



CS-3002: Information Systems Security

Lecture # 12: Malicious Software and Botnets

Prof. Dr. Sufian Hameed

Department of Computer Science

FAST-NUCES



FAST-NUCES

Overview

- *Introduction*
- *Malicious Software*
- *Botnets*
- *Botnets detection and Rootkits*



Lets start with a story

Takeover of Torpig Bot by security researchers from
University of California, Santa Barbara





The Botnet Threat

A network of compromised machines (bots) controlled by a botmaster

Responsible for (non-exhaustive list):

- Large-scale network probing (i.e., scanning activities)
- Launching Distributed Denial of Service (DDoS) attacks
- Sending large-scale unsolicited emails (SPAM)
- Click-fraud campaign
- Information theft

Shift from a for-fun activity towards a profit-oriented business

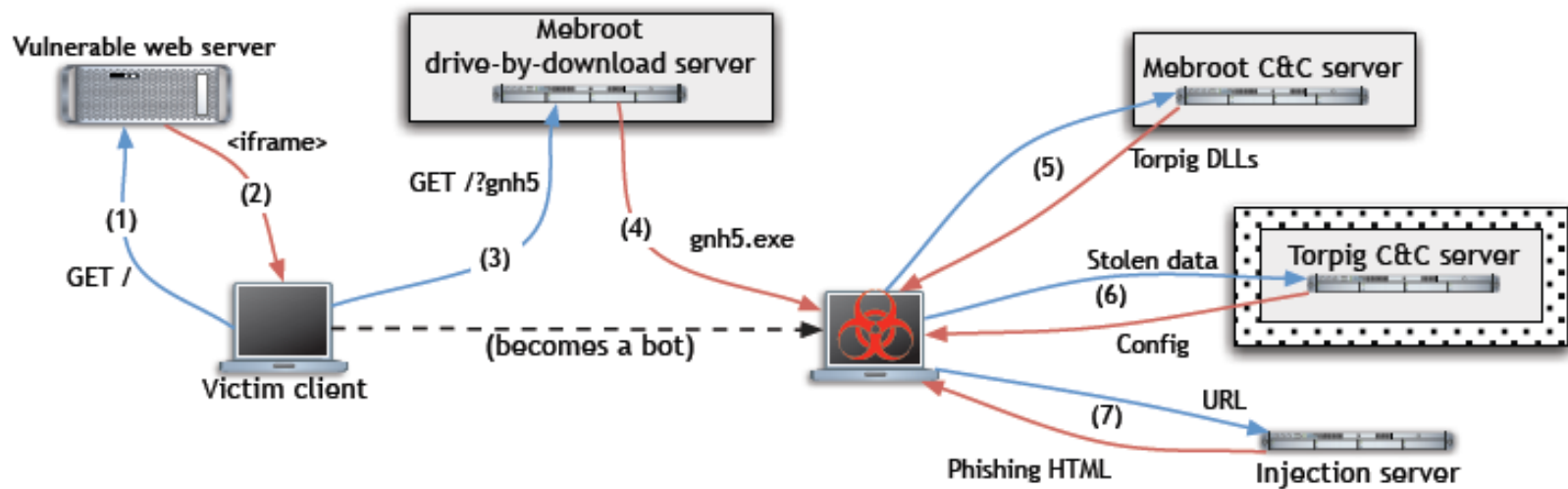


The Torpig Botnet

- Trojan horse
 - Distributed via the Mebroot “malware platform” as DLL
 - Injects itself into 29 different applications
 - Steals sensitive information (e.g., passwords, SSN, credit card numbers)
 - HTTP injection for phishing
 - Uses “encrypted” HTTP as Command & Control (C&C) protocol
 - Uses a resilient approach (domain flux) to contact a C&C server
- Mebroot
 - Sophisticated master boot record based Rootkit
 - Spreads via drive-by downloads (unintended downloads)



The Torpig Botnet



Data Collection Principles

Principle 1: the hijacked botnet should be operated so that any harm and/or damage to victims and targets of attacks would be minimized

- Always responded with okn message
- Never sent new/blank configuration file

Principle 2: the sinkholed botnet (bot manage by good guys) should collect enough information to enable notification and remediation of affected parties

- Worked with law enforcement (FBI and DoD Cybercrime units)
- Worked with bank security officers
- Worked with ISPs



Data Collection

Data Type	Data Items,(#)
Mailbox account	54,090
Email	1,258,862
Form data	11,966,532
HTTP account	411,039
FTP account	12,307
POP account	415,206
SMTP account	100,472
Windows password	1,235,122

Figure : Data items sent to our C&C server by Torpig bots.

Observations:

- Weak passwords
- Credential reuse
- Enables for more successful social engineering attacks



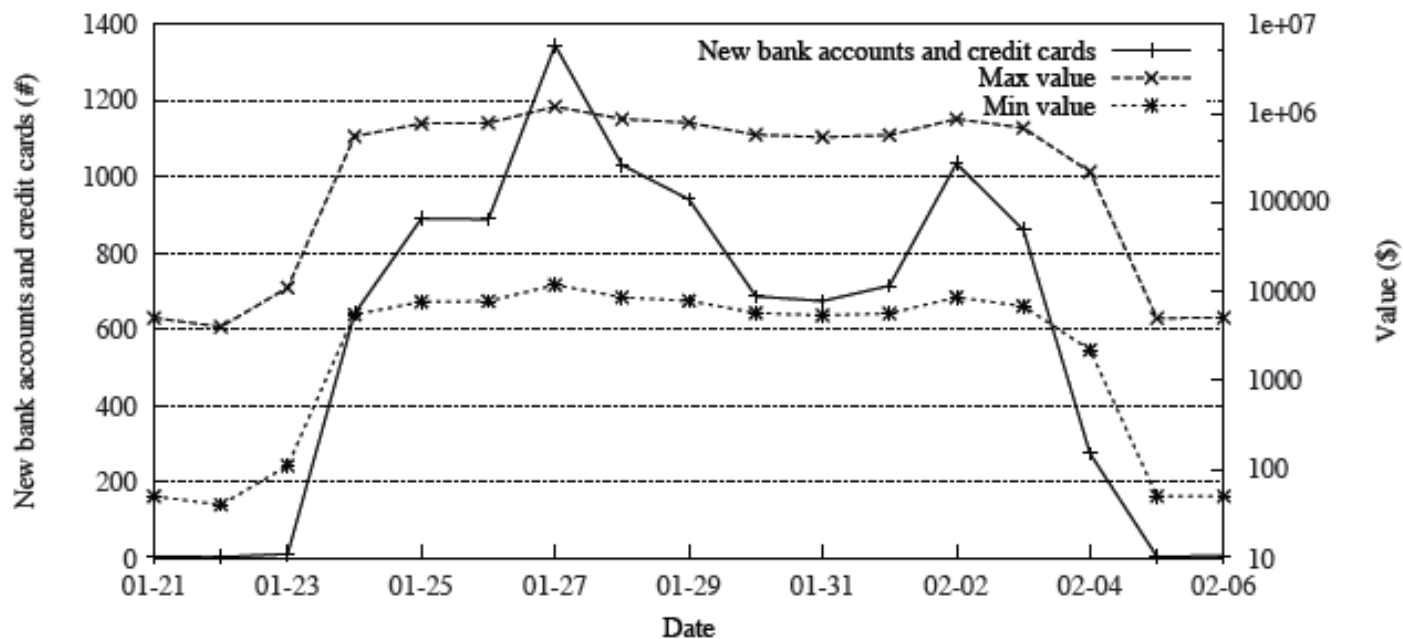
Threats (Theft of Financial Information)

- 8,310 unique accounts from 410 financial institutions
 - Top 5: PayPal (1,770), Poste Italiane, Capital One, E*Trade, Chase
 - 38% of credentials stolen from browsers password manager
- 1,660 credit cards
 - Top 3: Visa (1,056), Mastercard, American Express, Maestro, Discover
 - US (49%), Italy (12%), Spain (8%)
 - Typically, one CC per victim, but there are exceptions



Value of Financial Information

- In a 2008 report on the underground economy, Symantec estimates
 - Value of a stolen credit card details at \$.10 to \$25.00
 - Similarly, bank accounts at \$10.00 to \$1,000.00
- Using such values, 10 days of Torpig data valued at \$83K to \$8.3M



Lessons Learned



Lessons Learned

1. Most security threats start from the web
2. A malicious web page leverages a defect in a program to gain arbitrary code execution
3. The exploit downloads and installs a malware sample, that infects the victim



Malicious Software



Malicious Software

(Malware) refers to any unwanted software and executable code that is used to perform an unauthorized, often harmful, action on a computing device. It is an umbrella-term for various types of harmful software, including viruses, worms, Trojans, rootkits, and botnets.



Taxonomy I

Means of Distribution	Self-Spreading	Virus	Worm
	Non-Spreading	Root-kit Trojan horse	Dialer Spyware Keylogger
		Requires Host	Runs Independently
		Dependency on Host	



Taxonomy II

- Virus
 - Self-replicating
 - Needs a host to infect
 - Boot (Brain virus), overwrite, parasitic, cavity, entry point obfuscation, code integration (W95/Zmist virus)
- Worm
 - Self-replicating, spreads (autonomously) over network
 - Exploits vulnerabilities affecting a large number of hosts
 - Sends itself via email
 - e.g., Internet worm, Netsky, Sobig, Code Red, Blaster, Slammer



Taxonomy III

- Trojan horse
 - Malicious program disguised as a legitimate software
 - Many different malicious actions
 - Spy on sensitive user data
 - Hide presence (e.g., root-kit)
 - Allow remote access (e.g., Back Orifice, NetBus)
- Root-kit
 - Used to keep access to a compromised system
 - Usually hides files, processes, network connections
 - User- and kernel-level (very sophisticated and can hide drivers)



Fighting Malware I

Foremost Goals

- Understand malware behaviors
- to (automatically) identify and classify families of malware
- (to) automatically generate effective malware detection models
- Collect malware samples
 - How about infection strategies?
- Analyze samples
 - Static analysis
 - Studying a program's properties without executing it
 - Reverse engineering may be hampered (e.g., obfuscation, encryption)
- Dynamic analysis
 - Studying a program's properties by executing it
 - Environment-limited analysis



Fighting Malware II

- Extract (and generalize) malicious behavior
 - Host
 - Network
- Generate and deploy detection models

Hard Problems

- Lack of general definition of malicious behavior
- Cat-and-mouse game: attackers have much freedom
- Victims often (unwittingly) help attackers



Botnets



Botnets I

- Bot
 - Autonomous programs performing tasks
 - More recent trend in malicious development
- Benign bots
 - First bots were programs used for Internet Relay Chat (IRC)
 - React to events in IRC channels
 - Typically offer useful services

- Early definition of bot

An IRC 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.



Botnets II

- Eggdrop bot (1993)
 - Used to manage IRC chat channels when the operator was away
 - Still maintained, see <http://eggheads.org>
 - Malicious IRC bots started to evolve
 - Takeover wars to control certain IRC channels
 - Trash talking (flooding)
 - Also involved in Denial of Service (DoS) to force IRC net split (disconnection from previous connection)
 - IRC proxies to hide attackers' origin
 - A number of parallel, malicious developments



History I

- How did we get here?
 - Early 1990s: IRC bots
 - Automated management of IRC channels
 - 1999 --- 2000: Distributed DoS tools (distribution)
 - Trinoo, TFN2k, Stacheldraht
 - 1999 --- 2000: Trojan Horse (remote control)
 - BackOrice, BackOrice2k, SubSeven
 - 2001 --- today: Worms (spreading)
 - Code Red, Blaster, Sasser



History II

- Bots today
 - Malware (backdoor, Trojan) running on compromised machines
 - Incorporates different modules to carry out malicious tasks (spamming, DoS, . . .)
 - Remote controlled by criminal entity (called bot master or bot herder)
- Bots are incorporated in network of compromised machines
 - Botnets (sizes up to hundreds of thousands of infected machines)
- Botnets
 - Main vehicle for carrying out criminal activities
 - Financial motivation



Botnets

- How do botnets get created?
 - Infection and spreading
- How are bots (and botnets) controlled?
 - Command and control channel (C&C), robustness features (e.g., fast flux, domain flux, push/pull/P2P)
- What are botnets used for?
 - Criminal applications
- How can we mitigate the problem?



Creation

- Host infected by one of
 - Network worm (vulnerabilities)
 - Email attachment
 - Trojan version of program (ever heard of P2P?)
 - Drive-by-downloads (malicious web sites)
 - Existing backdoor (from previous infection)
- Specialized services (PPI, Exploit-as-a-Service)




Drive-By Downloads

- Malicious scripts
 - Injected into legitimate sites (e.g., via SQL injection)
 - Hosted on malicious sites (URLs distributed via spam)
 - Embedded into ads
- Drive-by downloads
 - Attacks against web browser and/or vulnerable plug-ins
 - Typically launched via client-side scripts (JavaScript, VBScript)
- Redirection
 - Landing page redirects to malicious site (e.g., via iframe)
 - Makes management easier
 - Customize exploits (browser version), serve each IP only once



Propagation Technique: Remote exploit + drive-by-download



BNET BUSINESS NETWORK: **BNET** | **TECHREPUBLIC** | **ZDNET**

ZDNet

Search:

[Members Log In](#) | [Newsletters](#) | [Site Assistance](#)

[Home](#) | [News & Blogs](#) | [Videos](#) | [White Papers](#) | [Downloads](#)

Zero Day

Ryan Naraine, Dancho Danchev & Adam O'Donnell

Get Zero Day via: [Mobile](#) [RSS](#) [Email Alerts](#) Bios: [Ryan's Bio](#) [Dancho's Bio](#)

Pick a blog category [view](#)

May 20th, 2008

Over 1.5 million pages affected by the recent SQL injection attacks

Posted by Dancho Danchev @ 4:05 pm

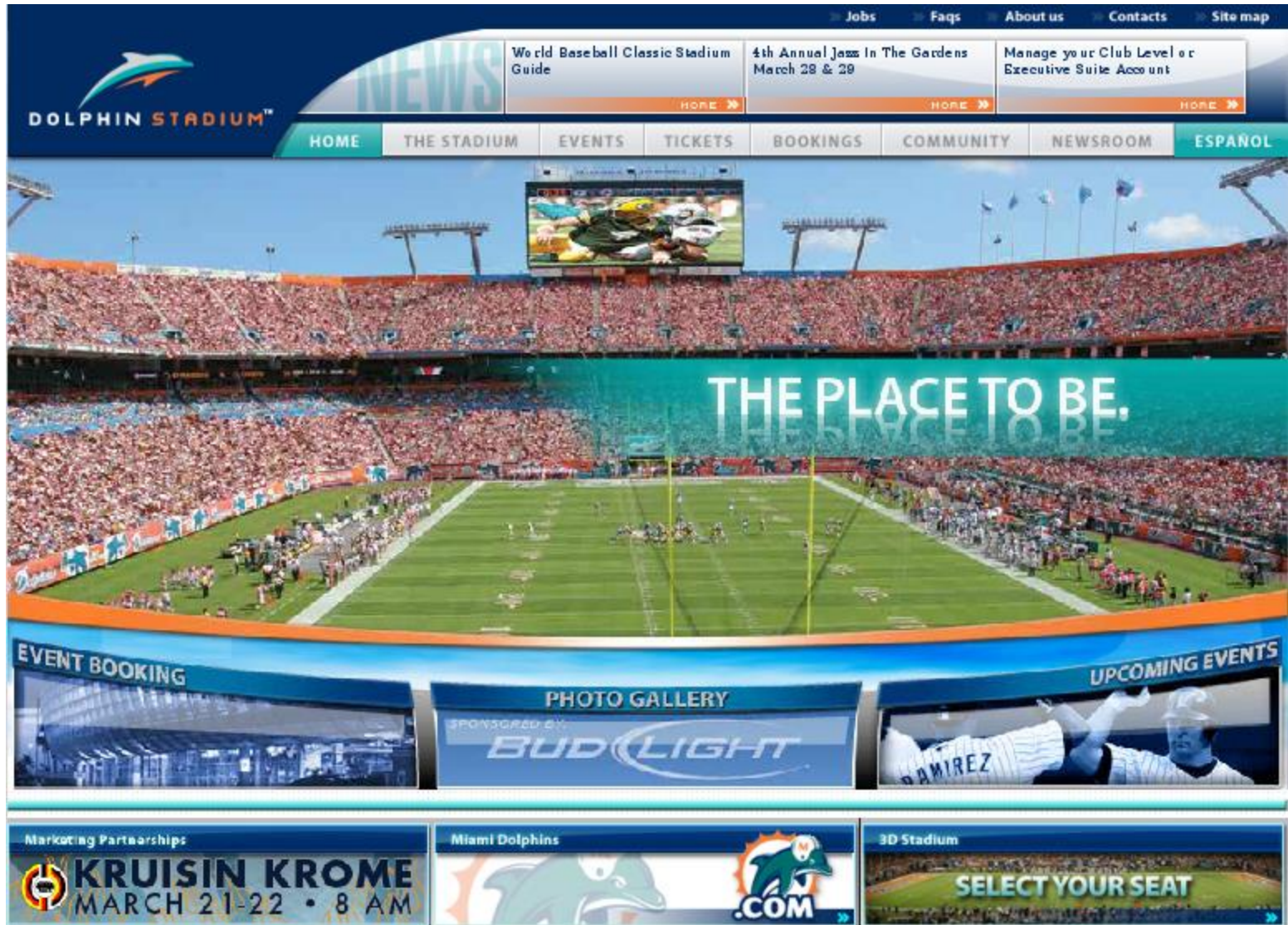
Categories: [Botnets](#), [Viruses and Worms](#)

Tags: [Shadowserver Foundation](#), [SQL Injection](#), [Malware Domains](#), [Dancho Danchev](#)

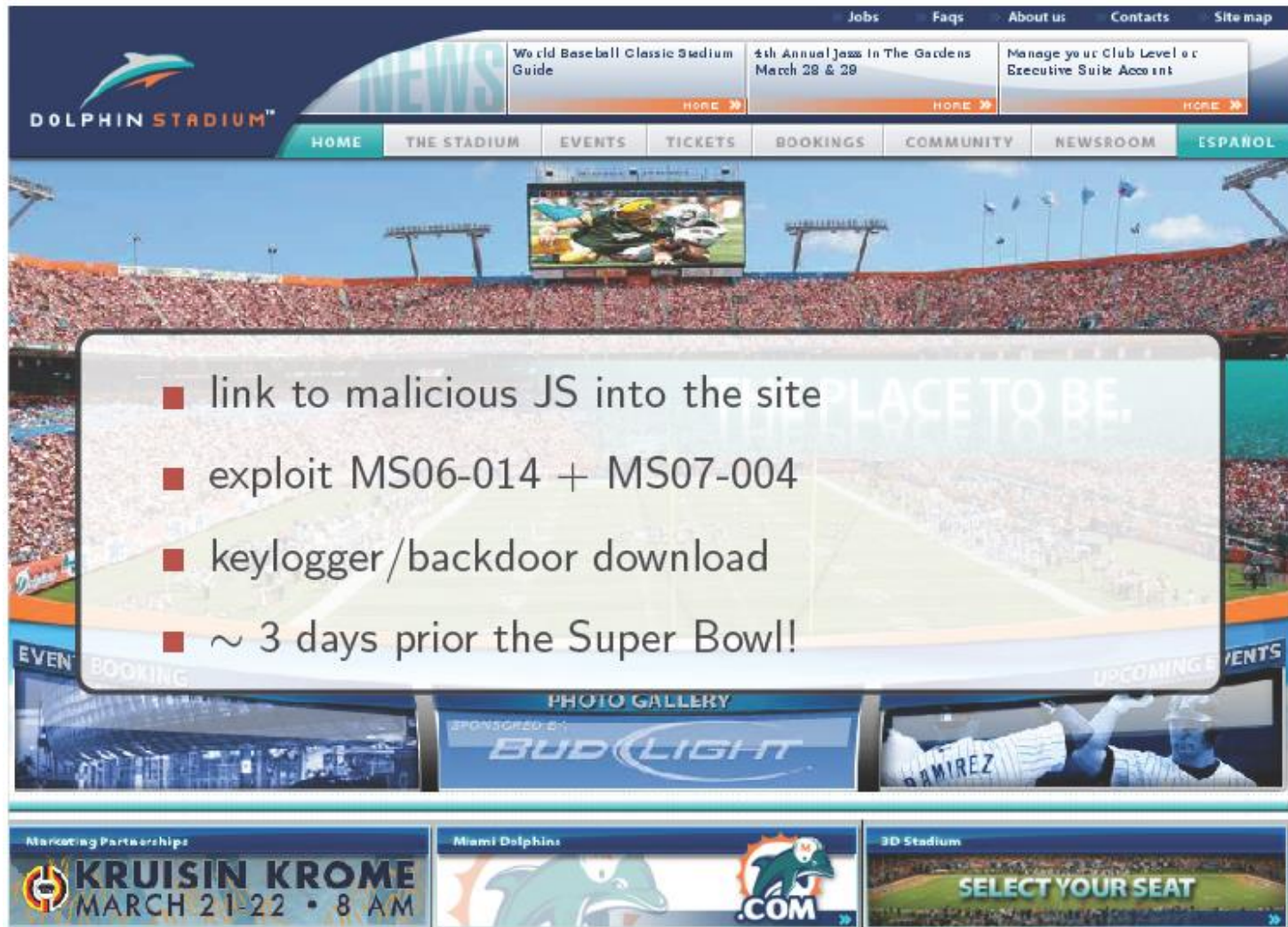
Qualified buyers now get
on most 2009



Propagation Technique: Remote exploit + drive-by-download



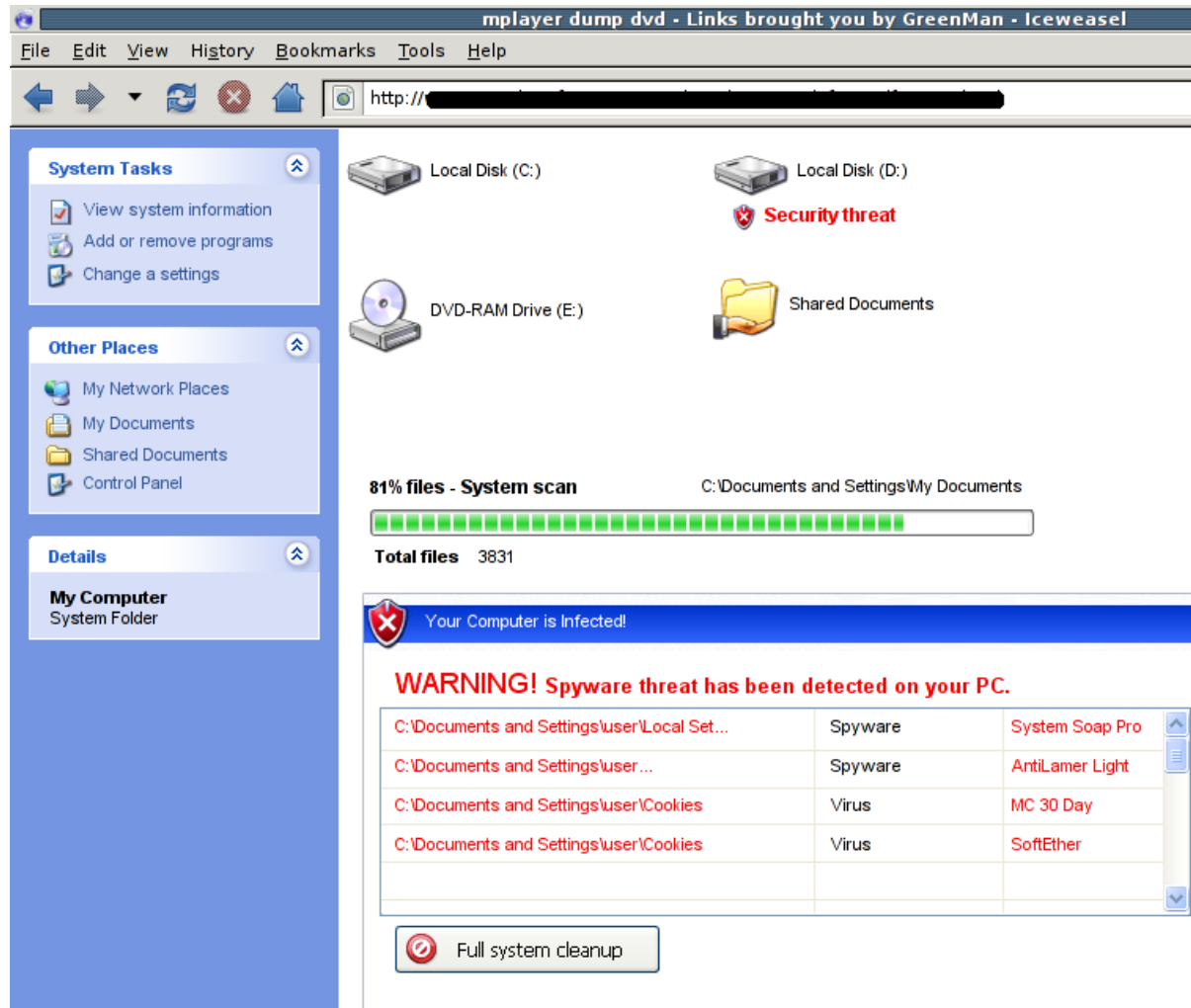
Propagation Technique: Remote exploit + drive-by-download



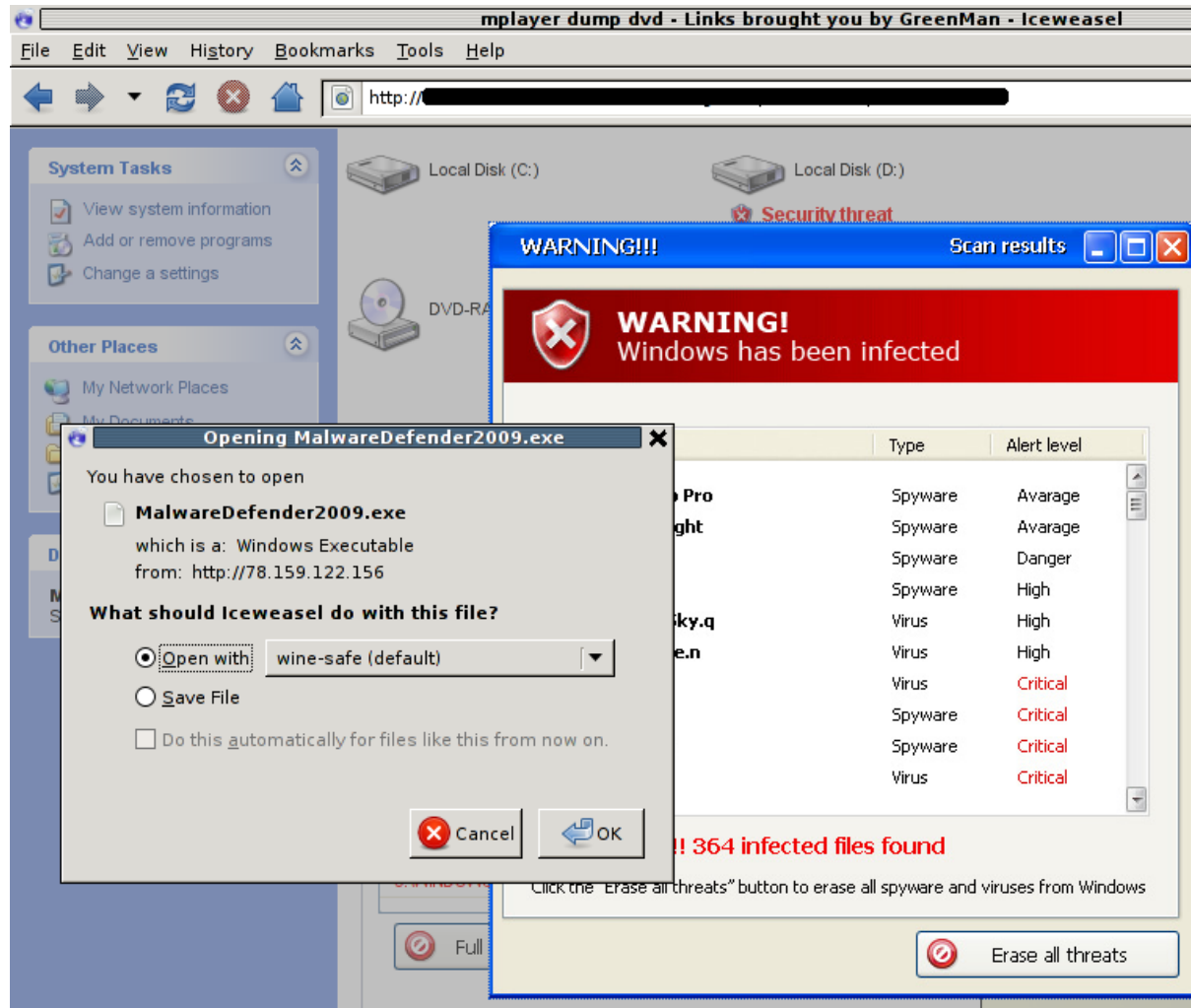
The screenshot shows the Dolphin Stadium website. At the top, there is a navigation bar with links: Jobs, Faqs, About us, Contacts, Site map. Below this is a header section with the Dolphin Stadium logo and three featured links: "World Baseball Classic Stadium Guide", "4th Annual Jazz In The Gardens March 28 & 29", and "Manage your Club Level or Executive Suite Account". A main navigation bar includes links: HOME, THE STADIUM, EVENTS, TICKETS, BOOKINGS, COMMUNITY, NEWSROOM, and ESPAÑOL. The main content area features a large image of the stadium filled with fans. Overlaid on this image is a semi-transparent box containing a list of propagation techniques. Below the main image, there are sections for "EVEN BOOKING", "PHOTO GALLERY" (sponsored by Bud Light), and "UPCOMING EVENTS". At the bottom, there are three banners: "Marketing Partnerships" for Kruisin Krome (March 21-22, 8 AM), "Miami Dolphins" with the team logo and ".COM", and "3D Stadium" with a "SELECT YOUR SEAT" button.

- link to malicious JS into the site
- exploit MS06-014 + MS07-004
- keylogger/backdoor download
- ~ 3 days prior the Super Bowl!

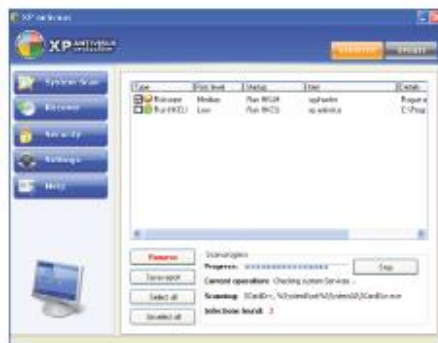
Propagation Techniques: Rogue Antivirus



Propagation Techniques: Rogue Antivirus



Propagation Techniques: Rogue Antivirus



Command & Control



Command & Control

Centralized control

IRC Commands published in IRC channels

(irc.krienaicw.pl #djdjeiu)

HTTP Commands published in web pages

(<http://www.myspace.com/angelairiejs/>)

Distributed control

P2P Commands and/or “commander” address published in the P2P network (more resilient to take down)

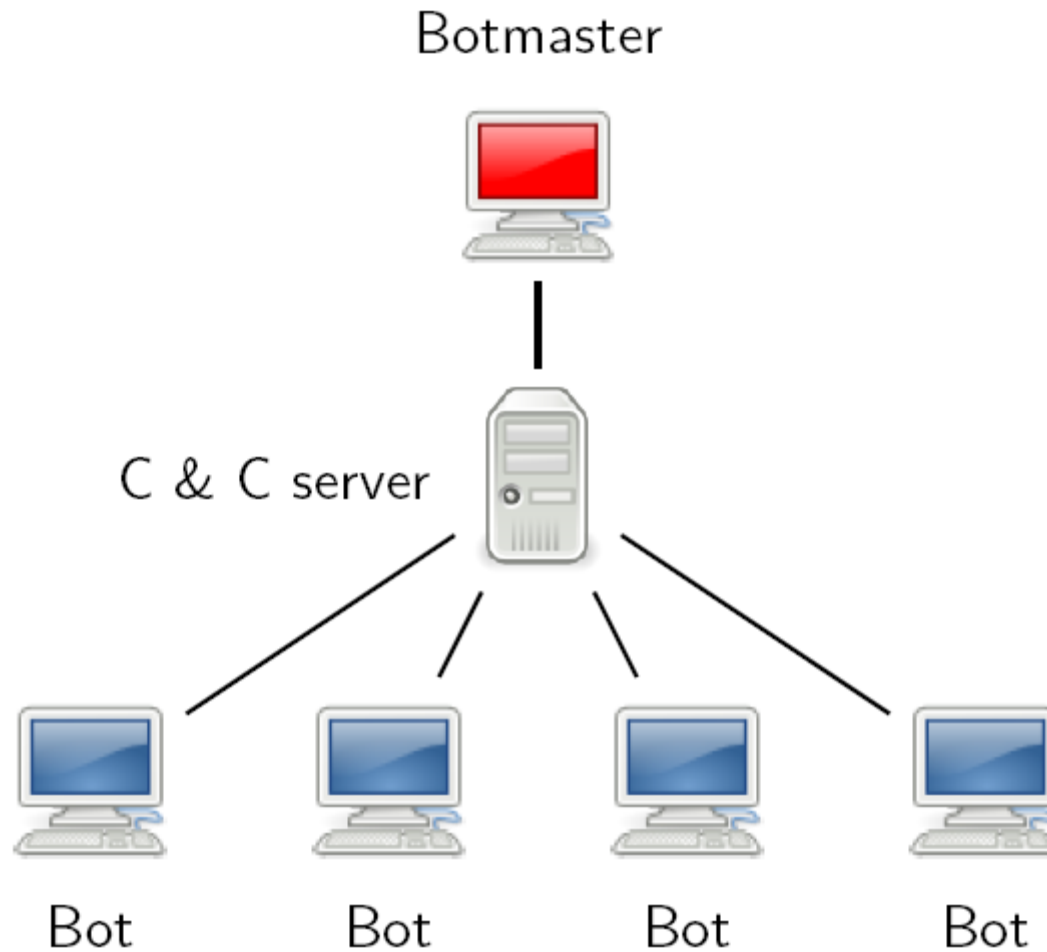
Push vs. Pull

Push The bot silently waits for commands from the “commander”

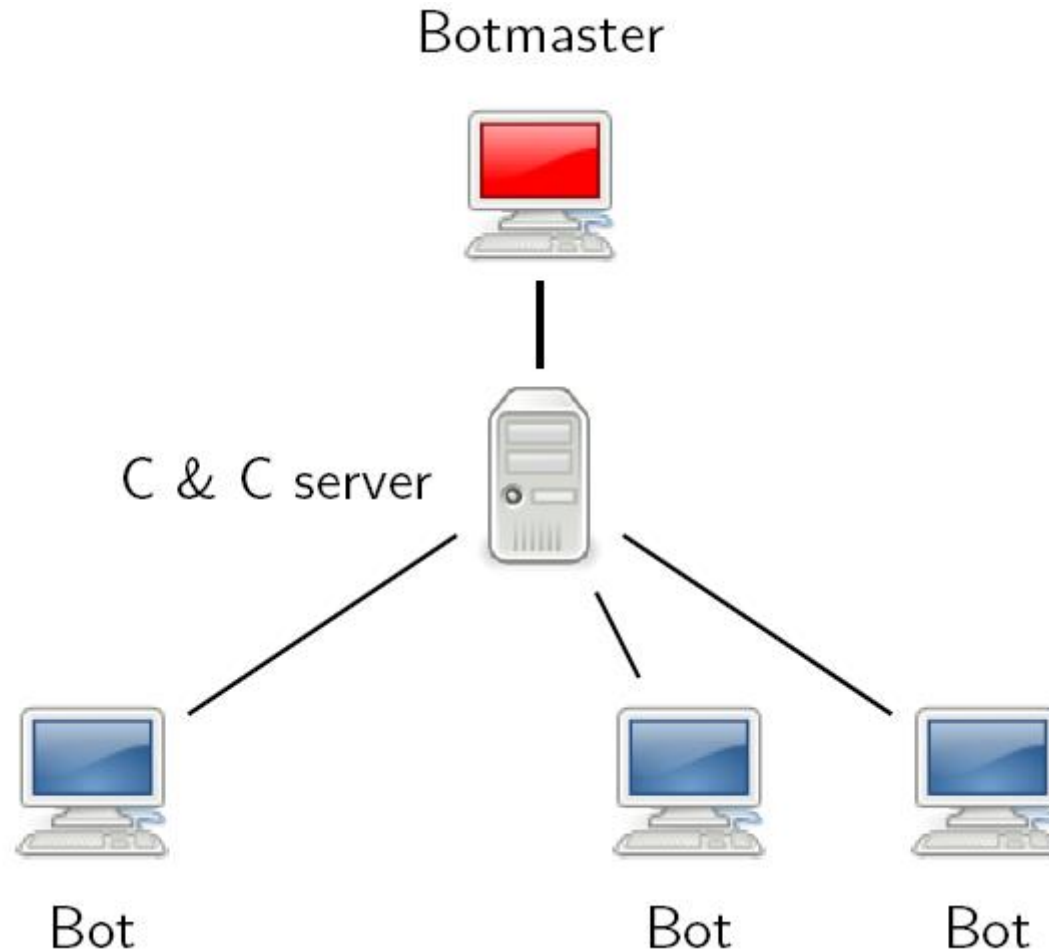
Pull The bot repeatedly queries the “commander” to see if there is a new work to do



IRC based botnet

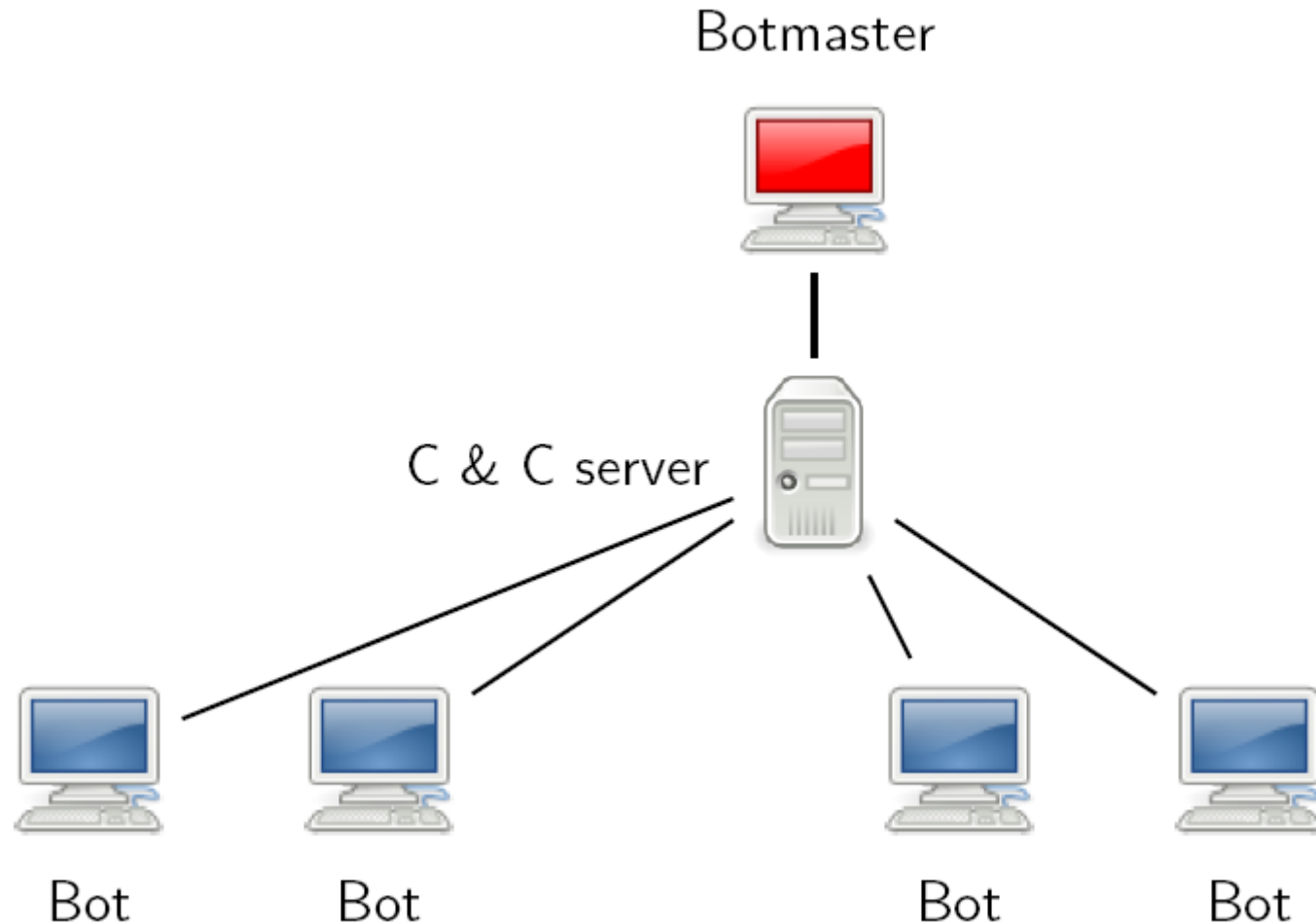


IRC based botnet



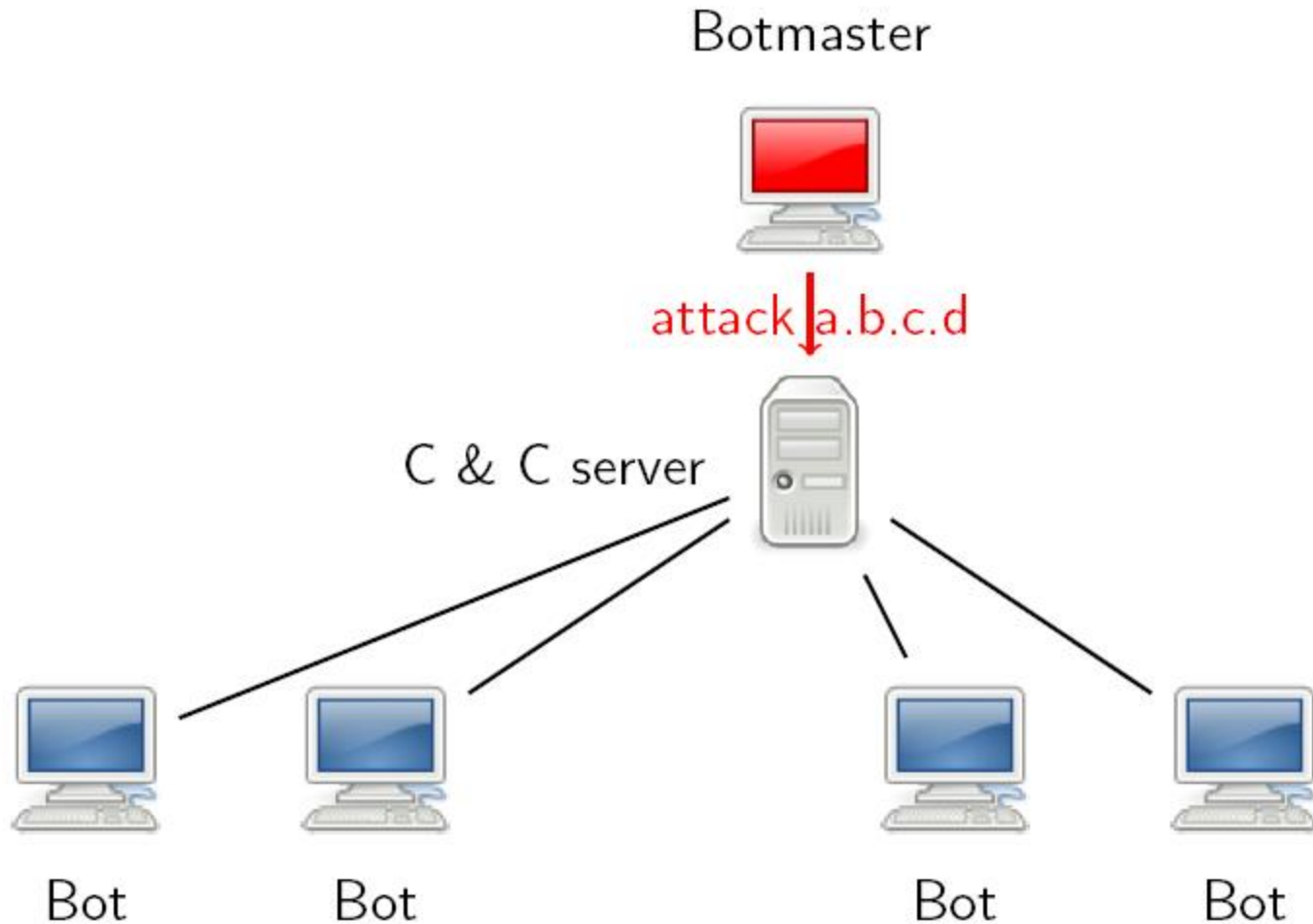
- The bots continuously (re)connect to the C&C server

IRC based botnet



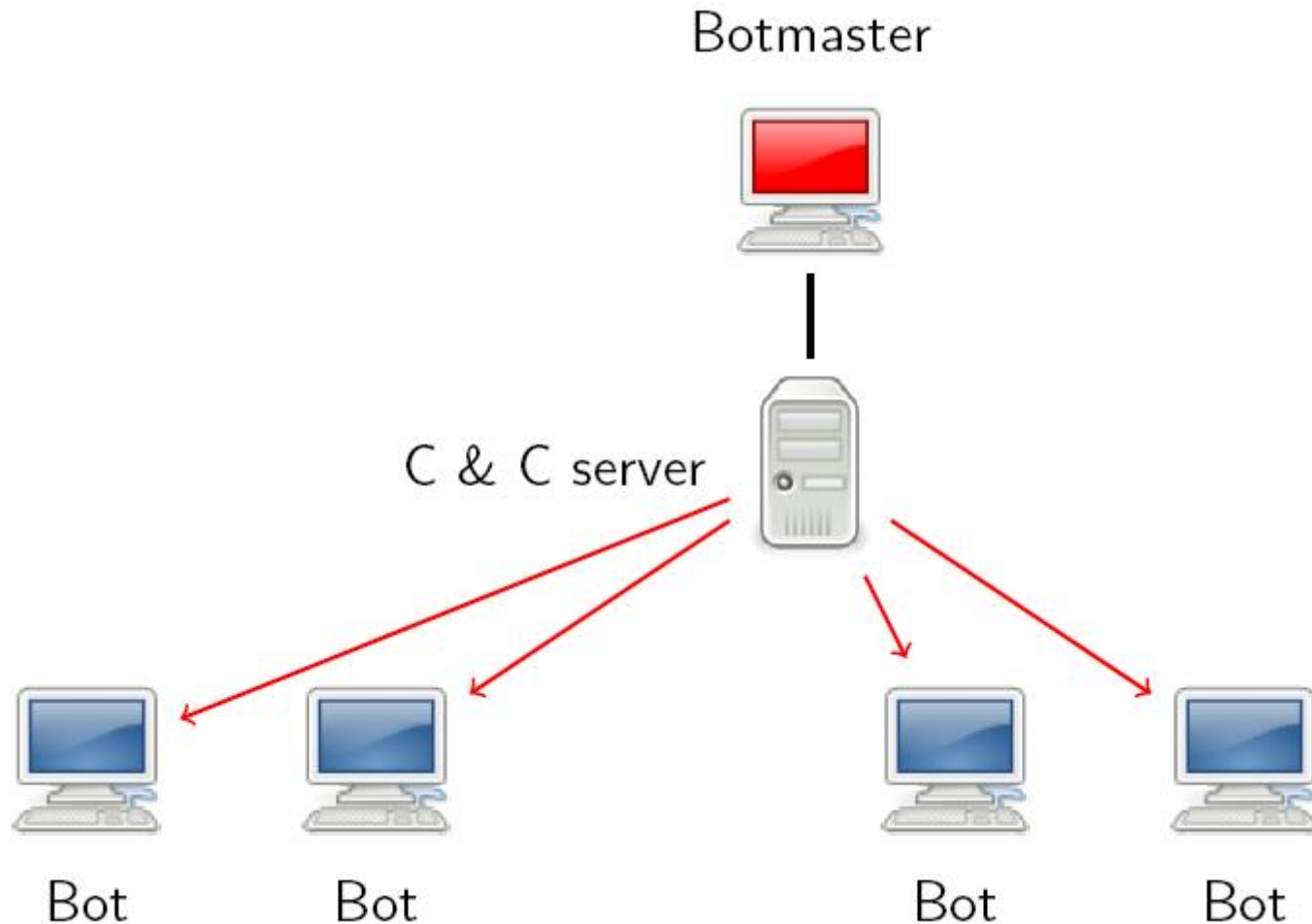
- The bots continuously (re)connect to the C&C server

IRC based botnet



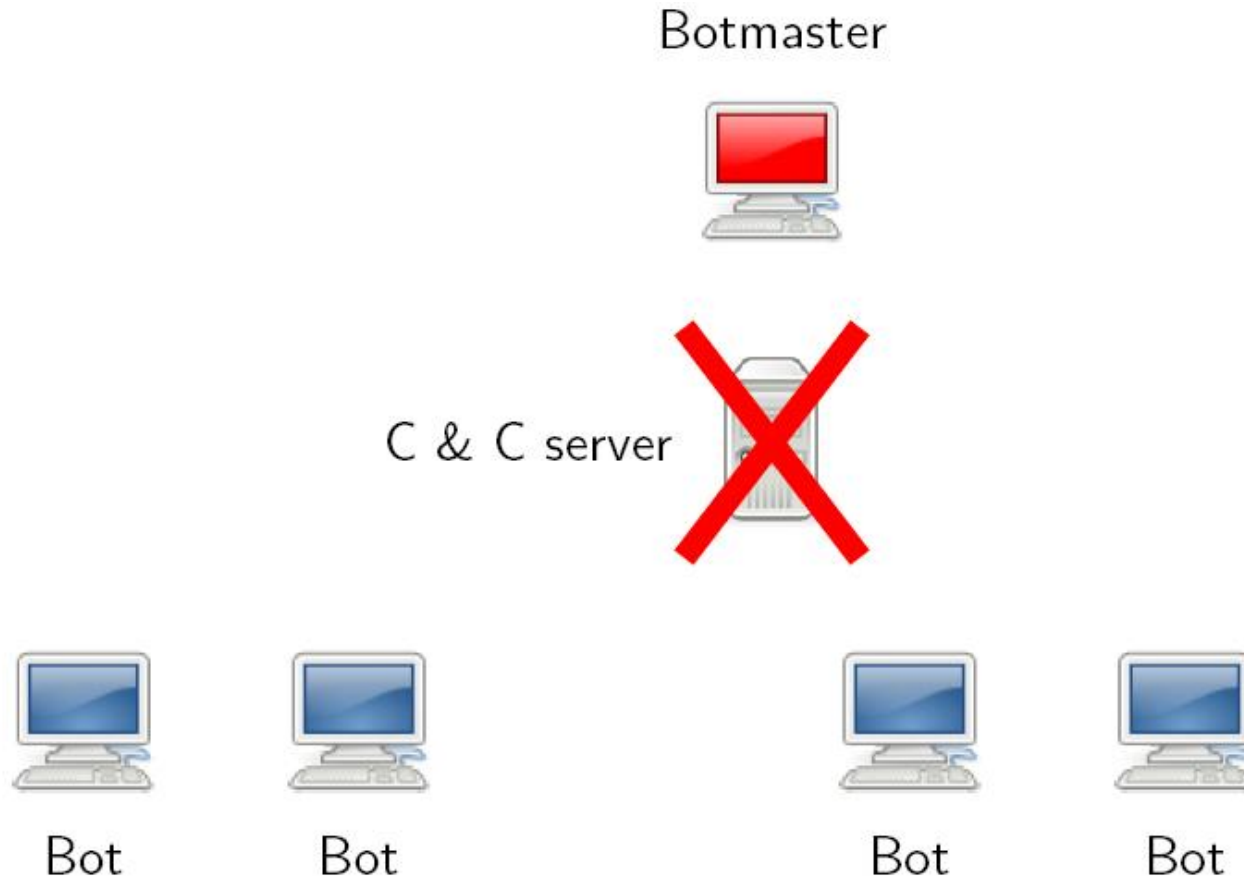
- The botmaster sends a command to the C&C server

IRC based botnet



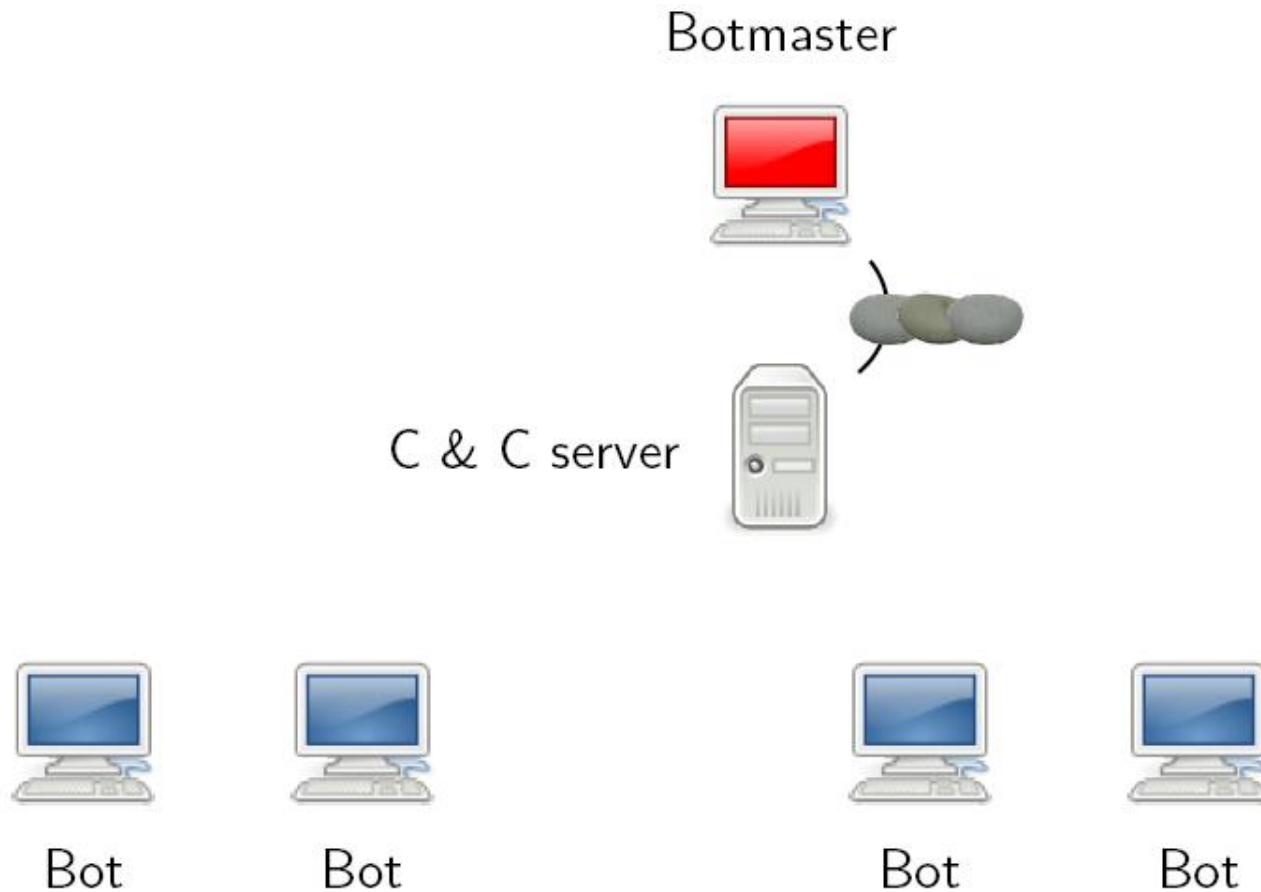
- The C&C server forwards the command to all connected bots

IRC based botnet



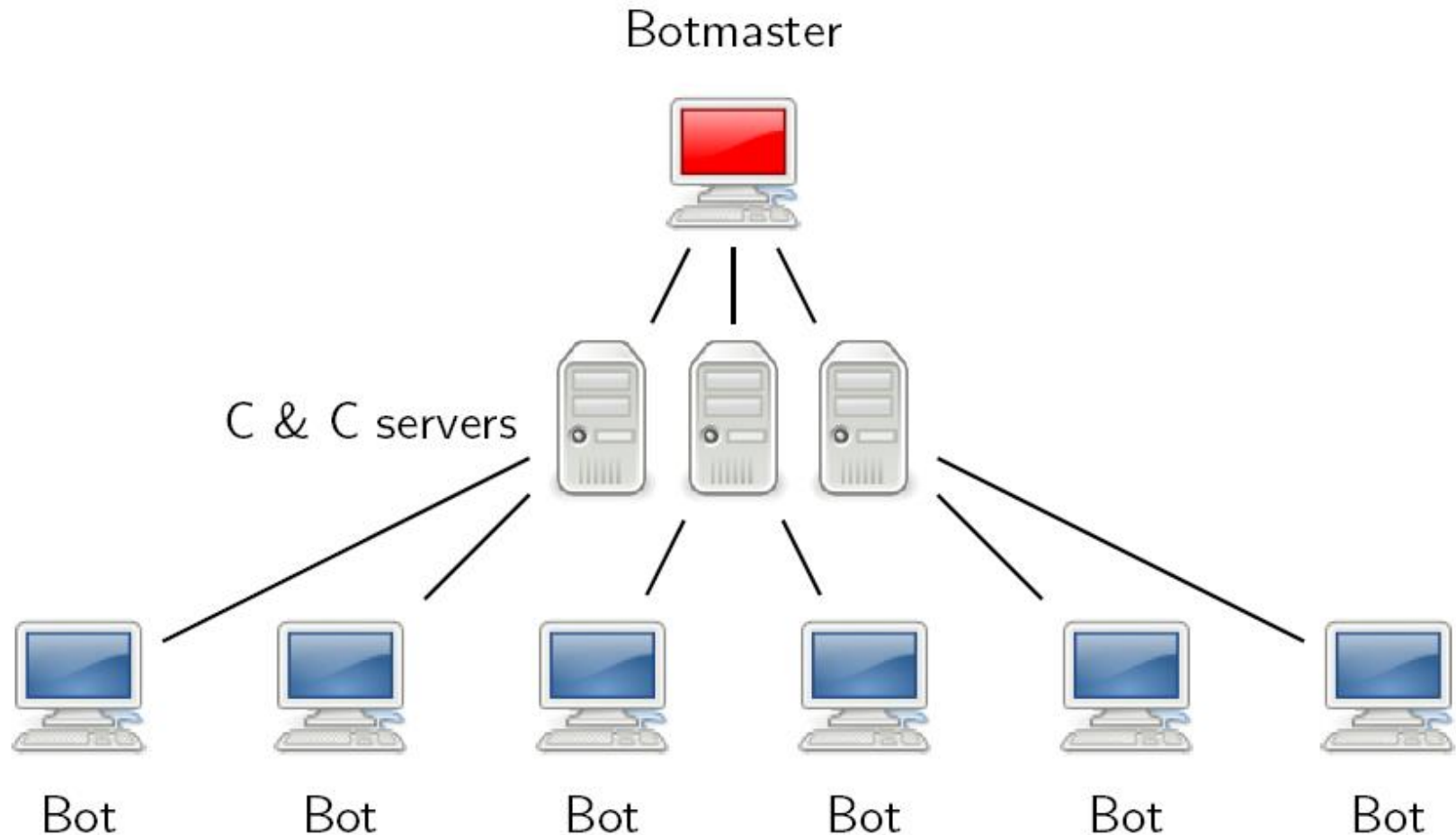
- If the C&C server is isolated, the botmaster loses control of all the bots

IRC based botnet



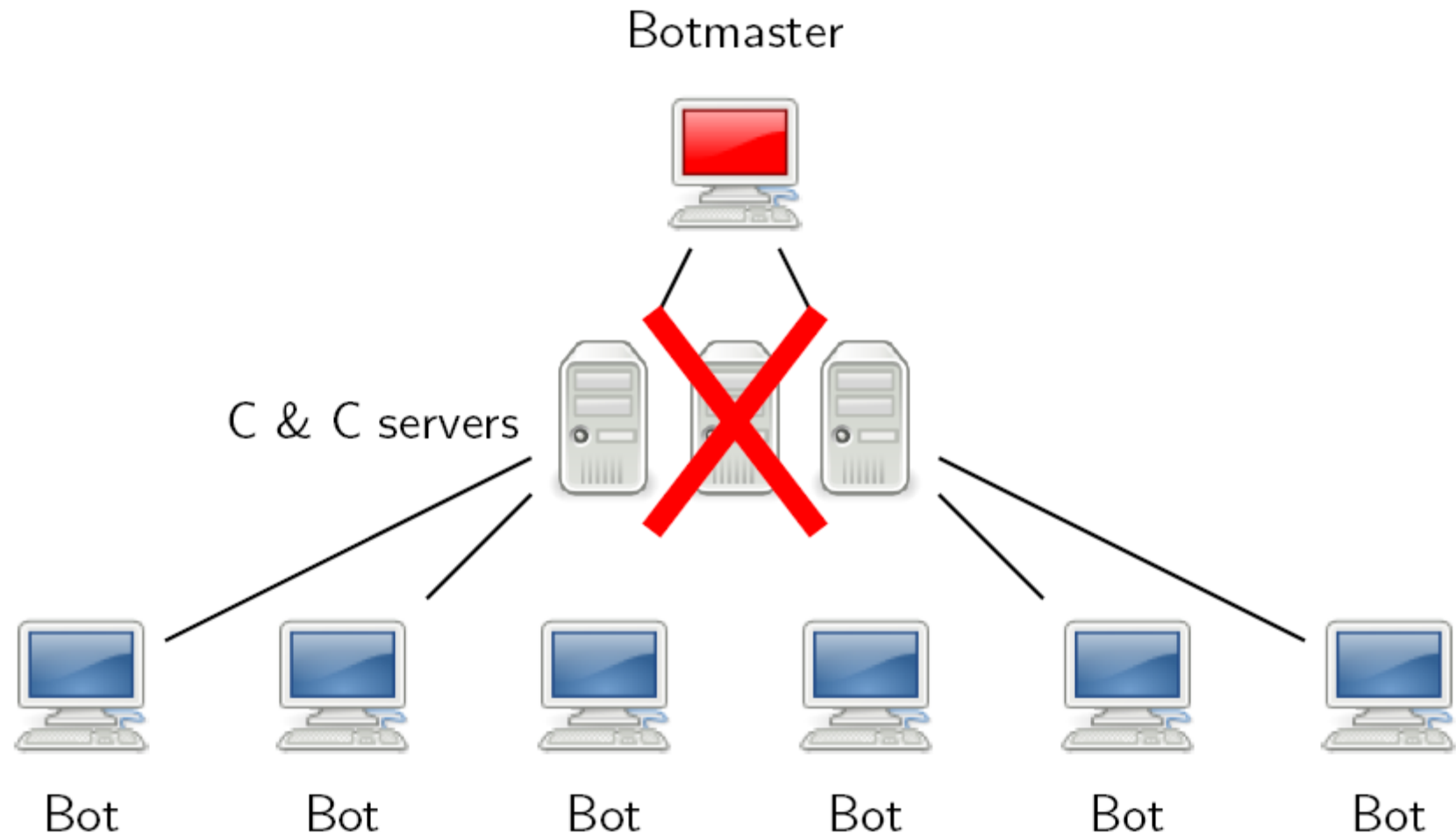
- It could be very difficult to identify the botmaster! (e.g., coffee shops or a chain of compromised machines)

IRC based botnet



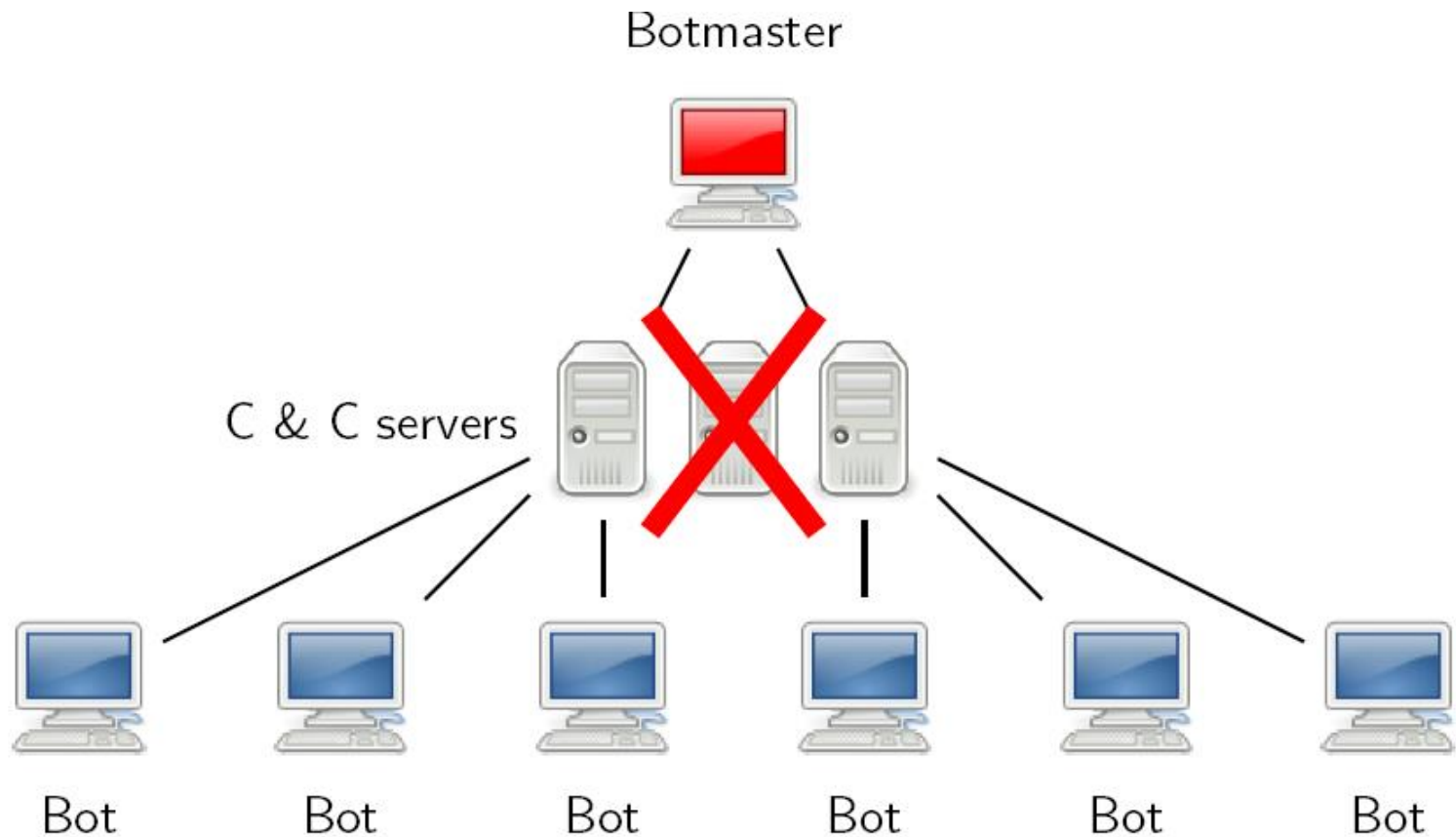
- Multiple C&C servers can be used simultaneously

IRC based botnet



- When a C&C server is isolated the bots connect automatically to the others

IRC based botnet



- The use of multiple C&C servers increases the lifetime of the botnet

HTTP based botnet

- Very similar to botnet based on IRC
- C&C in \pull" mode

```
http://91.207.4.122/spm/s_tasks.php?id=468831...  
http://akhadqwd.blogspot.com/  
http://www.myspace.com/sakjfuje/  
  
...  
<table>  
  <tr>  
    <td class="asjkdha">ddos www.bank.com</td>  
  </tr>  
</table>  
...
```

- C&C trace is difficult to identify
- Difficult to block at network level (e.g., firewall)
- Difficult to block at DNS level (e.g., domains blacklisting)



Dynamic rendez-vous points

How does a bot locate the C&C server?

1. Hardcoded IP address
2. FustFlux: Hardcoded FQDN or dynamically generated FQDNs (1 FQDN → 1 or more IP addresses)
3. DomainFlux: Hardcoded URL or dynamically generated URLs (generated different domains as time passes by)
4. Search keys in the P2P network

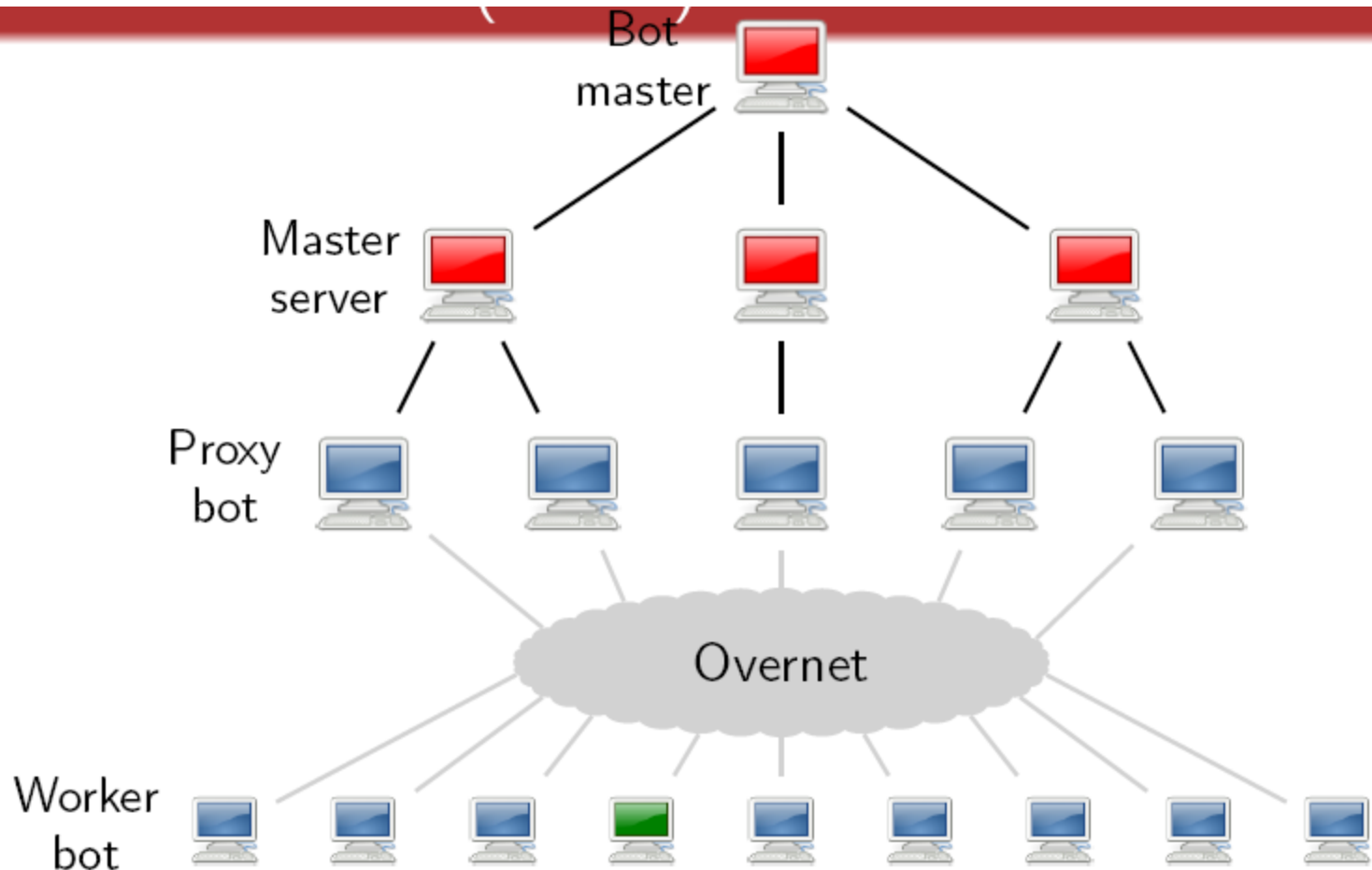


Preventing the rendez-vous

- Network level ACLs
- DNS ACLs
- HTTP ACLs
- DDoS against the C&C server?
- It is possible to perform clustering to determine that a domain is generated by a DomainFlux

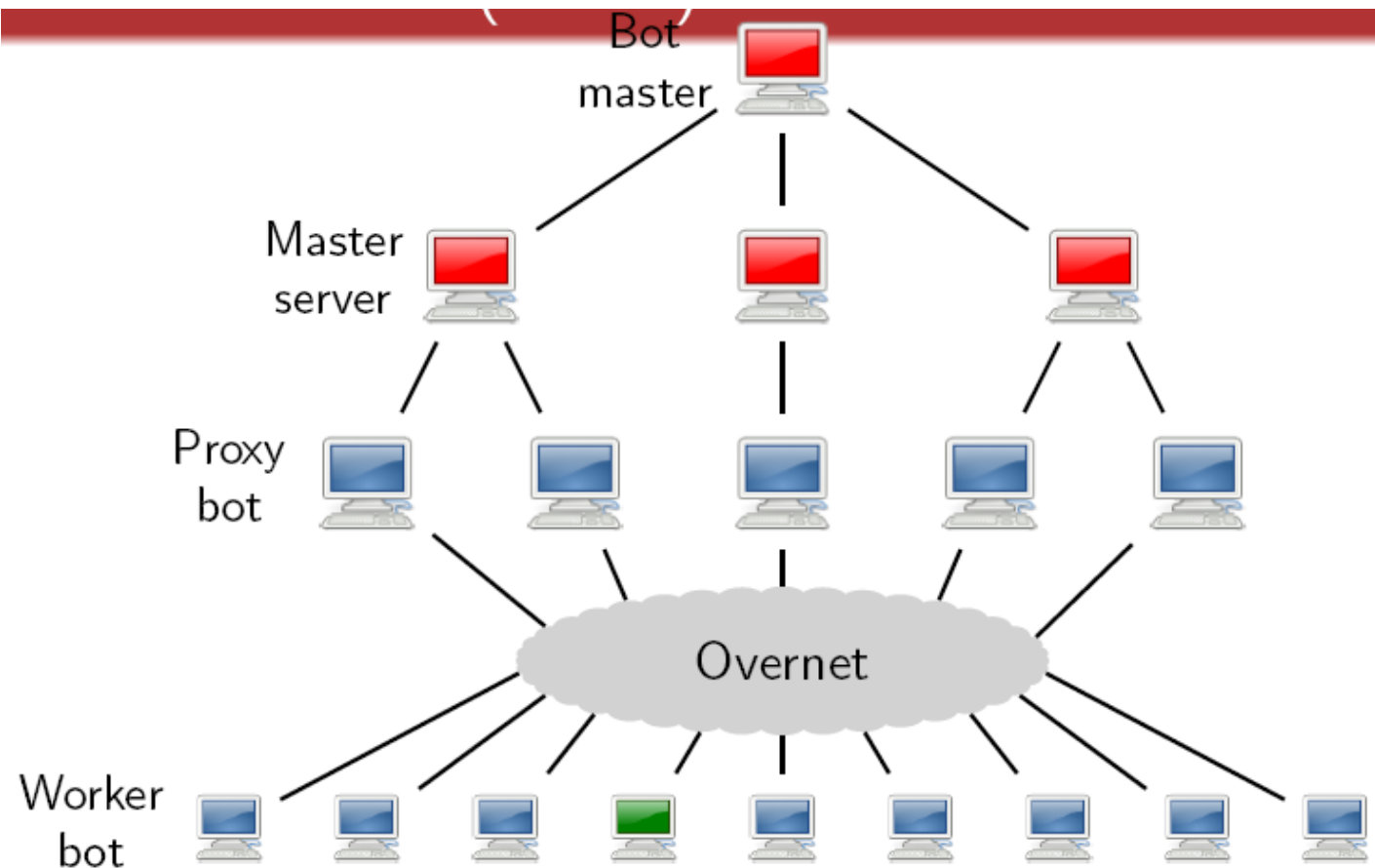


P2P based botnet (Storm)



- P2P protocol based on Overnet (derived from Kadmelia)

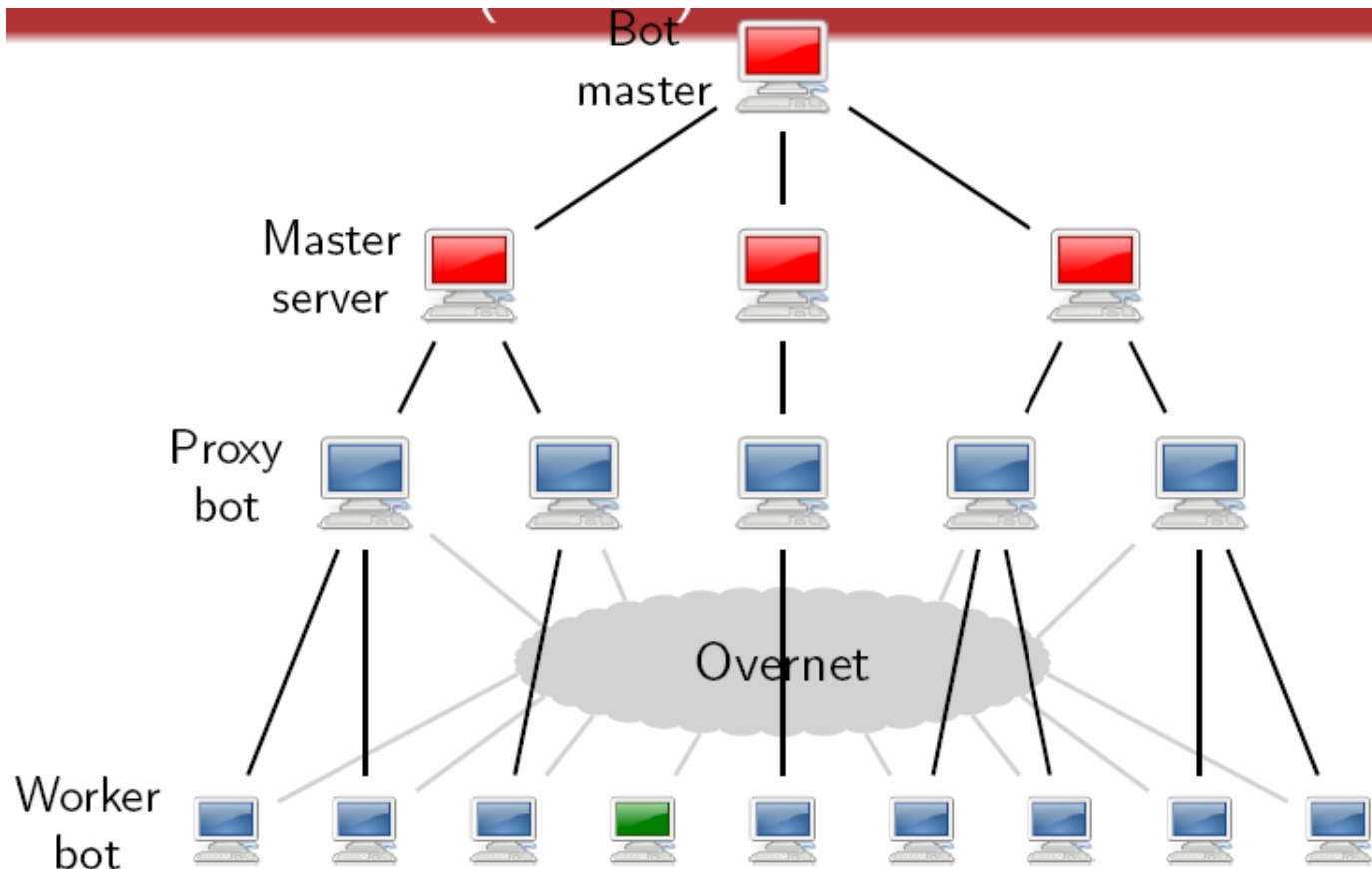
P2P based botnet (Storm)



- Search for keys in the P2P network to locate the proxies (keys are dynamic and published by the proxies through the P2P)

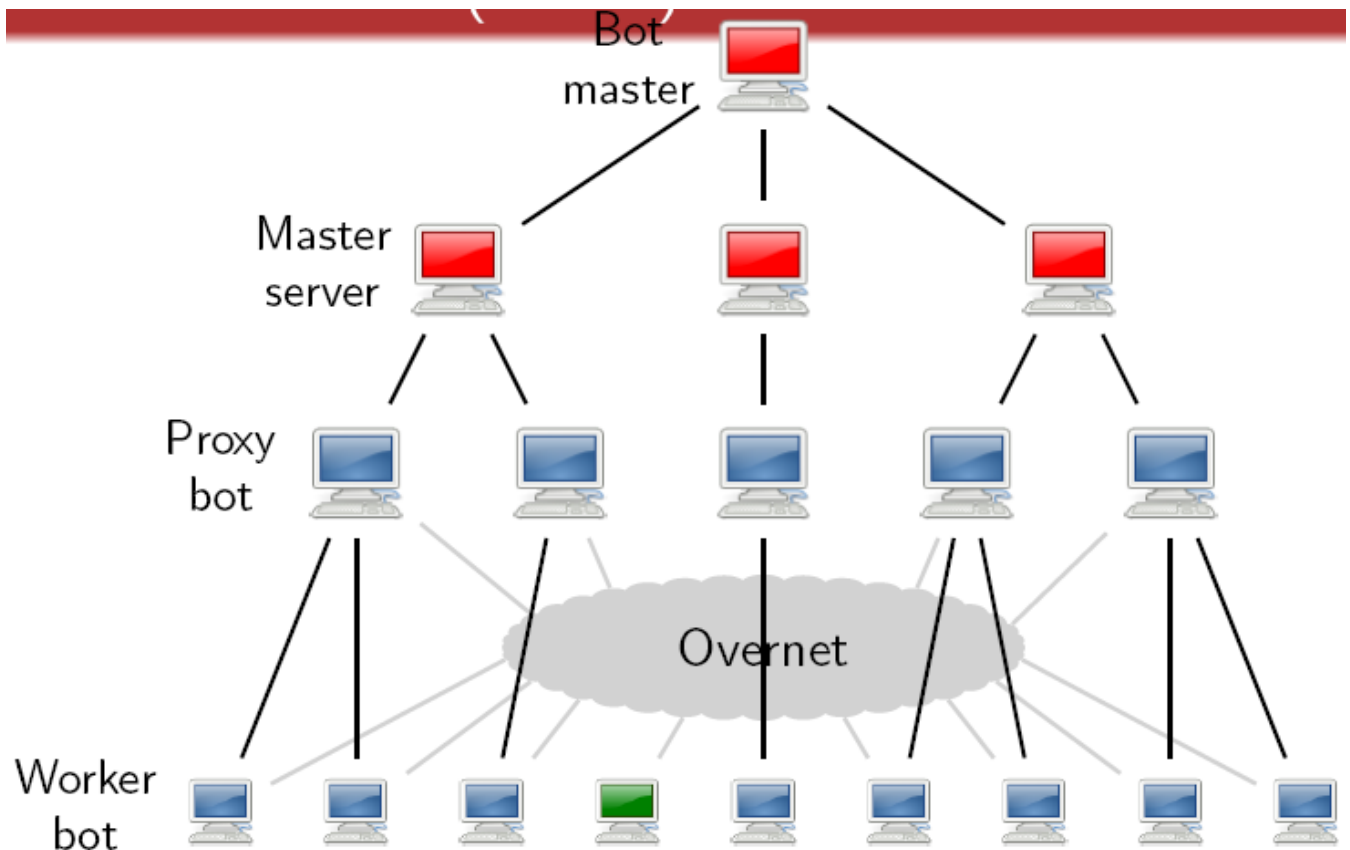
P2P based botnet (Storm)

- Connect to the proxy and wait for commands (each key is associated with the IP and the port of a new proxy)



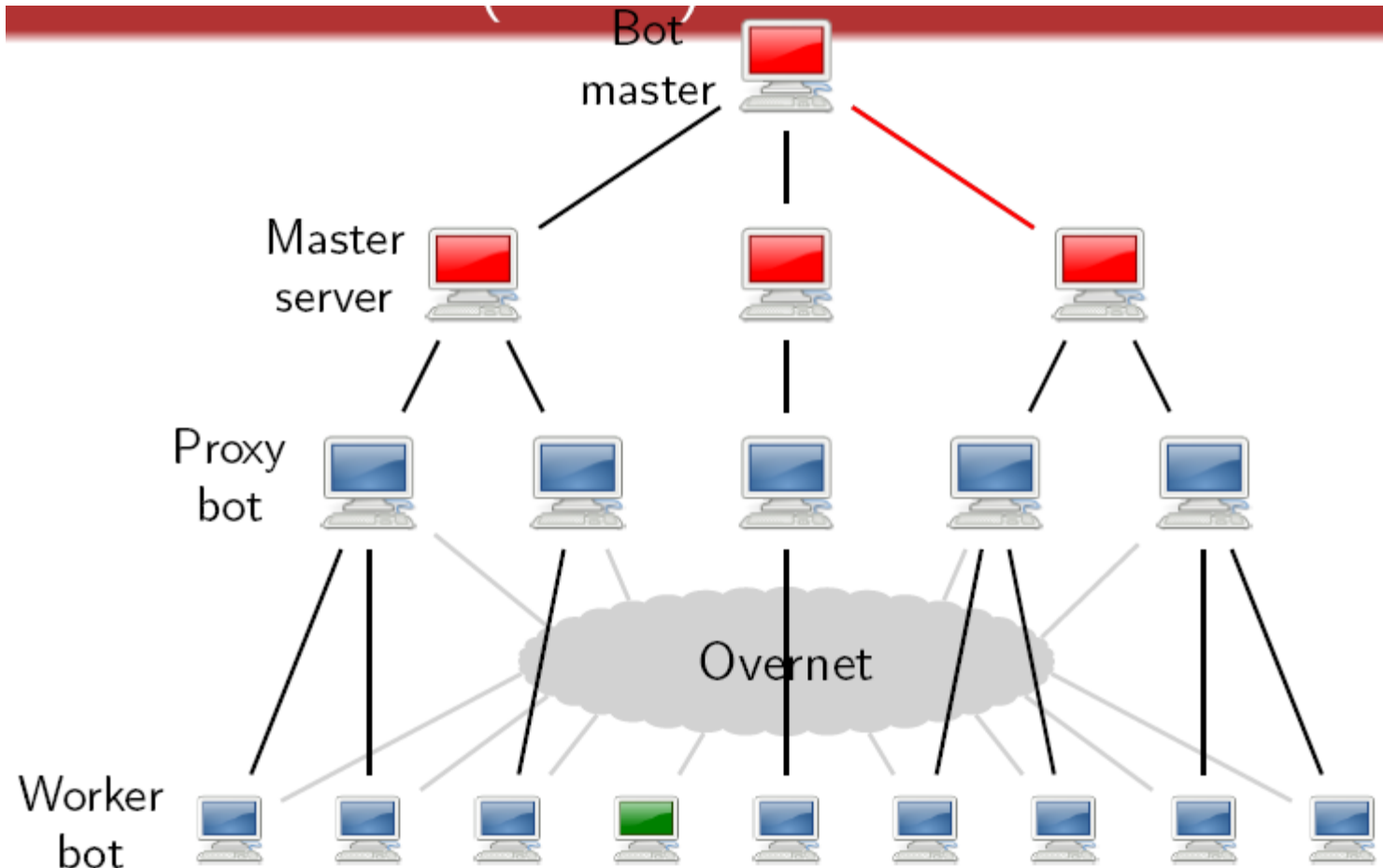
P2P based botnet (Storm)

- The worker bot connects to the proxy, authenticates itself and waits for commands



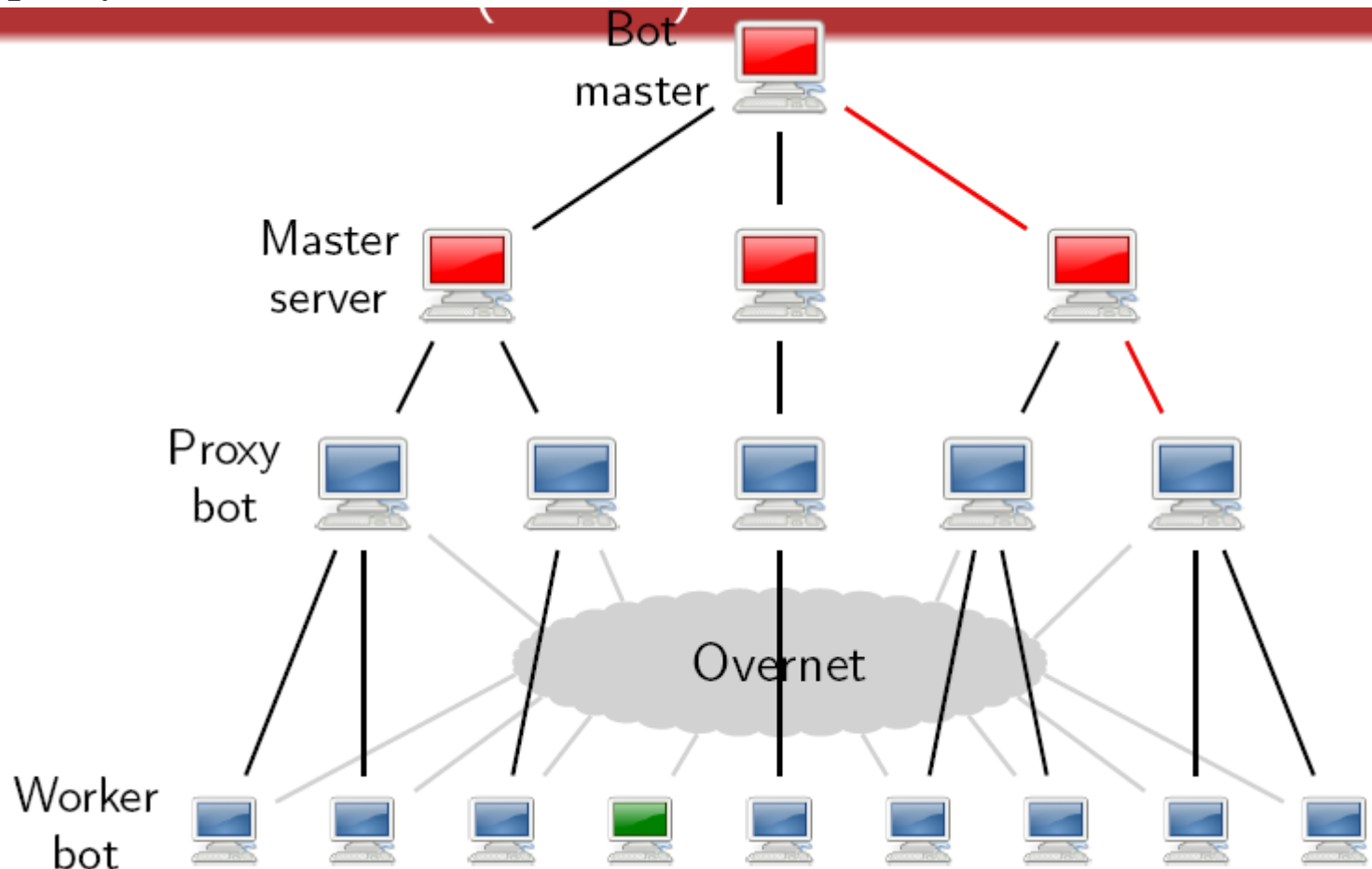
P2P based botnet (Storm)

- The proxy forwards command from the master to the workers and vice versa



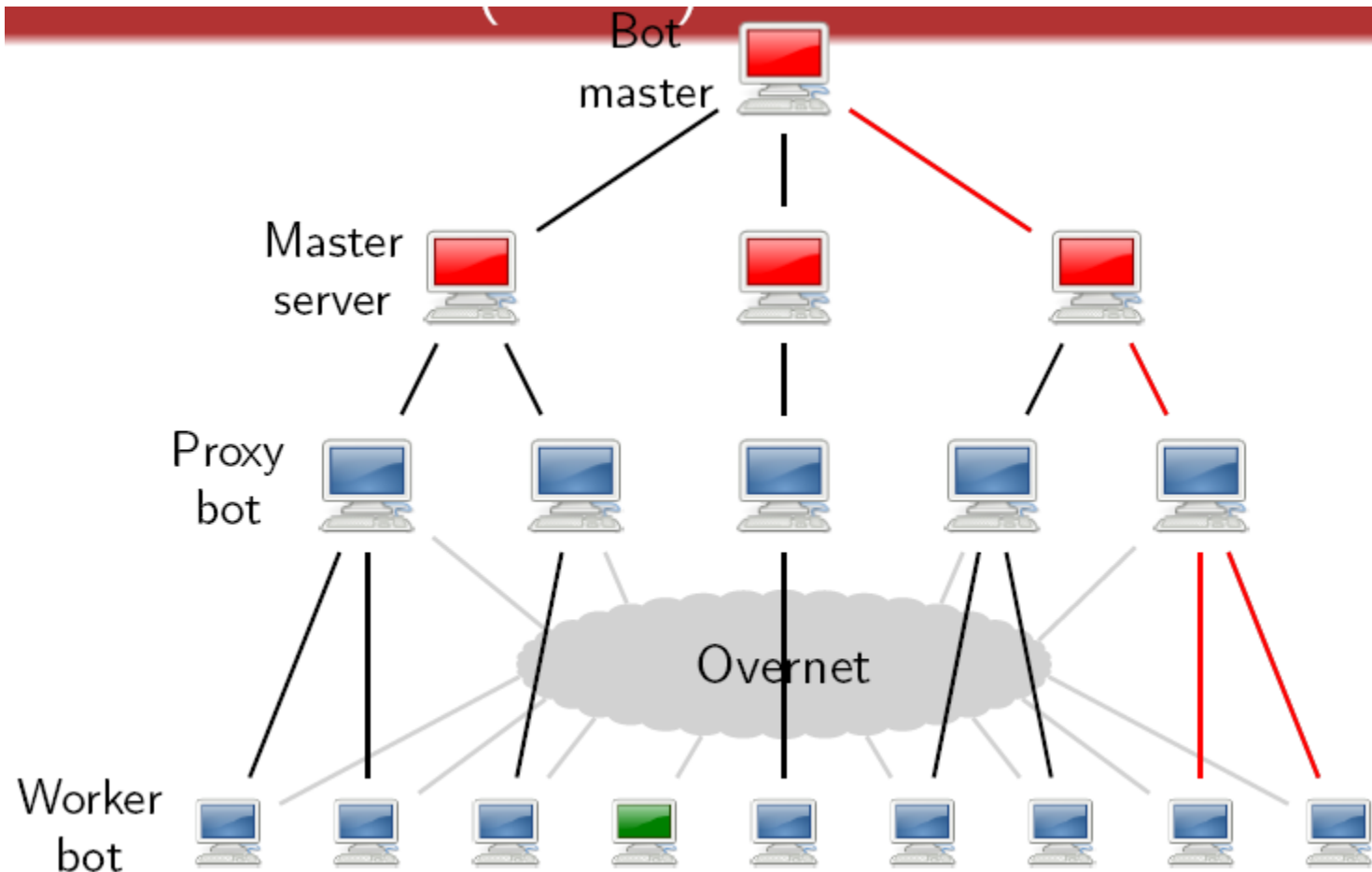
P2P based botnet (Storm)

- The proxy forwards command from the master to the workers and vice versa



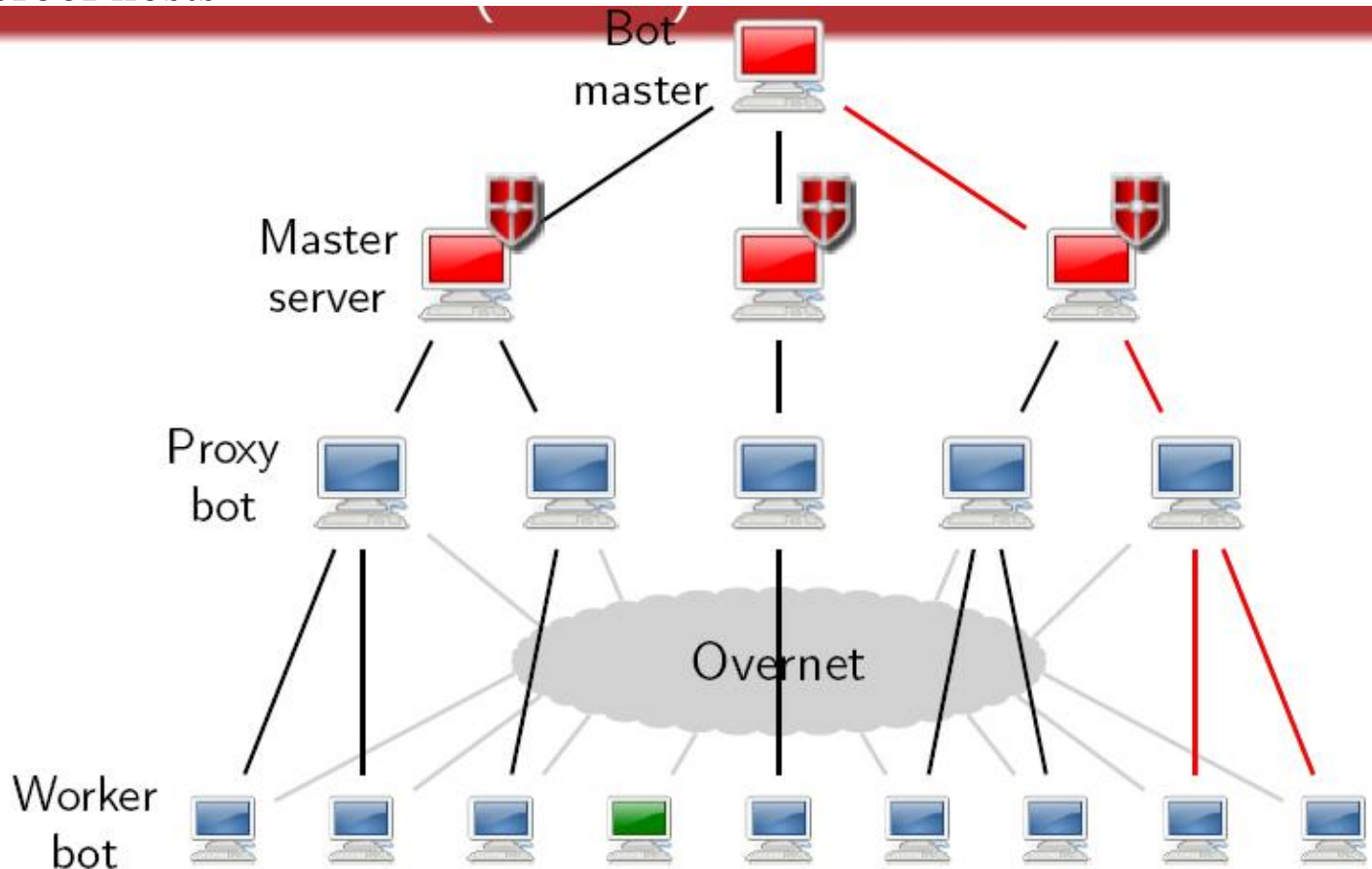
P2P based botnet (Storm)

- The proxy forwards command from the master to the workers and vice versa



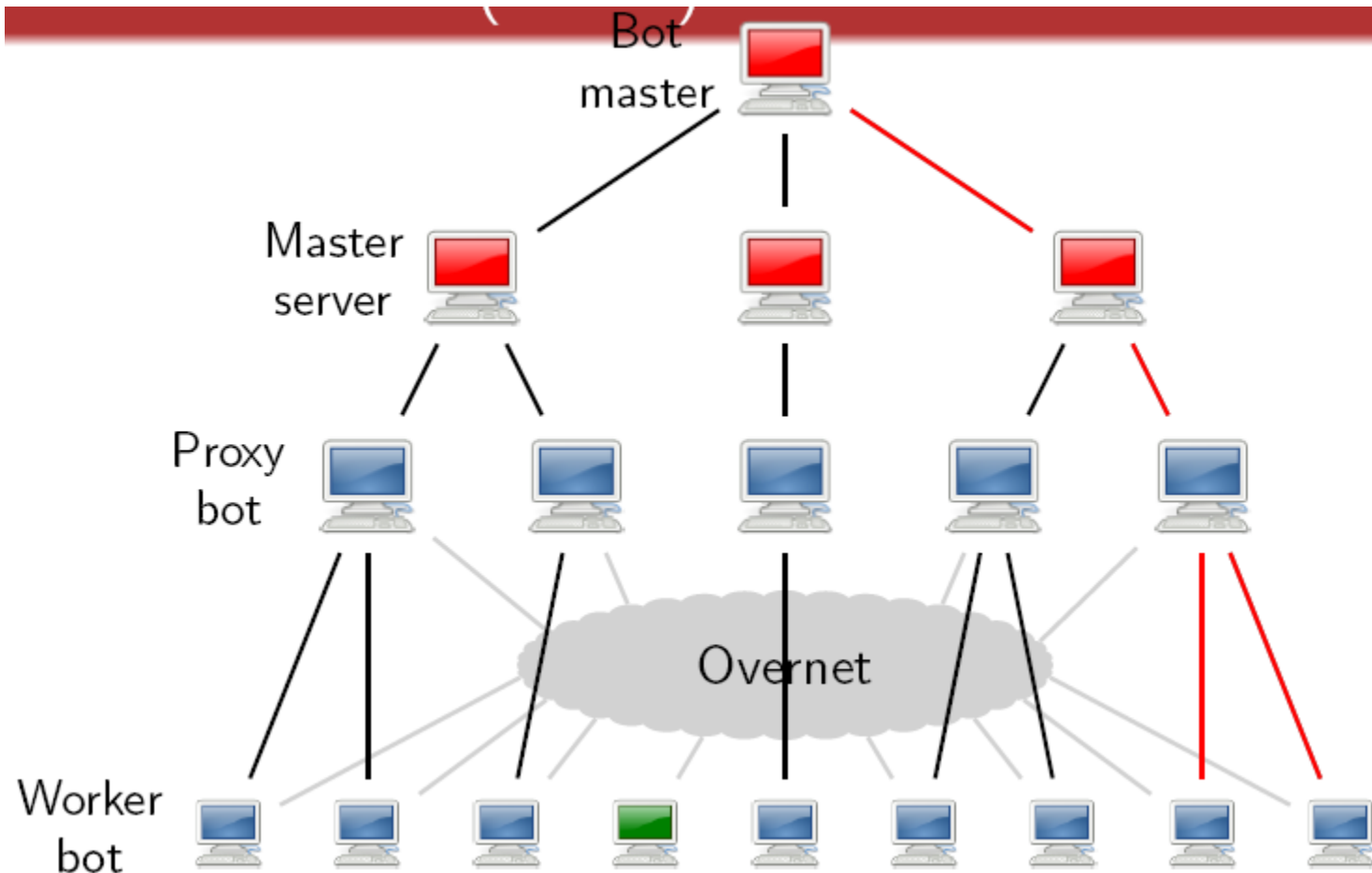
P2P based botnet (Storm)

- Master servers are controlled directed by the botmaster and are hosted on bullet-proof hosts

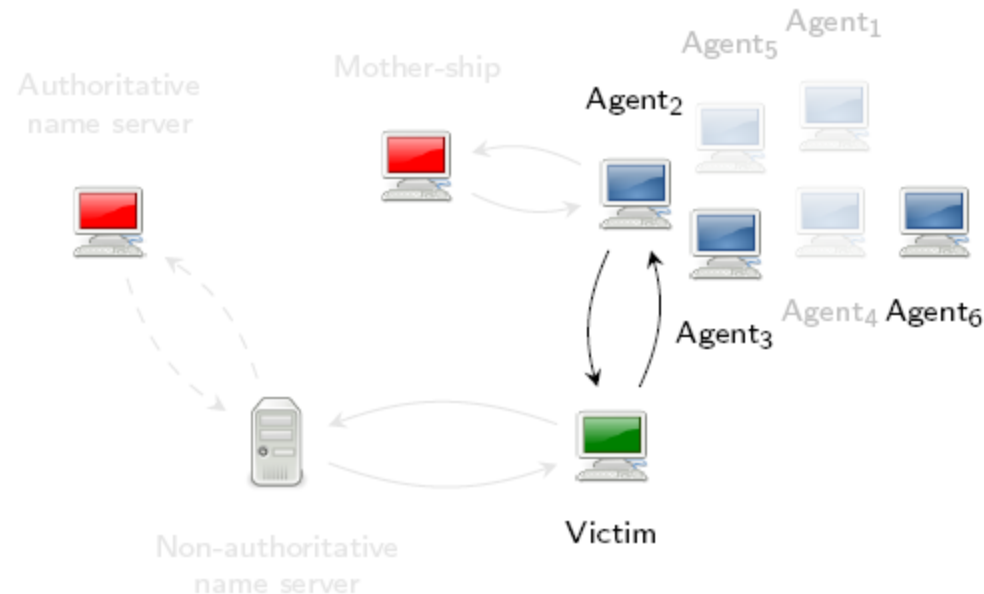


P2P based botnet (Storm)

- Workers with best resources are elected to proxies

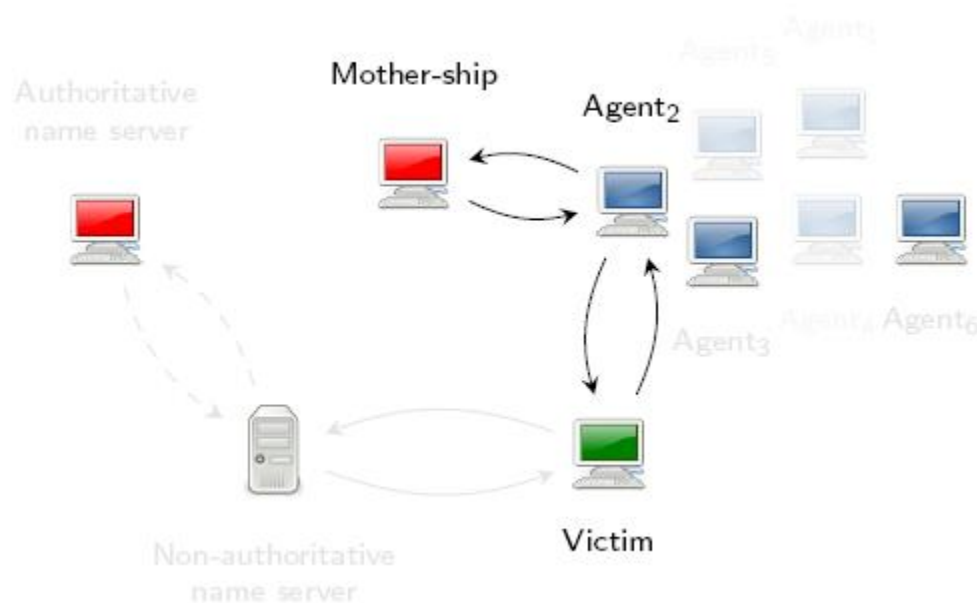


Fast-flux service network



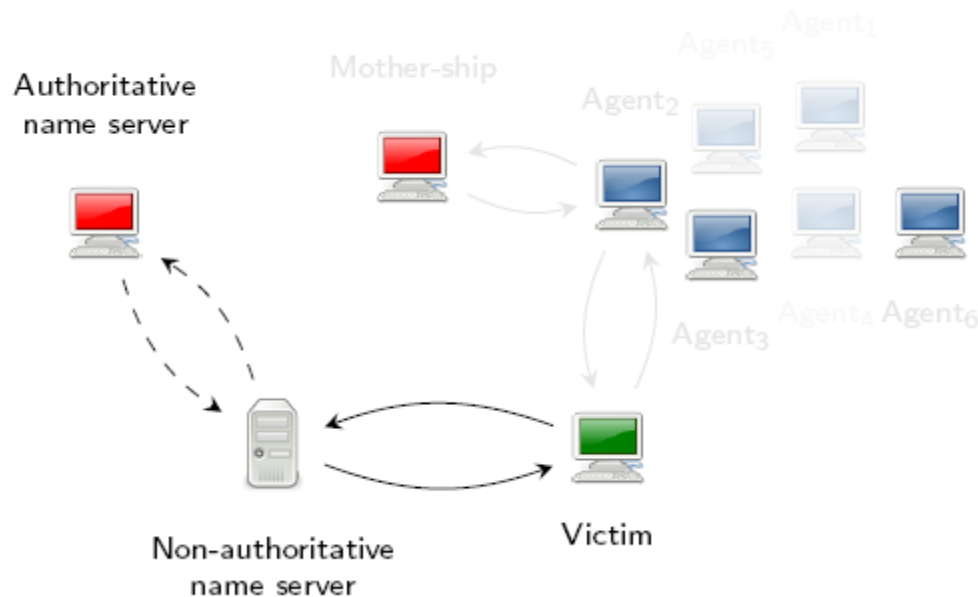
- Off line, disinfected, or problematic agents are replaced with others
- The botnet is typically composed of millions of agents
- The identity of the code components of the infrastructure is well protected
- Multiple domains are used by the same botnet (it is not sufficient to shut down a domain)

Fast-flux service network



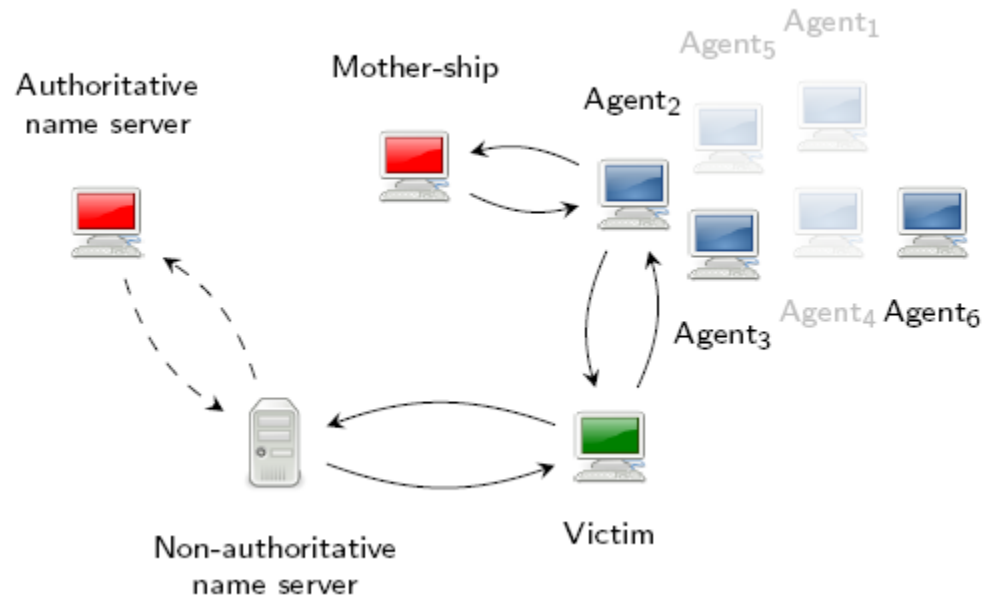
- Off line, disinfected, or problematic agents are replaced with others
- The botnet is typically composed of millions of agents
- The identity of the code components of the infrastructure is well protected
- Multiple domains are used by the same botnet (it is not sufficient to shut down a domain)

Fast-flux service network



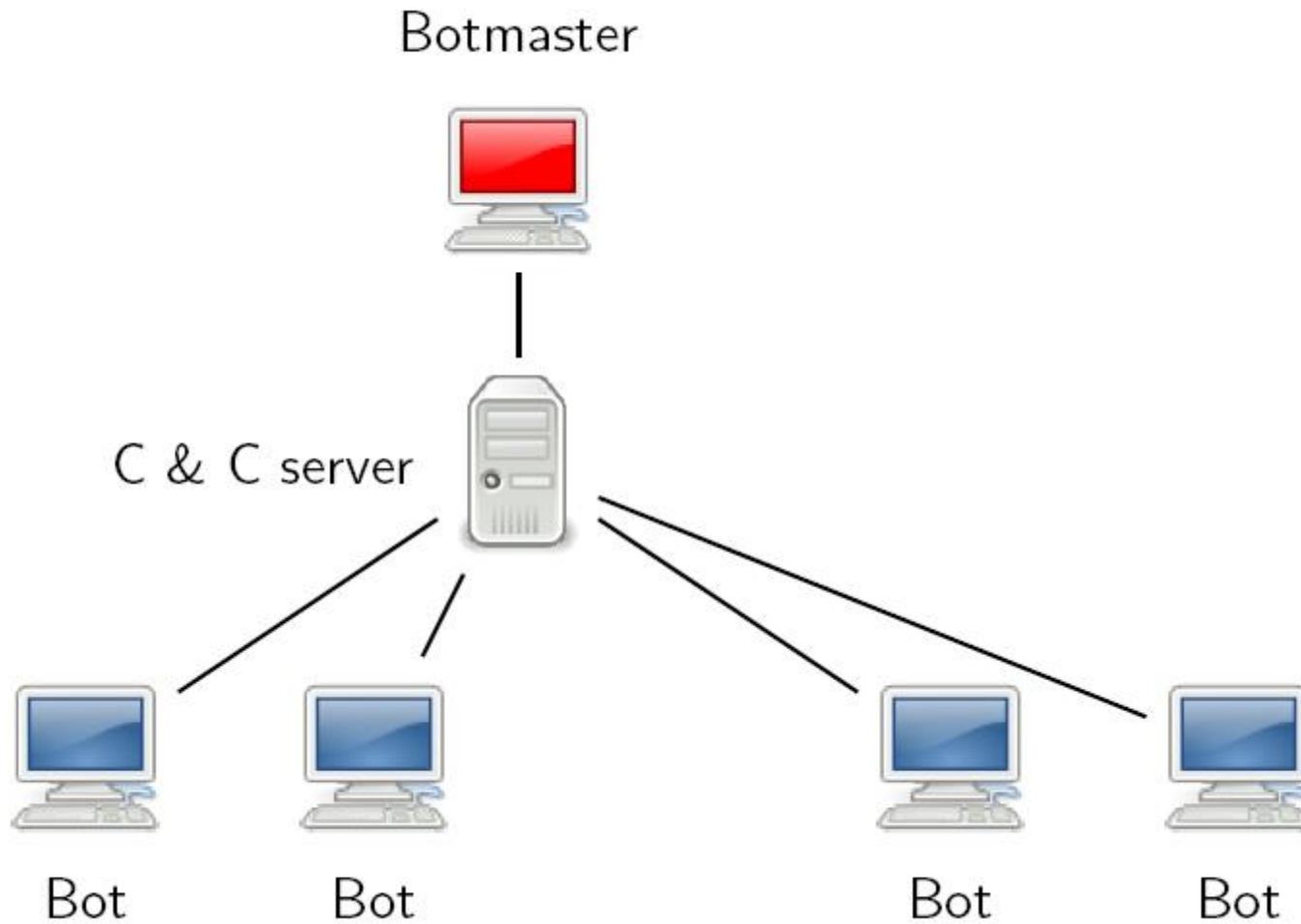
- Off line, disinfected, or problematic agents are replaced with others
- The botnet is typically composed of millions of agents
- The identity of the code components of the infrastructure is well protected
- Multiple domains are used by the same botnet (it is not sufficient to shut down a domain)

Fast-flux service network

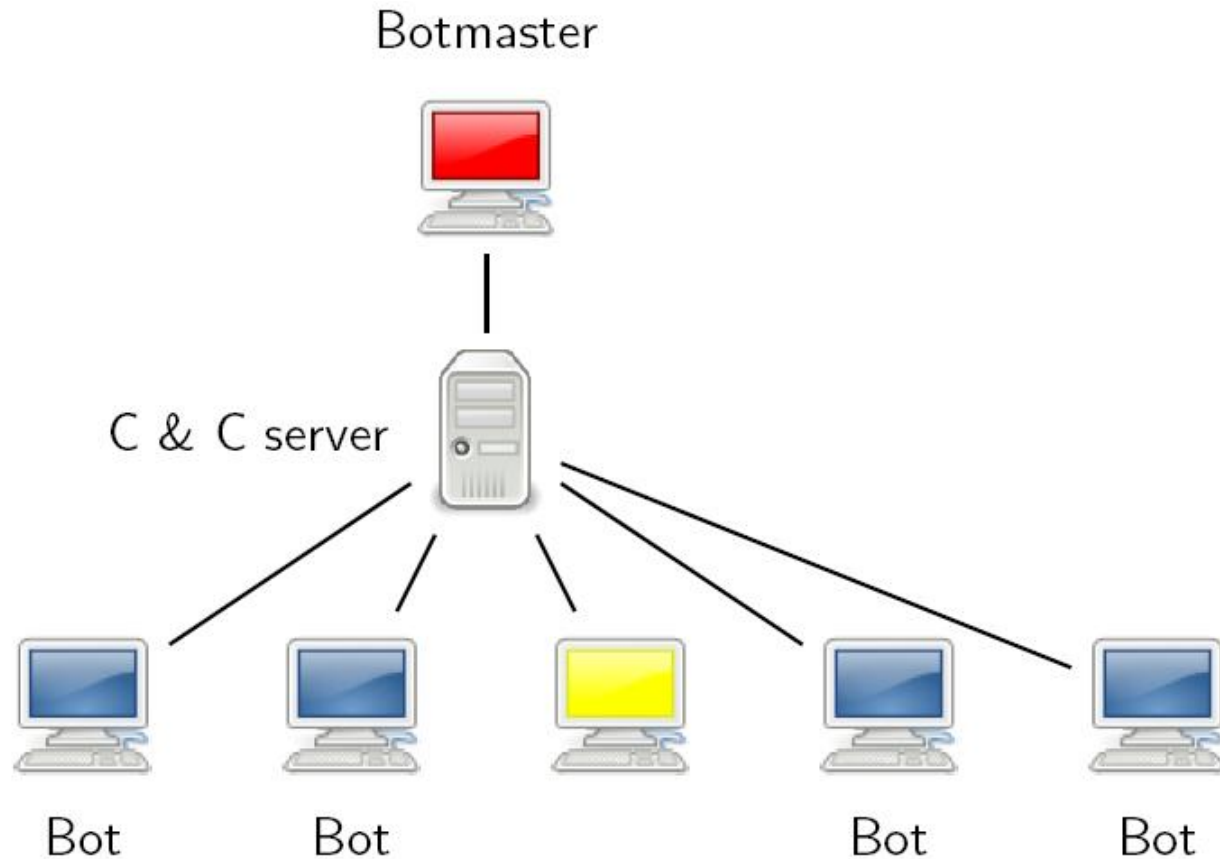


- Off line, disinfected, or problematic agents are replaced with others
- The botnet is typically composed of millions of agents
- The identity of the code components of the infrastructure is well protected
- Multiple domains are used by the same botnet (it is not sufficient to shut down a domain)

Infiltrate a botnet (HTTP C&C)



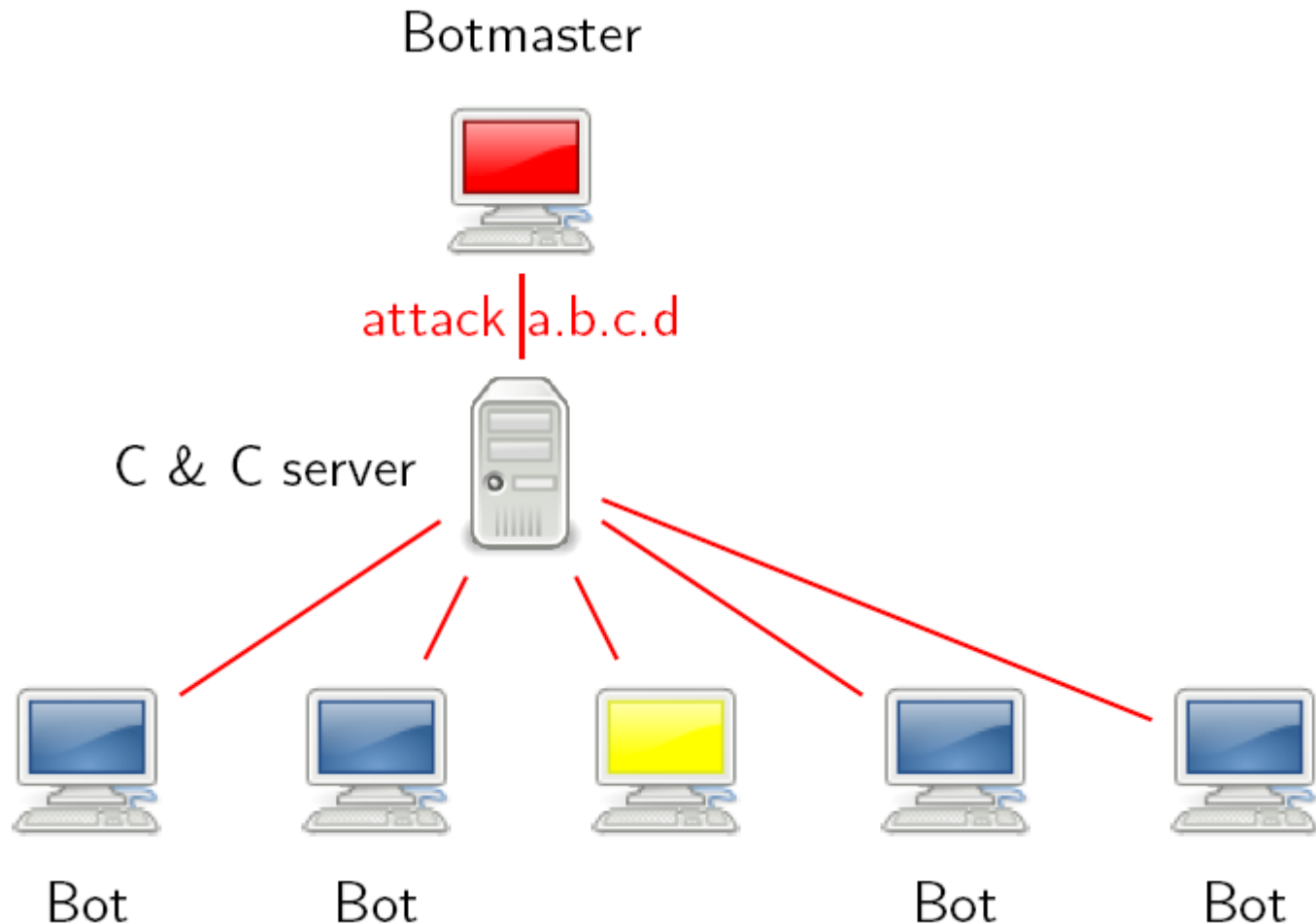
Infiltrate a botnet (HTTP C&C)



- The infiltrator connects to C&C server as a normal bot and can see all the bots connected

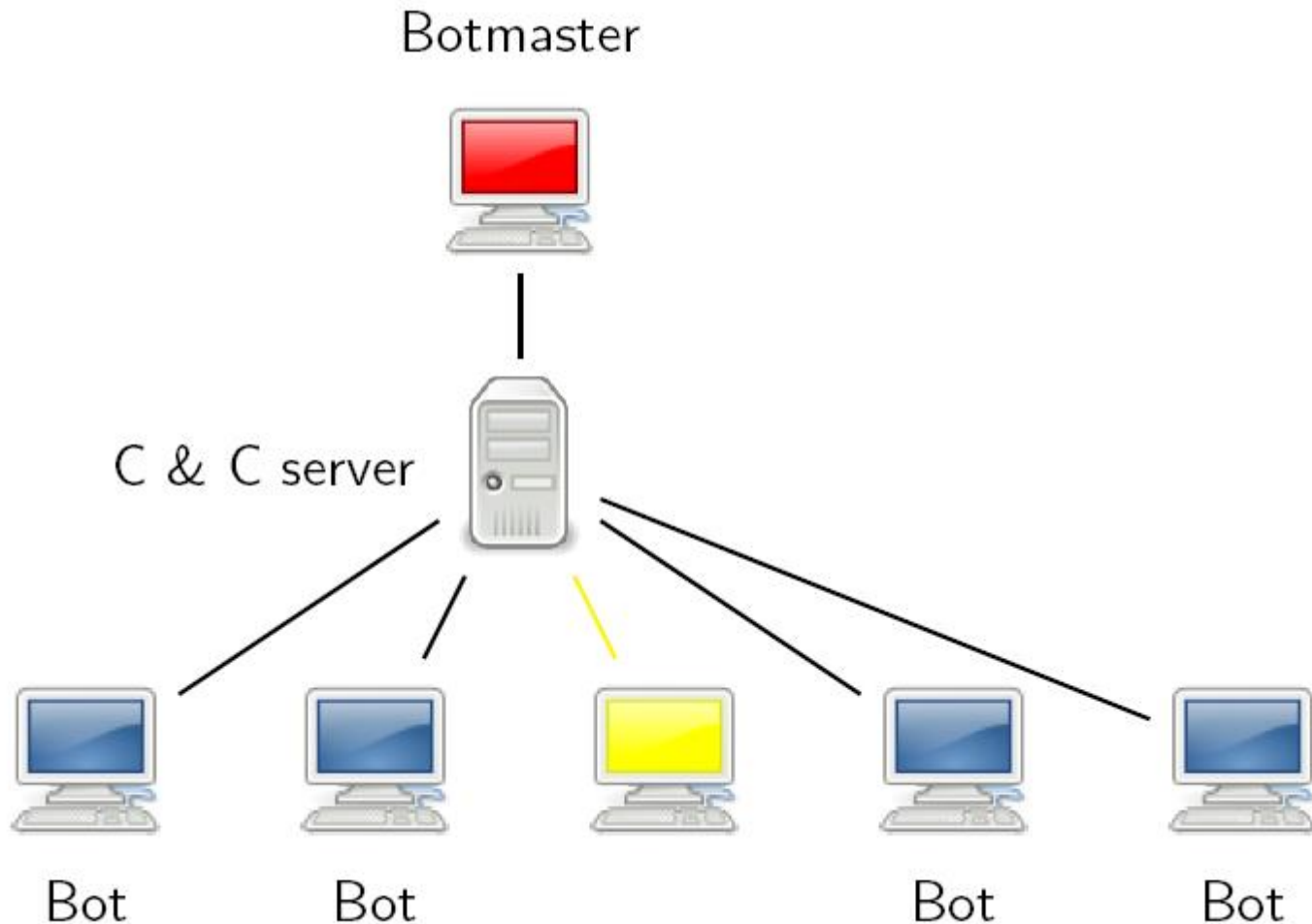
Infiltrate a botnet (HTTP C&C)

- The infiltrator receives commands from the botmaster as the other bots



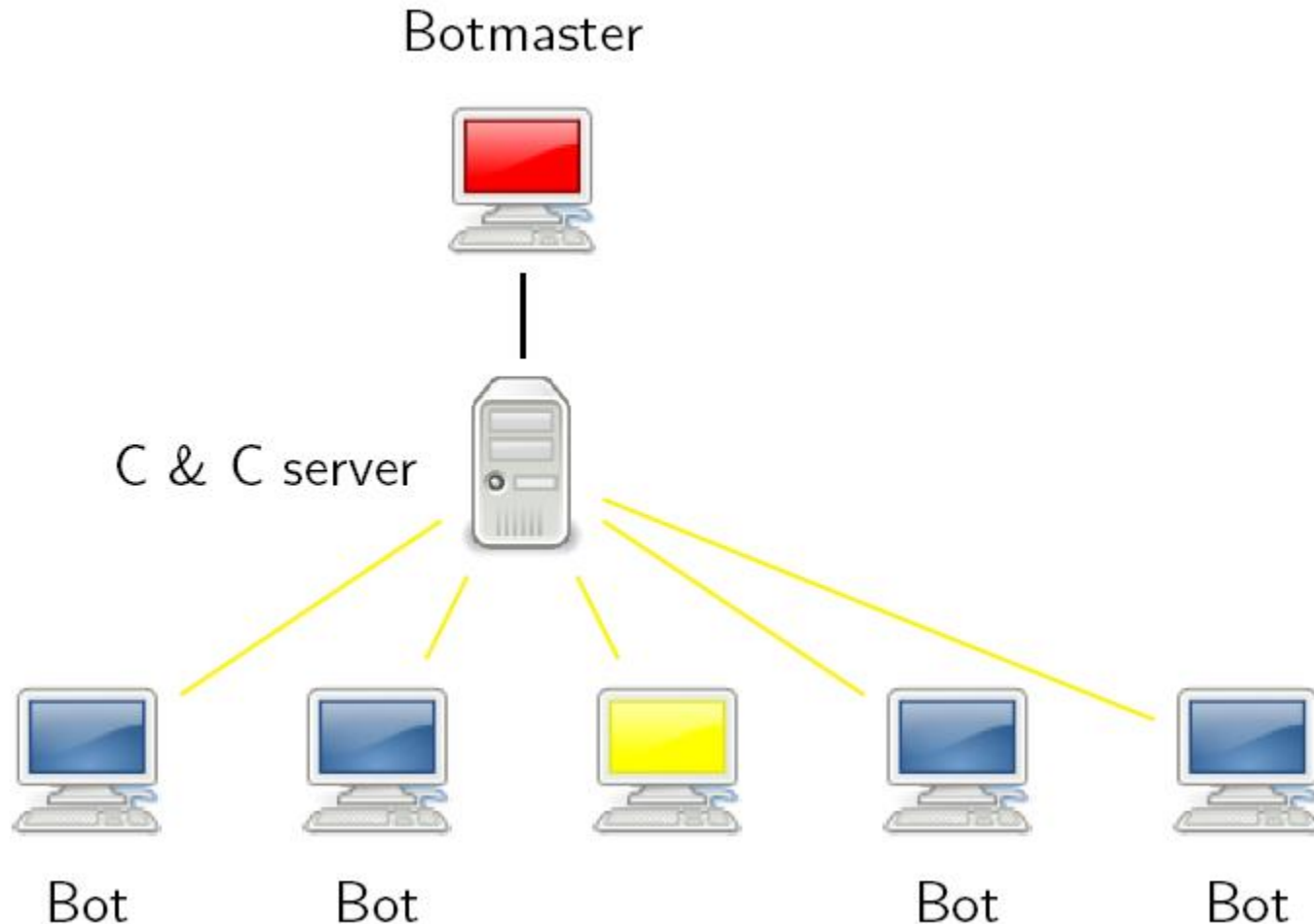
Infiltrate a botnet (HTTP C&C)

- The infiltrator has the ability to send commands to other bots

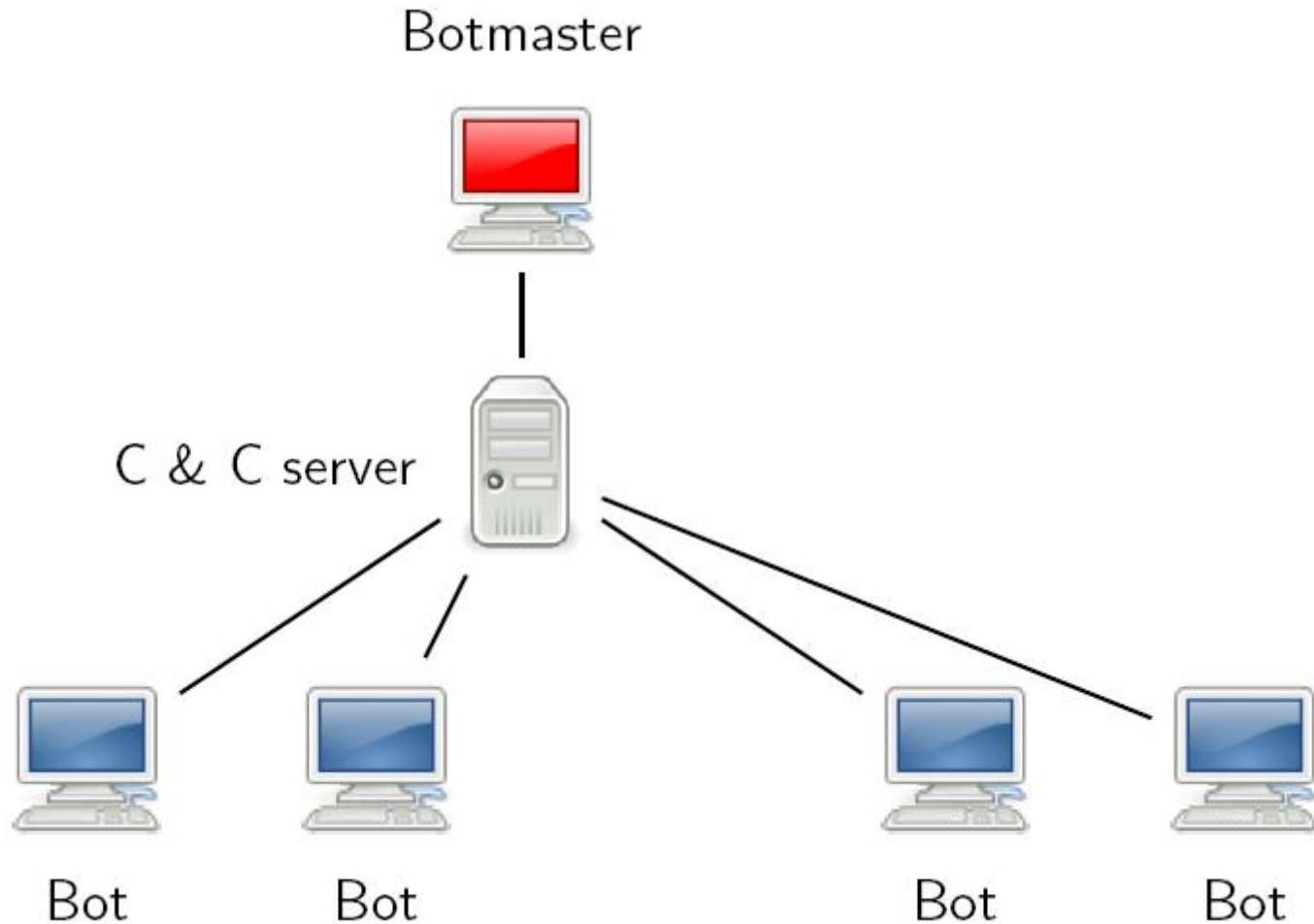


Infiltrate a botnet (HTTP C&C)

- The infiltrator has the ability to send commands to other bots

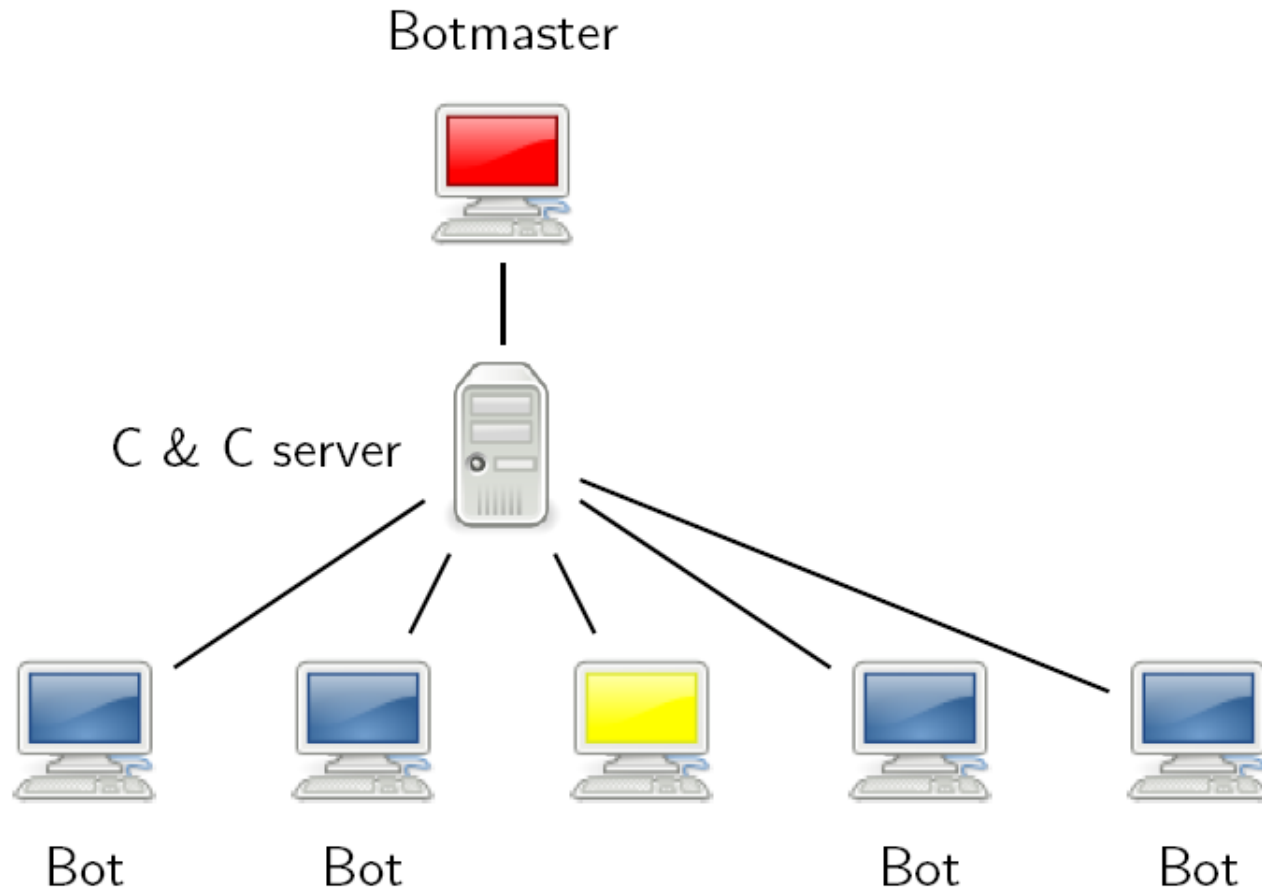


Infiltrate a botnet (HTTP C&C)



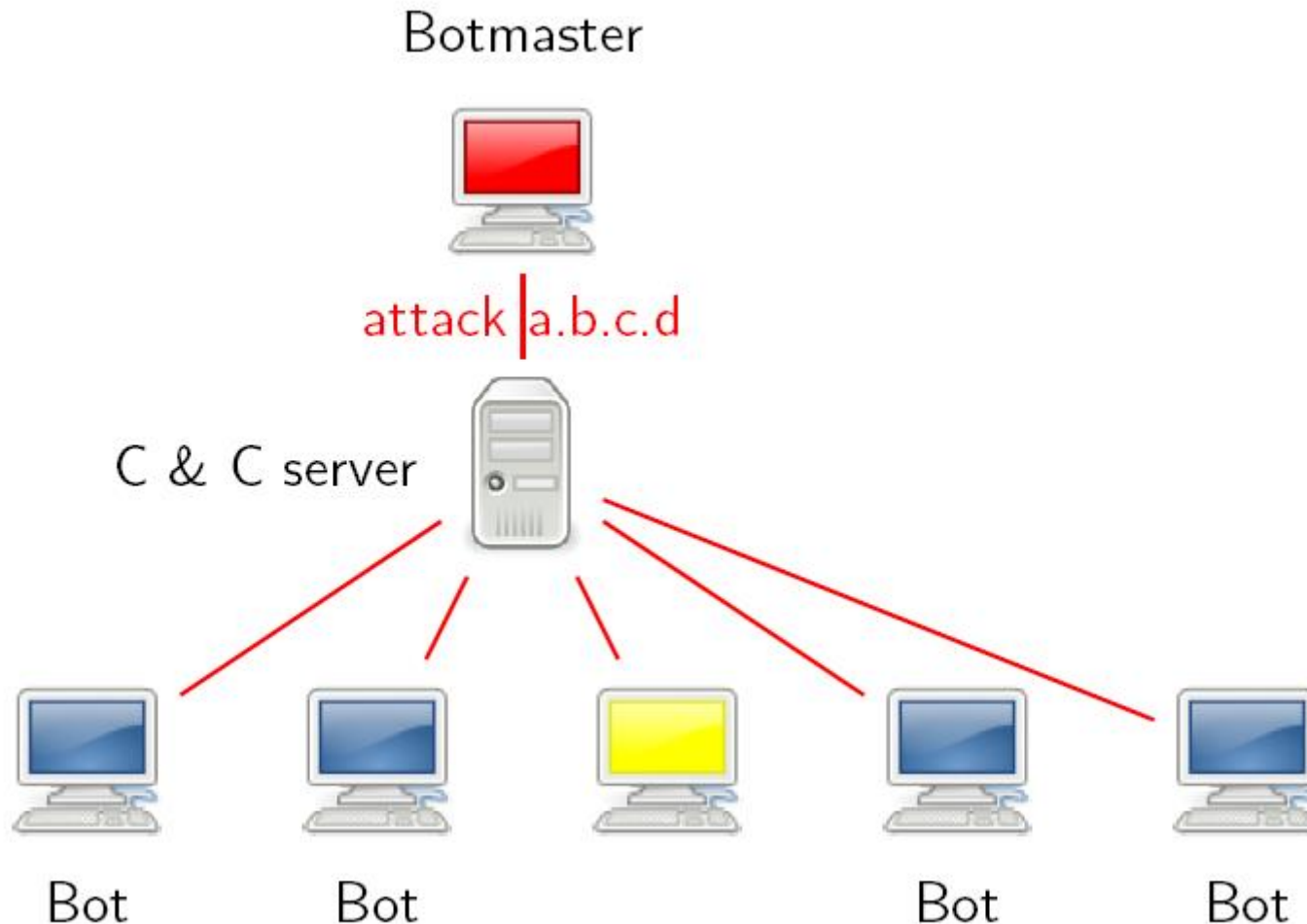
Infiltrate a botnet (HTTP C&C)

- The infiltrator connects to the C&C server as a normal bot but he cannot see the other bots connected (C&C pull)



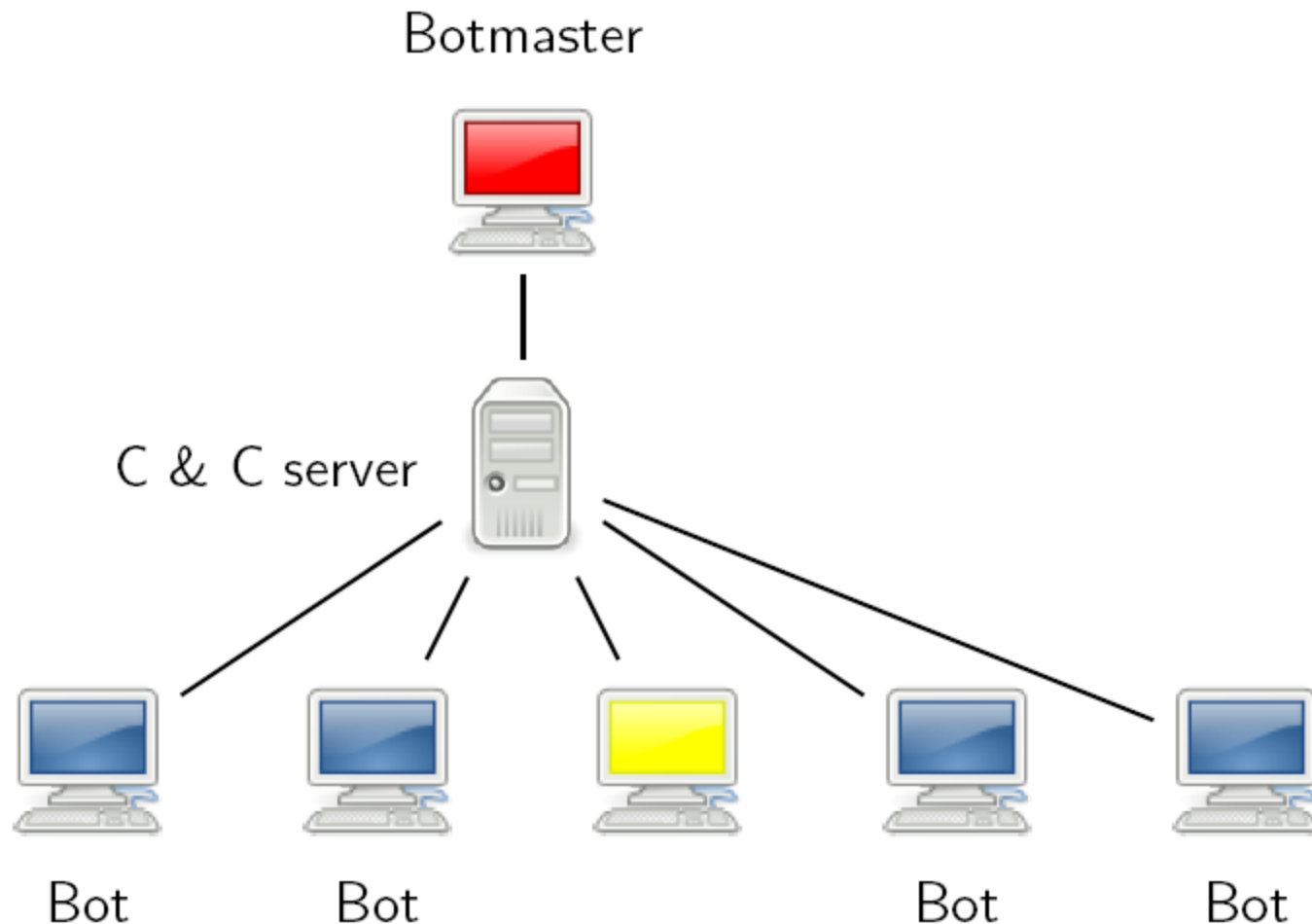
Infiltrate a botnet (HTTP C&C)

- The infiltrator receives commands from the botmaster as the other bots



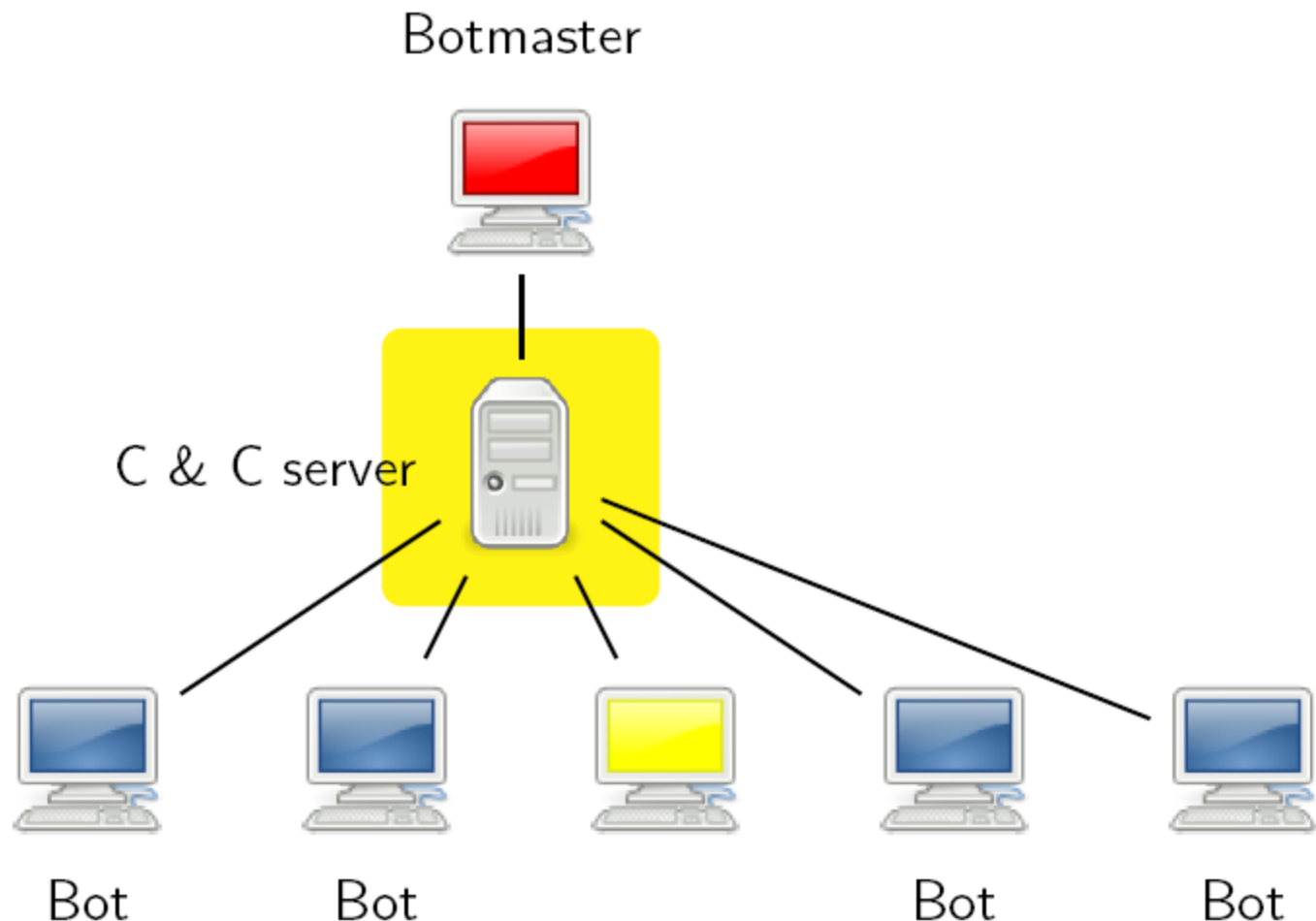
Infiltrate a botnet (HTTP C&C)

- The infiltrator cannot send commands to other bots



Infiltrate a botnet (HTTP C&C)

- The server-side infiltrator instead can see the other bots and send them commands



Infiltrate a botnet (HTTP C&C)

Botmaster



- **“Your Botnet is My Botnet: Analysis of a Botnet Takeover”**, Brett Stone-Gross, Marco Cova, Lorenzo Cavallaro, Bob Gilbert, Martin Szydlowski, Richard Kemmerer, Chris Kruegel, and Giovanni Vigna, In the Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS), 2009



Bot



Bot



Bot



Bot

Rootkits

(Hide Your Malware)

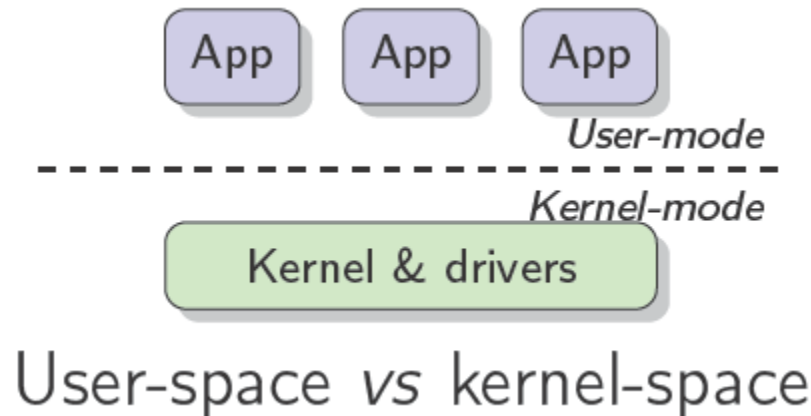


Rootkits

A rootkit is a tool used to hide, to the system administrator, the presence of a malware on the system

What to hide

- ★ Files
- ★ Registry keys
- ★ Services
- ★ Network connections
- ★ Processes
- ★ ...



Root-kit: hooking

Hijack the flow of the execution by modifying a code pointer

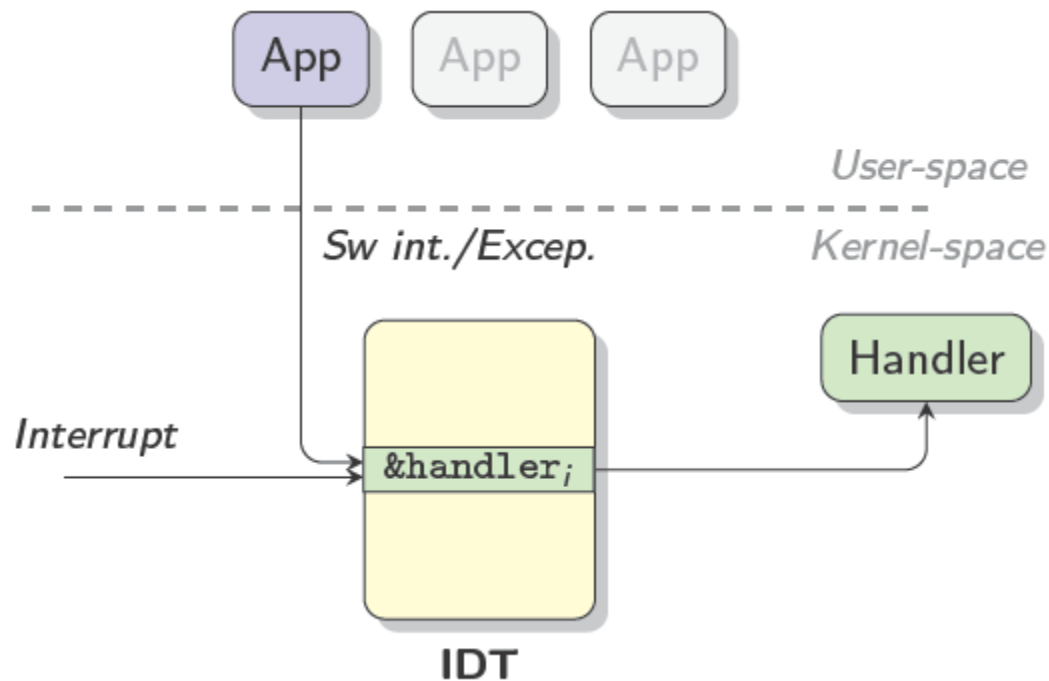
Examples

- ★ *User-space*: IAT
- ★ *Kernel-space*: IDT, MSR, SSDT



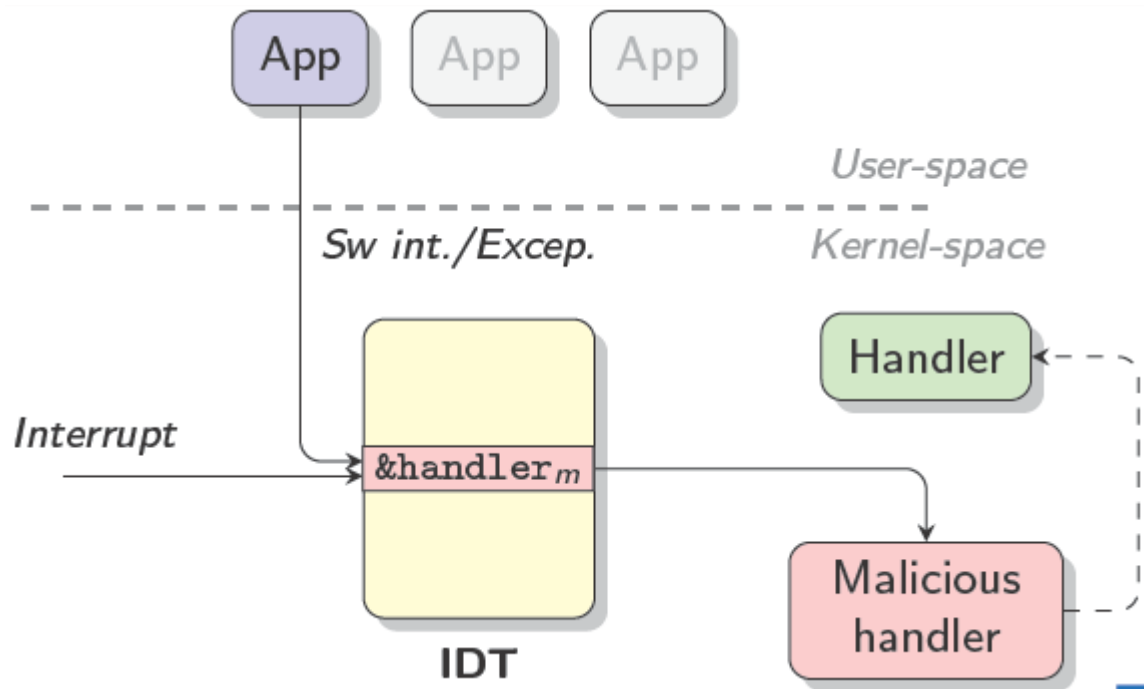
Root-kit: IDT hooking

- Interrupt Descriptor Table
- Interrupts and exceptions dispatching
- Hijacking of 1+ handlers (e.g., 0x2e → KiSystemService)



Root-kit: IDT hooking

- Interrupt Descriptor Table
- Interrupts and exceptions dispatching
- Hijacking of 1+ handlers (e.g., 0x2e → KiSystemService)



Root-kit: IDT hooking

- Interrupt Descriptor Table
- Interrupts and exceptions dispatching
- Hijacking of 1+ handlers (e.g., 0x2e → KiSystemService)

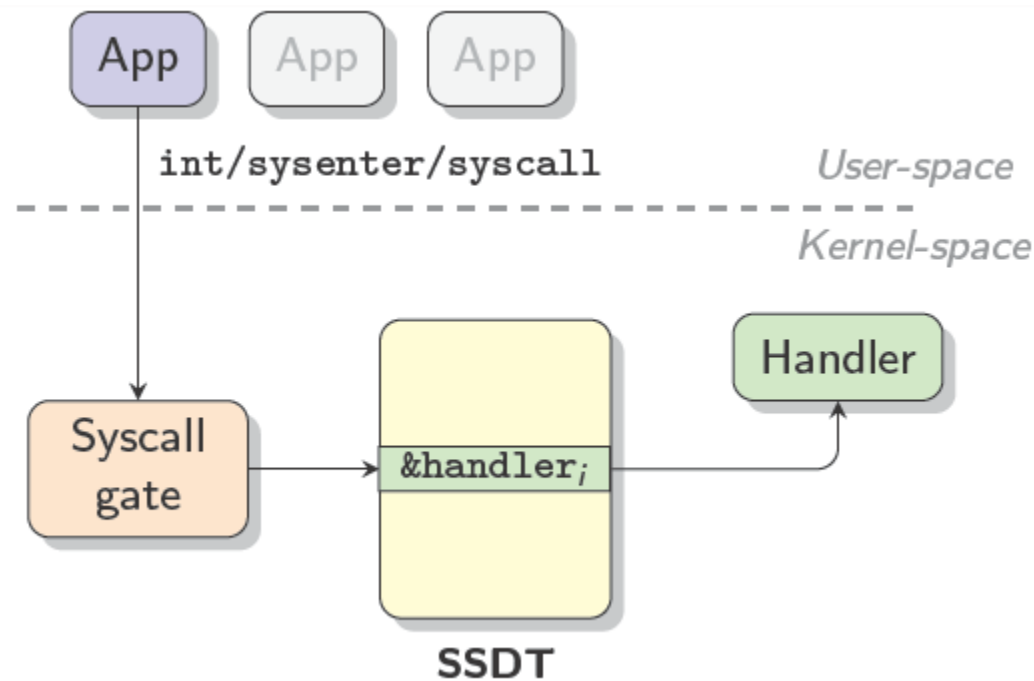
Problems

- It is not possible to do filtering
- System calls cannot be intercepted if sysenter/syscall are used
- Easy to detect:
 - if (IDT[0x2e] != KiSystemService) then ...



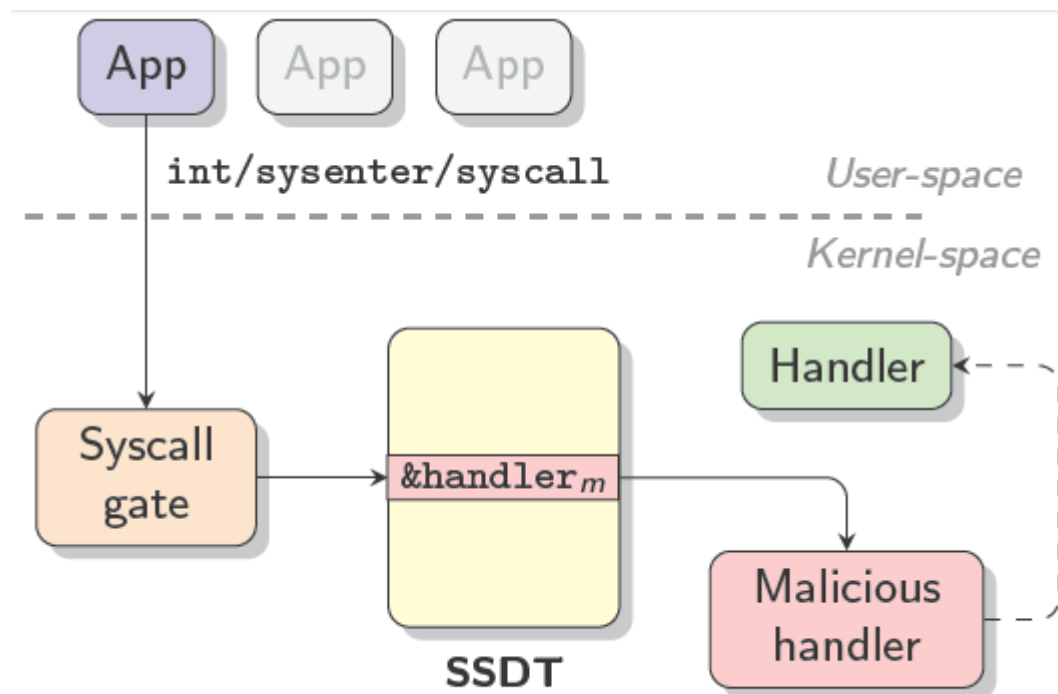
Root-kit: SSDT hooking

- System Service Descriptor Table
- System calls dispatching
- Hijacking of 1+ descriptors (e.g., 0x74 → NtOpenFile)



Root-kit: SSDT hooking

- System Service Descriptor Table
- System calls dispatching
- Hijacking of 1+ descriptors (e.g., 0x74 → NtOpenFile)



Root-kit: SSDT hooking

- System Service Descriptor Table
- System calls dispatching
- Hijacking of 1+ descriptors (e.g., 0x74 → NtOpenFile)

Problems

- Easy to detect:
- if (SSDT[0x74] != NtOpenFile) then ...



Root-kit: run-time patching

- It is possible to intercept the execution in multiple points
- More difficult to detect (no common hooking point)

```
...  
NtOpenFile:  
    movl $0x74, %eax  
    movl $0x7ffe0300, %edx  
    nop  
    call *%edx  
    ret $0x18  
...
```

Root-kit: run-time patching

- It is possible to intercept the execution in multiple points
- More difficult to detect (no common hooking point)

```
...  
NtOpenFile:  
    movl $0x74, %eax  
    pushl MaliciousHandler  
    ret  
    call *%edx  
    ret $0x18  
...
```

```
MaliciousHandler:  
    ...  
    movl $0x7ffe0300, %edx  
    nop  
    push addr  
    ret
```

Root-kit: run-time patching

- It is possible to intercept the execution in multiple points
- More difficult to detect (no common hooking point)

...
NtOpenFile:

→ **movl \$0x74, %eax**

pushl *MaliciousHandler*

ret

call *%edx

ret \$0x18

...

MaliciousHandler:

...

movl \$0x7ffe0300, %edx

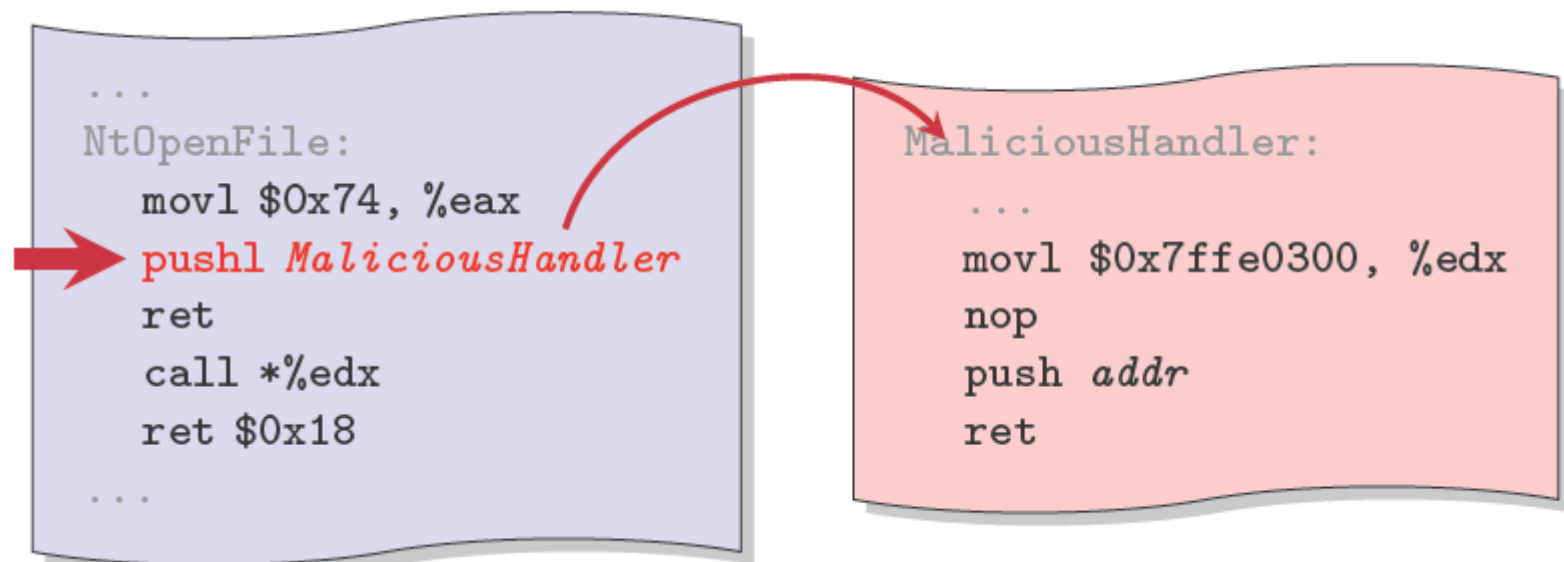
nop

push *addr*

ret


Root-kit: run-time patching

- It is possible to intercept the execution in multiple points
- More difficult to detect (no common hooking point)



Root-kit: run-time patching

- It is possible to intercept the execution in multiple points
- More difficult to detect (no common hooking point)


```
...  
NtOpenFile:  
    movl $0x74, %eax  
    pushl MaliciousHandler  
     ret  
    call *%edx  
    ret $0x18  
...
```

```
MaliciousHandler:  
    ...  
    movl $0x7ffe0300, %edx  
    nop  
    push addr  
    ret
```

Root-kit: run-time patching

- It is possible to intercept the execution in multiple points
- More difficult to detect (no common hooking point)

```
...  
NtOpenFile:  
    movl $0x74, %eax  
    pushl MaliciousHandler  
    ret  
    call *%edx  
    ret $0x18  
...
```



```
MaliciousHandler:  
...  
    movl $0x7ffe0300, %edx  
    nop  
    push addr  
    ret
```


Root-kit: run-time patching

- It is possible to intercept the execution in multiple points
- More difficult to detect (no common hooking point)

```
...  
NtOpenFile:  
    movl $0x74, %eax  
    pushl MaliciousHandler  
    ret  
    call *%edx  
    ret $0x18  
...
```

MaliciousHandler:


```
...  
→ movl $0x7ffe0300, %edx  
    nop  
    push addr  
    ret
```

Root-kit: run-time patching

- It is possible to intercept the execution in multiple points
- More difficult to detect (no common hooking point)

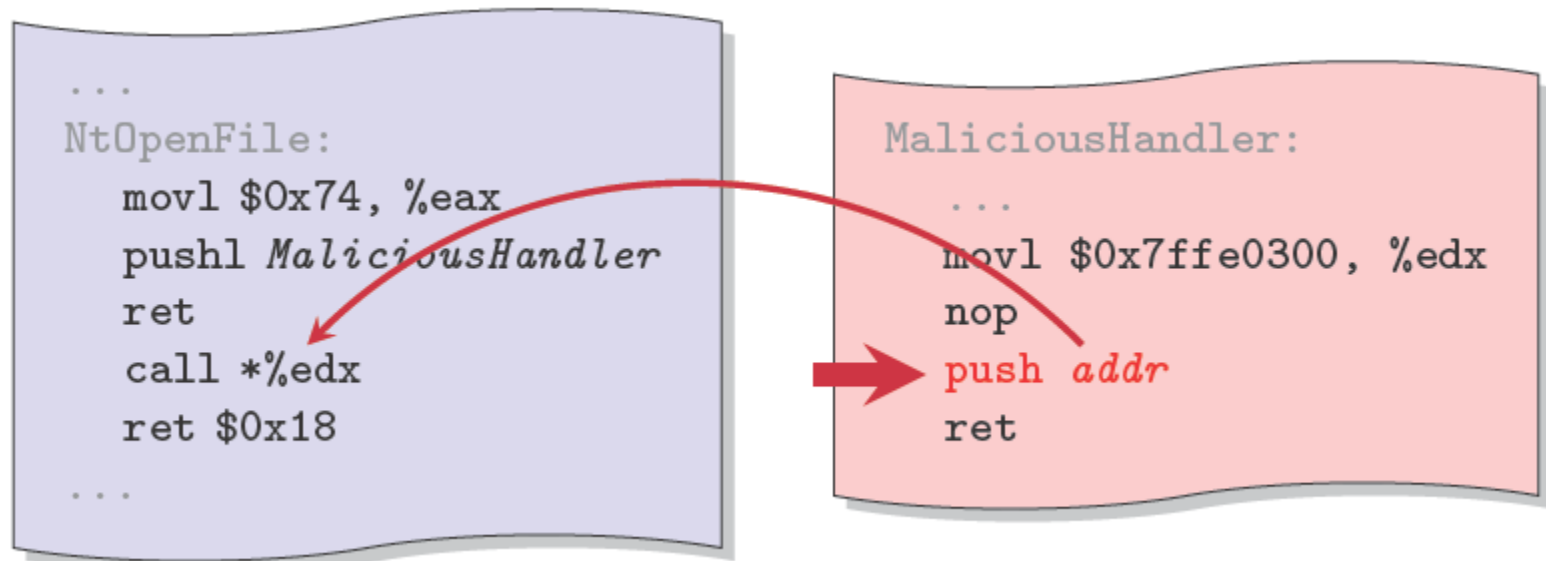
```
...  
NtOpenFile:  
    movl $0x74, %eax  
    pushl MaliciousHandler  
    ret  
    call *%edx  
    ret $0x18  
...
```

```
MaliciousHandler:  
    ...  
    movl $0x7ffe0300, %edx  
    nop  
    push addr  
    ret
```



Root-kit: run-time patching

- It is possible to intercept the execution in multiple points
- More difficult to detect (no common hooking point)




Root-kit: run-time patching

- It is possible to intercept the execution in multiple points
- More difficult to detect (no common hooking point)


```
...  
NtOpenFile:  
    movl $0x74, %eax  
    pushl MaliciousHandler  
    ret  
    call *%edx  
    ret $0x18  
...
```

```
MaliciousHandler:  
    ...  
    movl $0x7ffe0300, %edx  
    nop  
    push addr  
    ret
```



Root-kit: run-time patching

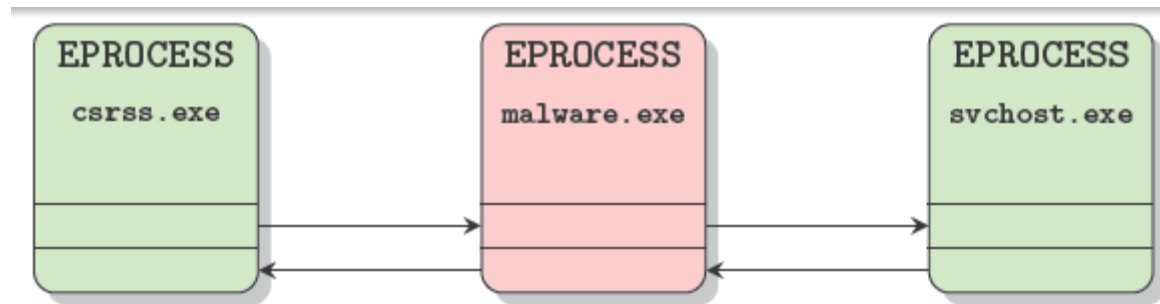
- It is possible to intercept the execution in multiple points
- More difficult to detect (no common hooking point)

```
...  
NtOpenFile:  
    movl $0x74, %eax  
    pushl MaliciousHandler  
    ret  
     call *%edx  
    ret $0x18  
...
```

```
MaliciousHandler:  
    ...  
    movl $0x7ffe0300, %edx  
    nop  
    push addr  
    ret
```

Root-kit: Direct Kernel Object Manipulation (DKOM)

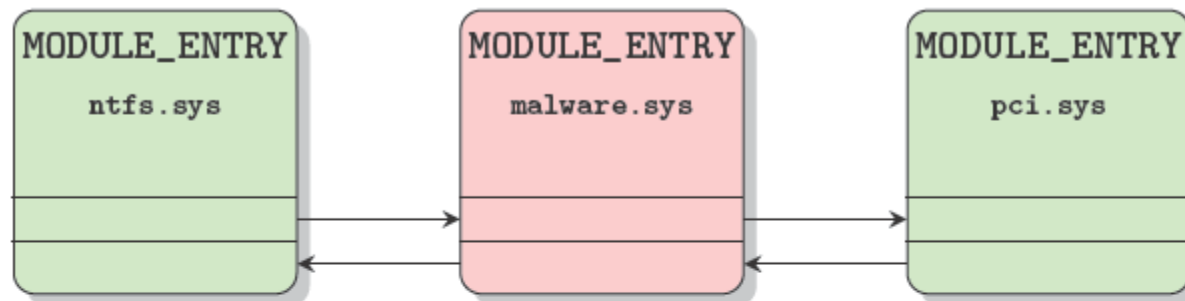
- In memory alteration of a kernel structure
- No hook or patch necessary



- malware.exe disappears from the list of running processes
- Scheduling is thread-based

Root-kit: Direct Kernel Object Manipulation (DKOM)

- In memory alteration of a kernel structure
- No hook or patch necessary



```
C:\WINDOWS> drivers.exe
```

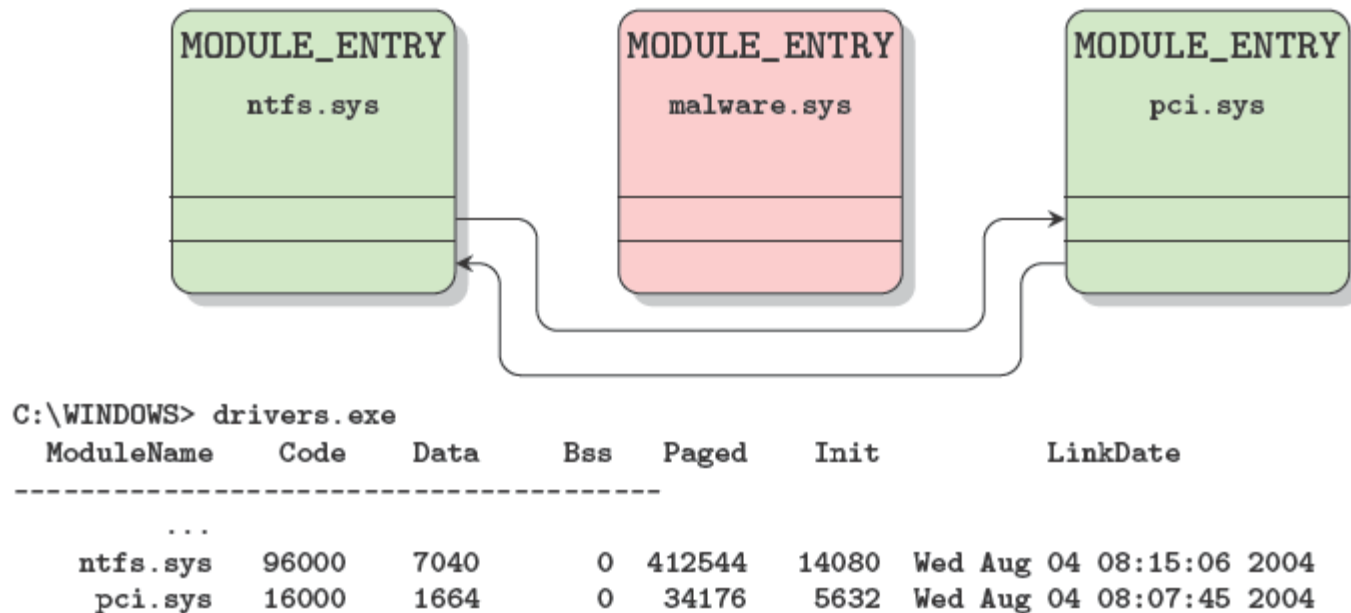
ModuleName	Code	Data	Bss	Paged	Init	LinkDate

...						
ntfs.sys	96000	7040	0	412544	14080	Wed Aug 04 08:15:06 2004
malware.sys	3903	0	0	0	0	Sat Mar 13 02:22:32 2010
pci.sys	16000	1664	0	34176	5632	Wed Aug 04 08:07:45 2004



Root-kit: Direct Kernel Object Manipulation (DKOM)

- In memory alteration of a kernel structure
- No hook or patch necessary



Acknowledgements

Material in this lecture are taken from the slides prepared by:

- Prof. Lorenzo Cavallaro (Information Security Group, Royal Holloway, University of London)

