Introduction to Information Security

—— Malicious Code

Dr. Tianlei HU

Associate Professor

College of Computer Science, Zhejiang Univ.

htl@zju.edu.cn

Agenda

- Type of Malicious Code —— Trojans, Viruses, Worms and Others
- Defenses of Malicious Code
- Botnets, Spam and DDoS

What is Malicious?

Malicious Logic:

- Set of instructions that cause site security policy to be violated

Example:

- Shell script on a UNIX system:
- cp /bin/sh /tmp/.xyzzy
- chmod u+s, o+x /tmp/.xyzzy
- rm ./ls
- ls $*
- Place in a program called "ls" and trick someone into executing it
- You now have a setuid-to-*them* shell!

Trojans, Viruses and Worms

Ghost on the Internet

Trojan Horse

Program with an *overt* purpose (known to the user) and a *covert* purpose (unknown to the user)

- Often called a Trojan
- Named by Dan Edwards in 1974 Anderson Report

Example: the previous script is Trojan horse

- Overt purpose: list files in a directory
- Covert purpose: create a setuid shell

Example: NetBus —— Designed for Windows NT system

The victim uploads and installs this

- Usually disguised as a game program or in one

Acts as a server, accepting and executing commands for remote administrator

- This includes intercepting keystrokes and mouse motions and sending them to the attacker.
- It also allows attackers to upload, and download files

Replicating Trojan Horse

Trojan horse that makes copies of itself

- Also called *propagating Trojan horse*

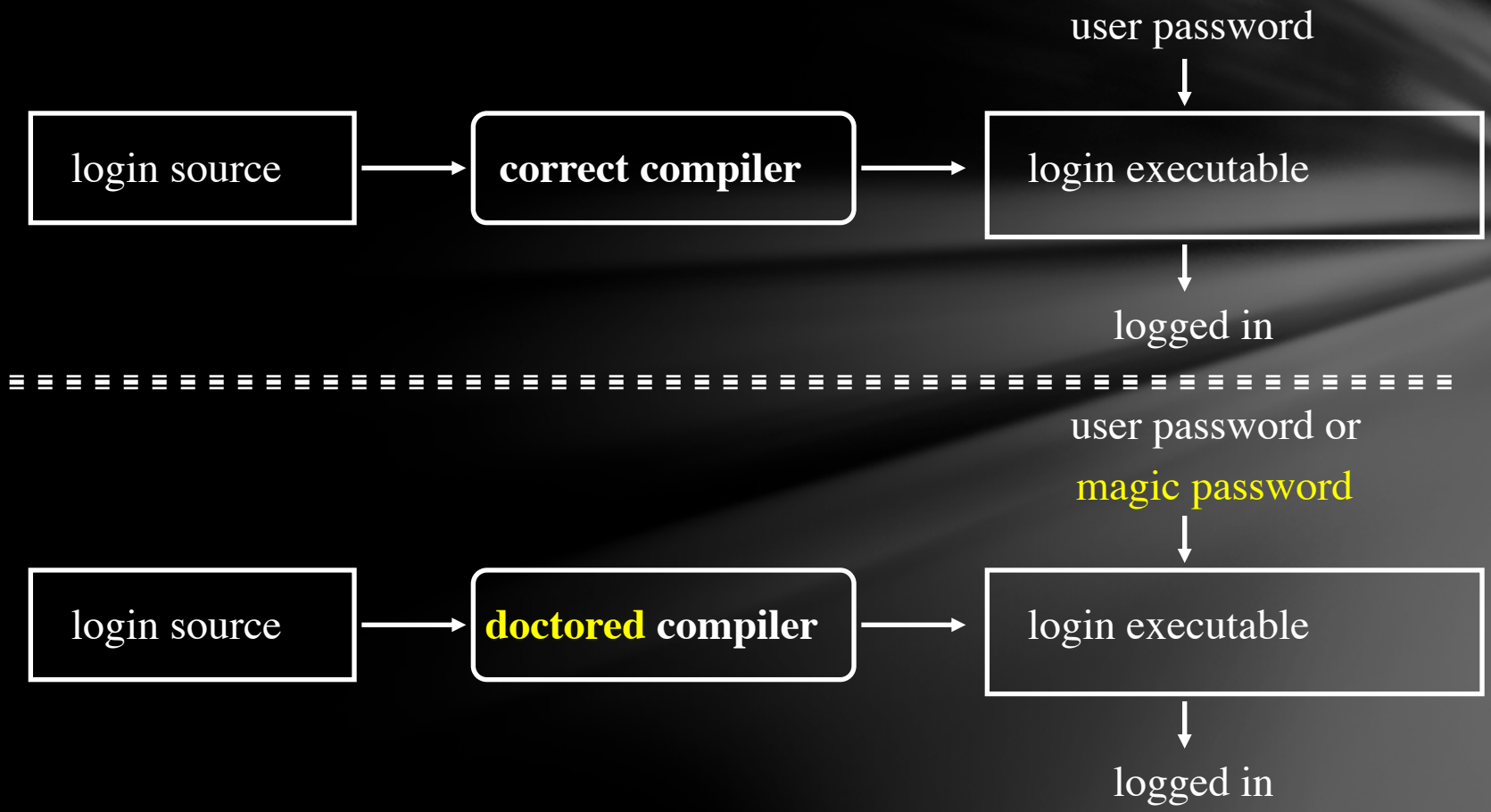
Hard to detect

- 1976: Karger and Schell suggested modifying the compiler to include a Trojan horse that copied itself into specific programs, including a later version of the compiler
- 1980: Thompson implements this

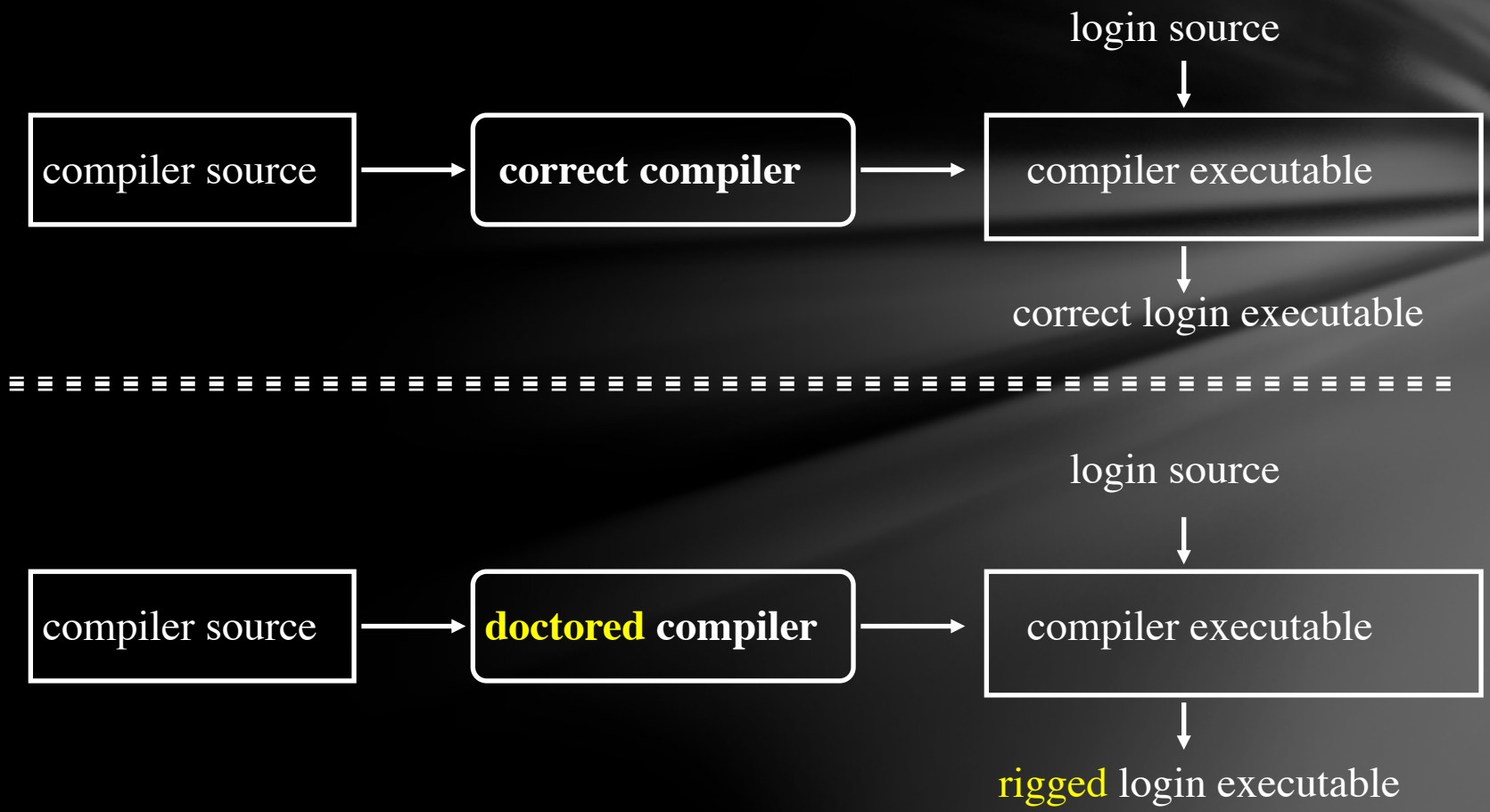
Thompson's Compiler

- Modify the compiler so that when it compiles *login*, *login* accepts the user's correct password or a fixed password (the same one for all users)
- Then modify the compiler again so that when it compiles a new version of the compiler, the extra code to do the first step is automatically inserted.
- Recompile the compiler
- Delete the source containing the modification and put the undoctored source back.

The Login Program



The Compiler



Comments on Thompson's Trojan

Great pains were taken to ensure the second version of the compiler was never released.

- We can only remove the trojan when a new compiler executable from a different system overwrote the doctored compiler.

The point: *no amount of source-level verification or scrutiny will protect you from using untrusted code*

- Also: having source code helps but does not ensure you're safe.

Trojan concealing—— Rootkit

- A rootkit is the binary executable code of a set of Trojan horse program
  - Main features: Stealth!
- A typical infection route:
  - Login into the system using password theft and dictionary attacks, etc.
  - gain root privileges by the buffer overflow in rdist, sendmail, loadmodule, rpc.ypupdated, lpr, or passwd and other procedures
  - Download rootkit from FTP, extract, compile, and install
- How to hide their presence?
  - Create a hidden directory
    - /dev/.lib, /usr/src/.poop, etc.
    - Usually using un-displayable characters in the directory name!!
  - Install system program of "black" version
    - Netstat, ps, ls, du, login
  - Make the checksum of the program equal to the original one

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

History of Rootkit

- For *nix:
  - Since 1980s, as wide use of *nix, rootkit became popular
  - Because of the relatively unified and open of *nix, the rootkit is easy to make
- For Windows:
  - In 1999, the first malicious rootkit for Windows appeared: NTRootkit
  - In 2003, another HackerDefender appeared.
- For others:
  - In 2009, for iOS
    - Considering personal privacy, never jailbreak IPHONE/IPAD ……
  - In 2010, Stuxnet for PLC（Programmable Logical Controllers）
- Famous incident——DRM backdoor of Sony BMG@2005
  - In 2005, a copyright protection software called Extended Copy Protection was used as part of the records of Sony BMG.
    - The software secretly installed a rootkit that restricted users' access to CDs.
    - After the Rootkit was publicly known, malicious software via the Rootkit appeared.
    - Sony released a patch to uninstall the rootkit but introduced more serious vulnerabilities.
      - At last, Sony recalled all CDs using the software.

Detect & Remove Rootkit/ Trojan

- To detect Rootkit/ Trojan:
  - Alternative trusted medium
  - behavior-based
  - signature-based
  - difference-based
  - integrity checking
  - memory dumps
- To remove Rootkit/ Trojan:
  - Removing Rootkit by hand is too hard for ordinary users, so tools provided by Security software manufacturers are widely used.
    - Some of the security software access files via Windows file system API, so it could not cope with the rootkit
    - For security, we have to access the raw file system to find and kill the Trojan
  - Many security experts insist that if antivirus software or other security software runs in an already infected OS, using them to remove Trojans is not dependable
  - So, the best/secure way to remove Rootkit/ Trojan is :
    - —— Reinstall the Operation System……
  - For PC, using the 360 security suites is safe enough.

Computer Virus

A program that inserts itself into one or more files and performs some action

- *The insertion phase* is inserting itself into a file
- *The execution phase* is performing some (possibly null) action

The insertion phase *must* be present

- Need not always be executed
- Lehigh virus inserted itself into the boot file only if not infected.

Pseudocode

```
beginvirus:
   if spread-condition then begin
      for some set of target files do begin
         if target is not infected then begin
            determine where to place virus instructions
            copy instructions from beginvirus to endvirus into target
            alter target to execute added instructions
         end;
      end;
   end;
   perform some action(s)
   goto beginning of infected program
endvirus:
```

History of Virus

Programmers for Apple II wrote some in 1980

- Not called viruses; they are very experimental

Fred Cohen

- A graduate student who described them in 1983
- The teacher (Adleman) named it a "computer virus."
- Tested idea on UNIX systems and UNIVAC 1108 system in 1984

Cohen's Experiments:

- UNIX systems: the goal was to get superuser privileges
  - Max time 60m, min time 5m, average 30m
  - A virus is small, so there is no degrading of response time
  - A virus is tagged so that it can be removed quickly
- UNIVAC 1108 system (MAC): the goal was to spread
  - Implemented simple security property of Bell-LaPadula
  - As writing is not inhibited (no *-property enforcement), viruses spread easily

Early Reports of Virus

Brain (Pakistani) virus (1986)

- Written for IBM PCs
- Alters boot sectors of floppies spread to other floppies

MacMag Peace virus (1987)

- Written for Macintosh
- Prints "universal message of peace" on March 2, 1988, and deletes itself

Duff's experiments (1987)

- Wrote a Bourne shell script virus
- A small virus placed on the UNIX system spread to 46 systems in 8 days

Highland's Lotus 1-2-3 virus (1989)

- Stored as a set of commands in a spreadsheet and loaded when the spreadsheet opened
- Changed a value in a specific row or column and spread it to other files.

Types of Viruses

- Boot sector infectors
- Executable infectors
- Multipartite viruses
- TSR viruses
- Stealth viruses
- Encrypted viruses
- Polymorphic viruses
- Macro viruses

Boot Sector Infectors

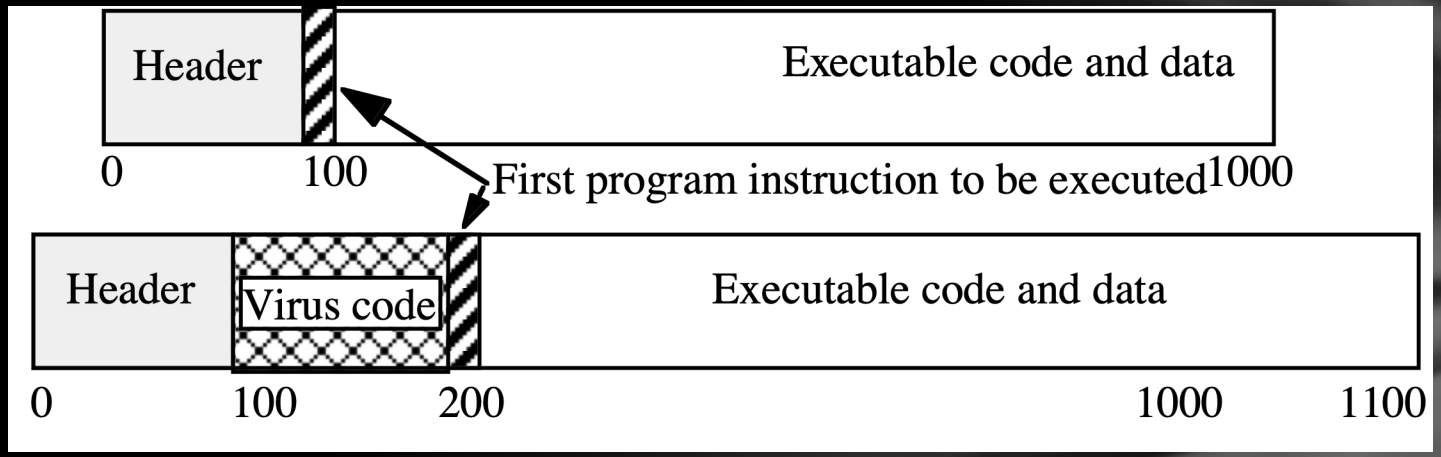
A virus that inserts itself into the boot sector of a disk

- Section of a disk containing code
- Executed when the system first "sees" the disk
  - Including at boot time …

Example: Brain virus

- Moves disk interrupt vector from 13H to 6DH
- Sets new interrupt vector to invoke Brain virus
- When a new floppy was seen, check for 1234H at location 4
  - If not there, it copies itself onto disk after saving the original boot block

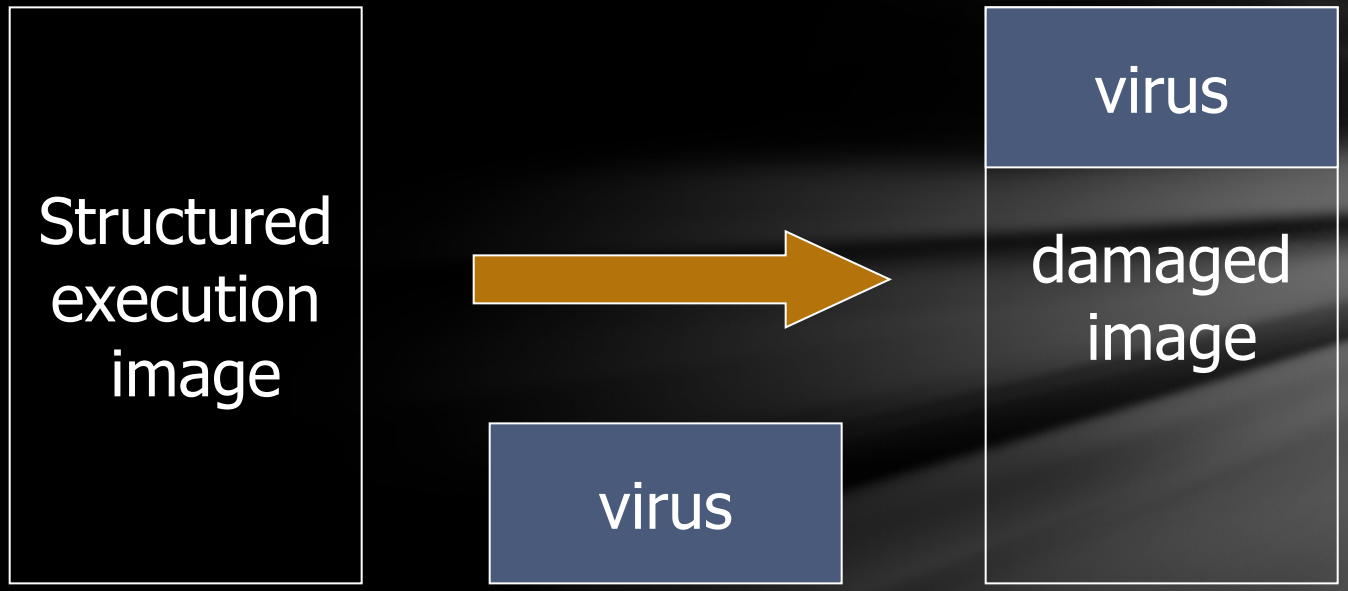
Executable Infectors



A virus that infects executable programs

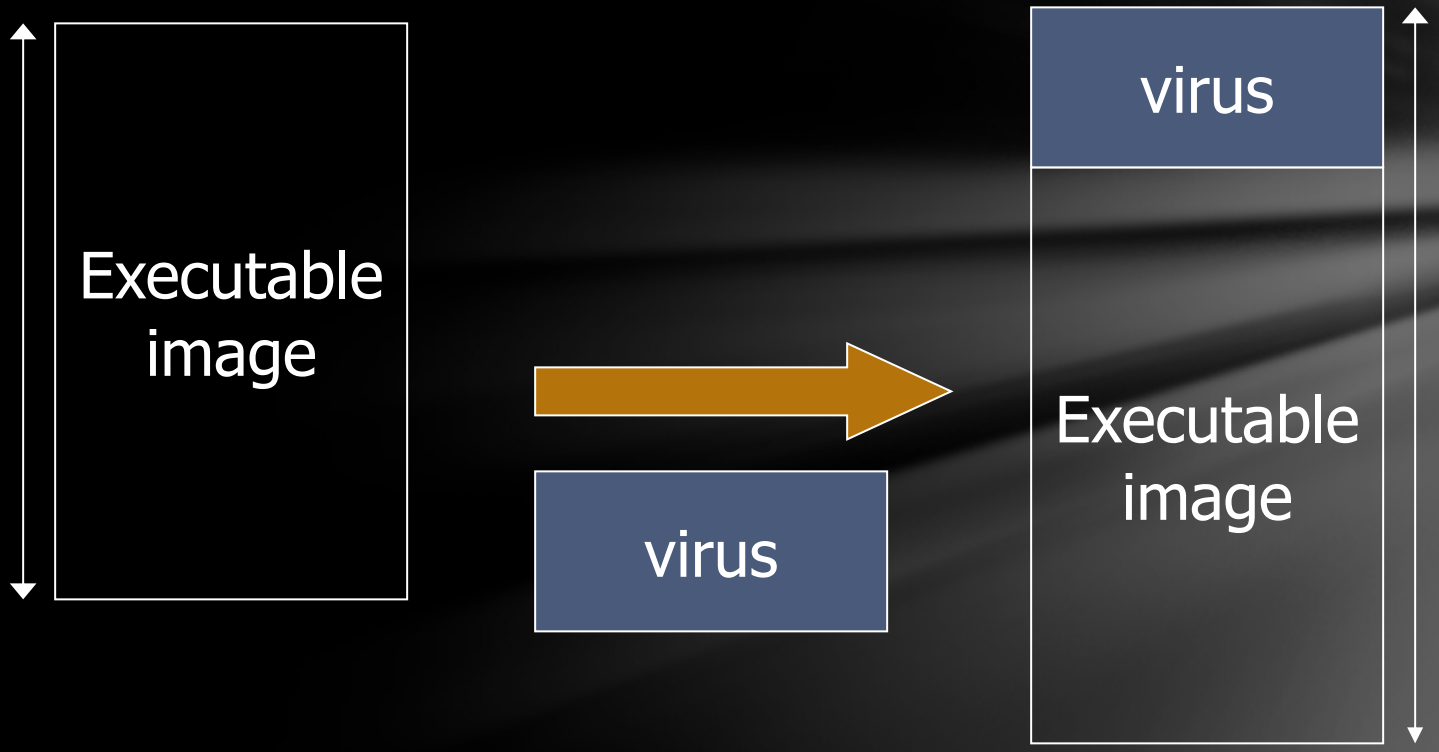
- It can infect either .EXE or .COM on PCs
- It may prepend itself (as shown) or put itself anywhere, fixing up binary so it is executed at some point.

Means of attaching: overwriting (virus *replaces* part of program)



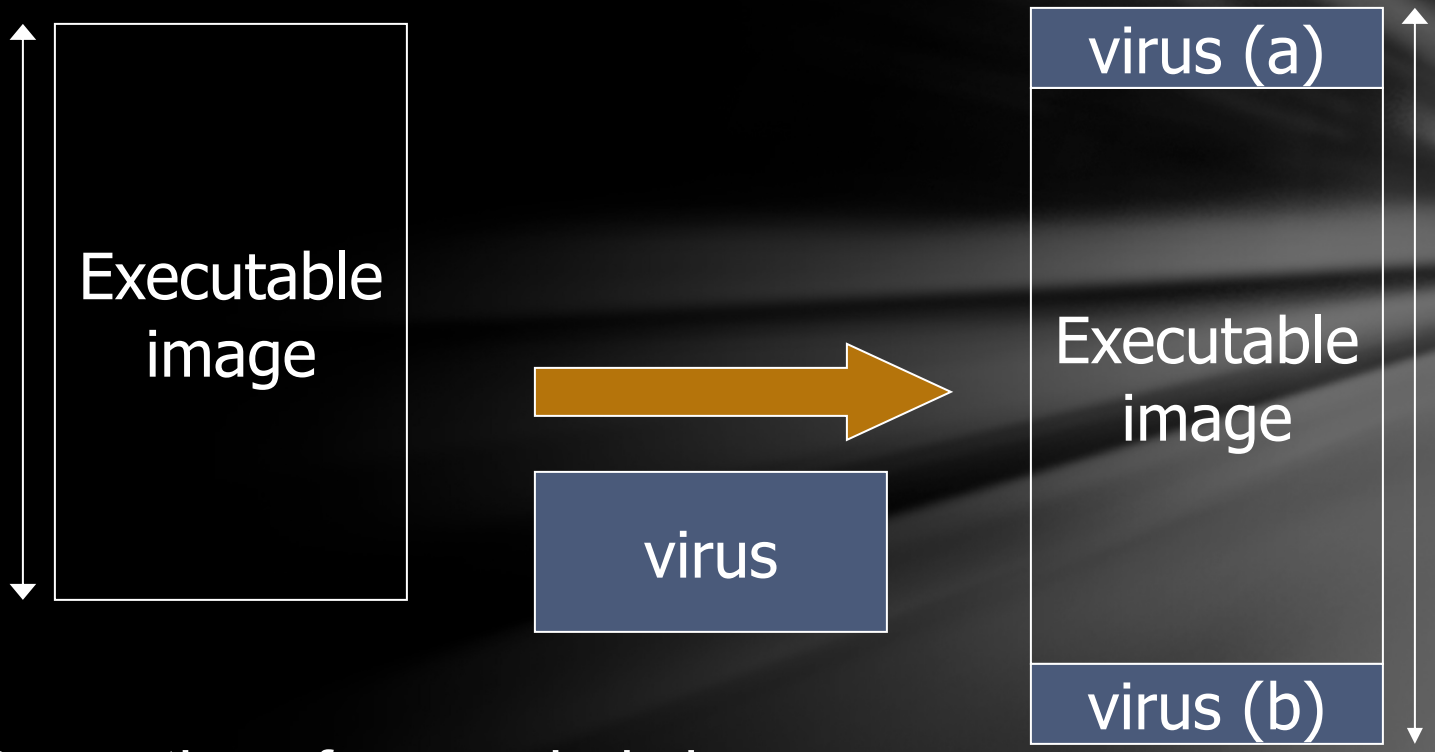
- A virus overwrites an executable file
- Easiest mechanism
- Since the original program is damaged easily detected

Means of attaching: at the beginning (virus is *appended* to program)



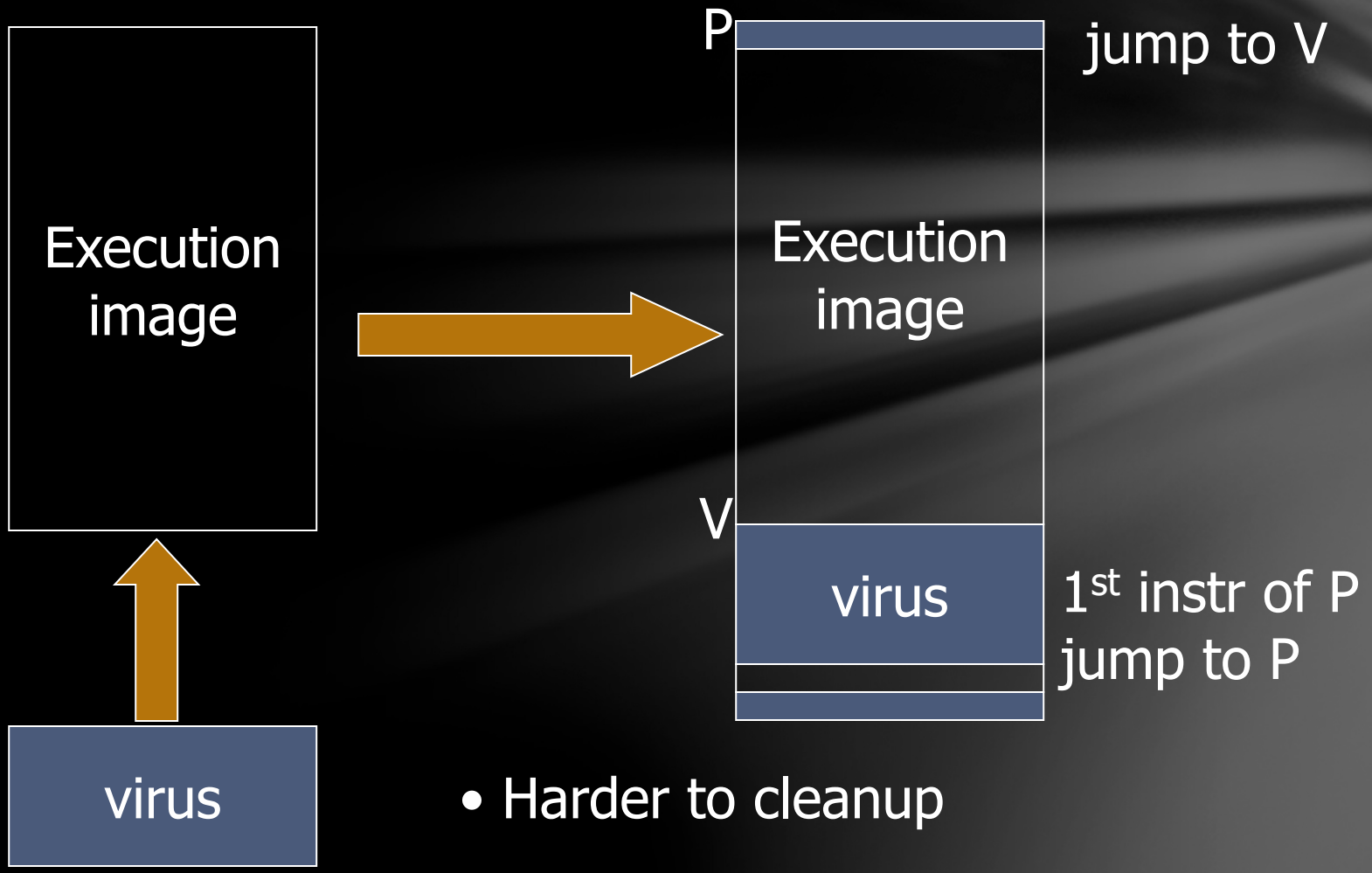
- Improved stealth because the original program is intact
- If the original program is large, copying it may be slow
- File size grows if multiple infections occur

Means of attaching: beginning and end (virus *surrounds* program)



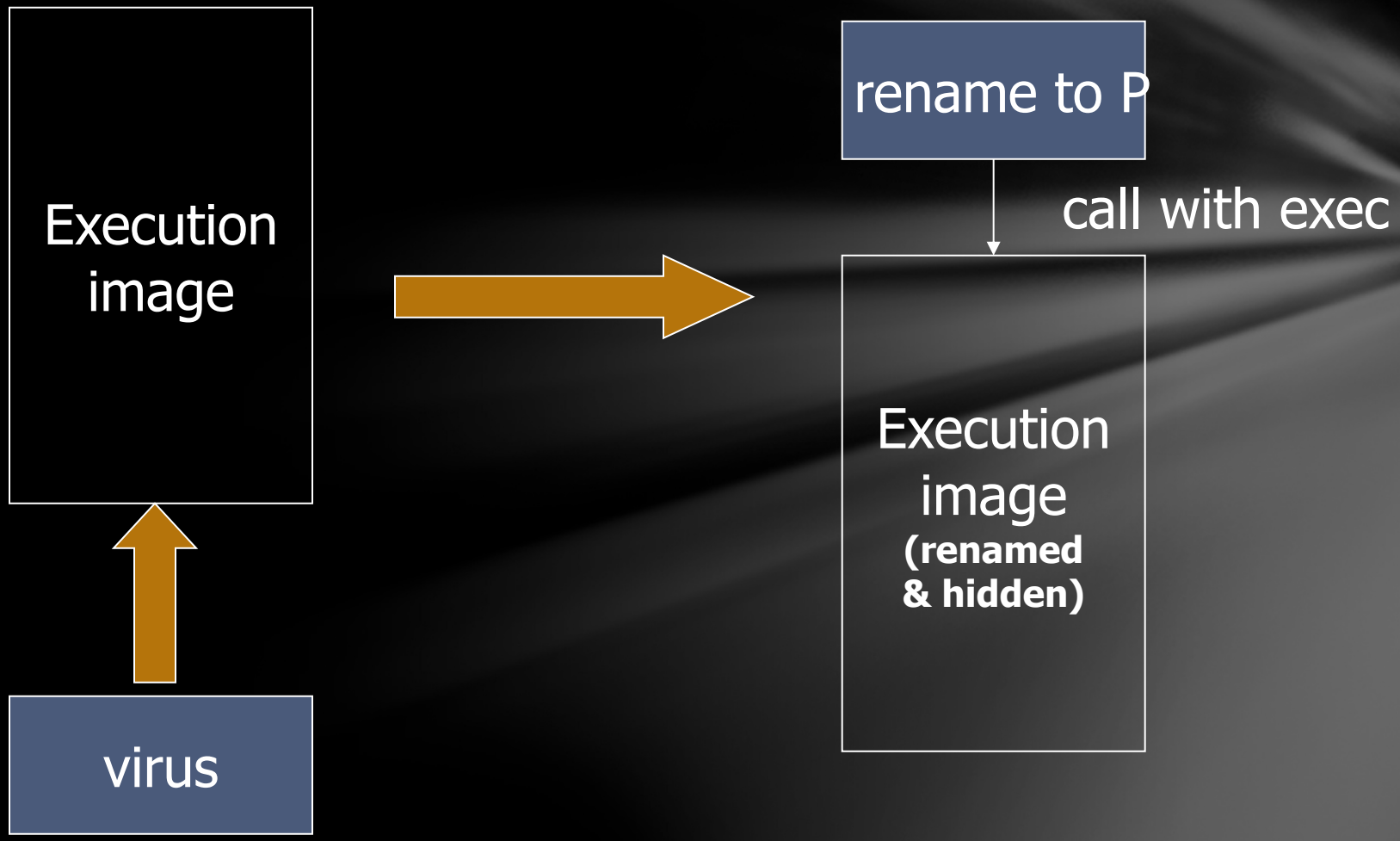
- Properties of appended virus +
- Ability to clean up and avoid detection
- Example: attach to the program that constructs file lists with sizes; modify after the program has ended

Means of attaching: intersperse (virus is *integrated* into program)



- Harder to cleanup

Means of attaching: companions



Executable Infectors (*con't*)

Jerusalem (Israeli) virus

- Checks if the system infected
  - If not, set up to respond to requests to execute files
- Checks date
  - If not 1987 or Friday 13th, set up to respond to clock interrupts and run the program.
  - Otherwise, set a destructive flag; it will delete, not infect, files
- Then: check all calls asking for files to be executed
  - Do nothing for COMMAND.COM
  - Otherwise, infect or delete
- BUG!: doesn't set signature when .EXE executes
  - So .EXE files are continually reinfected, and size goes up quickly

Multipartite Viruses

A virus that can infect either boot sectors or executables

Typically, two parts

- One part is a boot sector infector
- Another part is an executable infector

TSR Viruses

A virus that stays active in memory after the application (or bootstrapping or disk mounting) is completed

- TSR is "Terminate and Stay Resident"

Examples: Brain, Jerusalem viruses

- Stay in memory after a program or disk mount is completed

Stealth Viruses

A virus that conceals infection of files

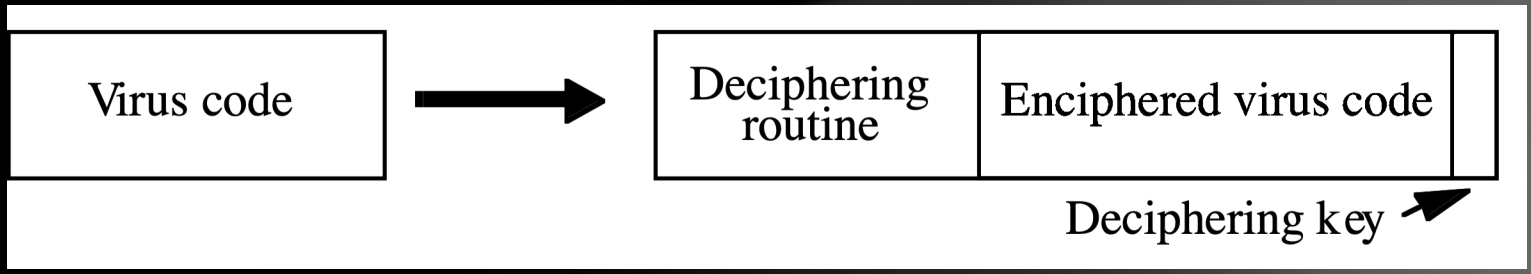
Example: IDF virus modifies DOS service interrupt handler as follows:

- Request for file length: return the size of the *uninfected* file
- Request to open file: temporarily disinfect file, and reinfect on closing
- Request to load file for execution: load infected file

Encrypted Viruses

A virus that is enciphered except for a small deciphering routine

- Detecting virus by signature is now much harder as most viruses are enciphered



Example

```
(* Decryption code of the 1260 virus *)
(* initialize the registers with the keys *)
rA = k1; rB = k2;
(* initialize rC with the virus; starts at sov,  ends at eov *)
rC = sov;
(* the encipherment loop *)
while (rC != eov) do begin
        (* encipher the byte of the message *)
        (*rC) = (*rC) xor rA xor rB;
        (* advance all the counters *)
        rC = rC + 1;
        rA = rA + 1;
end
```

Anti-virus tools' answer to encryption

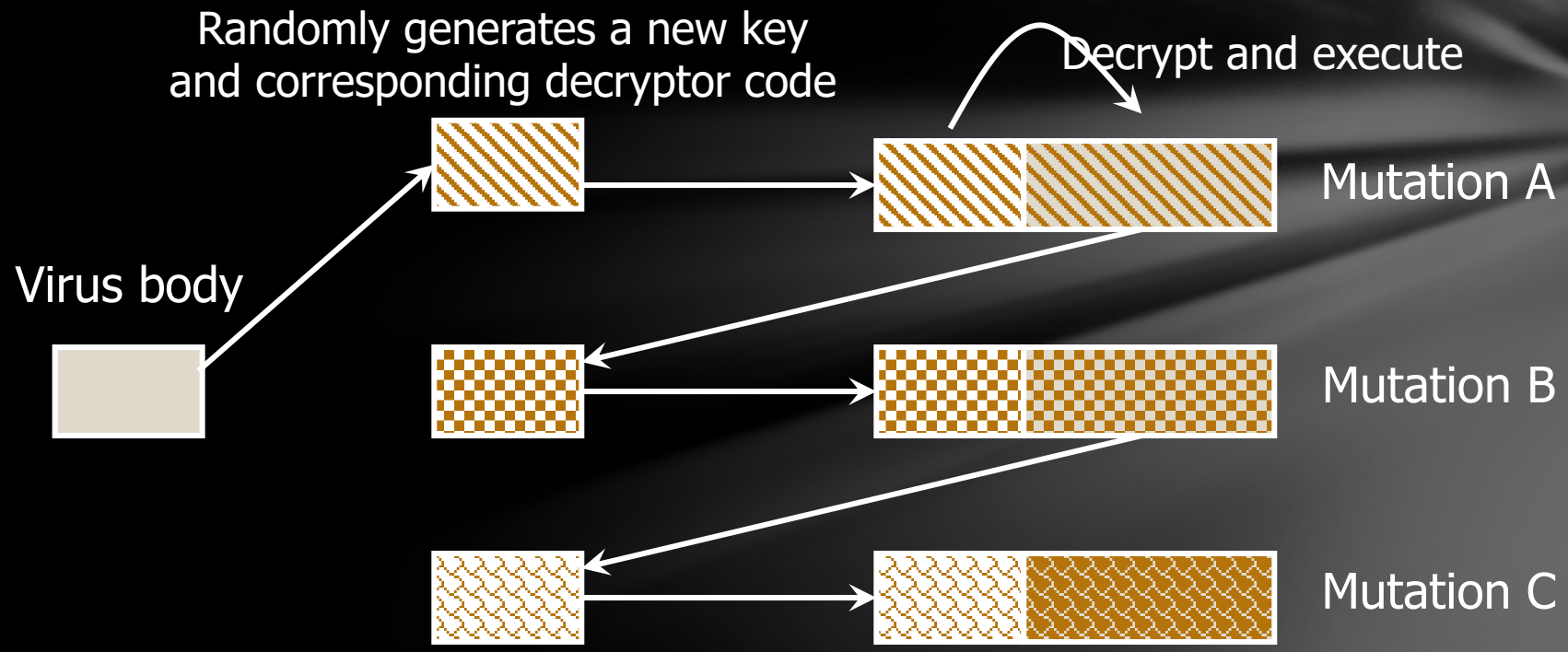
Select the signature from the unencrypted portion of the code, i.e., the decryption engine

Problems:

- Anti-virus tools usually want to determine which virus is present, not just determine that some virus is present (to "disinfect").
  - It can emulate the decryption and then further analyze the decrypted code.
- virus writers have responded by obscuring the encryption engine through mutations

It's a game of cat and mouse

Virus Emulation



Polymorphic Viruses

Polymorphic = “many forms.” A virus that changes its form each time it replicates

Prevent signature detection by changing the “signature” or instructions used for deciphering.

At the instruction level: substitute instructions

At the algorithm level: different algorithms to achieve the same purpose

- Given two code segments, evaluating their semantic equivalency is an undecidable problem!

Example

These are different instructions (with varying patterns of bit) but have the same effect:

- add o to a register
- subtract o from a register
- xor o with a register
- no-op

The polymorphic virus would pick randomly from among these instructions.

Example

```
(* Polymorphic code of the 1260 virus *)
(* initialize the registers with the keys *)
rA = k1; rB = k2;
rD = rD +1; (* random code*)
(* initialize rC with the virus;   starts at sov,  ends at eov *)
rC = sov;
rC = rC +1; (* random code*)
(* the encipherment loop *)
while (rC != eov) do begin
        rC = rC - 1; (* random code*)
        (* encipher the byte of the message *)
        (*rC) = (*rC) xor rA xor rB;
        (* advance all the counters *)
        rC = rC + 2;
        rD = rD + 1; (* random code*)
        rA = rA + 1;
end
While ( rC != sov) do begin (* random code*)
        rD = rD - 1; (* random code*)
        rC = rC - 1; (* random code*)
end (* random code*)
```

Worm / 蠕虫

- Differences among Worm, Trojan, and Virus:
  - Virus :
    - Existence: Usually inserted into host code (not a standalone program)
    - Propagation: Propagates by infecting other programs
  - Trojan :
    - Existence: It is a standalone executable program
    - Propagation: do not self-replicate
  - Worm :
    - Existence: It is a standalone executable program
    - Propagation: Propagates automatically by copying itself to target systems

Computer Worms Origins: distributed computations

Schoch & Hupp (1982): animations, broadcast messages

A segment is the part of the program copied onto a workstation

A segment processes data and communicates with the controller. Any activity on a workstation caused the segment to shut down.

Example: Christmas Worm

- Distributed in 1987, designed for IBM networks
- Electronic letter instructing the recipient to save it and run it as a program
  - Drew Christmas tree, printed "Merry Christmas!"
  - It also checked the address book and list of previously received emails and sent copies to each address.
- Shut down several IBM networks
- It is a macro worm
  - Written in a command language that was interpreted
  - Can cross different system platforms!

Morris Worm

- In 1988, Robert Morris released a worm.
  - The original goal is to slowly and harmlessly breed in the Internet to measure the size of the Internet.
  - However, due to a code error, worms create new copies as quickly as possible, and finally, the infected machine completely crashes.
  - The damage is between $ 10M ~ 100M
- Morris worm exploited fingerd in VAX systems, carrying out the buffer overflow attacks
  - send a special string to fingered, so be able to make it execute code to create a new copy of the worm
  - As it can not know the operating system in the target machine, the worm also attacks the BSD fingerd and crash it.
- Robert Morris:
  - Achieve Bachelor of Arts and Ph.D. in Applied Science from Harvard University
  - Son of the chief scientist of the U.S. National Security Agency
  - The first one be indicted based on the " Computer Fraud and Abuse Act, was sentenced to three years probation, 400 hours of community service and fined $ 10, 050
  - Currently, he is a professor of computer science at MIT

Code Red I

- Code Red I v1, @2001.07.31 ——the first worm in modern sense
  - Use a buffer overflow vulnerability in the IIS server
  - Spread between the 1st and 20th of every month
    - Obtained by randomly scanning IP for a new target
    - Start 99 threads to produce an address and find the presence of IIS
  - Attack between the 21st and the end of each month
    - Modify the page as“HELLO! Welcome to http: //www.worm.com! Hacked by Chinese!”
- Code Red I v2, @2001.07.19 —— almost the same code with Code Base I, but modify a bug when randomly generating IP
  - Attacked "all" IIS hosts on the Internet!
  - a large number of attacked hosts led to extremely rapid worm propagation
    - Exponential spread
    - 350, 000 hosts were infected in 14 hours!
  - The carried load: distributed DDOS on www.whitehouse.gov!
    - Due to code errors, it will die automatically on the 20th of each month.
    - However, if the host's clock is mistaken, it will be revived on the 1st of the month!

Code Red II

- 2001.8.4: Use the same IIS bug but with a completely different code
  - According to the code comments, known as the "Code Red II. "
  - only effectively running on Windows 2000, making Windows NT crash!
- Scanning algorithms tend to scan closer address
  - 1/2 chance of scanning the same Class A address
  - 3/8 chance of scanning the same Class B address
  - 1/8 chance of scanning the other address !
- Worm with a malicious program :
  - Installing a backdoor on the IIS server achieves unlimited remote access
- Set to die on October 1, 2001, automatically

Nimda

- On September 18, 2001, the outbreak of the multi-mode spreading worm
  - Attack the same IIS buffer overflow as Red Code I/II
  - Collect the email address on the host, send the email as an attachment
  - Copy through shared folders
  - insert malicious code to the attacked host's website, attack accessing web browser!
  - Check the backdoor left by the Red Code II
- Malicious payload: delete all data on your hard disk!
- Signature-based detection is no longer effective: Nimda can cross the firewall!!
  - A firewall generally does not check e-mail; the mail server filters the malicious e-mail
    - Most of the mail server filter scan only the signature
  - Nimda is a new worm; there is no "signature," so the scanner can not check out
  - Nimda caused the emergence of new security issues
    - Zero-day attack
    - Worms spread quickly; many machines are infected immediately before their signatures are updated and checked!

Slammer

- Break out on 24/25 January 2003
  - UDP worms, using a buffer overflow in SQL Server
  - Microsoft has resolved the buffer overflow vulnerability, but not everyone will install the patch!
- All of the code can be put into a 404-length UDP packet
  - including worms binary code and buffer overflow pointer to the code
- Classic random search methods:
  - Once the worm takes control, it randomly generates IP addresses and sends copies of itself to the addresses' 1434 port
    - MS-SQL listening port is 1434
- Slammer's propagation: Search 55, 000, 000 IP addresses per second, 30, 000 copies of the worm per second
  - Double within 8.5 seconds after initial infection, while Code Red took 37 minutes
  - Within 10 minutes, the package generated by Slammer saturates the Internet
    - 75000 SQL Server are infected
    - Moreover, the pseudo-random number generator used in the code has a problem.

Slammer's influence

- $1.25 billion loss
- Temporarily crashes all kinds of critical infrastructure equipment
  - ATM Network of America bank
  - All mobile phone networks in Korea
  - Five root DNS servers
  - U.S. Continental Airlines ticketing system
- The worm even does not contain any malicious code but only runs out of the network bandwidth and the host's resources completely!
- Why does Slammer spread so quickly?
  - Old worms (Code Red) create a thread to connect the target host. If successfully connected, it sends copies of itself using TCP.
    - Limited by network latency - three-way handshake
  - Slammer is a connectionless UDP worm
    - Do not establish a connection; just send 404 bytes udp packet
    - Limited only by network bandwidth!!

Blaster and Welchia/Nachia

- Blaster Worm
  - On August 11, 2003, used the RPC service's vulnerability in Windows XP/2000
  - Randomly generating the first address, then sequentially scanning IP addresses upwards.
  - Payload: DoS of the Windows Update service, install backdoors at the same time
- Welchia/Nachia: Originally designed for anti-worm
  - Randomly start to scan and use ICMP to determine whether the host is alive
  - If alive, copy itself and fix RPC vulnerability, and kill the Blaster worm
  - However, it causes greater damage. . . Network bandwidth is completely exhausted!

Other Types of Mal. Code —— Rabbits, Bacteria

A program that absorbs all of some class of resources

Example: for UNIX system, shell commands [Dennis 1979]:

```
while true
do
   mkdir x;  chdir x
done
```

Exhaust disk space or file allocation table (inode) space may crash the system. But very easy to trace back!

Countermeasures: Quote for each user

Other Types of Mal. Code —— Logic Bombs

A program that performs an action that violates the site security policy when some external event occurs (Ex. Special Holiday, Special Name, …)

Example:

A program that deletes the company's payroll records when one particular record is deleted

- The "particular record" is usually that of the person writing the logic bomb
- The idea is if (when) they are fired, and the payroll record is deleted, the company loses *all* those records

Defenses of Mal. Code

How Hard Is It to Write a Virus?

2268 matches for the "virus creation tool" in CA's Spyware Information Center

- Including dozens of poly- and metamorphic engines

OverWritting Virus Construction Toolkit

- "The perfect choice for beginners"

Biological Warfare Virus Creation Kit

- Note: Norton Anti-Virus will detect all viruses

VBS Worm Generator (for Visual Basic worms)

- Used to create the Anna Kournikova worm

Many others

Theory of Virus Detection

There is NO algorithm to detect all possible viruses/malicious code.

It was proved by Cohen in "Computers and Security" in 1989.

No perfect solutions: Now what?

- Signature-based antivirus
  - Look for known patterns in malicious code
  - Always a battle with the attacker
  - *Great business model!*

Virus Analysis

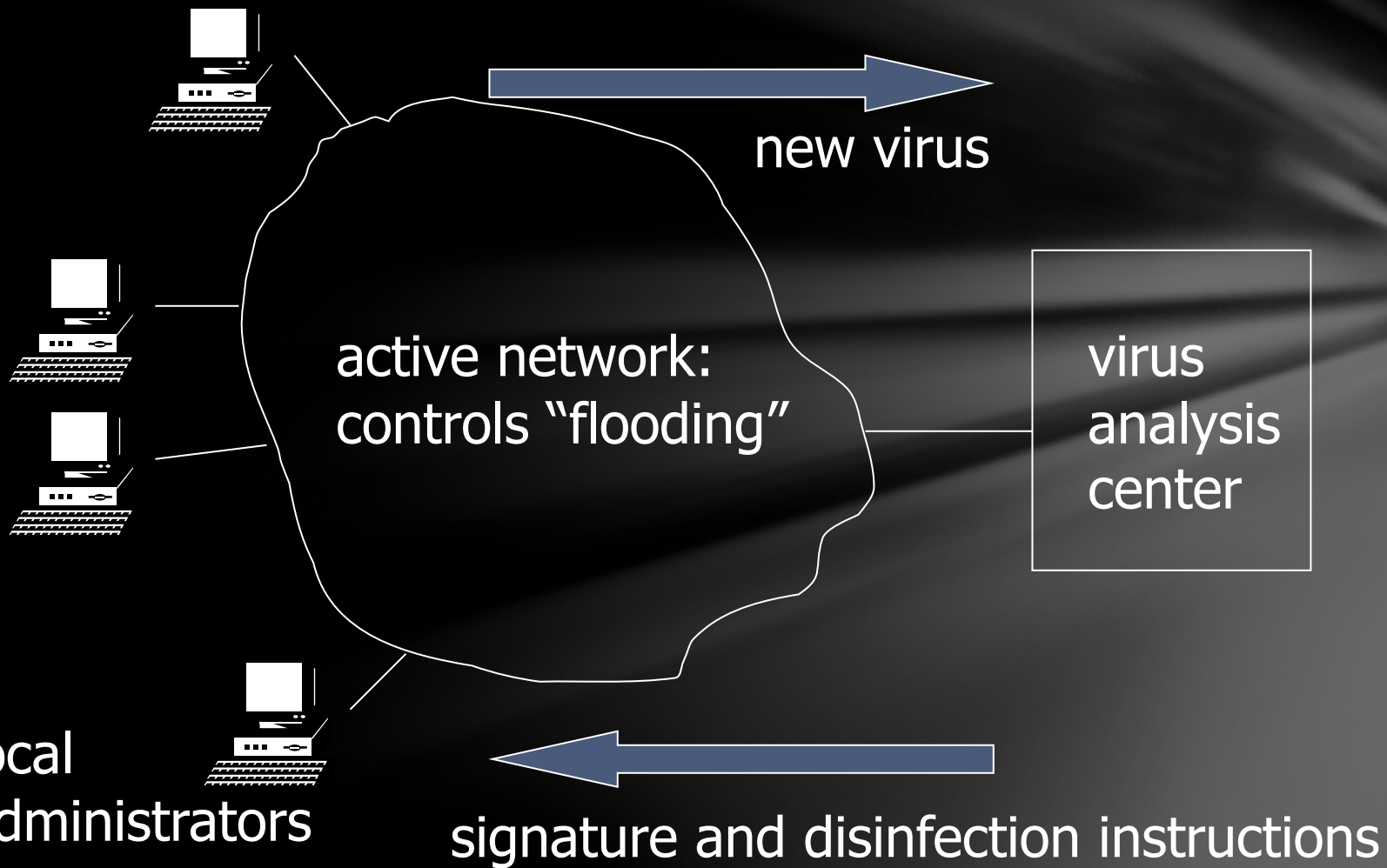
Analysis of the virus by a human expert

- slow: by the time signature has been extracted, posted to the AV tool database, and downloaded to users, the virus may have spread widely.
  - pre-1995: 6 months to a year for a virus to spread worldwide
  - mid-90's: a few months
  - now: days or hours
- labor-intensive: too many new viruses
  - currently, 8-10 new viruses per day
- can't handle epidemics:
  - The queue of viruses to be analyzed overflows
  - heavy demand on the server that posts signatures & fixes

Automated analysis, e.g., "Immune System."

- developed at IBM Research
- licensed to Symantec

Immune System Architecture



Signature Extraction at VAC

Virus allowed (encouraged) to replicate in a controlled environment in an immune center

This yields a collection of infected files

In addition, a collection of “clean” files is available

Machine learning techniques are used to find strings that appear in most infected files and few clean files, e.g.:

- search files for candidate strings
  - add points if found in an infected file
  - subtract points if found in a clean file

Defenses Means

- Distinguish between data, instructions
- Limit objects accessible to processes
- Inhibit sharing
- Detect altering of files
- Detect actions beyond specifications
- Analyze statistical characteristics

Clear Distinction between Data and Executable

The virus must write to the program

- A program is only allowed to write data

Must execute to spread/act

- Data not allowed to execute

Auditable action is required to change data to an executable

Example: LOCK

Logical Coprocessor Kernel

- Designed to be certified at the TCSEC A1 level

Compiled programs are typed "data."

- A sequence of specific, auditable events is required to change the type to "executable."

Cannot modify "executable" objects

- So viruses can't insert themselves into programs (no infection phase)

Example: Duff and UNIX

Observation: users with execute permission usually have read permission, too

- So files with "execute" permission have the type "executable"; those without it type "data."
- Executable files can be altered, but the type is immediately changed to "data."
  - Implemented by turning off execute permission
  - Certifier can change them back
    - So the virus can spread only if run as a certifier

Defenses Means

- Distinguish between data, instructions
- Limit objects accessible to processes
- Inhibit sharing
- Detect altering of files
- Detect actions beyond specifications
- Analyze statistical characteristics

Limiting Accessibility

Information Flow

- Malicious code usurps the authority of a user
- Limit information flow between users
  - If *A* talks to *B*, *B* can no longer talk to *C*
- Limits the spread of the virus
- Problem: Tracking information flow

Application of principle of least privilege

Basic idea: remove rights from the process so it can only perform its function

- Warning: if that function requires it to write, it can write anything
- But you can make sure it writes only to those objects you expect

Guardians, Watchdogs

The system intercepts all requests to open a file

Program invoked to determine if access is to be allowed

- These are *guardians* or *watchdogs*

Effectively redefines system (or library) calls

Sandbox / Virtual Machine

Run in a protected area.

Libraries/system calls replaced with limited privilege set.

Defenses Means

- Distinguish between data, instructions
- Limit objects accessible to processes
- Inhibit sharing
- Detect altering of files
- Detect actions beyond specifications
- Analyze statistical characteristics

Inhibit Sharing

Use separation implicit in integrity policies.

Example: LOCK keeps a single copy of the shared procedure in memory

- Master directory associates a unique owner with each procedure and a list of trusted users with each user
- Before executing any procedure, the system checks that the user running the procedure trusts the procedure owner

Use Multilevel Security Mechanisms

Use Multi-Level Security Mechanisms

- Place programs at the lowest level
- Don't allow users to operate at that level
- *Prevents writes by malicious code*

Example: DG/UX system

- All executables in the "virus protection region" below user and administrative regions

Defenses Means

- Distinguish between data, instructions
- Limit objects accessible to processes
- Inhibit sharing
- Detect altering of files
- Detect actions beyond specifications
- Analyze statistical characteristics

Detect Alteration of Files

Compute manipulation detection code (MDC) to generate a signature block for each file, and save it

Later, recompute MDC and compare it to stored MDC

- If different, the file has changed

Example: tripwire

- The signature consists of file attributes (size, owner id, protection mode, inode no.), cryptographic checksums chosen from MD4, MD5, HAVAL, SHS, CRC-16, CRC-32, etc.)

Antivirus Programs

Look for specific sequences of bytes (called “virus signature” in files

- If found, warn the user and/or disinfect the file

Each agent must look for a known set of viruses

Cannot deal with viruses not yet analyzed

- Due in part to the undecidability of whether a generic program is a virus

Defenses Means

- Distinguish between data, instructions
- Limit objects accessible to processes
- Inhibit sharing
- Detect altering of files
- Detect actions beyond specifications
- Analyze statistical characteristics

Detect Actions Beyond Spec

Treat the infection as an error and apply fault-tolerant techniques when executing.

Example: break the program into sequences of nonbranching instructions

- Checksum each sequence, encrypt the result
- When run, the processor recomputes the checksum, and at each branch, the co-processor compares the computed checksum with the stored one.
  - If they are different, an error occurs.

N-Version Programming

Implement several different versions of the algorithm

Run them concurrently

- Check intermediate results periodically
- If there is disagreement, the majority wins

Assumptions

- Majority of programs not infected
- The underlying operating system secure

Conflicts:

- To control the propagation of the virus, all the algorithms must vote for each file access.
- They have to be almost the same!

Proof-Carrying Code

Code consumer (user) specifies safety requirement

Code producer (author) generates proof code meets this requirement

Binary (code + proof) delivered to the consumer

Consumer validates proof

- Changing the code will make the validation process fail so that the consumer will reject the code.

Example statistics on Berkeley Packet Filter [Necula & Lee 1996]: proofs 300–900 bytes, validated in 0.3 –1.3 ms

- The startup cost is higher, and the runtime cost is considerably lower.

Defenses Means

- Distinguish between data, instructions
- Limit objects accessible to processes
- Inhibit sharing
- Detect altering of files
- Detect actions beyond specifications
- Analyze statistical characteristics

Detecting Statistical Changes

Based on: each programmer has their coding style. Statistical data can collect in this style.

Example: application had three programmers working on it, but statistical analysis shows code from a fourth person—may be from a Trojan horse or virus!

- Source code level: format, comments, etc
- Binary code level: data structure, algorithm, etc

Application Statistical

- High/low number of files read/written
- An unusual amount of data transferred
- Abnormal usage of CPU time
- Denning: use an intrusion detection system to detect these
- *It only works after the damage is done!*

Summary – the Key Points

A perplexing problem

- How do you tell if what the user asked for is *not* what the user intended?

File scanners are the most popular anti-virus agents

- Must be updated as new viruses come out

Botnets, DDoS and SPAM

Ghosts on the Internet

Botnets

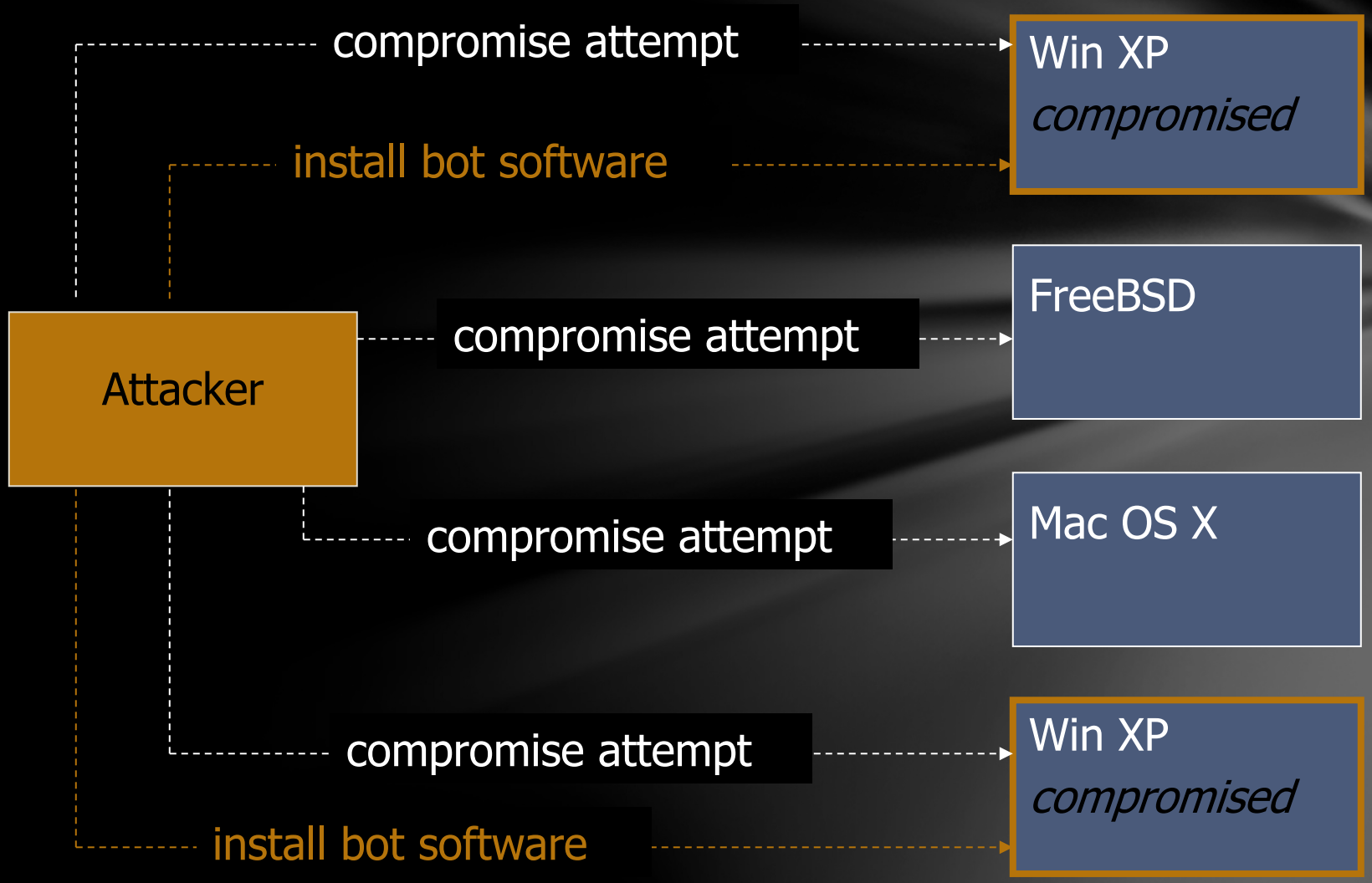
Botnet = network of autonomous programs capable of acting on instructions

- Typically large (up to several hundred thousand) groups of remotely controlled "zombie" systems
  - Machine owners are not aware they have been compromised
- Controlled and upgraded via IRC or P2P

Used as the platform for various attacks

- Distributed denial of service (DDoS)
- Spam and click fraud
- Launching pad for new exploits/worms

Building a Botnet



Typical Infection Path

Exploit a vulnerability to execute a short program (shellcode) on the victim's machine

- Buffer overflows, email viruses, etc.

Shellcode downloads and installs the actual bot

Bot disables firewall and antivirus software

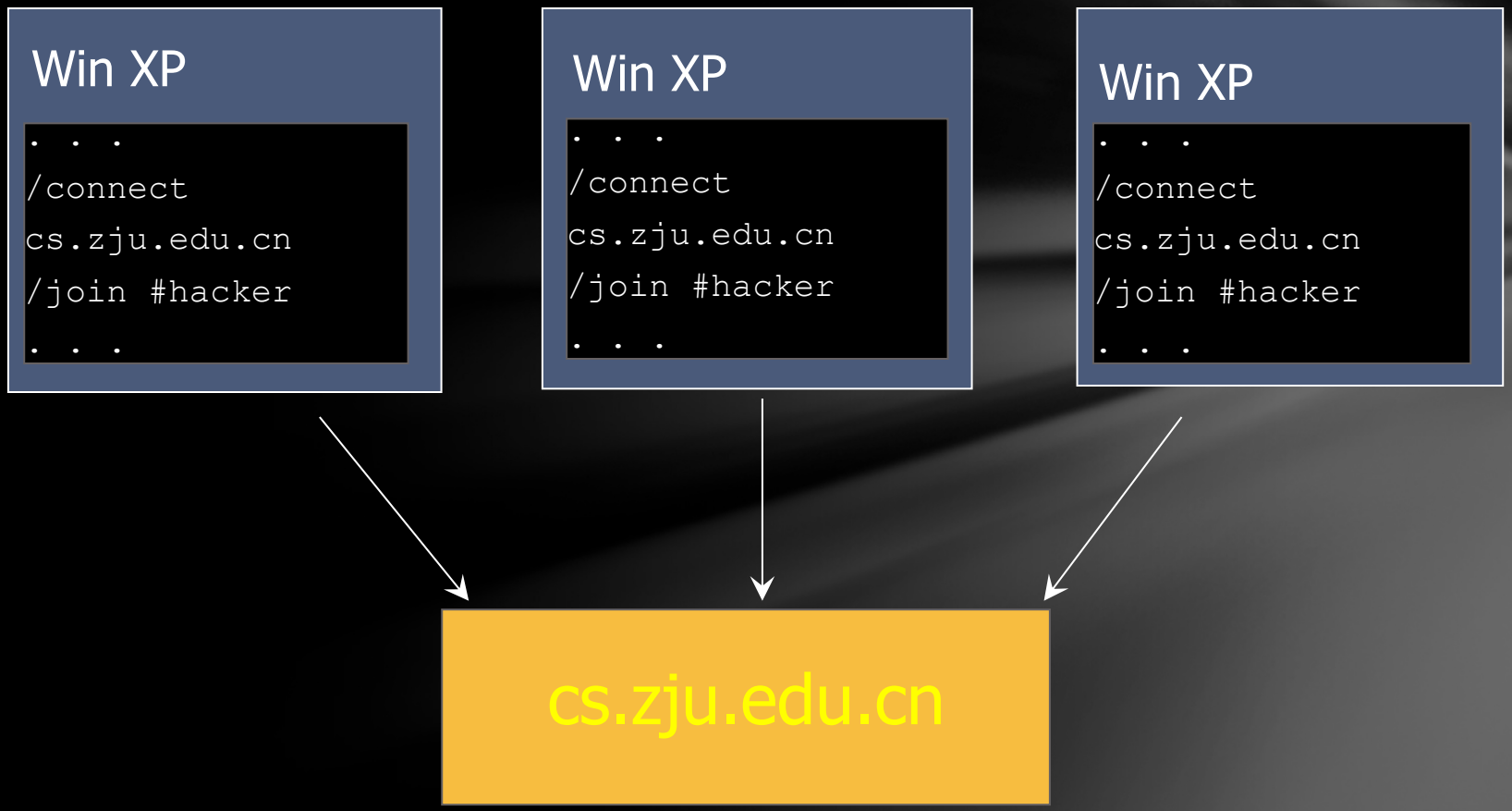
Bot locates IRC server, connects, joins a channel

- Typically need DNS to find out the server's IP address
  - Specially if the server's original IP address has been blacklisted
- Authentication passwords are often stored in bot binary

Botmaster issues authenticated commands

Like an Army!

Joining the IRC Channel



Command and Control

```
(12: 59: 27pm) -- A9-pcgbdv (A9-pcgbdv@140.134.36.124) has
joined (#owned) Users :  1646

(12: 59: 27pm) (@Attacker) .ddos.synflood 216.209.82.62

(12: 59: 27pm) -- A6-bpxufrd (A6-bpxufrd@wp95-
81.introweb.nl) has joined (#owned) Users :  1647

(12: 59: 27pm) -- A9-nzmpah (A9-nzmpah@140.122.200.221)
has left IRC (Connection reset by peer)

(12: 59: 28pm) (@Attacker) .scan.enable DOMAIN

(12: 59: 28pm) -- A9-tzrkeasv (A9-tzrkeas@220.89.66.93)
has joined (#owned) Users :  1650
```

Botnet Propagation

[Abu Rajab et al.]

Each bot can scan IP space for new victims

- Automatically
- On-command: target specific /8 or /16 prefixes
  - Botmasters share information about prefixes to avoid

Active botnet management

- Detect non-responding bots, and identify "super bots."

Evidence of botnet-on-botnet warfare

- DoS server by multiple IRC connections ("cloning")
- N-to-N architecture, see the architecture in DDoS

Reproduction and Development of Botnet

Date created	Date dismantled	Name	Estimated no. of bots	Spam capacity	Aliases
2009 (May)	2010-Oct (partial)	BredoLab	30,000,000[13]	3.6 billion/day	Oficla
2008 (around)	2009-Dec	Mariposa	12,000,000[14]	?	
2008 (November)		Conficker	10,500,000+[15]	10 billion/day	DownUp, DownAndUp, DownAdUp, Kido
2010 (around)		TDL4	4,500,000[16]	?	TDSS, Alureon
?		Zeus	3,600,000 (US Only)[17]	n/a	Zbot, PRG, Wsnpoem, Gorhax, Kneber
2007 (Around)		Cutwail	1,500,000[18]	74 billion/day	Pandex, Mutant (related to: Wigon, Pushdo)
2008 (Around)		Sality	1,000,000[19]	?	Sector, Kuku
2009 (Around)	2012-07-19	Grum	560,000[20]	39.9 billion/day	Tedroo
?		Mega-D	509,000[21]	10 billion/day	Ozdok
?		Kraken	495,000[22]	9 billion/day	Kracken
2007 (March)		Srizbi	450,000[23]	60 billion/day	Cbeplay, Exchanger
?		Lethic	260,000[24]	2 billion/day	none
2004 (Early)		Bagle	230,000[24]	5.7 billion/day	Beagle, Mitglieder, Lodeight
?		Bobax	185,000[24]	9 billion/day	Bobic, Oderoor, Cotmonger, Hacktool.Spammer, Kraken
?		Torpig	180,000[25]	n/a	Sinowal, Anserin
?		Storm	160,000[26]	3 billion/day	Nuwar, Peacomm, Zhelatin
2006 (Around)		Rustock	150,000[27]	30 billion/day	RKRustok, Costrat
?		Donbot	125,000[28]	0.8 billion/day	Buzus, Bachsoy

Denial of Service (DoS) Redux

Goal: overwhelm the victim machine and deny service to its legitimate clients

DoS often exploits networking protocols

- Smurf: ICMP echo request to broadcast address with spoofed victim's address as a source
- Ping of death: ICMP packets with payloads greater than 64K crash older versions of Windows
- SYN flood: "open TCP connection" request from a spoofed address
- UDP flood: exhaust bandwidth by sending thousands of bogus UDP packets

Distributed Denial of Service (DDoS)

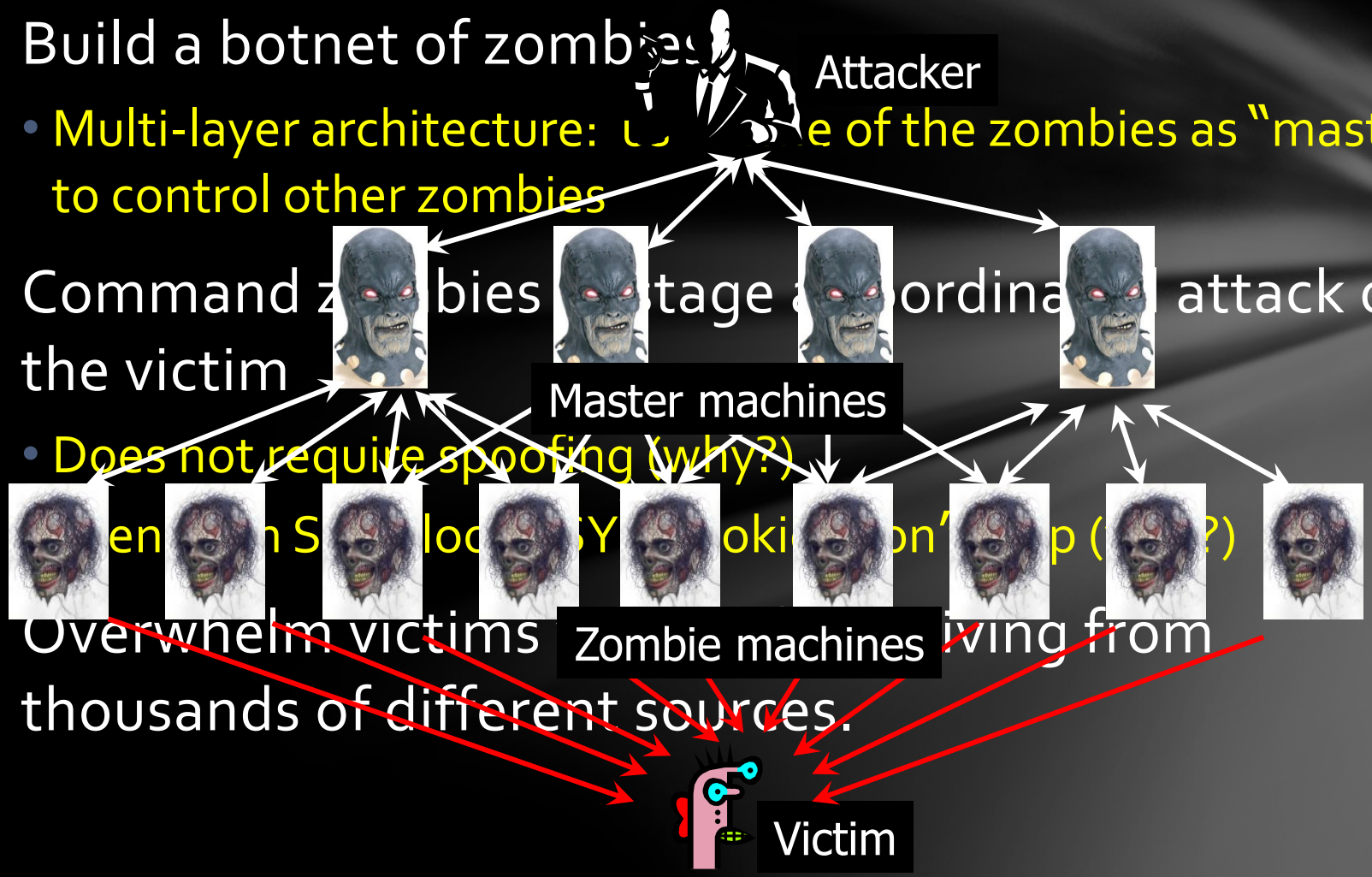
Build a botnet of zombies

- Multi-layer architecture: use some of the zombies as "masters" to control other zombies

Command zombies to stage a coordinated attack on the victim

- Does not require spoofing (why?)
- Often with SYN flooding, smurf, etc. (why?)

Overwhelm victim's resources by receiving from thousands of different sources.



Detecting Botnets

Today's bots are controlled via IRC and DNS

- IRC used to issue commands to zombies
- DNS is used by zombies to find the master and by the master to see if a zombie has been blacklisted

IRC/DNS activity is very visible in the network

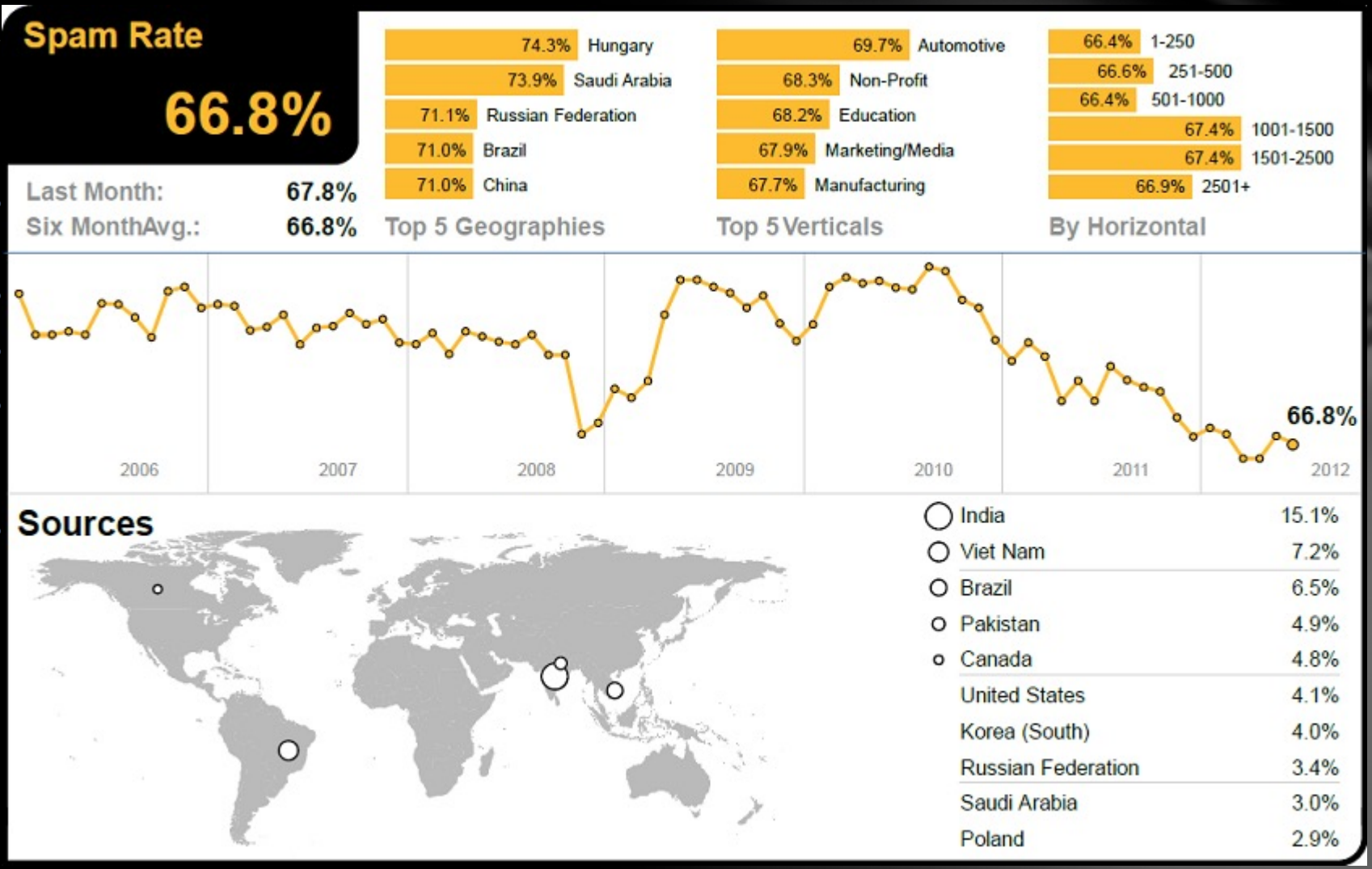
- Look for hosts performing scans and for IRC channels with a high percentage of such hosts.
  - Used with success at Portland State University
- Look for hosts who ask many DNS queries, but receive few queries about themselves.

Easily evaded by using encryption and P2P. ☹

SPAM

Email Spam

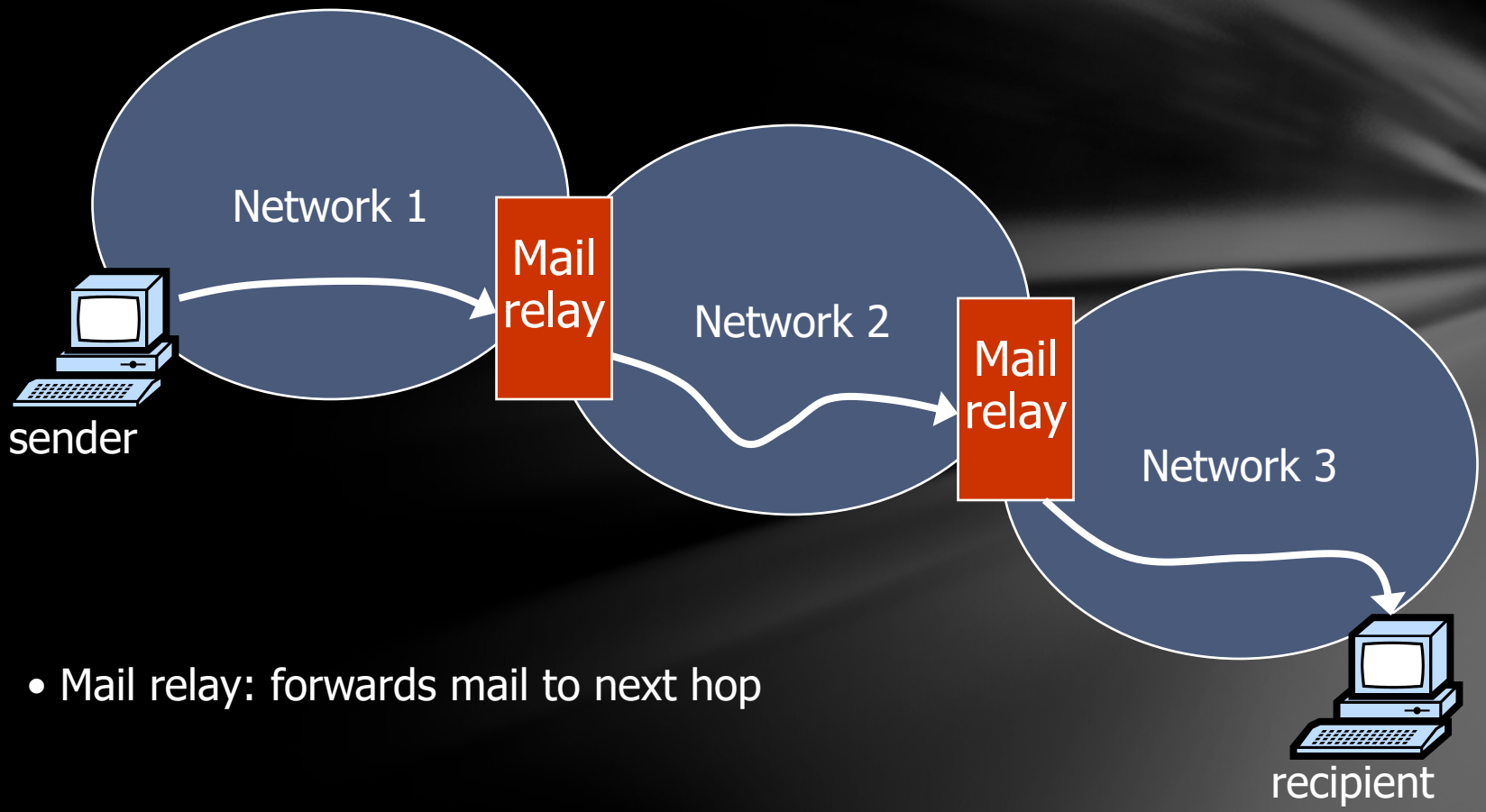
- 定


Spam Rate
66.8%
Last Month: 67.8%
Six MonthAvg.: 66.8%
74.3% Hungary
73.9% Saudi Arabia
71.1% Russian Federation
71.0% Brazil
71.0% China
Top 5 Geographies
69.7% Automotive
68.3% Non-Profit
68.2% Education
67.9% Marketing/Media
67.7% Manufacturing
Top 5 Verticals
66.4% 1-250
66.6% 251-500
66.4% 501-1000
67.4% 1001-1500
67.4% 1501-2500
66.9% 2501+
By Horizontal
2006
2007
2008
2009
2010
2011
2012
66.8%
Sources
India 15.1%
Viet Nam 7.2%
Brazil 6.5%
Pakistan 4.9%
Canada 4.8%
United States 4.1%
Korea (South) 4.0%
Russian Federation 3.4%
Saudi Arabia 3.0%
Poland 2.9%

Why so many spams?

- Driven by business benefits
- Simple manufacturing and sending
  - AddressBase: Email address crawler 、 All kinds of email AddressBase
  - Sending tools: 百度“邮件群发工具”、Botnets
- Lots of wrong configurations
  - Early SMTP don't need validation: a lot of Open Relay exist
  - So far, the Mail server is the most difficult server to configure correctly.

Email in the Early 1980s



- Mail relay: forwards mail to next hop

Email Spoofing

Mail is sent via SMTP protocol

- No built-in authentication

The sender sets the MAIL FROM field

- A classic example of improper input validation

The recipient's mail server only sees the IP address of the direct peer from whom it received the msg.

Open Relays

SMTP relay forwards mail to the destination

1. The bulk email tool connects via SMTP (port 25)
2. Sends list of recipients via RCPT TO command
3. Sends email body (once for all recipients!)
4. Relay delivers message

Honest relay adds correct Received: header revealing source IP

Hacked relay does not

A Closer Look at Spam

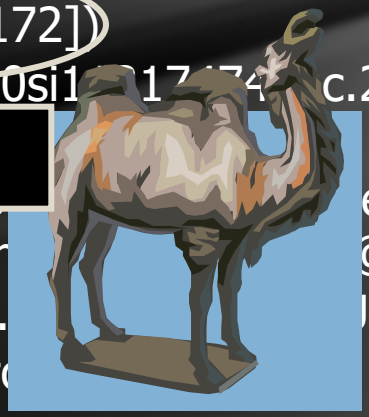
Inserted by relays

Received: by 10.78.68.6 [illegible] 3hua;
Mon, 12 Feb 2007 06: 43: 30 -0800 (PST)
Received: by 1 [illegible] SMTP id l18mr17307116agc.1171291410432;
Mon [illegible] : 43: 30 -0800 (PST)
Return-Path: <[illegible]>
Received: from onelinkpr.net ([203.169.49.172])
by mx.google.com with ESMTP id 30si1 [illegible] c.2007.02.12.06.43.18;
[illegible] 6:

Bogus!

Puerto Rico

Mongolia



Rec [illegible] er permitted nor
[illegible] by best guess record for dor [illegible] @aviva.ro)
Mes [illegible] 0050057765.stank.203.169. [illegible] >
From: "Barclay Morales" <wvnlwee@aviva.r[illegible]
To: <raykwatts@gmail.com>
Subject: You can order both Viagra and Cialis.

Why Hide Sources of Spam?

Many email providers blacklist servers and ISPs that generate a lot of spam

- Use info from spamhaus.org, spamcop.net

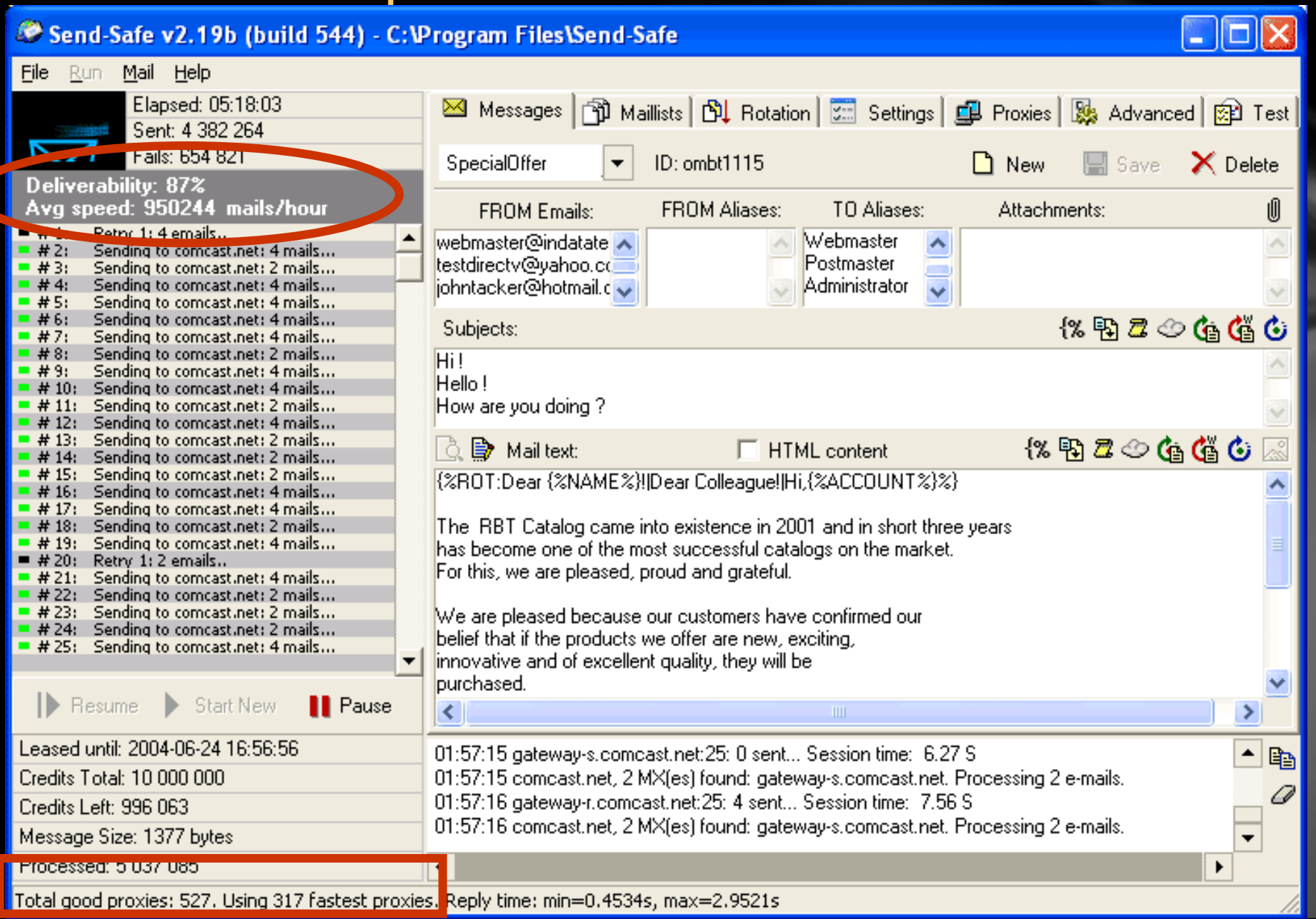
Real-time blacklists stop 15-25% of spam at SMTP connection time

- 85% after message body URI checks

Spammers' objective: evade blacklists

- Botnets come in very handy!

Send-Safe Spam Tool



Where Does Spam Come From?

IP addresses of spam sources are widely distributed across the Internet

- In tracking experiments, most IP addresses appear once or twice; 60-80% are not reachable by traceroute

The vast majority of spam originates from a small fraction of IP address space

- The same fraction that most legitimate email comes from

Spammers exploit routing infrastructure

- Create a short-lived connection to the mail relay, then disappear
- Hijack a large chunk of unallocated "dark" space

Anti-Spam —— by techniques

- Filter:
  - Keyword Filtering 、 Black List and White List、 HASH Filtering 、 Rule-based Filtering 、 Bayesian Filtering
- Sender verification:
  - SMTP Verification 、 Reverse Lookup、DKIM、SenderID、FairUCE、email Fingerprint
- Challenge Authentication:
  - Challenge-response 、 Computational challenge
- References:
  - http://baike.baidu.com/view/1484964.htm
  - http://en.wikipedia.org/wiki/Anti-spam

Anti-Spam —— by the law

- The USA: CAN-SPAM Act of 2003
  - Must not use false or misleading header information. Must not use deceptive subject lines. Must not send email via an agent.
  - Must honor opt-out requests promptly. Must identify the message as an ad.
  - An FTC（Federal Trade Commission） report analyzed the effectiveness（2005.12）.
    - There were 50 prosecutions against spam in the USA.
    - But invalid for spam outside the USA（60%）
    - It is very difficult to collect evidence because of open relay built on Botnets
- 中国: 互联网电子邮件服务管理办法, 2006年3月30日开始实施
  - 禁止未经授权利用他人的计算机系统发送互联网电子邮件;
  - 禁止将采用在线自动收集、字母或者数字任意组合等手段获得的他人的互联网电子邮件地址用于出售、共享、交换或者向通过上述方式获得的电子邮件地址发送互联网电子邮件;
  - 禁止故意隐匿或者伪造互联网电子邮件信封信息;
  - 未经互联网电子邮件接收者明确同意, 不得向其发送包含商业广告内容的互联网电子邮件;
  - 发送包含商业广告内容的电子邮件时, 应当在互联网电子邮件标题信息前注明“广告”或“ＡＤ”字样.

Anti-Spam —— by the law

In April 2005, two siblings were convicted because of spam in Virginia. The brother Jeremy Jaynes was sentenced to 9 years in prison, and the sister Jessica DeGroot was sentenced to a fine of $ 7500.

In May 2007, "Spam King" Robert Alan Soloway was arrested. On that day, the spam under global monitored dropped by 8%. He was sentenced to 47 months imprisonment in 2008, a fine of $ 700 000. He was released on bail in March 2011, and the terms demanded monitoring all his web and email.

In December 2010, San Francisco resident Daniel Balsam gained more than 1 million U.S. dollars via the prosecution of the spam company in 8 years.

In December 2011, the judge of the New York federal district court announced the decision of the case of forging Yahoo mail to a fraud user on Monday. In case the spam disseminator was absent, it ruled that the aggrieved party, yahoo won $610 million in damages. Due to the loss caused by spam dissemination, Facebook was awarded $360 million in legal damages early this year. In the other lawsuit, the Facebook spammer was fined $1 billion.

Review

- The Concept of Trojans, Viruses, and Worms
- Defenses of Malicious Code
- The Concept of Botnet, Spam and DDOS