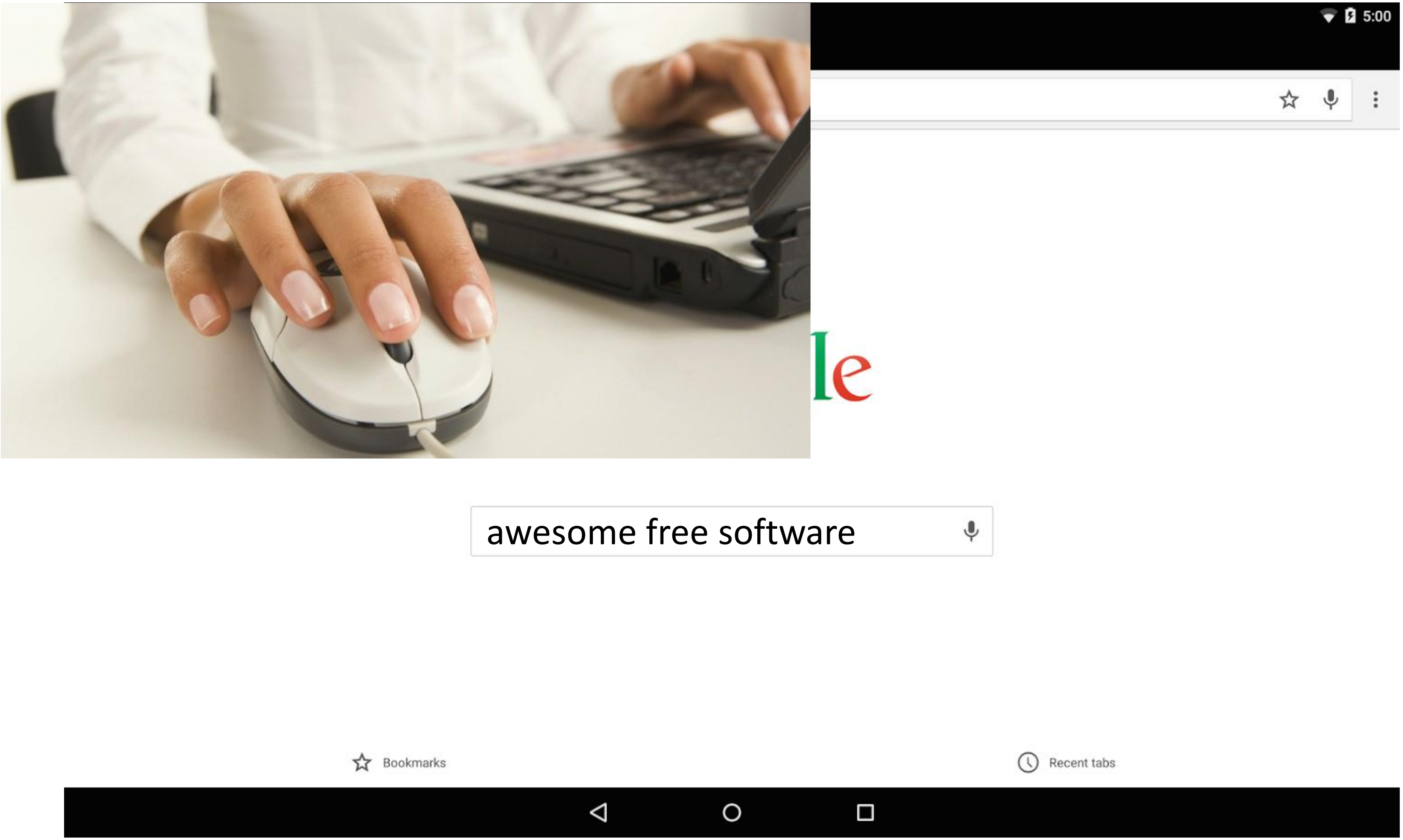
- Packing, polymorphism, obfuscation, anti-VM/debugging, …

Infection Vectors

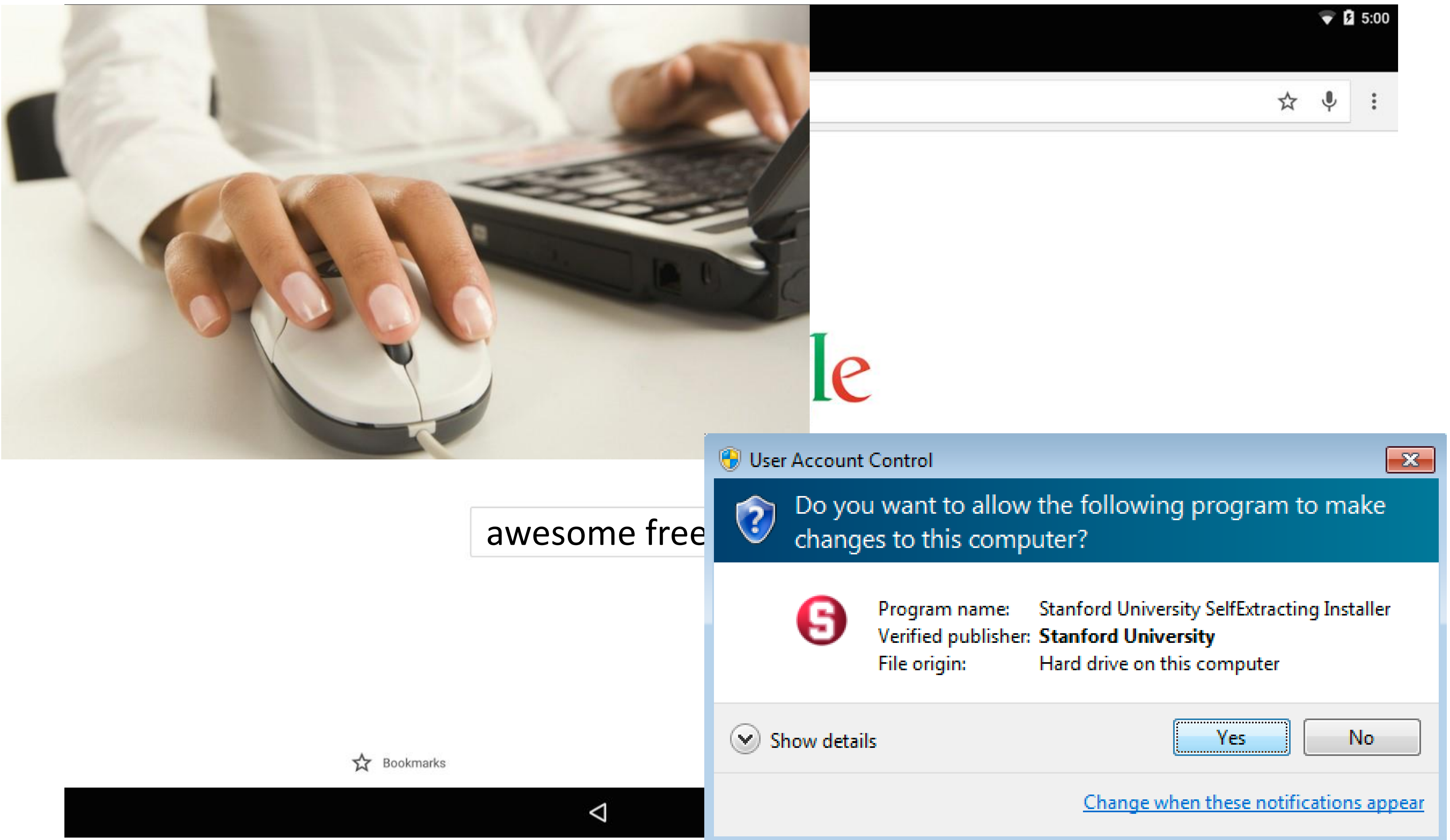
Installed by User

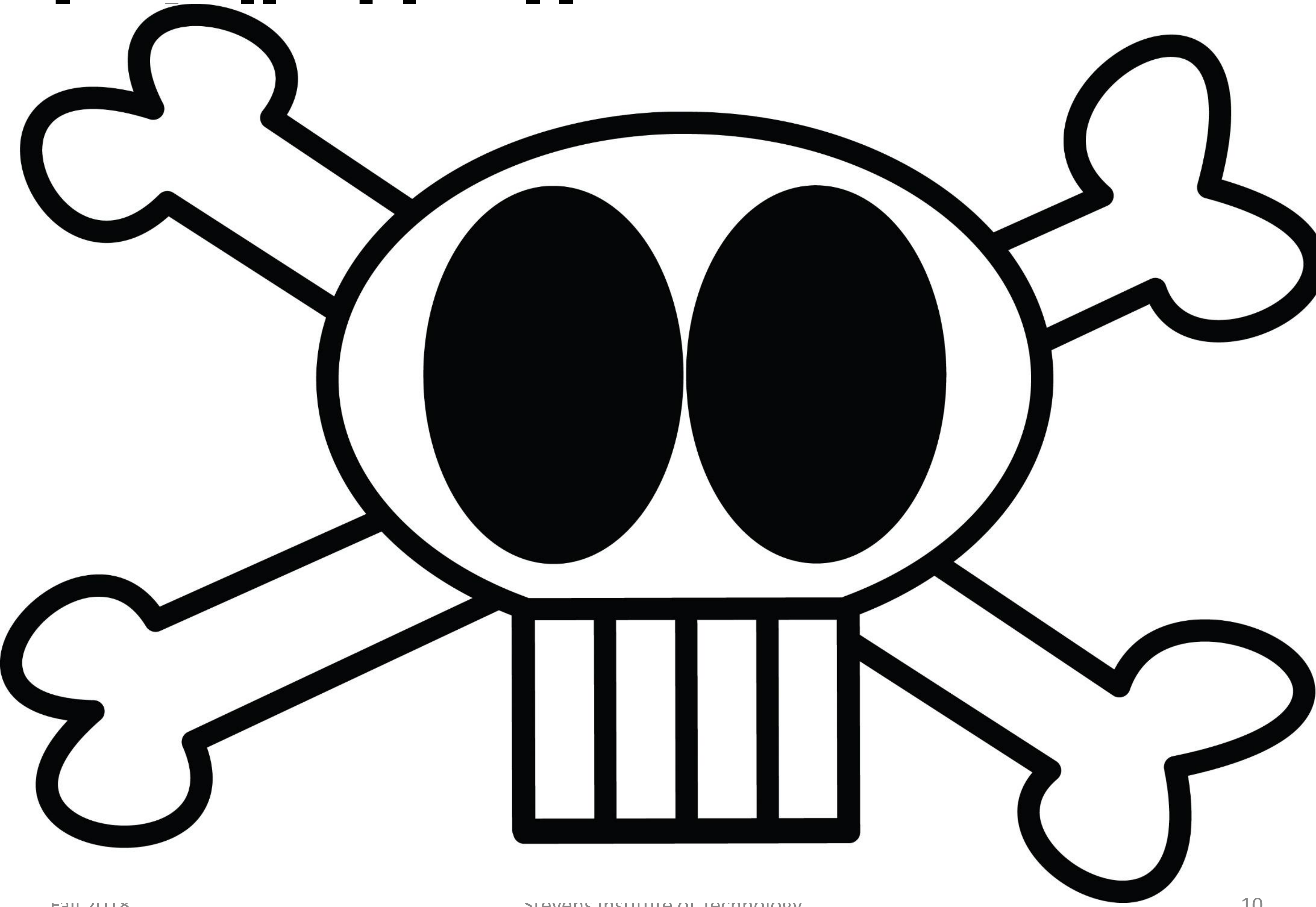


Installed by User



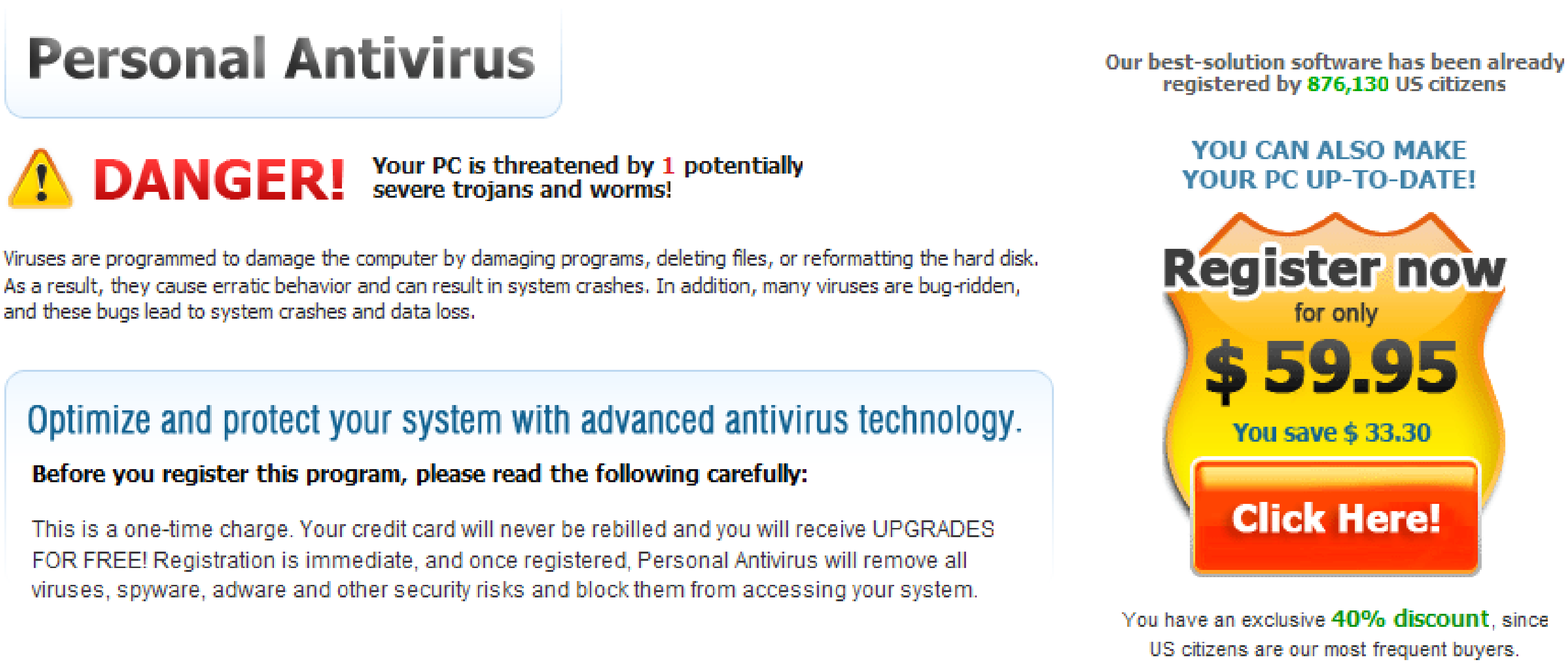
Installed by User





Malware Anti-Malware

ScareWare


Personal Antivirus
DANGER!
Your PC is threatened by 1 potentially severe trojans and worms!
Viruses are programmed to damage the computer by damaging programs, deleting files, or reformatting the hard disk. As a result, they cause erratic behavior and can result in system crashes. In addition, many viruses are bug-ridden, and these bugs lead to system crashes and data loss.
Optimize and protect your system with advanced antivirus technology.
Before you register this program, please read the following carefully:
This is a one-time charge. Your credit card will never be rebilled and you will receive UPGRADES FOR FREE! Registration is immediate, and once registered, Personal Antivirus will remove all viruses, spyware, adware and other security risks and block them from accessing your system.
Our best-solution software has been already registered by 876,130 US citizens
YOU CAN ALSO MAKE YOUR PC UP-TO-DATE!
Register now
for only
$ 59.95
You save $ 33.30
Click Here!
You have an exclusive 40% discount, since US citizens are our most frequent buyers.

The Malware Is Hidden In the Application

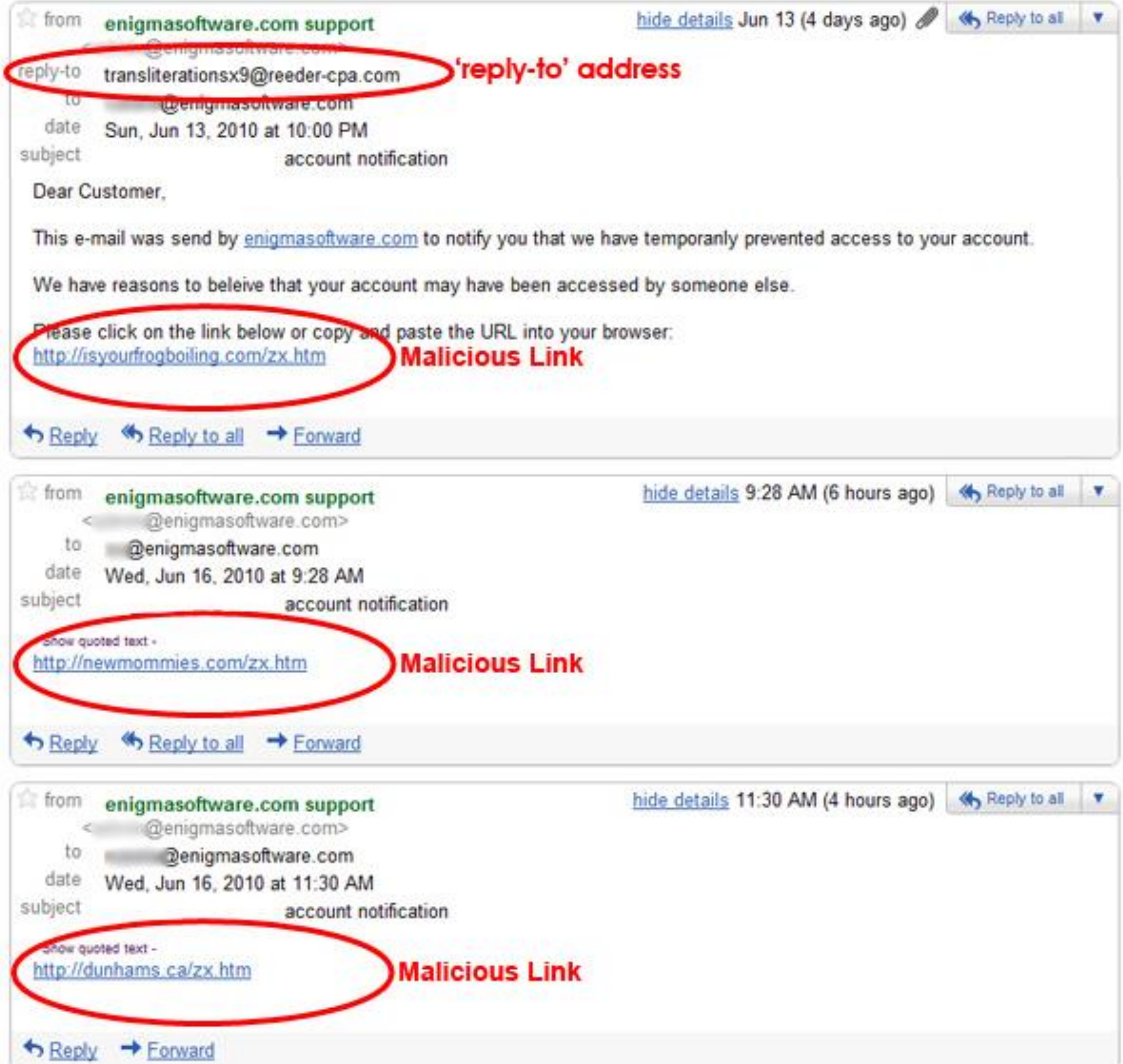
Obtained awesome game from shady source

You Got Mail

Receiving a malicious executable or link over email

The obvious

- A malicious executable as an attachment
  - Usually compressed
- A link to a website serving malicious executables



from enigmasoftware.com support
hide details Jun 13 (4 days ago)
Reply to all
reply-to transliterationsx9@reeder-cpa.com
'reply-to' address
to @enigmasoftware.com
date Sun, Jun 13, 2010 at 10:00 PM
subject account notification
Dear Customer,
This e-mail was send by enigmasoftware.com to notify you that we have temporarily prevented access to your account.
We have reasons to beleive that your account may have been accessed by someone else.
Please click on the link below or copy and paste the URL into your browser:
http://isyourfrogboiling.com/zx.htm
Malicious Link
Reply Reply to all Forward
from enigmasoftware.com support
@enigmasoftware.com>
hide details 9:28 AM (6 hours ago)
Reply to all
to @enigmasoftware.com
date Wed, Jun 16, 2010 at 9:28 AM
subject account notification
- Show quoted text -
http://newmommies.com/zx.htm
Malicious Link
Reply Reply to all Forward
from enigmasoftware.com support
@enigmasoftware.com>
hide details 11:30 AM (4 hours ago)
Reply to all
to @enigmasoftware.com
date Wed, Jun 16, 2010 at 11:30 AM
subject account notification
- Show quoted text -
http://dunhams.ca/zx.htm
Malicious Link
Reply Forward


Notice of appearance in Court #000215737 - Mozilla Thunderbird
File Edit View Go Message Tools Help
Reply Reply All Forward More
From State Court <adam.kaiser@c1291.colo.hc.ru>
Subject Notice of appearance in Court #000215737
09/08/2015 10:37 PM
To bizdev@rackspace.co.uk
Notice to Appear,
This is to inform you to appear in the Court on the September 14 for your case hearing.
Please, prepare all the documents relating to the case and bring them to Court on the specified date.
Note: If you do not come, the case will be heard in your absence.
You can review complete details of the Court Notice in the attachment.
Yours faithfully,
Adam Kaiser,
District Clerk.
1 attachment: Notice_to_Appear_000215737.zip 3.6 KB
Save
Notice_to_Appear_000215737.zip 3.6 KB

You Got Mail

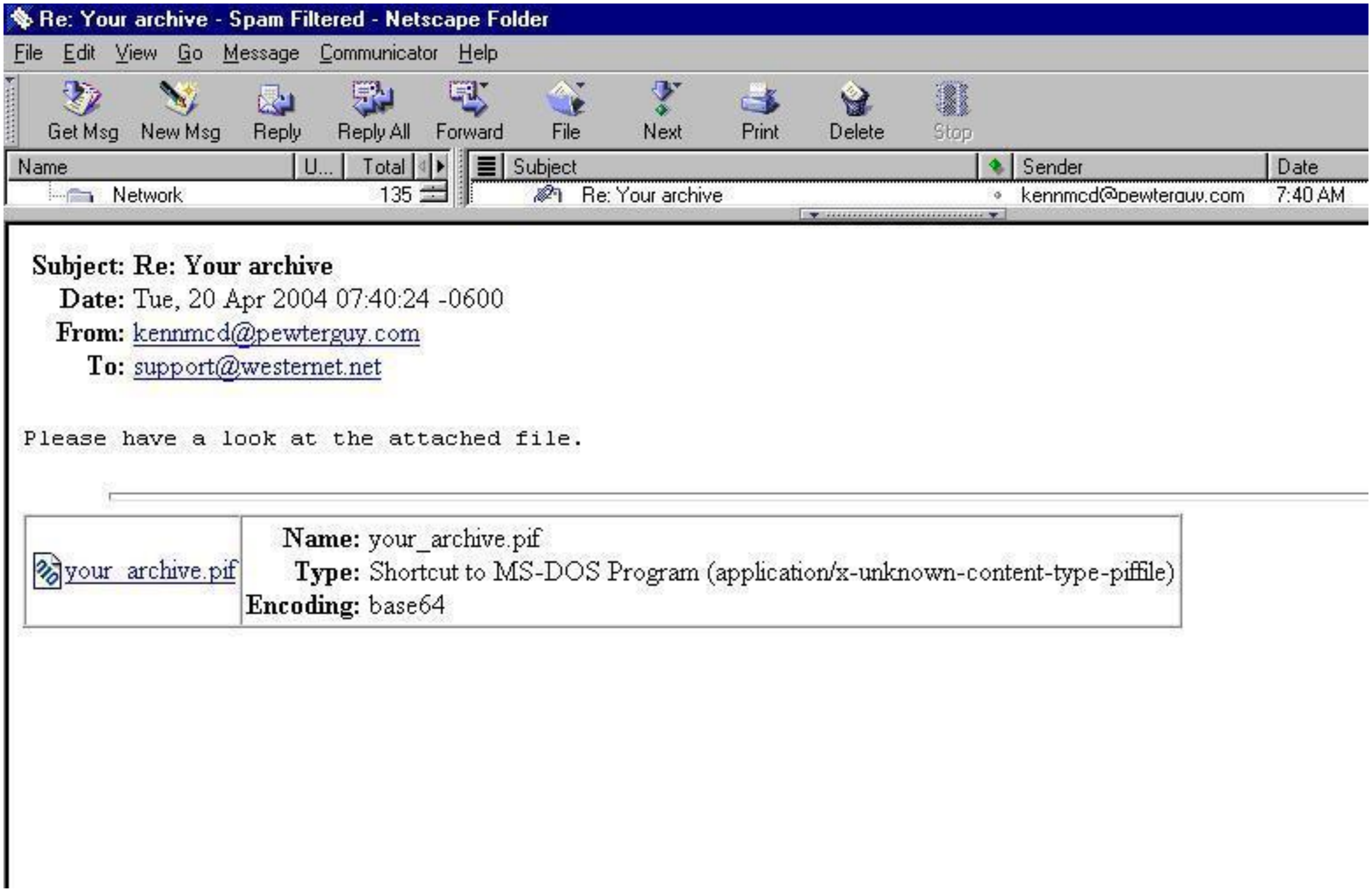
Receiving a malicious executable or link over email

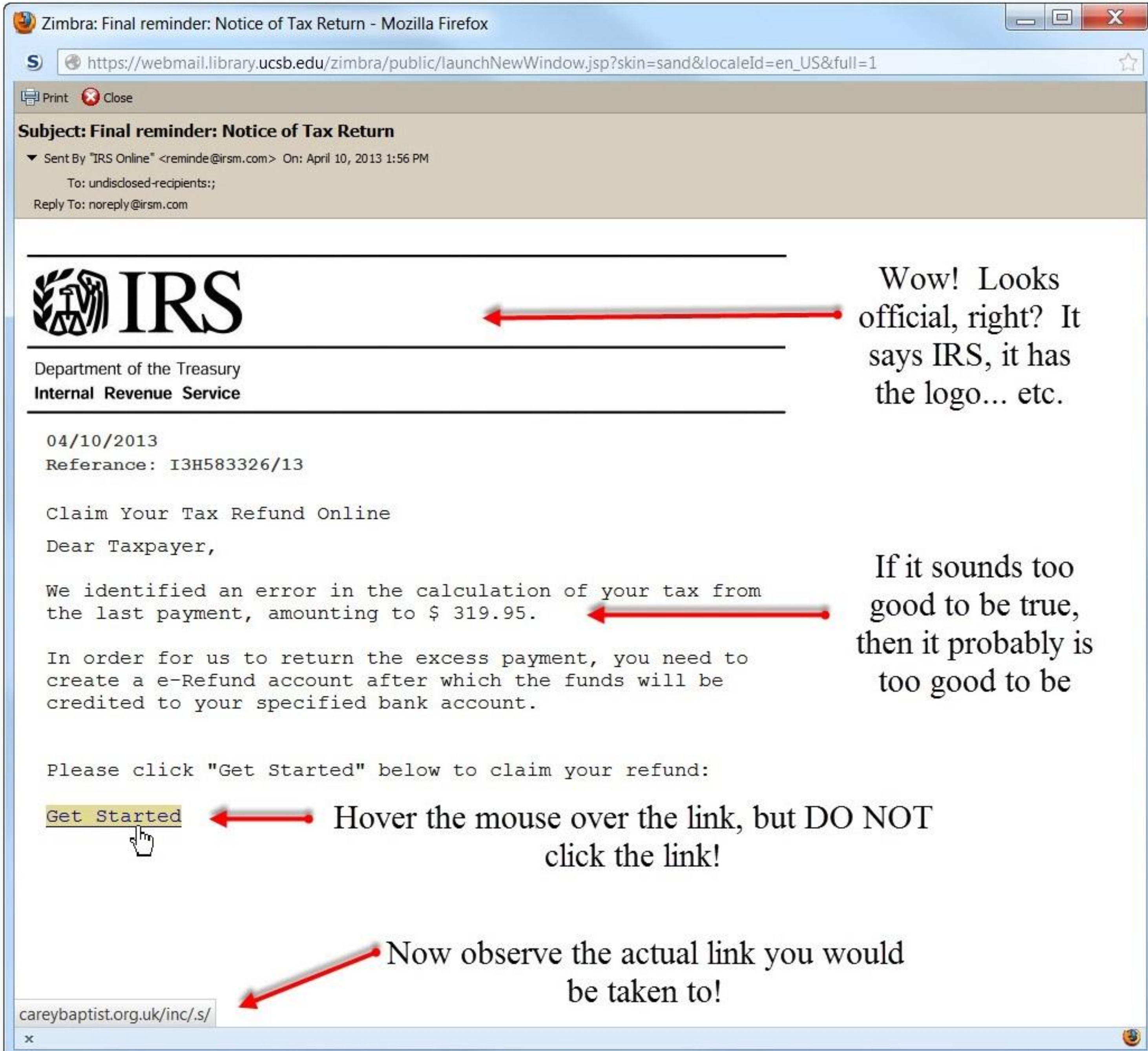
The obvious

- A malicious executable as an attachment
  - Usually compressed
- A link to a website serving malicious executables

Less obvious

- **The attachment has non-executable extension (e.g., .gif), but will be executed when opened**
  - **The file magic number is used instead of the extension**
- **Two file extensions are used and system hides known extensions**
  - **Example: “Image.gif.exe”**
- **HTML emails with “hidden” URLs**


Re: Your archive - Spam Filtered - Netscape Folder
File Edit View Go Message Communicator Help
Get Msg New Msg Reply Reply All Forward File Next Print Delete Stop
Name U... Total
Network 135
Subject Sender Date
Re: Your archive kennmcd@pewterguy.com 7:40 AM
Subject: Re: Your archive
Date: Tue, 20 Apr 2004 07:40:24 -0600
From: kennmcd@pewterguy.com
To: support@westernet.net
Please have a look at the attached file.
your_archive.pif
Name: your_archive.pif
Type: Shortcut to MS-DOS Program (application/x-unknown-content-type-piffile)
Encoding: base64


Zimbra: Final reminder: Notice of Tax Return - Mozilla Firefox

https://webmail.library.ucsb.edu/zimbra/public/launchNewWindow.jsp?skin=sand&localeId=en_US&full=1

Print Close

Subject: Final reminder: Notice of Tax Return

Sent By "IRS Online" <reminde@irsm.com> On: April 10, 2013 1:56 PM

To: undisclosed-recipients:;

Reply To: noreply@irsm.com

IRS

Department of the Treasury
Internal Revenue Service

04/10/2013
Referance: I3H583326/13

Claim Your Tax Refund Online

Dear Taxpayer,

We identified an error in the calculation of your tax from the last payment, amounting to $ 319.95.

In order for us to return the excess payment, you need to create a e-Refund account after which the funds will be credited to your specified bank account.

Please click "Get Started" below to claim your refund:

Get Started

careybaptist.org.uk/inc/.s/

In All of The Above Cases…

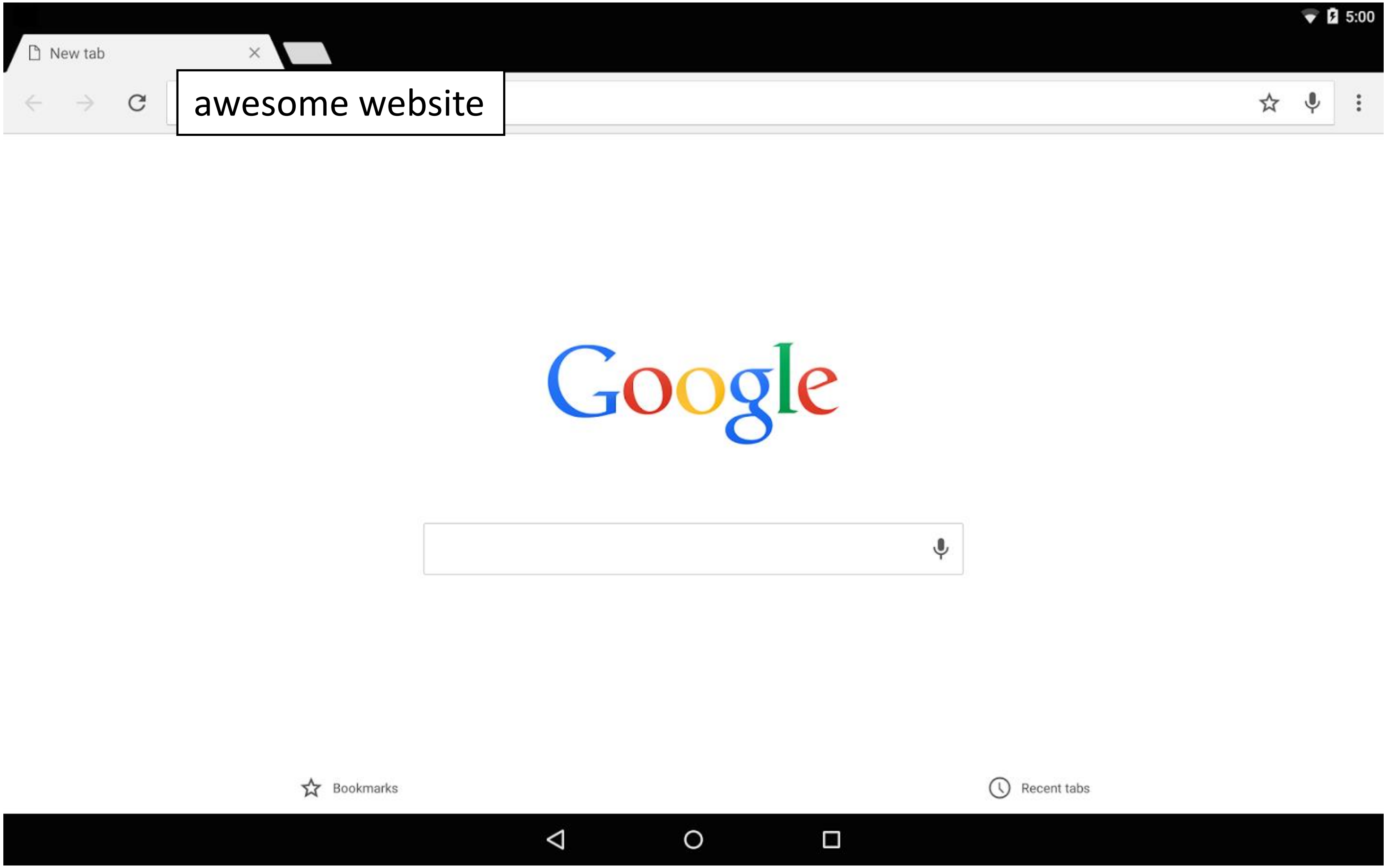
The malware is …

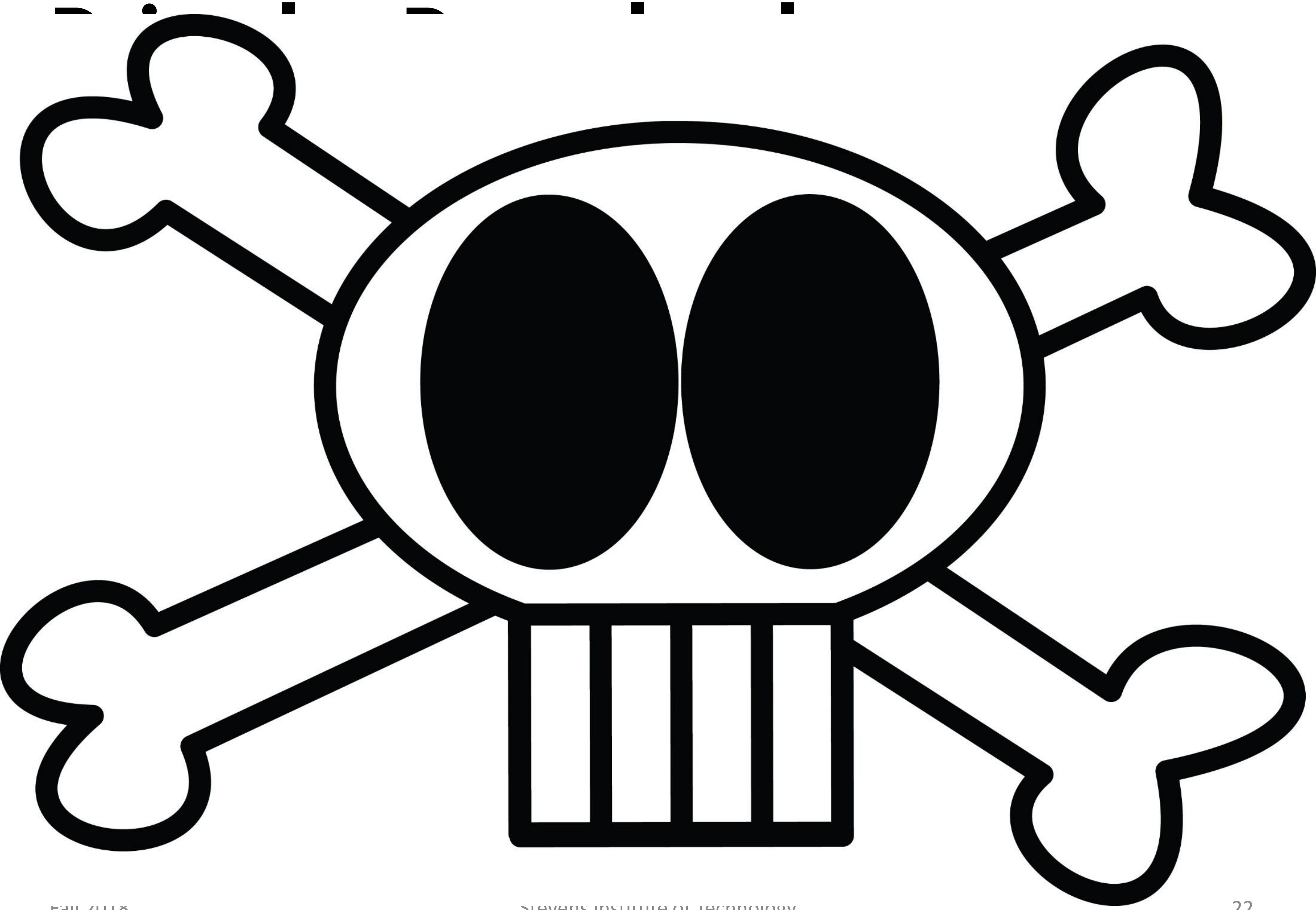
…the application

…embedded in the application

Users initiate download and execution of the malware

Drive-by-Downloads





Typical Drive-by-Download

User visits a website

↓

The website exploits a vulnerability in the browser

↓

The compromised browser downloads and installs malware

Drive-by-Downloads

No interaction is required beyond visiting the website, hence, drive-by

The visited domain does not have to be malicious

- For example, it could include a malicious ad or other 3rd party content

Exploited vulnerabilities

- Memory corruptions vulnerabilities in browser or other native-code component (e.g., Flash)
- Plugins that execute code over the network (e.g., ActiveX)

Exploits Delivered Over Email

Email includes document carrying exploit

- PDF, doc, etc.

The document exploits a vulnerability in the application used to opend the

Or it can include malicious interpreted code

- Scripts, macros, etc.

Exploits Delivered Over Email

Email includes malicious document

↓

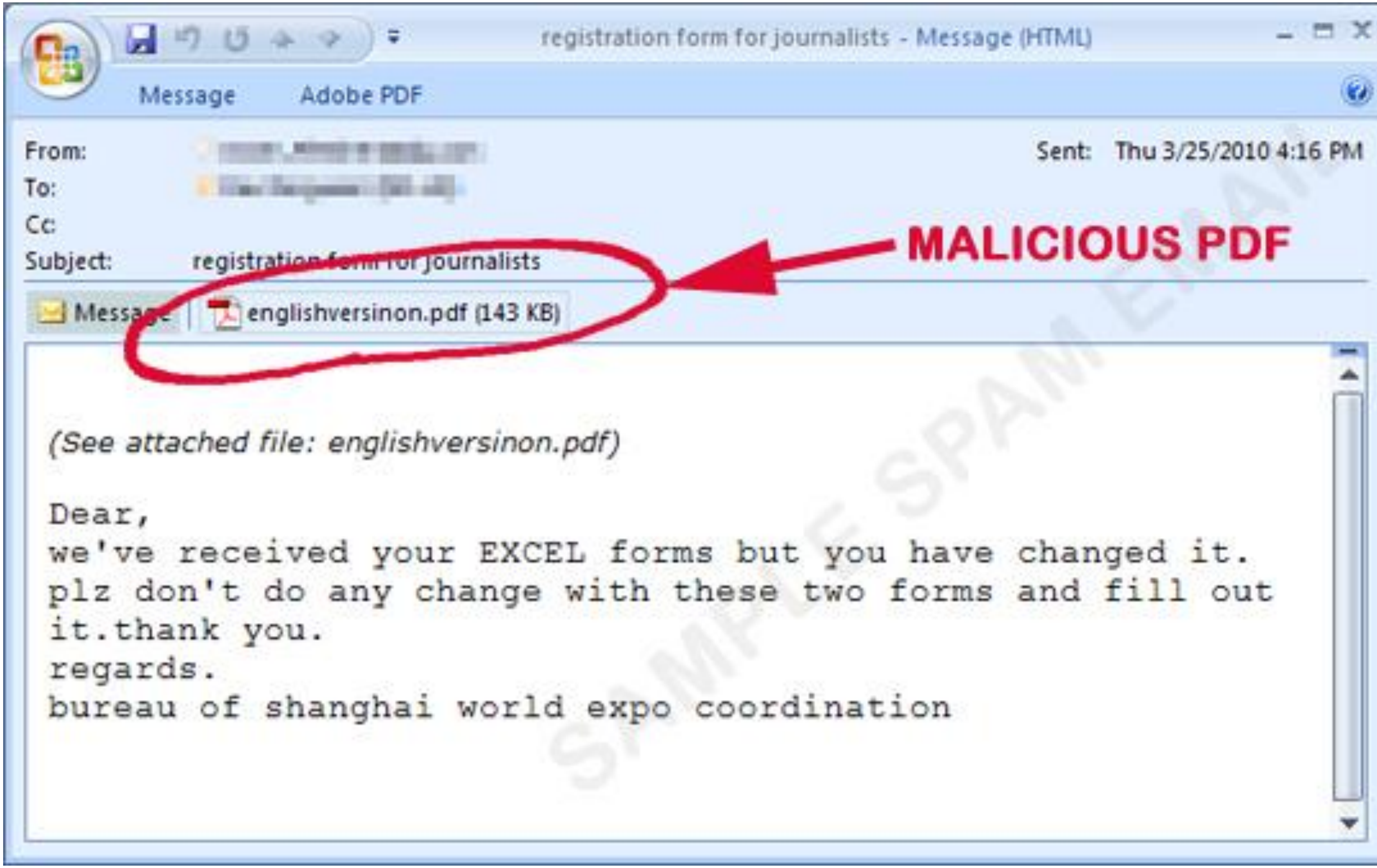
The user opens the attachment

↓

Attachment exploits a vulnerability in the application used

↓

The compromised application downloads and installs malware


registration form for journalists - Message (HTML)
Message
Adobe PDF
From:
To:
Cc:
Subject: registration form for journalists
Sent: Thu 3/25/2010 4:16 PM
MALICIOUS PDF
englishversinon.pdf (143 KB)
(See attached file: englishversinon.pdf)
Dear,
we've received your EXCEL forms but you have changed it.
plz don't do any change with these two forms and fill out
it.thank you.
regards.
bureau of shanghai world expo coordination

Exploits Against Servers

Connect to server port

↓

Exploit server vulnerability

↓

The compromised server downloads and installs malware

Computer Worms

Malware that exploits software vulnerabilities in client or server programs

Main difference from viruses: it actively seeks out vulnerable hosts to infect → self replicates

Propagation vectors include:

- The network (network shared, server vulnerabilities, email)
- Physical media (USB drives, CD, DVD data disks)

Network Target Discovery

Random

- **Generate random IP addresses and probe them**

Local subnet

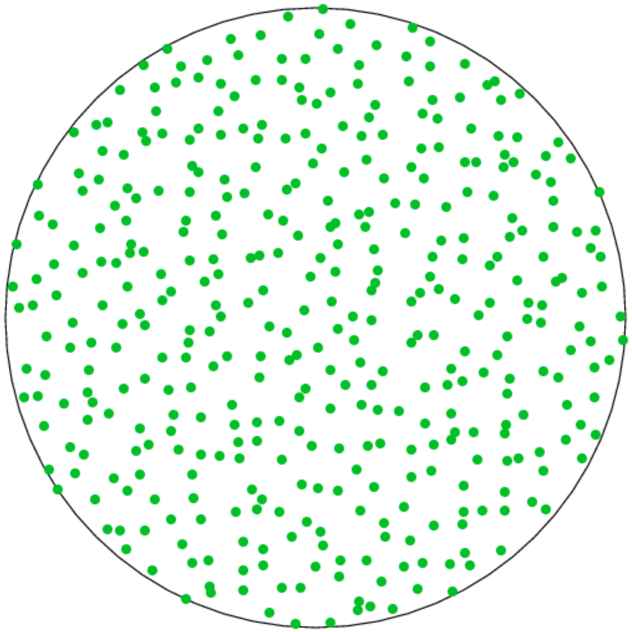
- First scan the local subnet for targets
  - Bypasses network-periphery defenses
  - Faster than random

Hit-list

- Compile a long list of potentially vulnerable machines
- Parts of the list are distributed to "siblings" of the worm
- Fast!

Topological

- Learn new targets from infected hosts



Network Target Discovery

Random

- Generate random IP addresses and probe them

Local subnet

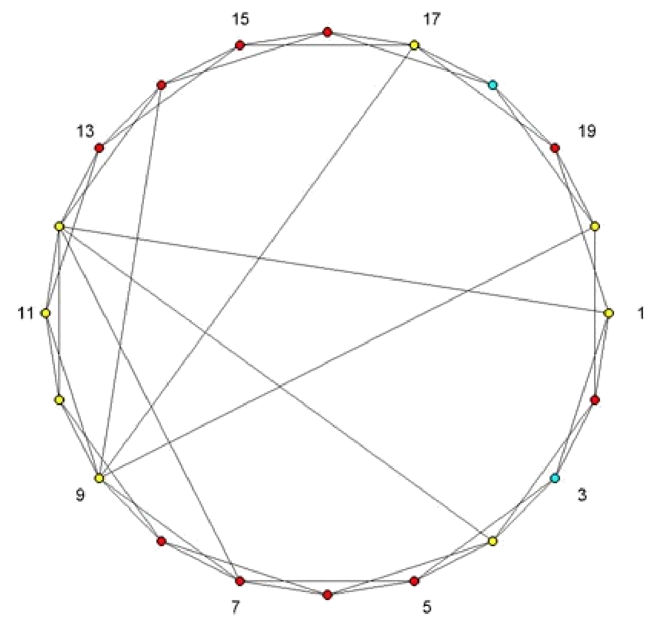
- **First scan the local subnet for targets**
  - **Bypasses network-periphery defenses**
  - **Faster than random**

Hit-list

- Compile a long list of potentially vulnerable machines
- Parts of the list are distributed to "siblings" of the worm
- Fast!

Topological

- Learn new targets from infected hosts



Network Target Discovery

Random

- Generate random IP addresses and probe them

Local subnet

- First scan the local subnet for targets
  - Bypasses network-periphery defenses
  - Faster than random

Hit-list

- **Compile a long list of potentially vulnerable machines**
- **Parts of the list are distributed to "siblings" of the worm**
- **Fast!**

Topological

- Learn new targets from infected hosts

Network Target Discovery

Random
- Generate random IP addresses and probe them

Local subnet
- First scan the local subnet for targets
  - Bypasses network-periphery defenses
  - Faster than random

Hit-list
- Compile a long list of potentially vulnerable machines
- Parts of the list are distributed to “siblings” of the worm
- Fast!

Topological
- **Learn new targets from infected hosts**

Worms and Epidemiology

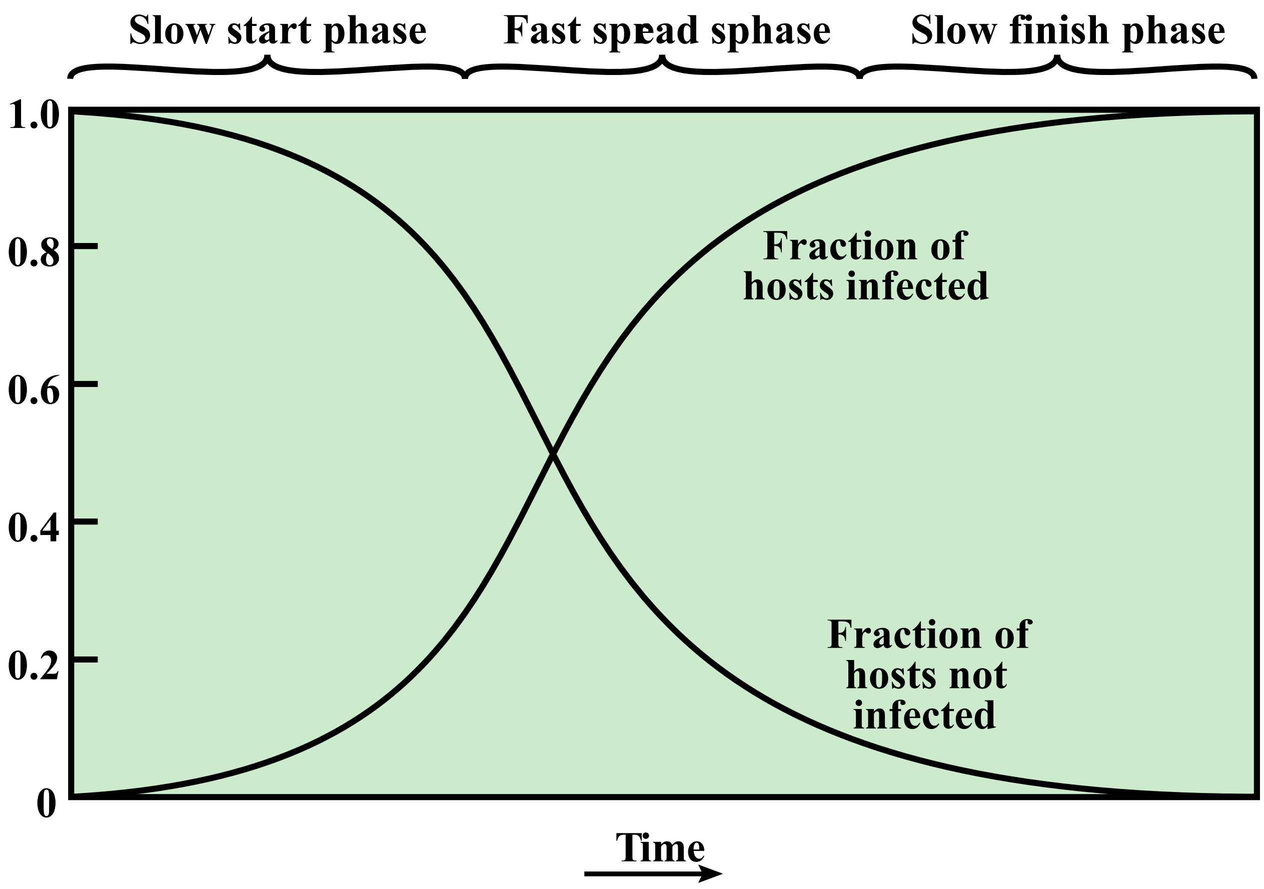
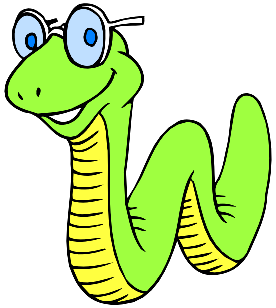


Figure 6.3 Worm Propagation Model

Case Study: Morris Worm



Earliest significant worm infection

Released by Robert Morris in 1988

Facts about 1988:

- The Internet consisted of about 60,000 computers
- They were connected using TCP/IP
- Mostly run BSD Unix

Vectors of Infection

Attempted to crack local password file to use login/password to logon to other systems

Exploited a bug (buffer overflow) in the finger daemon

Exploited a trapdoor in the debug option of the remote process that receives and sends mail

- DEBUG function enables one to run a program at the host
  - Added by the author of sendmail to remotely troubleshoot systems
- Who needs shellcode! Morris sent a C program in an email, compiled it, and executed it to fetch the rest of the worm from the already infected host

Target Acquisition

Internet too sparse for random scanning

On SUN and VAX architectures other hosts can be found by looking in

- /etc/hosts.equiv
- /.rhosts
- .forward
- ~/.rhosts
- Routing tables
- End points of point-to-point interfaces

What It Never Did

Gain privileged access

Destroy data

Install time bombs or backdoors

Brute force the root account

Flaws

Never checked if a host is already infected

Routines would exit with errors while leaving a copy of the virus running:

- When multiple instances of the worm attempted to infect a clean host concurrently
- When multiple instances of the worm attempted to infect and already compromised host
- When a machine is heavily loaded (remember it is actually compiling)

Infection rate was proportional to number of instances of the worm running on a host

Bad Target Finding

Worm checked for telnet or rsh ports to determine if a host is running UNIX

- Some systems only run sendmail!

It did not utilize DNS

First Reactions

Administrators cut off the sendmail service

Big mistake!

It shut off communication channels necessary to fix the error

The worm had alternate ways to propagate

The Author

Robert Morris was a student at Cornell University

He was identified, tried, and convicted in 1990

- 3 years of probation, fines, etc.

Received a PhD from Harvard

Now a professor at MIT

More Examples: Code Red

13 July 2001 – CodeRed: Buffer overflow in Microsoft IIS

```
GET /default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN%u9090%u6858%ucbd3%u7801%u
9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u9090%u8190%u00c3%u0003%u8b00%u531b%
u53ff%u0078%u0000%u00=a HTTP/1.0
```

Defaced affected website:

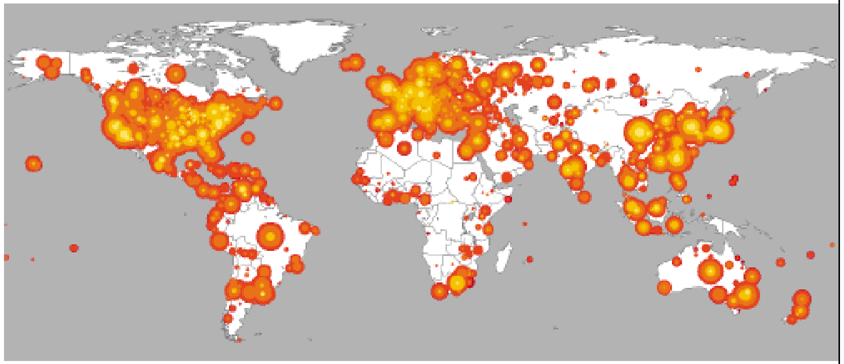
```
HELLO! Welcome to http://www.worm.com! Hacked By Chinese!
```

Days 1–19: propagate through random scanning

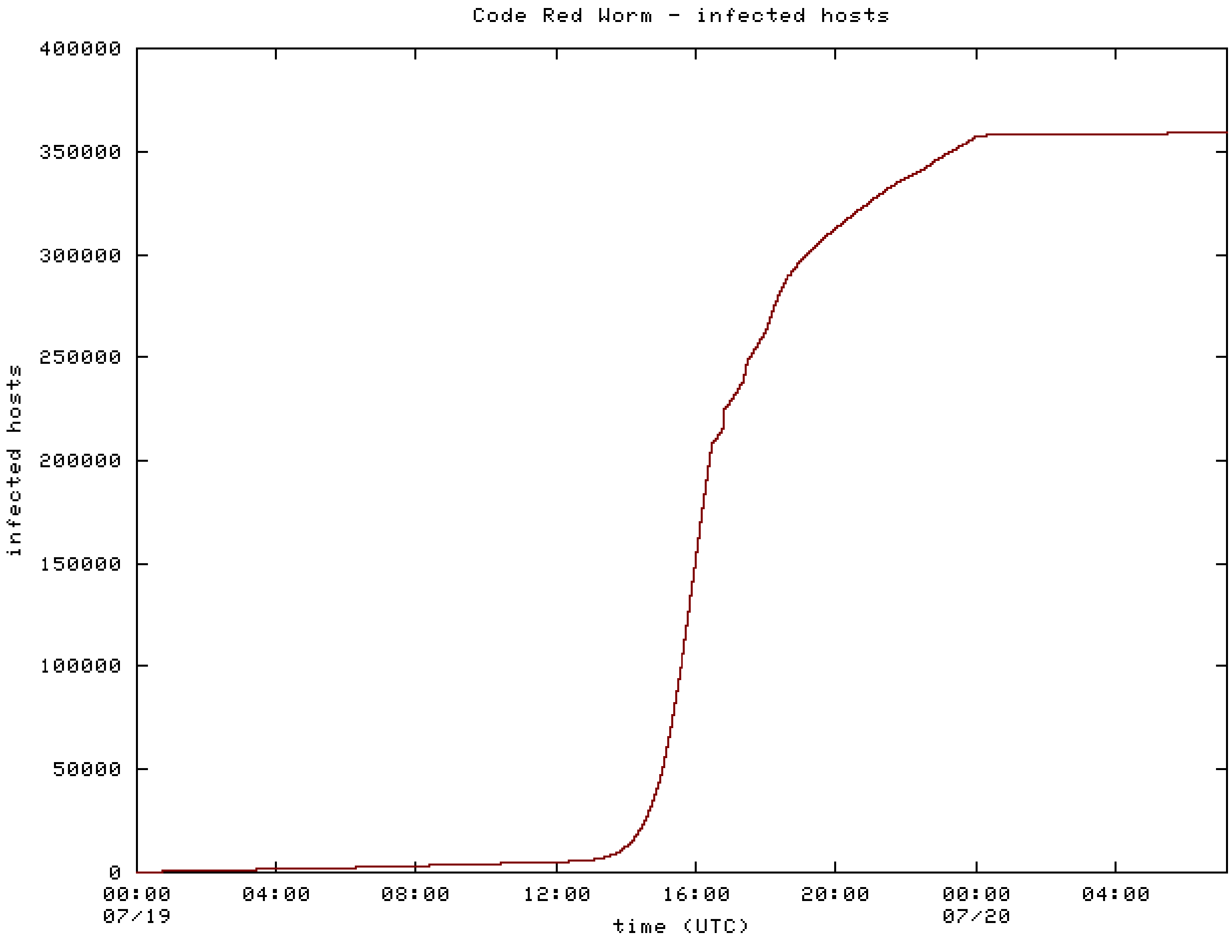
Days 20–27: DoS attack against www.whitehouse.gov

4 August 2001 – CodeRed II

Localized scanning



Code Red Propagation

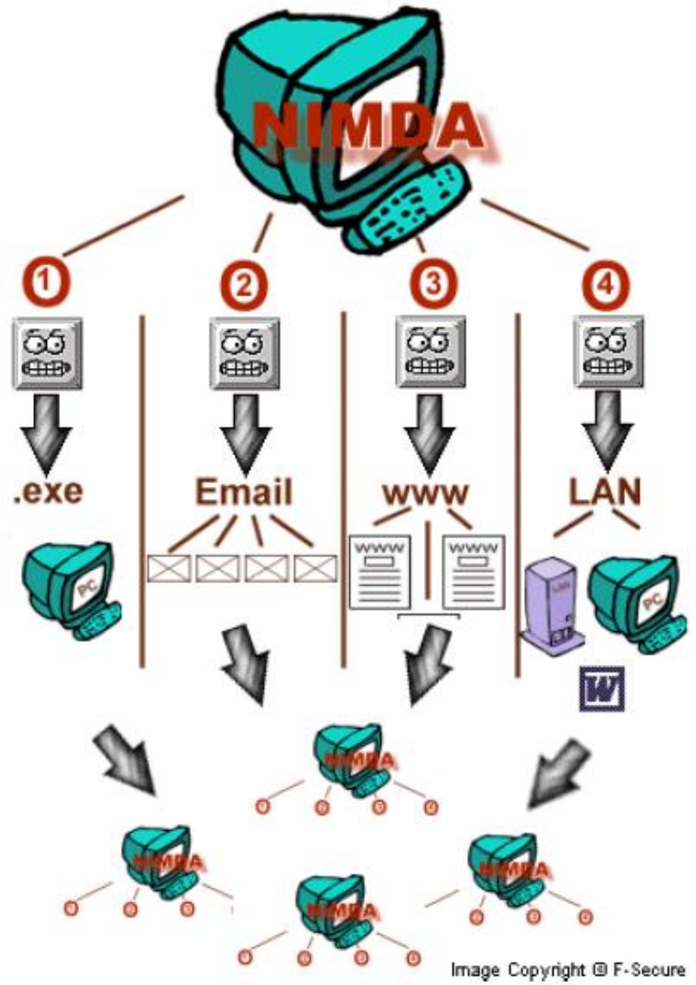


Example: Nimda Worm

18/9/2001 – Nimda

Many infection vectors

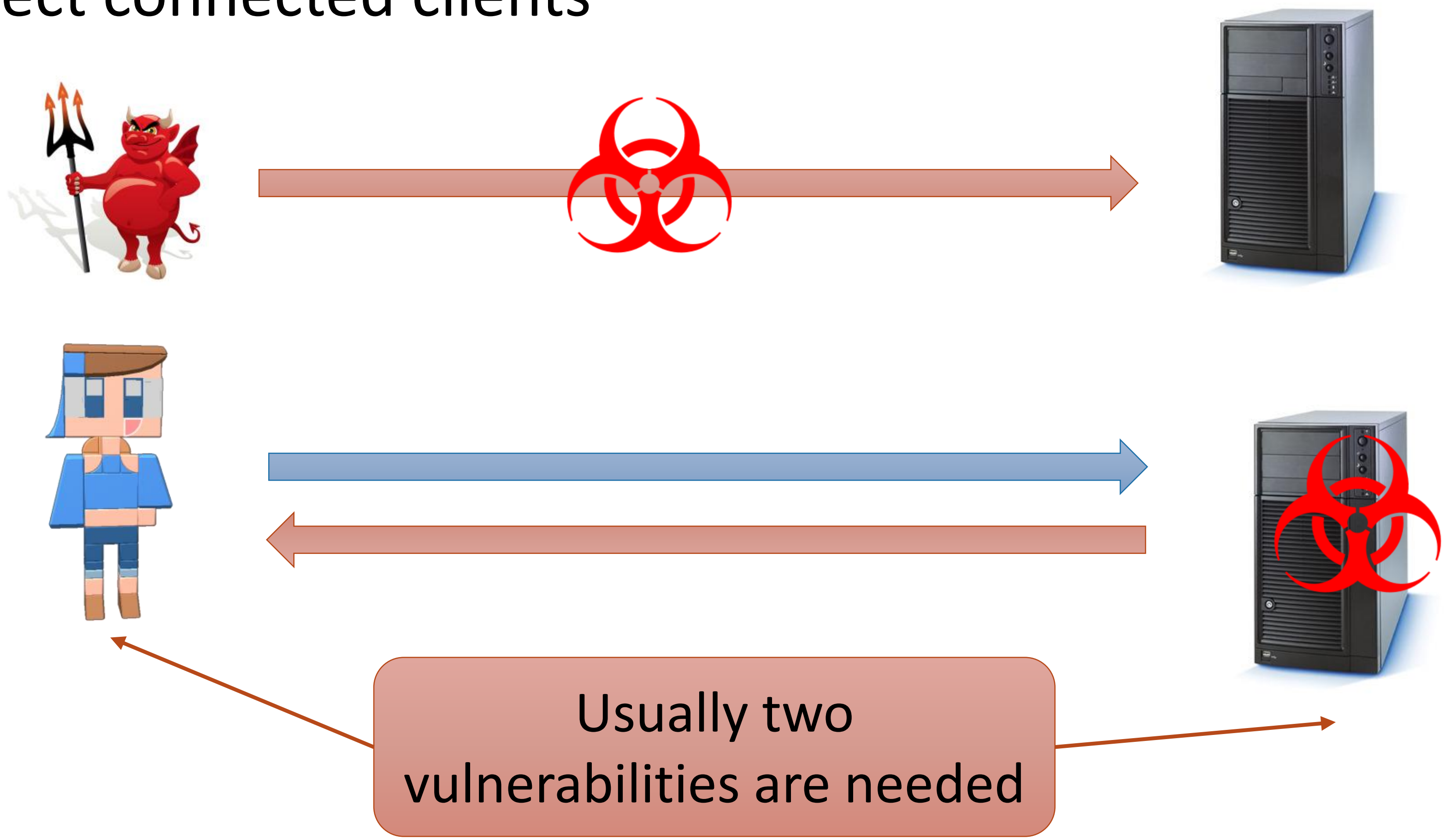
- Code Red IIS buffer overflow
- Bulk email to harvested addresses from victim host
- Open network shares
- Infect visitors of compromised web sites
- Microsoft IIS 4.0/5.0 directory traversal vulnerabilities
- Backdoors left behind by the Code Red II and Sadmind/IIS worms



Passive Propagation

Infect service and wait for clients to connect

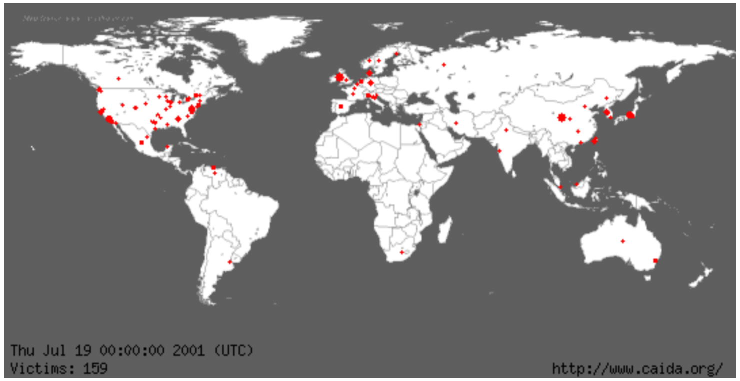
Infect connected clients



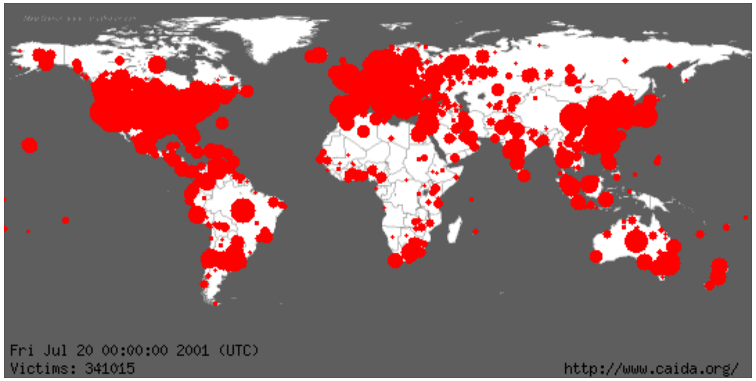
Recent Worm Attacks

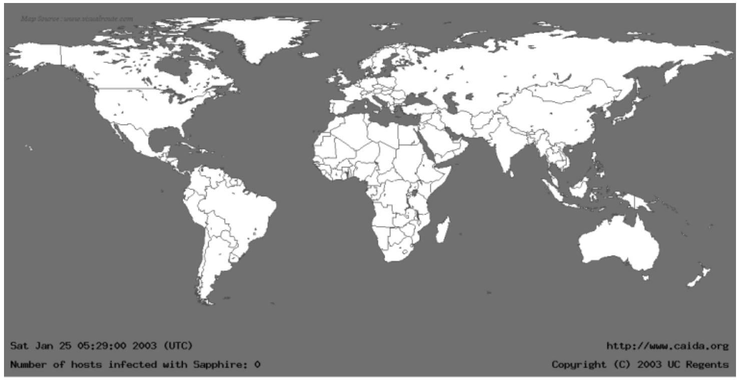
Melissa	1998	e-mail worm first to include virus, worm and Trojan in one package
Code Red	July 2001	exploited Microsoft IIS bug probes random IP addresses consumes significant Internet capacity when active
Code Red II	August 2001	also targeted Microsoft IIS installs a backdoor for access
Nimda	September 2001	had worm, virus and mobile code characteristics spread using e-mail, Windows shares, Web servers, Web clients, backdoors
SQL Slammer	Early 2003	exploited a buffer overflow vulnerability in SQL server compact and spread rapidly
Sobig.F	Late 2003	exploited open proxy servers to turn infected machines into spam engines
Mydoom	2004	mass-mailing e-mail worm installed a backdoor in infected machines
Warezov	2006	creates executables in system directories sends itself as an e-mail attachment can disable security related products
Conficker (Downadup)	November 2008	exploits a Windows buffer overflow vulnerability most widespread infection since SQL Slammer
Stuxnet	2010	restricted rate of spread to reduce chance of detection targeted industrial control systems

Fast Worms

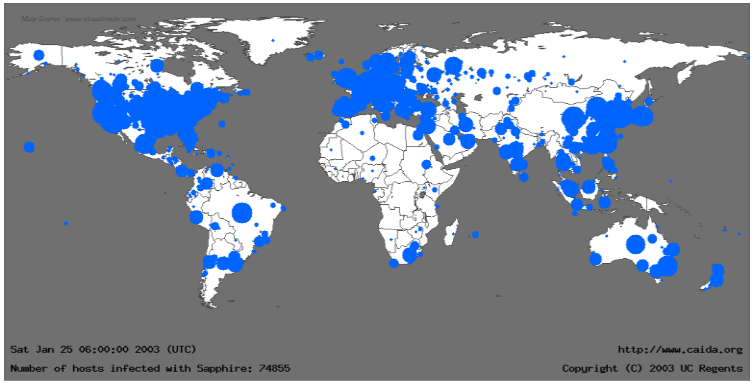


July 19, 2001
spread of
CODE RED
in **24 hours!**





Jan 29, 2003
spread of
SLAMMER
in **30 minutes!**



Payloads

Payloads

Click fraud

Bank fraud

Phishing

Spamming

Spreading malware

Data theft

Identity theft

Extortion

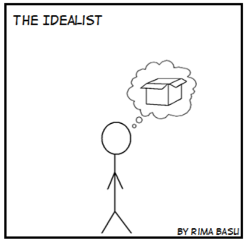
DDoS

Espionage

Sabotage

Sources/Motivation

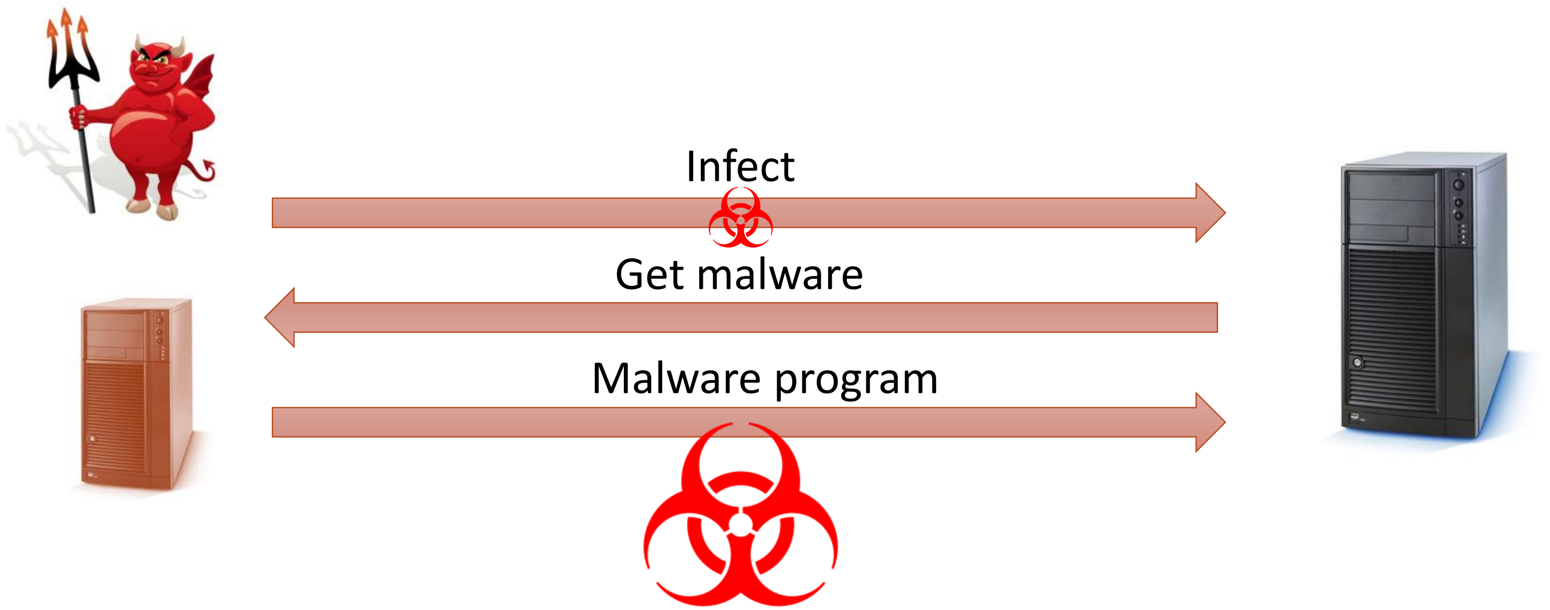
- Organized crime
- Criminals
- Politically motivated attackers
- Organizations that sell their services to companies and nations
- National government agencies



Two Stage Payloads

The more complex the payload the larger the malware

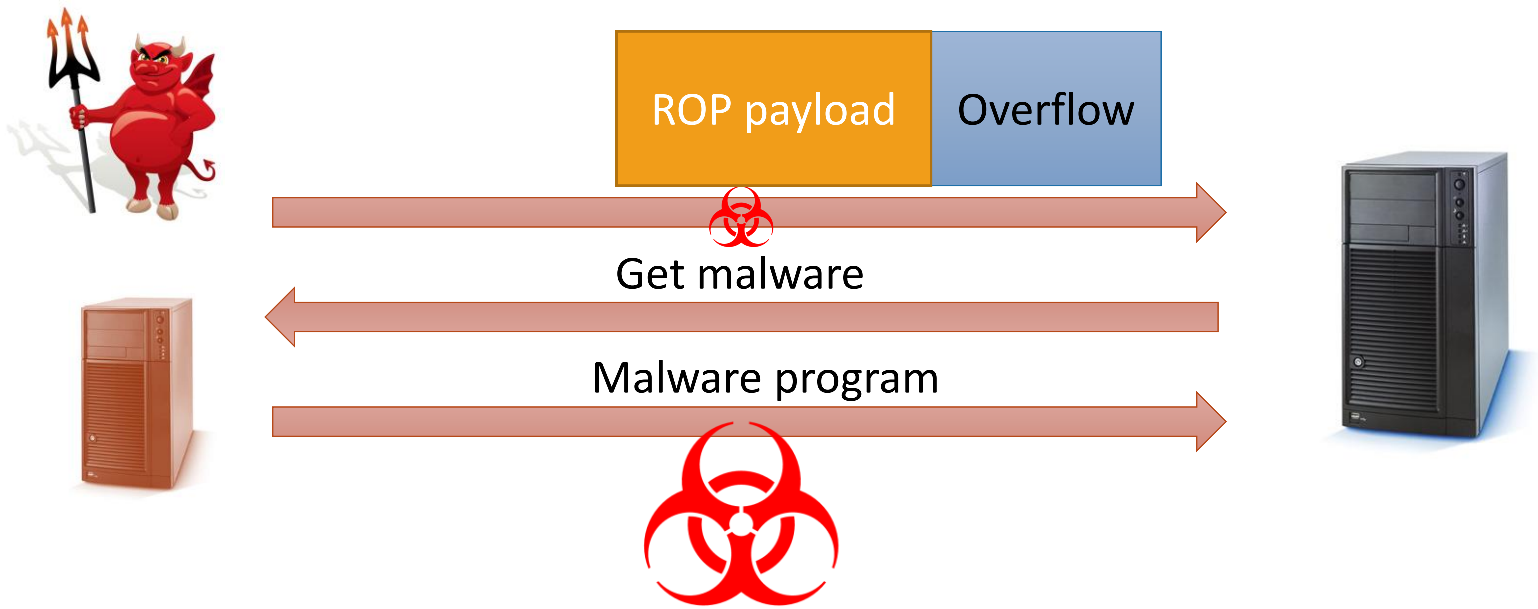
Maybe too large to send along with the infection



Example: Two-stage Attack with ROP

Use a ROP payload to download and execute malware

- system(“wget …”) -> system(“malware …”)



Exploit Kits

Exploit Kits

Initially the development and deployment of malware required considerable technical skill by software authors

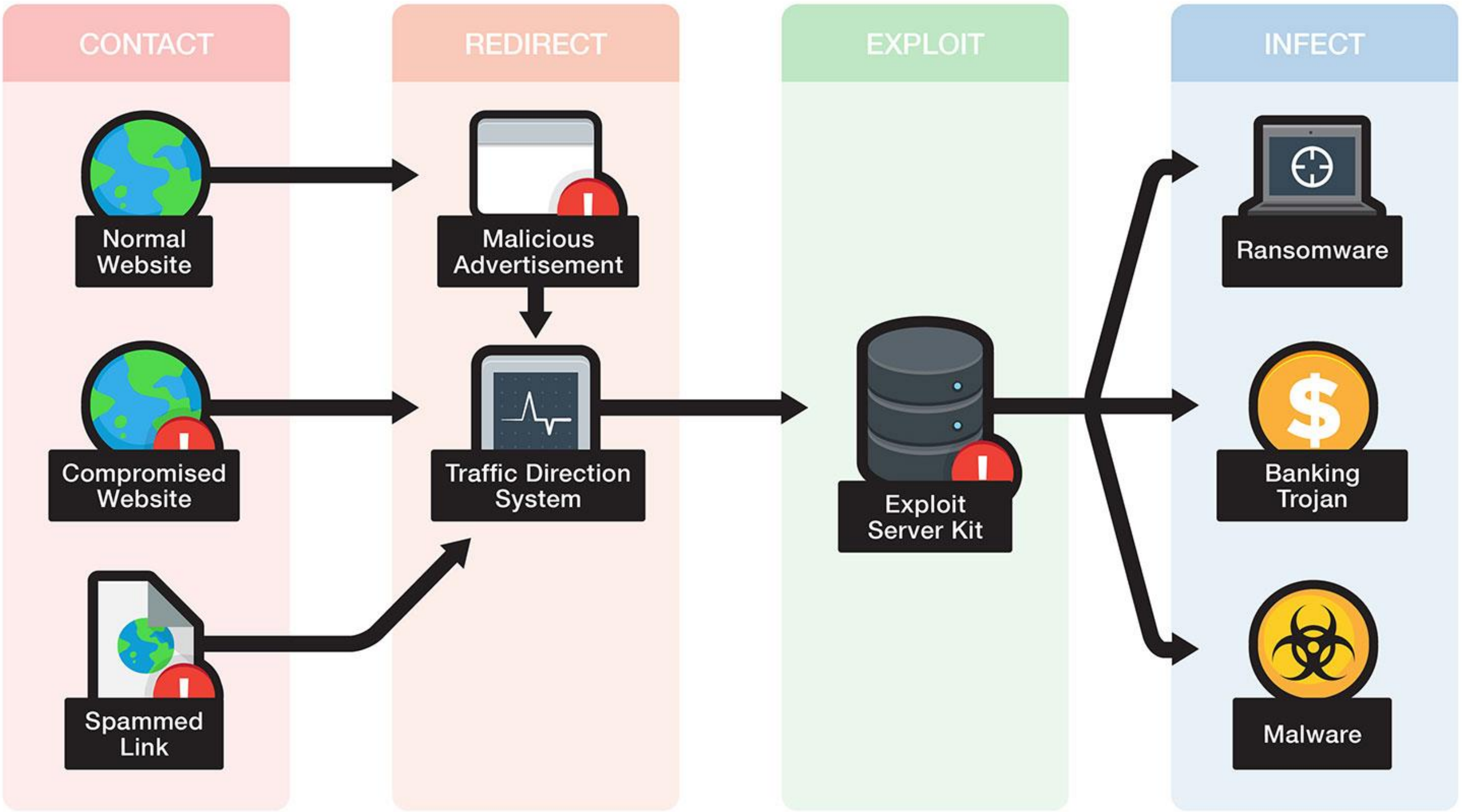
The development of virus-creation toolkits in the early 1990s and then more general attack kits in the 2000s greatly assisted in the development and deployment of malware

- Toolkits are often known as “crimeware”
- Widely used toolkits include: Zeus, Blackhole, Sakura, Phoenix

Include a variety of propagation mechanisms and payload modules that even novices can deploy

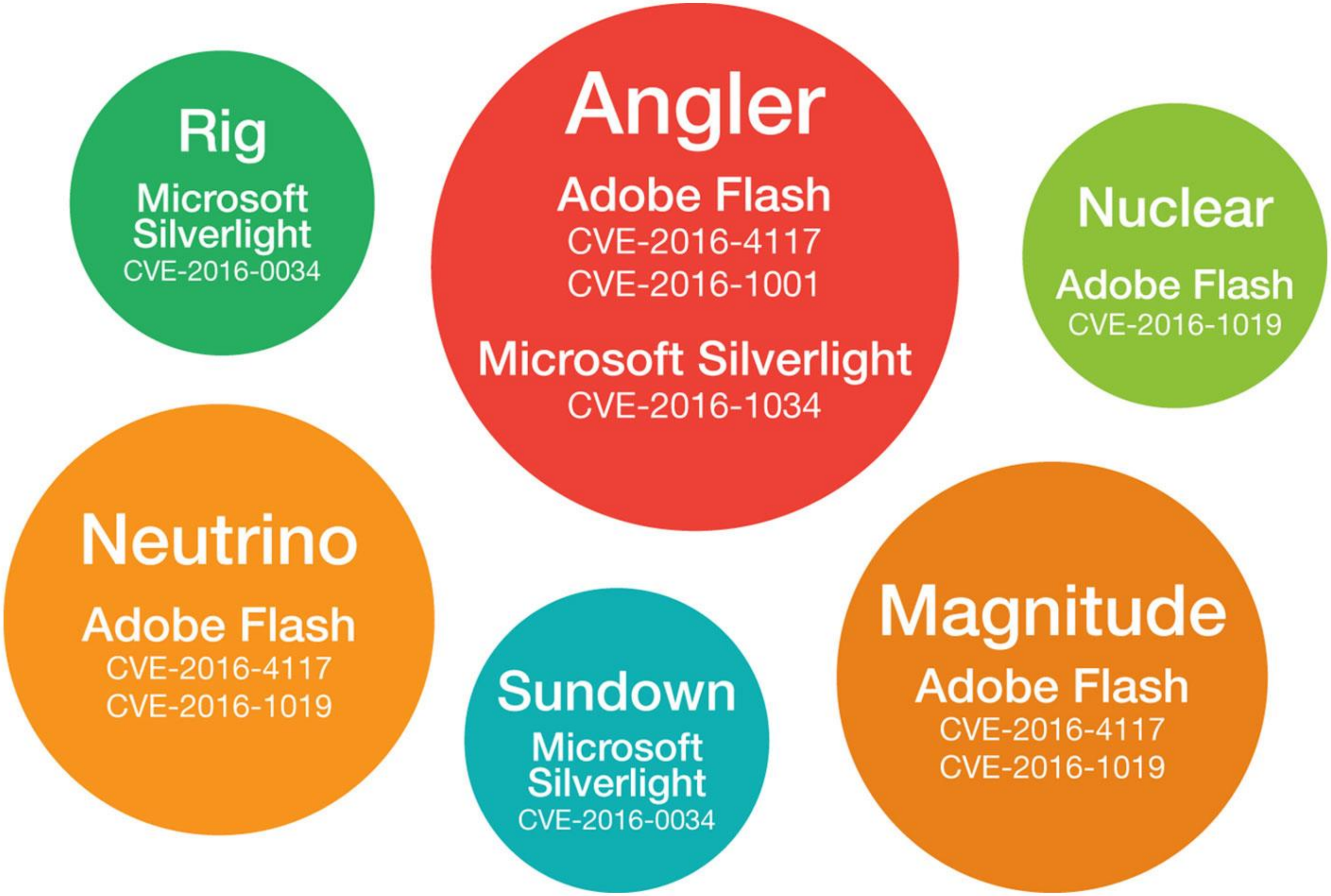
Variants that can be generated by attackers using these toolkits creates a significant problem for those defending systems against them

Exploit Kit Infection Chain



http://www.trendmicro.com/vinfo/us/security/definition/exploit-kit

Exploit Arsenal in Popular Kits



http://www.trendmicro.com/vinfo/us/security/definition/exploit-kit

Malware Analysis

Why Do We Analyze Malware?

To access damage

To identify infection signs for discovering other compromised hosts

To determine how it can be removed

To generate a “vaccine” for the infection

Types of Analysis

Static

Analyze the code without executing it

Disassemble

Higher coverage

- Prone to obfuscation

Dynamic

Analyze the code while executing it

Monitor execution

Lower coverage, what I see is what is executed

Examples of extracted data from malware:

Instructions

Control flow graph

Function call graph

Invoked APIs

Disrupting Disassembly

Common obfuscation approaches:

Use indirect control transfers

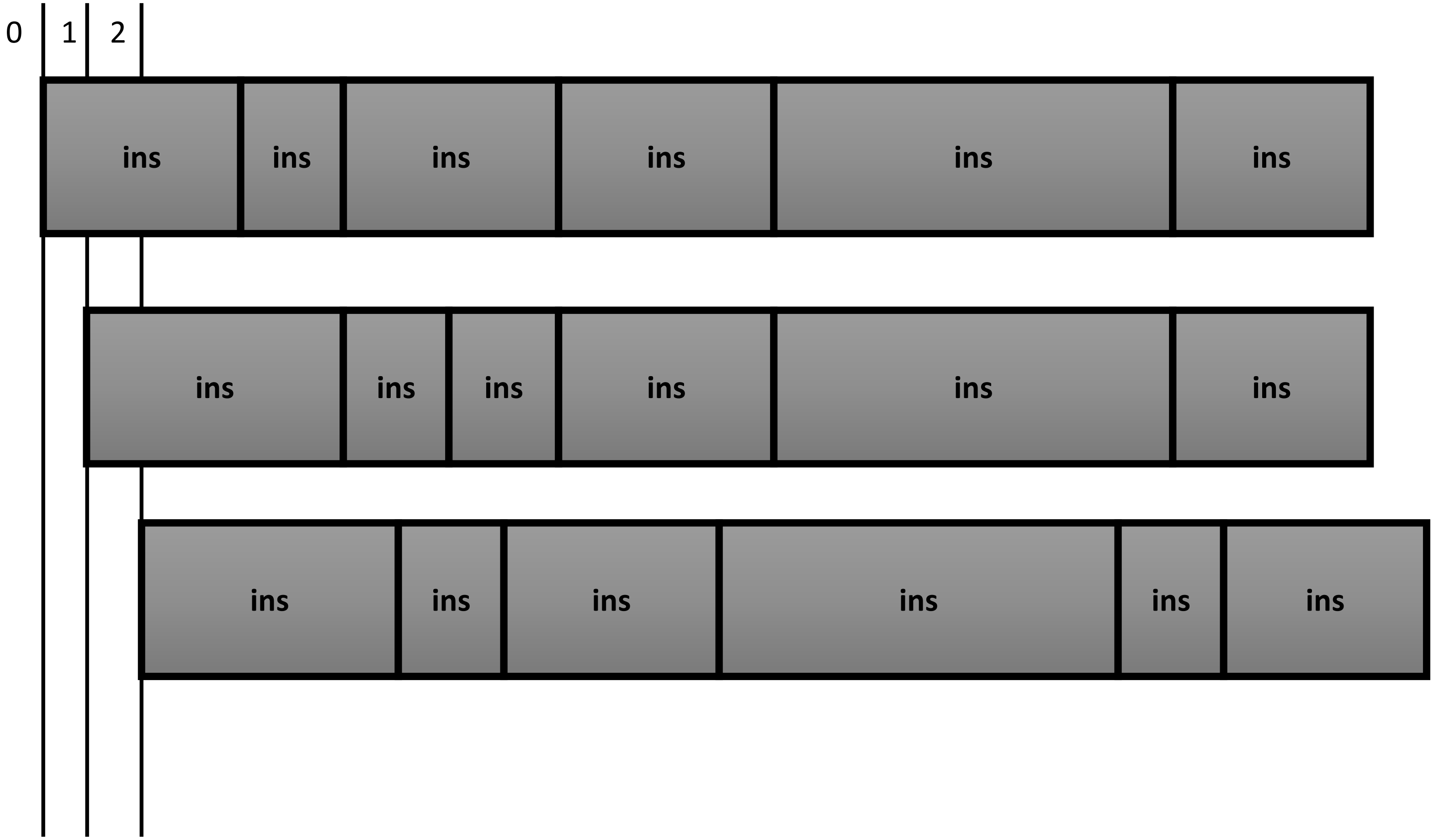
Overlapping instructions

- x86
- Dalvik -- http://www.dexlabs.org/blog/bytecode-obfuscation

Return-oriented programming

- https://www.usenix.org/system/files/conference/usenixsecurity13/sec13-paper_wang-updated-8-23-13.pdf

Different Code Based on Where you Start From



Combining both static and dynamic analysis is frequently necessary

Detecting Virtual Machines and Debuggers

If a program is not run natively on a machine, chances are high that it

- is being analyzed (in a security lab)
- scanned (inside a sandbox of an Antivirus product)
- debugged (by a security specialist)

Modern malware detect execution environment to complicate analysis

- Hide or alter its functionality
- Attempt to breakout

Detecting the ~~Matrix~~ VM

Look for VM artifacts in processes, file system, and/or registry

- Special files and processes (e.g., VMtools)

Look for VM artifacts in memory

- How do you peak into memory?

Look for VM-specific virtual hardware

- Device names, device parameters (e.g., MAC address)

Look for VM-specific processor instructions and capabilities

- Extra instructions added by VM for guest-host communication
- VMs frequently don't support obscure instructions or have limited support for instructions
- `The red pill'

Remotely by inspecting IP packets

Detecting VMs with the Red Pill

Proposed by Joanna Rutkowska

Runs a single instruction

- SIDT → Store Interrupt Descriptor Table
  - Store IDT in memory

Facts and observations

- On VMware guest machines, the IDT is typically located at 0xffXXXXXX
- On VirtualPC guests, it is located at 0xe8XXXXXX
- On host operating systems, it is located lower than that, typically around 0x80ffffff (Windows) and 0xc0ffffff (Linux)

It is sufficient to look at the first byte of the address

- If it’s greater than 0xd0, you've got a virtual machine
- If it is less than or equal to 0xd0, you are in a real machine

Red Pill++

Other OS features can be obtained with other instructions

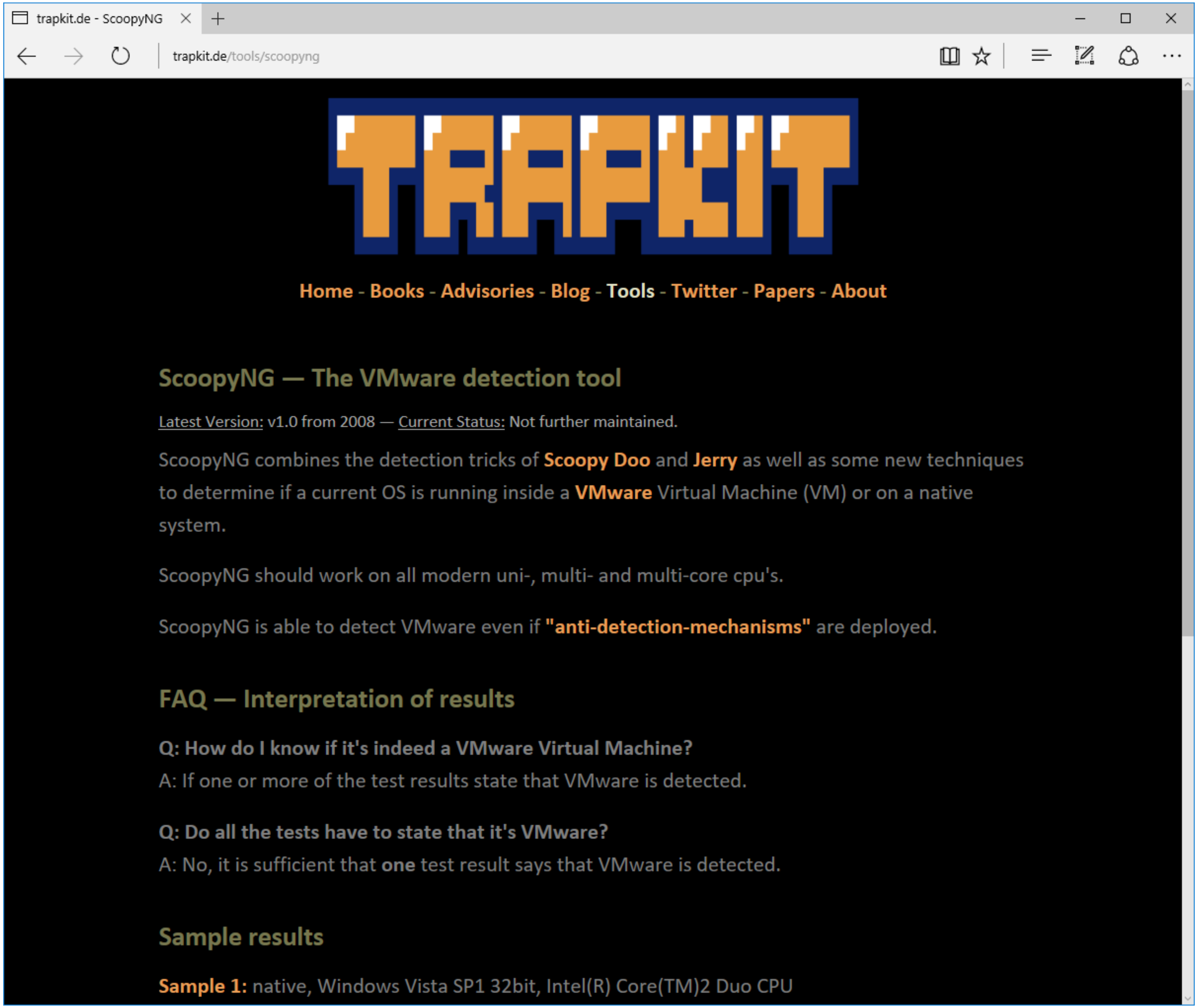
- The Global Descriptor Table (GDT), measured by the SGDT instruction
- The Local Descriptor Table (LDT), measured by the SLDT instruction

Anti-Debugging

Detecting a debugger is easy

- Windows: IsDebuggerPresent()
- Linux: ptrace()

```
if (ptrace(PTRACE_TRACEME, 0, NULL, 0) == -1)
  printf("traced!\n");
```



http://www.trapkit.de/tools/scoopyng/

Malware Detection

So you managed to collect:

- Instructions, function call graphs, control-flow graphs, APIs

For many years (static) malware signatures were used to identify them

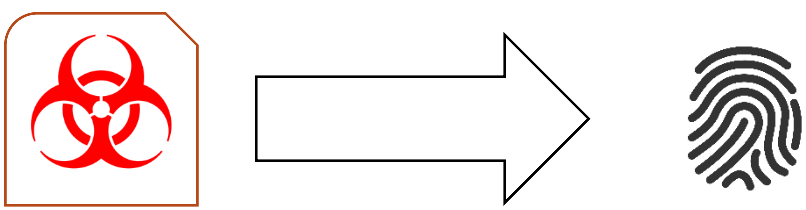
Malware Signatures

Signature → a sequence of bytes (usually code) that uniquely identifies software as malicious

Examples:

- The hash of the entire executable
- An expression that matches part of an executable

Static signatures have proven not to be a good way to identify malware



ce14856cf5fce0b4401
fac00d50e1ce82b641b
e19a2566121045c9bcd
30b4664

Polymorphism

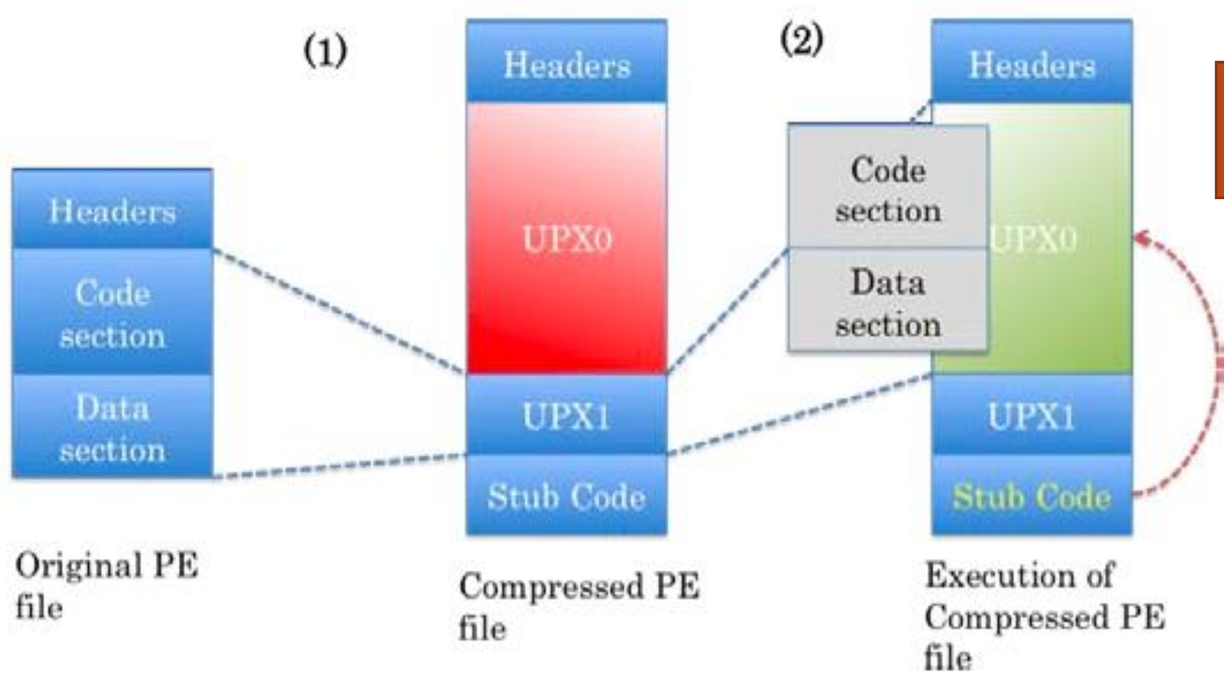
Avoid detection by changing (morphing) the bytes of the malware on each infection

- The actual payload can remain the same

Most popular method: encrypt or encode the payload using a different key for each infection

Creating Variants Using Packers

Examples: UPX, Aspack, FSG, PE Compact, …



Key within binary or fetched from the network

Weak packers can just compress the payload

Metamorphism

Create different "versions" of the program code that look different but have the same semantics (i.e., do the same thing)

Example techniques:

Dead-code insertion

Instruction reordering

Instruction substitution

Register substitution

…

Dead Code Insertion

5B	pop ebx
8D 4B 42	lea ecx, [ebx + 42h]
51	push ecx
50	push eax
90	**nop**
50	push eax
40	**inc eax**
0F 01 4C 24 FE	sidt [esp - 02h]
48	**dec eax**
5B	pop ebx
83 C3 1C	add ebx, 1Ch
FA	cli
8B 2B	mov ebp, [ebx]

```
5B 8D 4B 42 51 50 90 50 40 0F 01 4C 24 FE 48 5B 83 C3
1C FA 8B 2B
```

Instruction Reordering

```
5B                   pop ebx                          1
EB 09                jmp <S1>
                     S2:
50                   push eax
0F 01 4C 24 FE       sidt [esp - 02h]                 3
5B                   pop ebx
EB 07                jmp <S3>
                     S1:
8D 4B 42             lea ecx, [ebx + 42h]
51                   push ecx                         2
50                   push eax
EB F0                jmp <S2>
                     S3:
83 C3 1C             add ebx, 1Ch
FA                   cli                              4
8B 2B                mov ebp, [ebx]
```

```
5B EB 09 1 | 50 0F 01 4C 24 FE 5B EB 07 3 | 8D
4B 42 51 50 EB F0 2 | 83 C3 1C FA 8B 2B 4
```

Instruction Substitution

pop ebx	5B	pop ebx
lea ecx, [ebx + 42h]	8D 4B 42	lea ecx, [ebx + 42h]
push ecx	51	push ecx
push eax	50	push eax
push eax	**83 EC 04**	**sub esp, 04h**
	89 04 24	**mov [esp], eax**
sidt [esp - 02h]	0F 01 4C 24 FE	sidt [esp - 02h]
pop ebx	**83 04 24 1C**	**add [esp], 1Ch**
add ebx, 1Ch	5B	pop ebx
cli	FA	cli
mov ebp, [ebx]	8B 2B	mov ebp, [ebx]

```
5B 8D 4B 42 51 50 83 EC 04 89 04 24 0F 01 4C 24 FE
83 04 24 1C 5B 8B 2B
```

Behavior-based Malware Detection

Focus on the malicious behaviors demonstrated by malware instead

Click fraud	Identity theft
Bank fraud	Extortion
Phishing	DDoS
Spamming	Espionage
Spreading malware	Sabotage
Data theft	

Advanced Persistent Threats

Well-resourced attacks targeting high-value targets

Aim to gain persistent presence in a system or network

- By utilizing multiple attack vectors
- By employing **hiding** techniques

High profile attacks include Aurora, RSA, APT1, and Stuxnet

How Would you Hide?

Deception

Present a fake image of how things are (Potemkin village)

Methods:

Masquerade

Lie

- Modify programs to lie
- Modify the kernel to lie
- Modify VM to lie
- Modify the HW to lie?

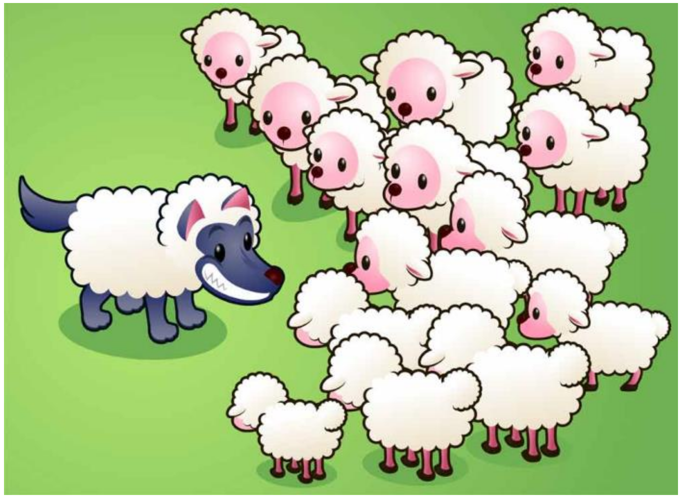
Masquerading

Assume unsuspecting filename

Stealthy operation

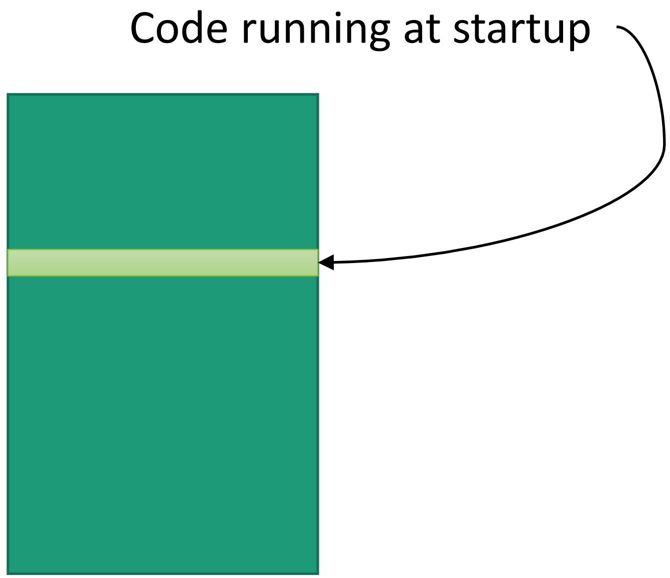
- Piggyback communications
- Small footprint

Infect a legitimate application

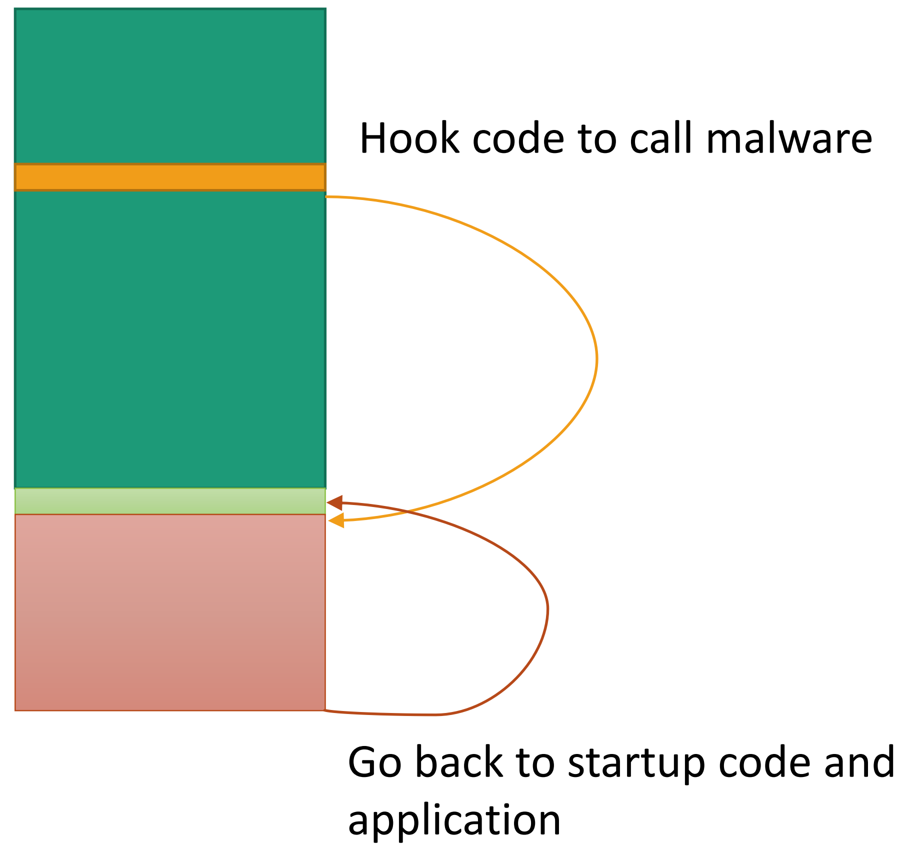


Application Infection

Before



After



Hiding Between Functions

Function 1:
…
…
return

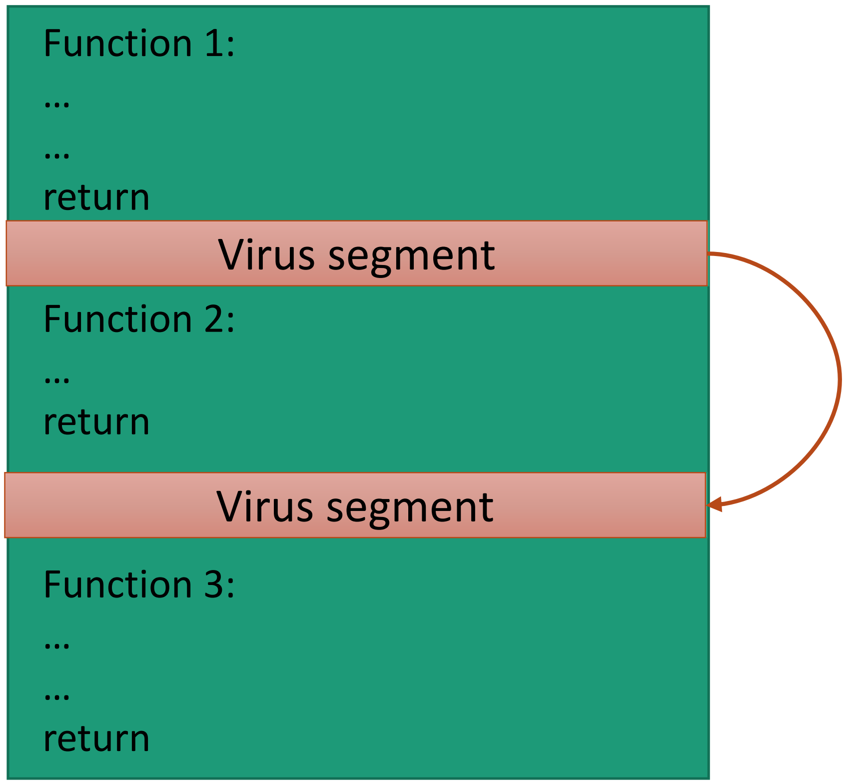
Gap left for alignment

Function 2:
…
return

Gap left for alignment

Function 3:
…
…
return

Hiding Between Functions



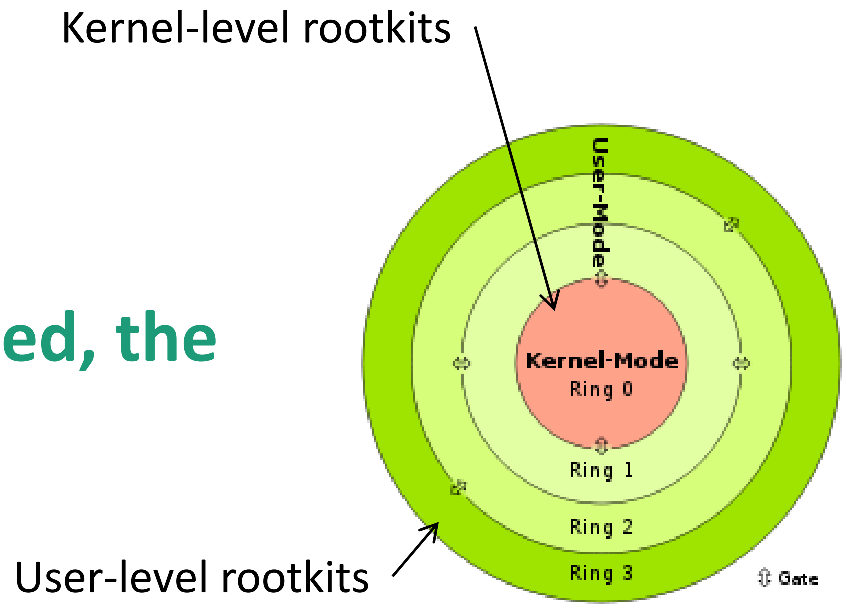
Malware jumps from segment to segment to compose its functionality

Lying

Rootkits → Malicious software designed to hide malware related data

- Files
- Processes
- Logins
- Network connections

The inner the level controlled, the better!



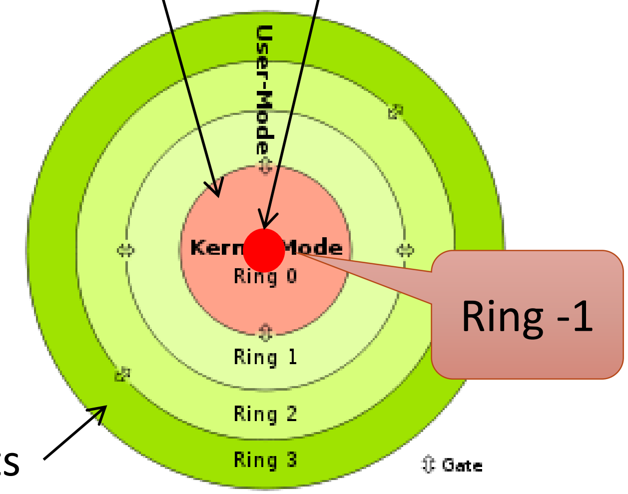
Lying

Rootkits → Malicious software designed to hide malware related data

- Files
- Processes
- Logins
- Network connections

- Hypervisor-level rootkits
- Bootkits
- Firmware-level bootkits

Kernel-level rootkits



User-level rootkits

The inner the level controlled, the better!

User-level Rootkits

Modify:

Utilities → ps, netstat, top, sshd

Applications → Alter behavior (e.g., modify Windows Explorer to hide a file)

API hooks → replace system calls, etc.

Kernel-level Rootkits

Modify or add:

Kernel code (*Phantasmagoria* adds instructions in system calls)

Kernel data structures (remove malware from process lists, *FU*)

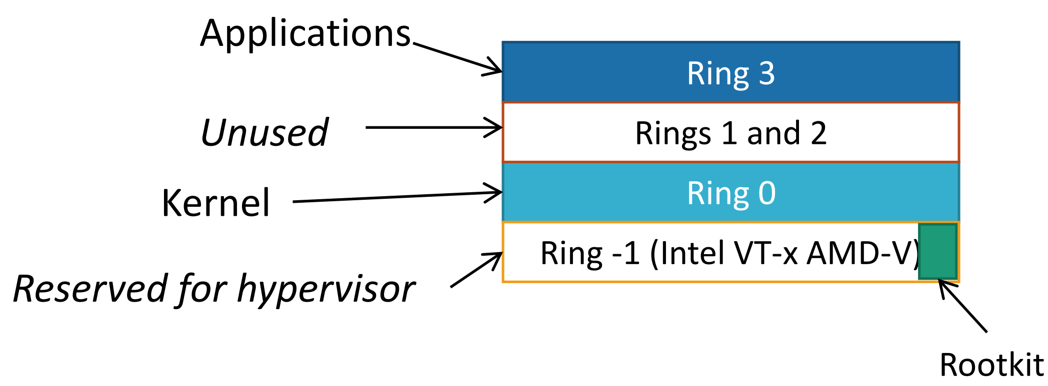
APIs (*Knark* adds entries in the proc file system, SuckIT adds new system calls)

Mostly implemented as ***Loadable Kernel Modules***

Hypervisor Rootkits

Runs with higher privilege than the kernel

Developed in academia (SubVirst, Blue Pill)



Firmware-level Rootkits

Firmware is the lowest-level of software that controls certain operations of hardware

Till recently the integrity of firmware was not checked

- Companies have started using **signed** firmware updates

Examples:

- Organized crime tampers with European card swipe device
  http://www.theregister.co.uk/2008/10/10/organized_crime_doctors_chip_and_pin_machines
- Attacks on BIOS anti-theft devices turn them into rootkits
  http://www.blackhat.com/presentations/bh-usa-09/ORTEGA/BHUSA09-Ortega-DeactivateRootkit-PAPER.pdf

(Some) Defenses

Check for file integrity → Tripwire, chkrootkit

Check for divergent results → checkps

Protecting hooks → system calls, internal kernel APIs

Code integrity checks → page-level signing

File Integrity Testing

Create checksums (hashes) of binaries on the system

Periodically check installed binaries vs stored checksums

Application signing

Challenges:

Storing the checksums out of reach

Keeping up with updates

Storing the tools out of reach!

Looking for Divergent Results

Run binaries and collect results

- ps, top, netcat

Collect results from other sources

- Directly access /proc filesystem

Compare results to find discrepancies

Challenges:

- Find other sources of information
- False alerts, system state is dynamic
- Storing the tools out of reach!

Monitor API Hooks

Store currently used, good set of hooks

Periodically read the values of hooks

Compare values to identify hooks being replaced

Challenges:

- Which APIs should be monitored
- False alerts, hooks can be placed for legitimate reasons
  - That's usually the problem with running multiple antivirus engines on your PC

Code-integrity Checking

Upon loading a page of code hash its contents

Periodically re-hash every page and check it against previously taken hash

Can be done

- By the kernel
- A hypervisor
- A coprocessor

Challenges:

- Storing the hashes out of reach
- Keeping up with code updates
- Code provenance/generated code
- Pages containing both code and data