Malicious Applications

Malware

- Insider Attacks
  - Backdoors/trapdoors
  - Logic Bombs
- Virus
  - Polymorphic and Metamorphic Virus
- Trojan Horses
- Privacy-Invasive Software
  - Adware/Spyware
- Worm
- Botnet

Backdoor/Trapdoor

- Secret entry point into a system
  - Specific user identifier or password that circumvents normal security procedures
- Commonly used by developers
  - Could be included in a compiler

XcodeGhost

Logic Bomb

- A logic bomb is a class of malicious code that is activated when a specific condition occurs.
  - Certain file is created
  - Certain time is reached
  - Certain user logins into the system
  - ……
- A logic bomb could represent
  - an “insider” threat
  - inherently malicious code

Virus

- Self-replicating code
  - Malicious functions + self-replicating (with users' involvement)
  - "Normal Code" => "Normal Code + Malicious Code"
- Stealthy
  - Attempts to evade detection
- Operates when infected code executed
  - Propagates
  - Performs malicious actions
  - Redirect to the normal code

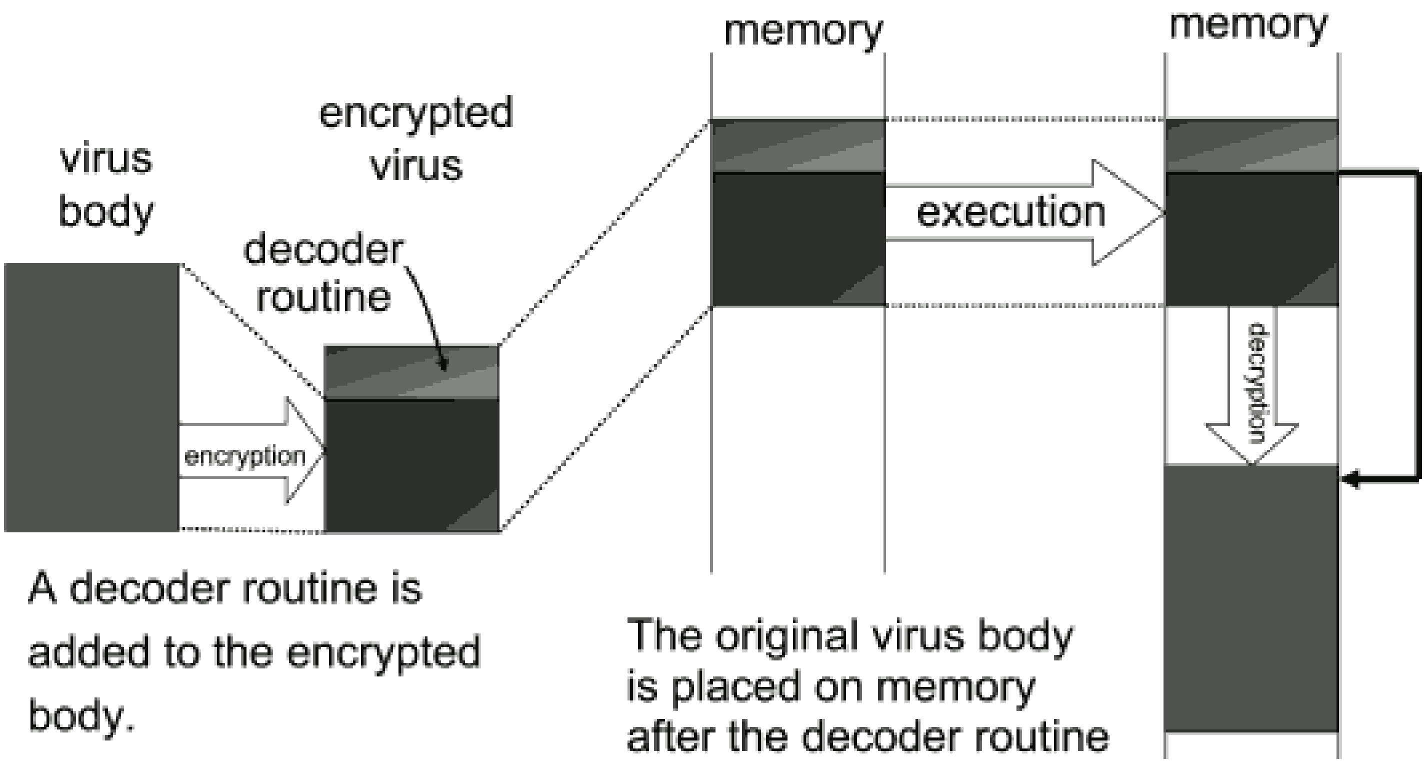
Virus Infection Vectors

- Boot Sector
  - Loaded when the system is booted
- Executable
  - Activated when an executable file is activated
- Macro files
  - Triggered when a document is loaded

Virus Characteristics

- Terminate and Stay Resident
  - Stays active in memory after application completes
  - Allow infection of previously unknown files
    - Traps calls that execute a program
- Stealthy
  - Conceal Infection
    - Trap read and disinfect
    - Let executable call infected file
  - Polymorphic/Metamorphic Virus
    - Change virus code to circumvent misuse detection

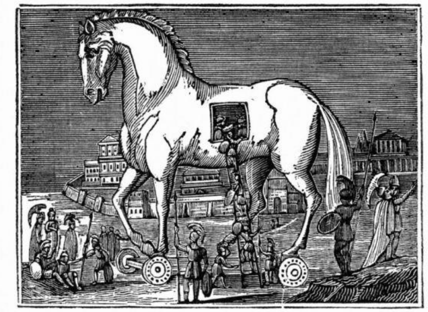
Polymorphic/Metamorphic Virus



A decoder routine is added to the encrypted body.

The original virus body is placed on memory after the decoder routine is finished, which is then executed.

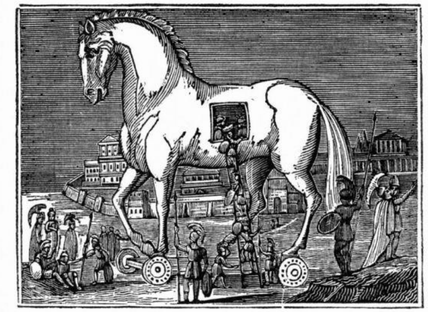
Trojan Horse

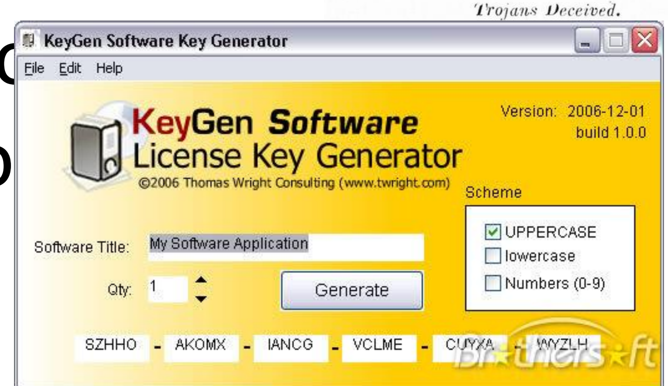

Trojans Deceived.

- A Trojan Horse is a malicious program that is disguised as legitimate software

- The gift horse left outside the gate of Troy by the Greeks
  - Appear to be interesting and innocent
  - Actually harmful

Trojan Horse


Trojans Deceived.



– Actually harmful

Privacy-Invasive Software

- Malware with specific malicious objectives
  - Adware
    - Pop up advertisements
  - Spyware
    - Key logging, screen capturing

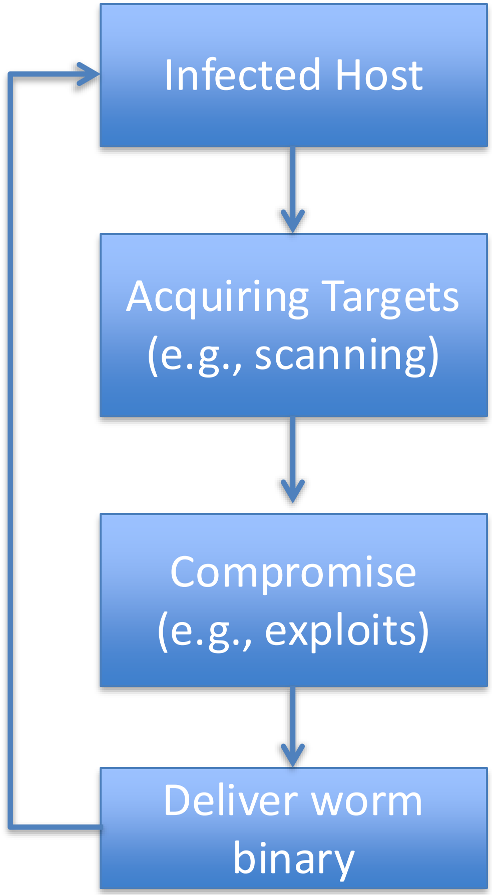
Worms

- Definition: Programs that self-propagate across the internet by exploiting security flaws in widely used services

Virus v.s. Worms

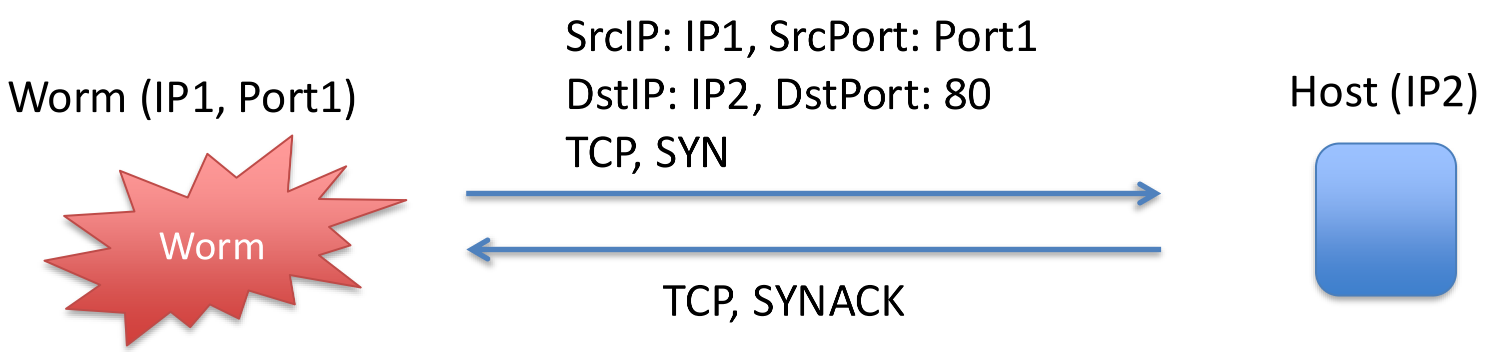
- Virus
  - A malicious program
  - Propagates depending on the user intervention
    - File sharing
    - Execute an infected file
- Worm
  - A malicious program
  - Propagates automatously
    - Self-replicate
    - No user intervention is required

An Abstract Worm Model



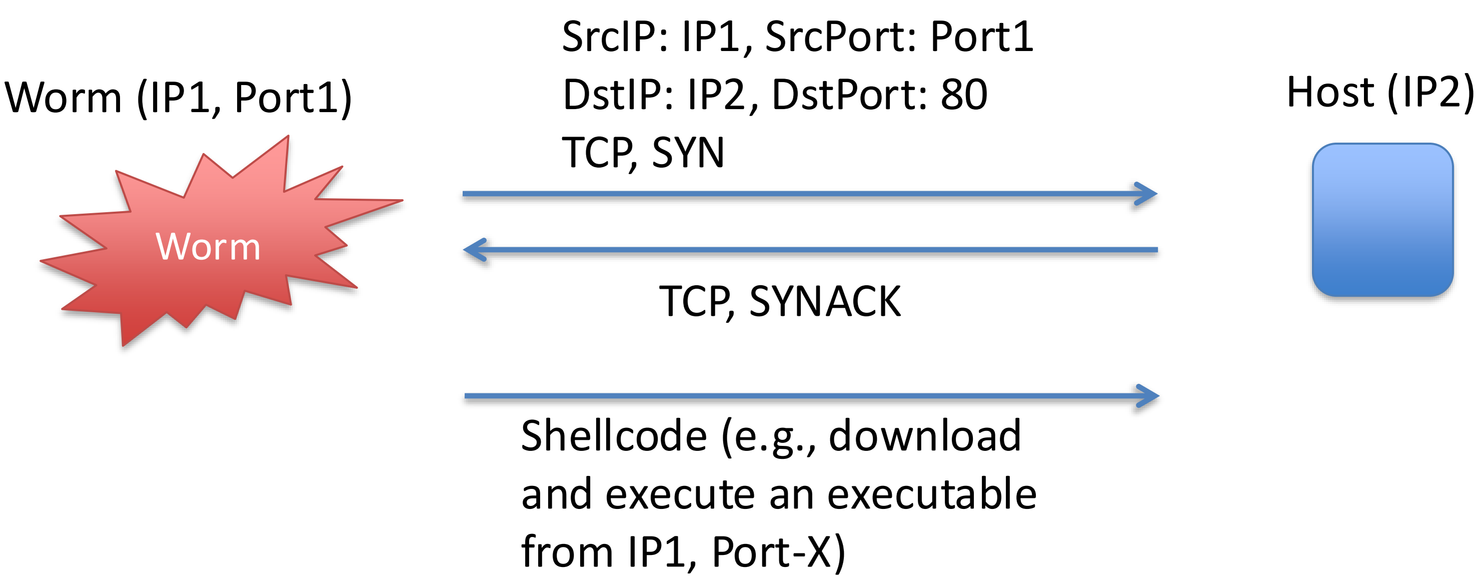
Examples

- Scanning
  - Does this host provide such service?
    - E.g., HTTP service (TCP, Port 80)



Examples

- Exploiting
  - Get access to the remote host
    - E.g., buffer overflow attack



Examples

- Deliver Worm Executable
  - Obtain and execute the worm executable

A Brief History of Worms

- Worms that affect the operation of the whole Internet
  - Morris Worm (1988)
  - Code Red (2001)
  - Nimda (2001)
  - Blaster (2003)
  - SQL Slammer (2003)

Morris Worm

- A program of only 99 lines
- 6000 computers in just few hours
- Disrupted the Internet at that time
- Was not really “malicious” (only propagate, no damage to the data)
- Exploits
  - BOF in Fingerd
  - Vulnerability in sendmail, which allows the execution of arbitrary commands
  - Brute force attacks to login (432 frequently used passwords)

Morris Worm

- The positive impacts
  - Computer Emergency Response Team (CERT)
    - Reacted to the damage and disruption caused by Morris worm
    - Becomes a leading center on information sharing with respect to software vulnerability and malware
  - Raise attentions to cyber-security

The Code Red Worm

- Rapid propagation
  - > 2,000 hosts/min
  - Code Red II Took about 14 hours to fully propagate
- 359,000 hosts are infected
- Exploits
  - BOF in Microsoft IIS web server (enabled by default)

The Nimda Worm

- Rapid Propagation
  - Became the Internet's most widespread worm within 22 minutes
- Exploits
  - MS IIS vulnerability (CVE-2000-0884)
  - Email itself as attachment based on email addresses harvested from the infected machine
  - Copy itself across open network shares
  - Add exploit code to web pages on compromised servers in order to infect clients that browse the pages
  - Leverage the backdoors left behind by Code Red II.

Blaster Worm

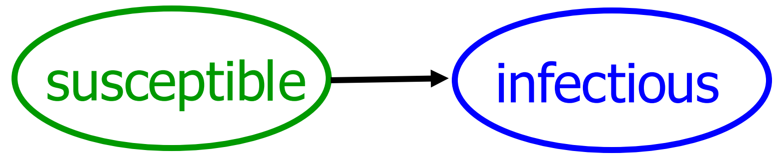
- Exploits
  - BOF in MS OS
  - The patch had been released one month earlier than the incident
    - “The original Blaster was created after a Chinese cracking collective called Xfocus reverse engineered the original Microsoft patch that allowed for execution of the attack” from wiki
- Attack
  - Launched DDoS attacks against windowsupdate.com

SQL Slammer Worm

- Rapid Propagation
  - Leverage UDP, which is connectionless
  - Infected 75,000, 90% within 10 minutes
- Exploits
  - BOF in MS SQL server

Deterministic epidemic models — Simple epidemic model

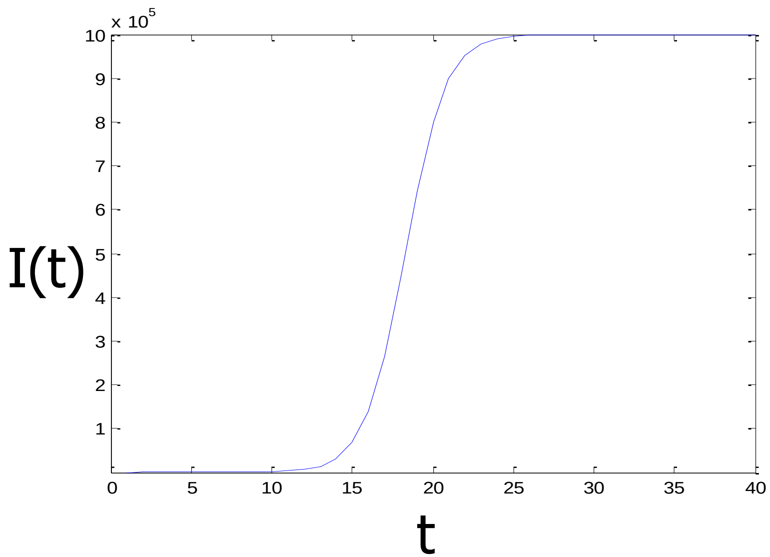
- State transition:



N: population; S(t): susceptible hosts; I(t): infectious hosts

$$dI(t)/dt = \beta\, S(t)\, I(t)$$

$$S(t) + I(t) = N$$



Build “Better” Worms

- “Better”
  - ***Faster: so fast that the deployed detectors do not have enough time to respond***
  - Stealthier: so stealthy that the worms can circumvent the detectors

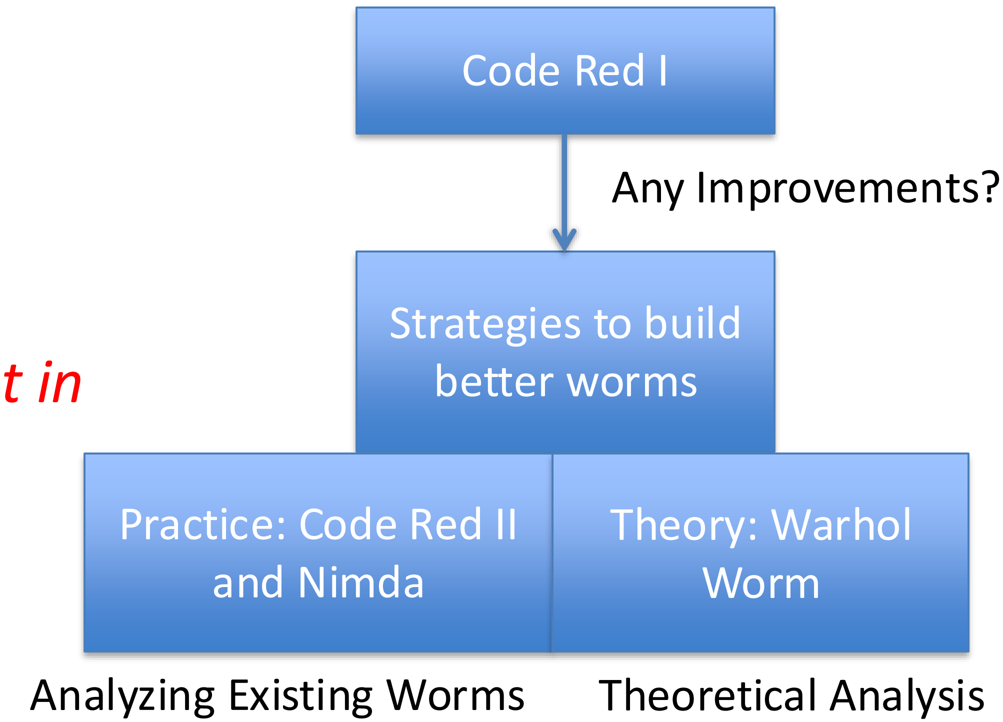
S. Staniford, V. Paxson, and N Weaver, “*How to 0wn the Internet in Your Spare Time*”, Usenix 2002.

The Roadmap

Timeline

- Morris Worm (1988)
- Code Red I (2001)
- Code Red II (2001)
- Nimda (2001)
- *"How to Own the Internet in Your Spare Time"* (2002)
- Blaster (2003)
- SQL Slammer (2003)

Structure

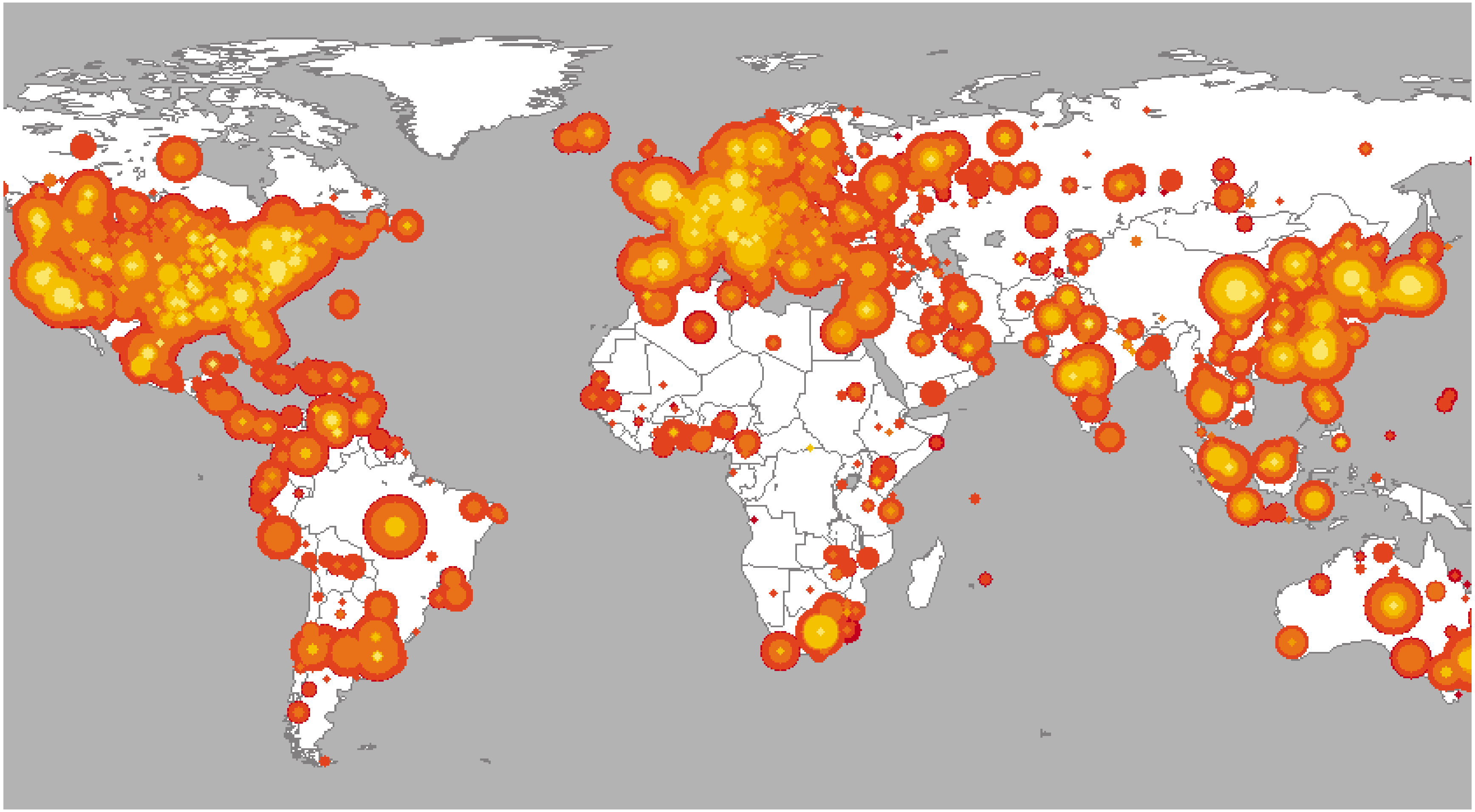


The Code Red I

- On July 19, 2001 more than 359,000 computers were infected with the Code Red I Version 2 (CRv2) worm in less than 14 hours.
  – 43% of all infected hosts were in the United States,
  – 11% originated in Korea followed
  – 5% in China

- There is a bug in the initial version of Code Red I Version 1 (CRv1).

The Code Red I



The Code Red I

- Its Scanning Strategy
  - Launch 99 threads and each thread generated random IP addresses in the whole IP space (2^32)
  - Probe each IP to determine whether the vulnerable service is available on the host.
  - If so, infect the host.

Build “Better” Worms - Practice

- Code Red II
- Nimda

Code Red II

- Code Red II
  - CRII was released on August 4th, 2001.
  - Died by design on October 1th, 2001.

- Localized Scanning Strategy
  - 3/8 probability: a random IP from the B (/16 network) address space of the infected machine
  - ½ probability: … from the A (/8 network) address….
  - 1/8 probability: … from the whole Internet

Code Red II

- Advantages of Localized Scanning
  - Facilitate propagation within certain internal networks
    - Firewall
    - NAT
  - Expedite propagation
    - Host with similar IP addresses imply a small network topological distance
- Consequence
  - Rapid infection

Nimda

- Nimda
  - Nimda began on September 18th, 2001.
  - It maintained itself on the Internet for months after it started
- Multi-vector infection
  - Web server vulnerability (similar to the Code Red)
  - Bulk emailing of itself as an attachment
  - Copying itself across open network shares
  - Adding exploit code to Web pages on compromised servers in order to infect clients which browse the page
  - By using the backdoors left behind by Code Red II

Nimda

- Enable propagation within internal networks (behind firewall or NATs)
    - Email
    - Network share
    - Exploit code in the compromised server
- Magnify the effectiveness using multiple ways
- Results
    – Became the Internet’s most widespread worm with in 22 minutes

Build “Better” Worms - Theory

- Hit-list scanning
- Permutation scanning
- Topological aware worms
- Internet scale hit-lists

Hit-list scanning

- "…the time needed to infect say the first 10,000 hosts dominates the infection time."

- Hit-list scanning
  - Before the worm is released, the worm author collects a list of say 10,000 to 50,000 potentially vulnerable machines
  - The initial worm focuses on infecting hosts on the list
  - When it infects a machine, it divides the hit-list in half, communicating half to the recipient worm, keeping the other half

Hit-list scanning

- How to collect hit-list in practice?
  - Stealthy scans
  - Distributed scanning
  - DNS searches
  - Spiders
    - 33% of automated search engine queries are looking for vulnerable Internet services.
  - Public surveys
  - Just listen

Permutation Scanning

- Random scanning
  - The Code Red I is a salient example to use random scanning
  - Disadvantage: many addresses are probed multiple times

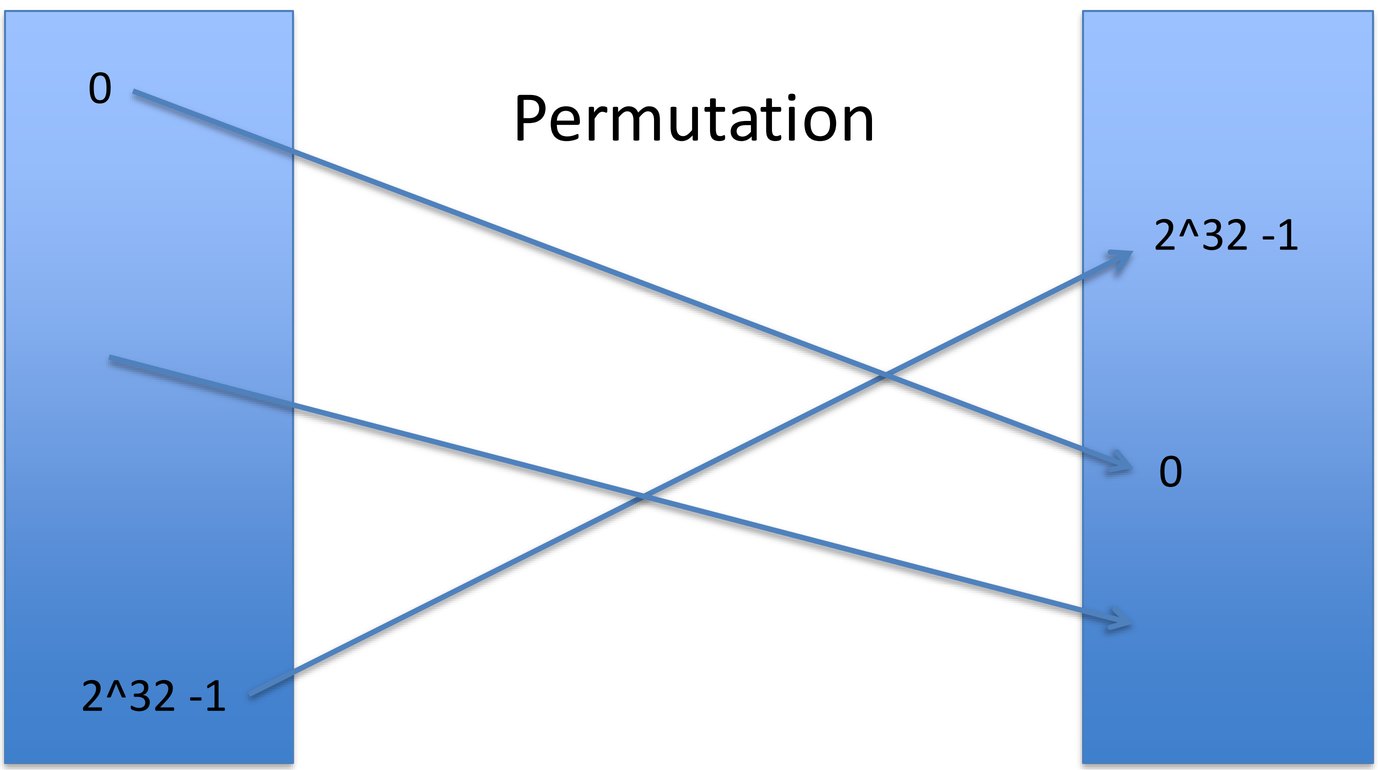
- Permutation Scanning
  - Objective: provide a self-coordinated, comprehensive scan while maintaining the benefits of random probing
  - Assumption: a worm can detect that a particular target is already infected

Permutation Scanning

- All worms share a common pseudo random permutation of the IP address space
- Any machine infected during the hit-list phase (or local subnet scanning) starts scanning after its point in the permutation
- Whenever the worm sees an already infected machine, it chooses a new, random start point and proceeds from there
- Worms infected by permutation scanning would start at a random point

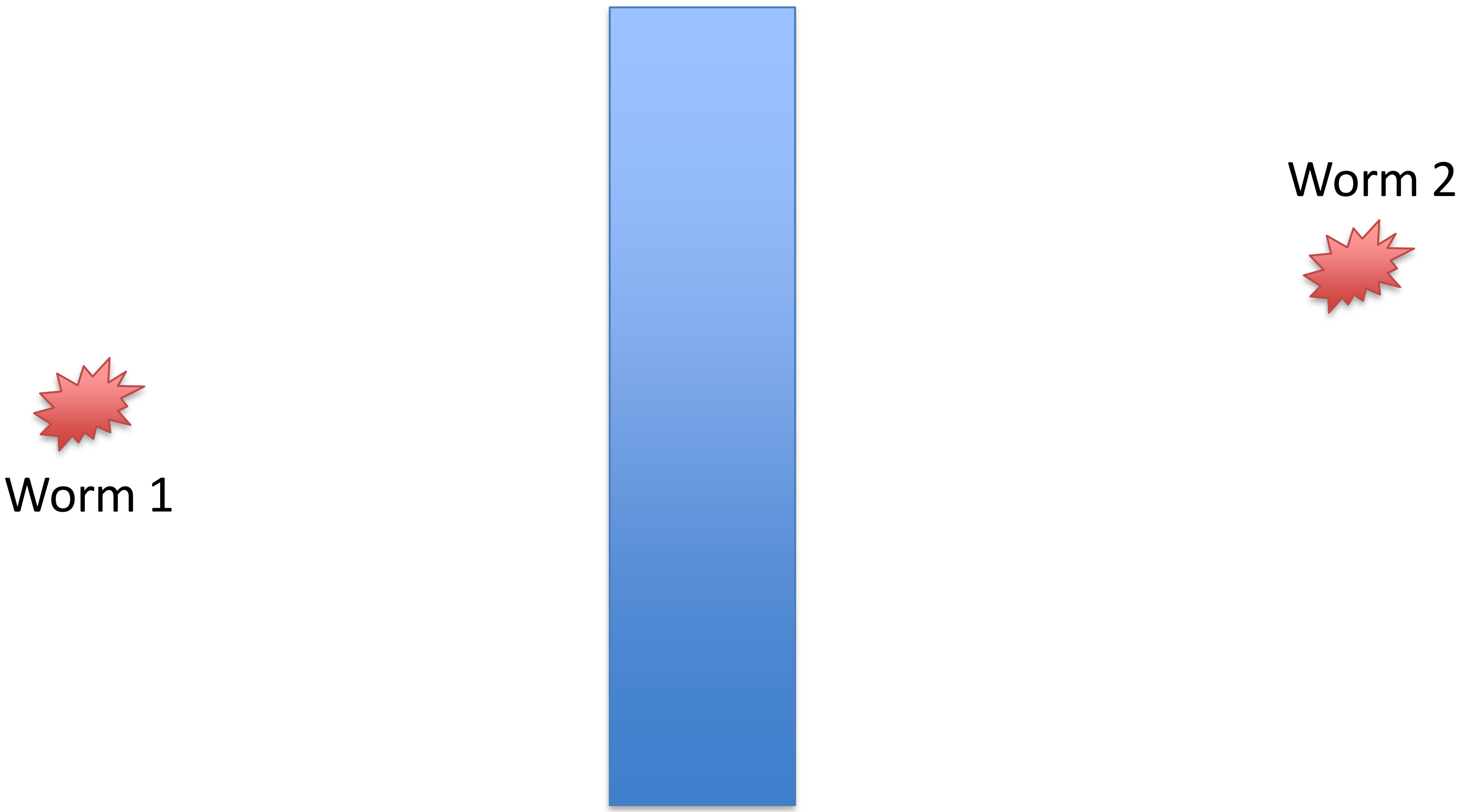
Permutation Scanning

- A Pseudo Random Permutation of the IP Address Space



Permutation Scanning

- A Pseudo Random Permutation of the IP Address Space



IP address space after permutation,
common for all worms

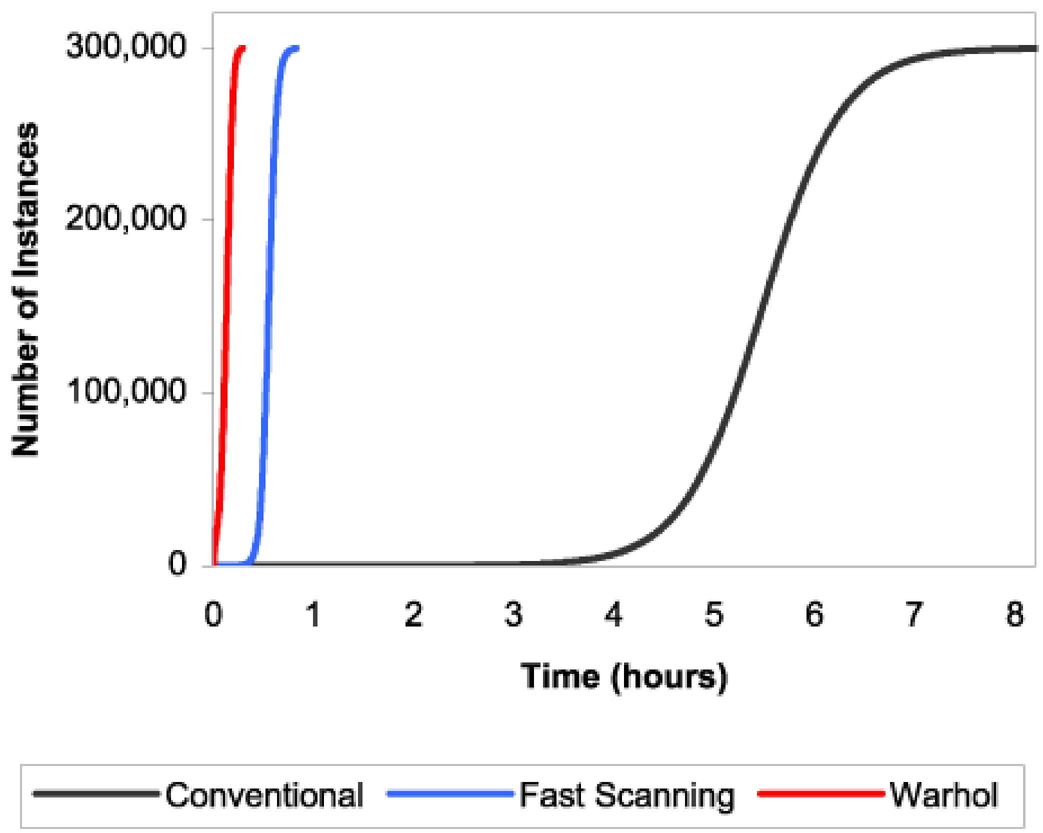
Warhol Worm

- Warhol = hit-list + Permutation

- User simulation to demonstrate its effectiveness
  - Assumptions: complete connectivity within 2^32 IP address space
  - Parameters
    - The number of vulnerable machines
    - Scan per second
    - The time to infect a machine
    - Number infected during the hit-list phase

Warhol Worm

- Simulation Results



Topological Scanning

- Leverage information contained on the victim machine in order to select new targets

- Very effective when the vulnerable targets exhibit a very sparse address space (compared to the whole address space)
  - Email worm
  - Perhaps IPv6? (You can investigate it!)

- Examples:
  - Nimda
  - A worm attacking flaws of P2P applications
  - Use web URLs visited by the compromised machine

Flash Worm

- Extend the hit-list worm
  - The hit-list contains a list of all or most Internet connected addresses with the relevant service(s) open
  - Capable of infecting the Internet in tens of seconds
- Two challenges
  - A large hit-list (12.6 million web servers => 48 MB hit-list)
  - High-bandwidth link for attackers to identify all hosts providing the service(s)

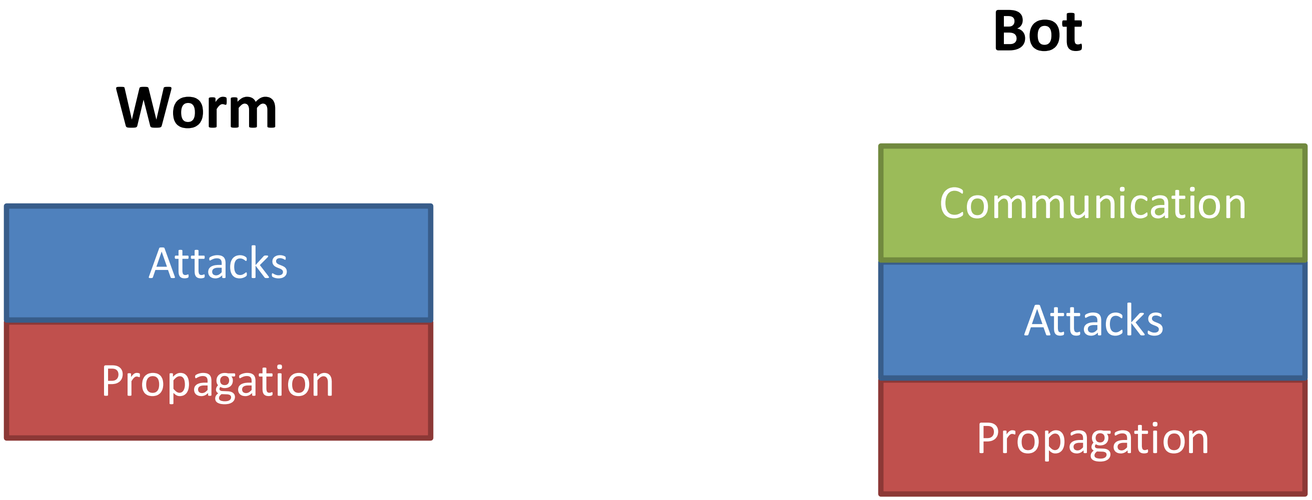
Detection

- Detect via honeypots
  - Any outbound connection from honeyfarm = worm.
  - Extract *signature* from inbound/outbound traffic.

- Detect via failed connections
  - network elements that identify the hosts that make failed connection attempts to too many other hosts.

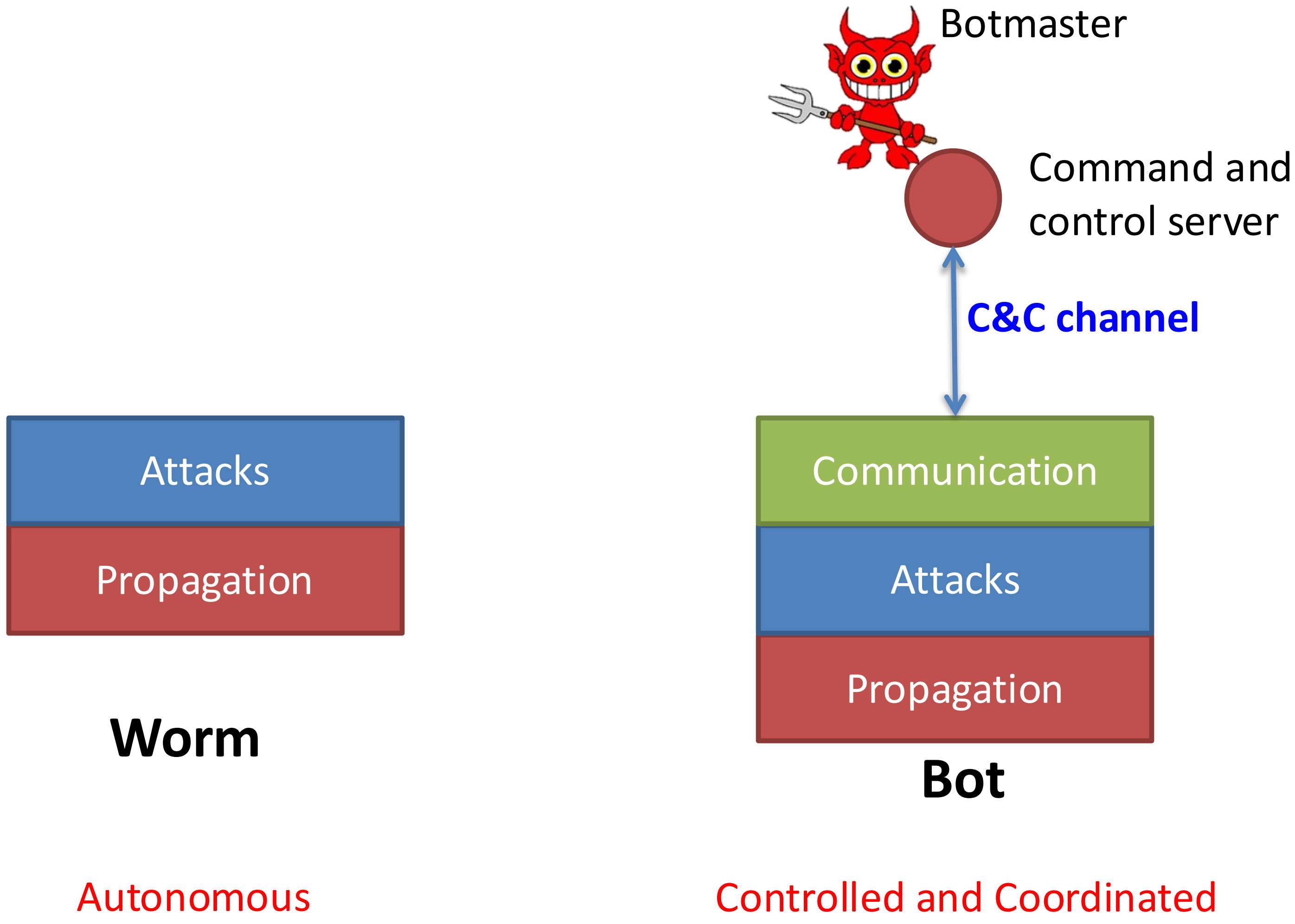
- Detect via exploit content
  - BOF
  - Spam

Worm => Bot



Worm
Attacks
Propagation
Bot
Communication
Attacks
Propagation

Worm => Bot


Botmaster
Command and control server
C&C channel
Attacks
Propagation
Worm
Communication
Attacks
Propagation
Bot
Autonomous
Controlled and Coordinated

Worm => Bot

worm

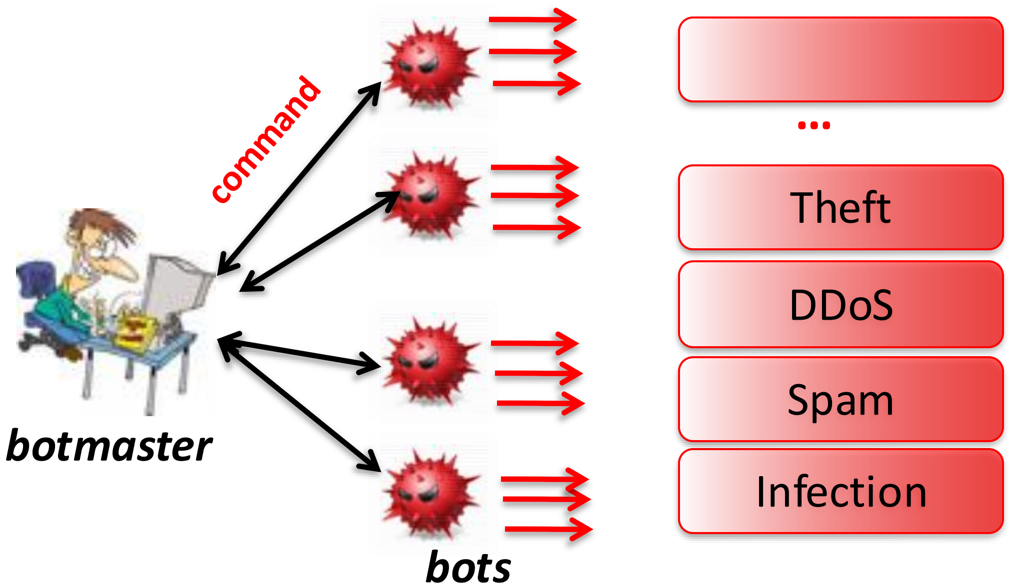
```
main(){
        foreach(h in ScanList){
                infect(h);
        }
}
```

bot

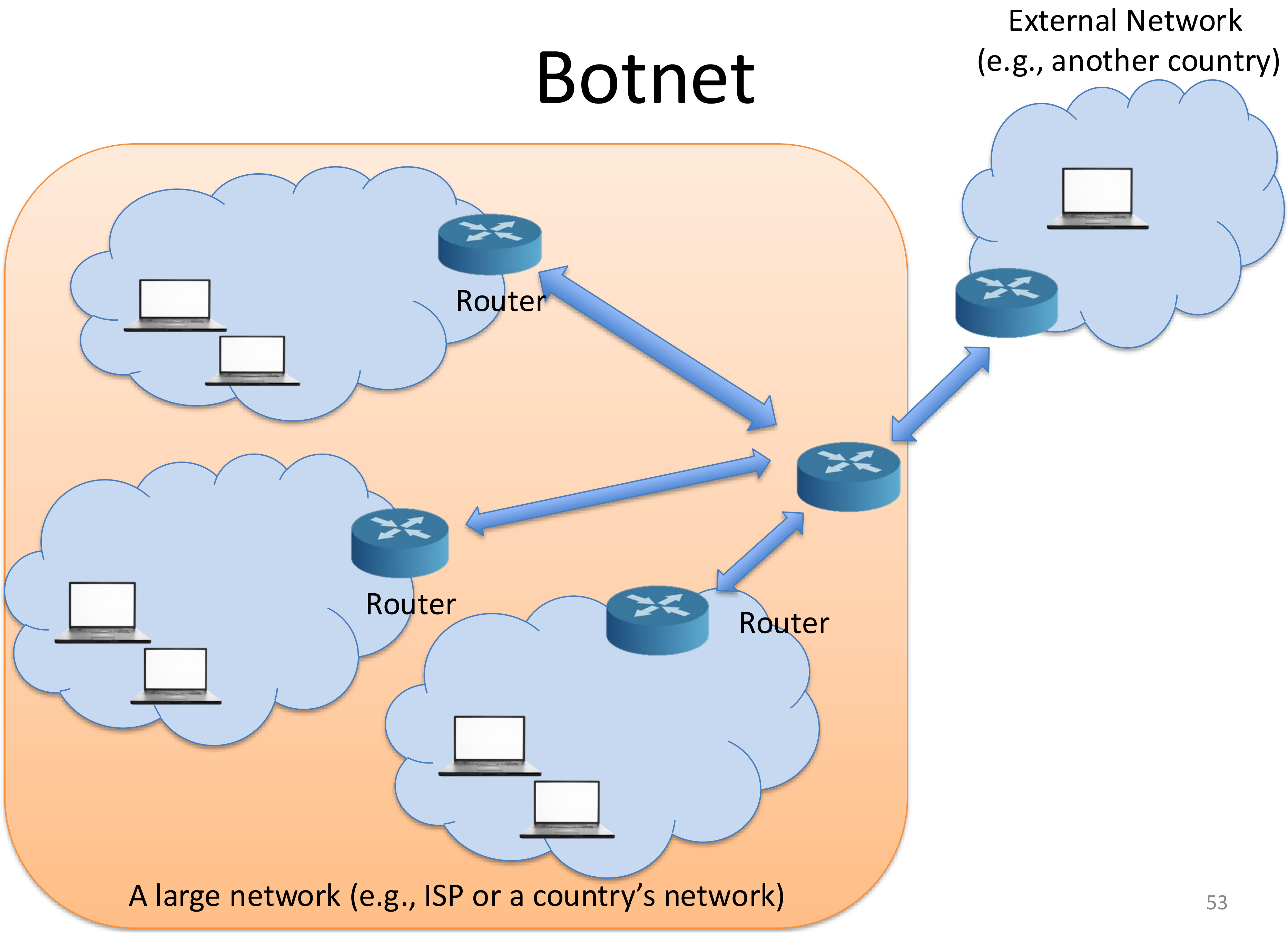
```
main(){
        cnc = connect(C&C server);
        while(every 5 minutes){
                msg = cnc.read();
                cmd = msg.cmd;
                par = msg.parameter;

                if(cmd.equal("scan")){
                        scanlist = par;
                        foreach(h in ScanList){
                                infect(h);
                        }
                }

                if(cmd.equal("spam")){
                        //send spam…..
                }
        }
}
```

Botnet

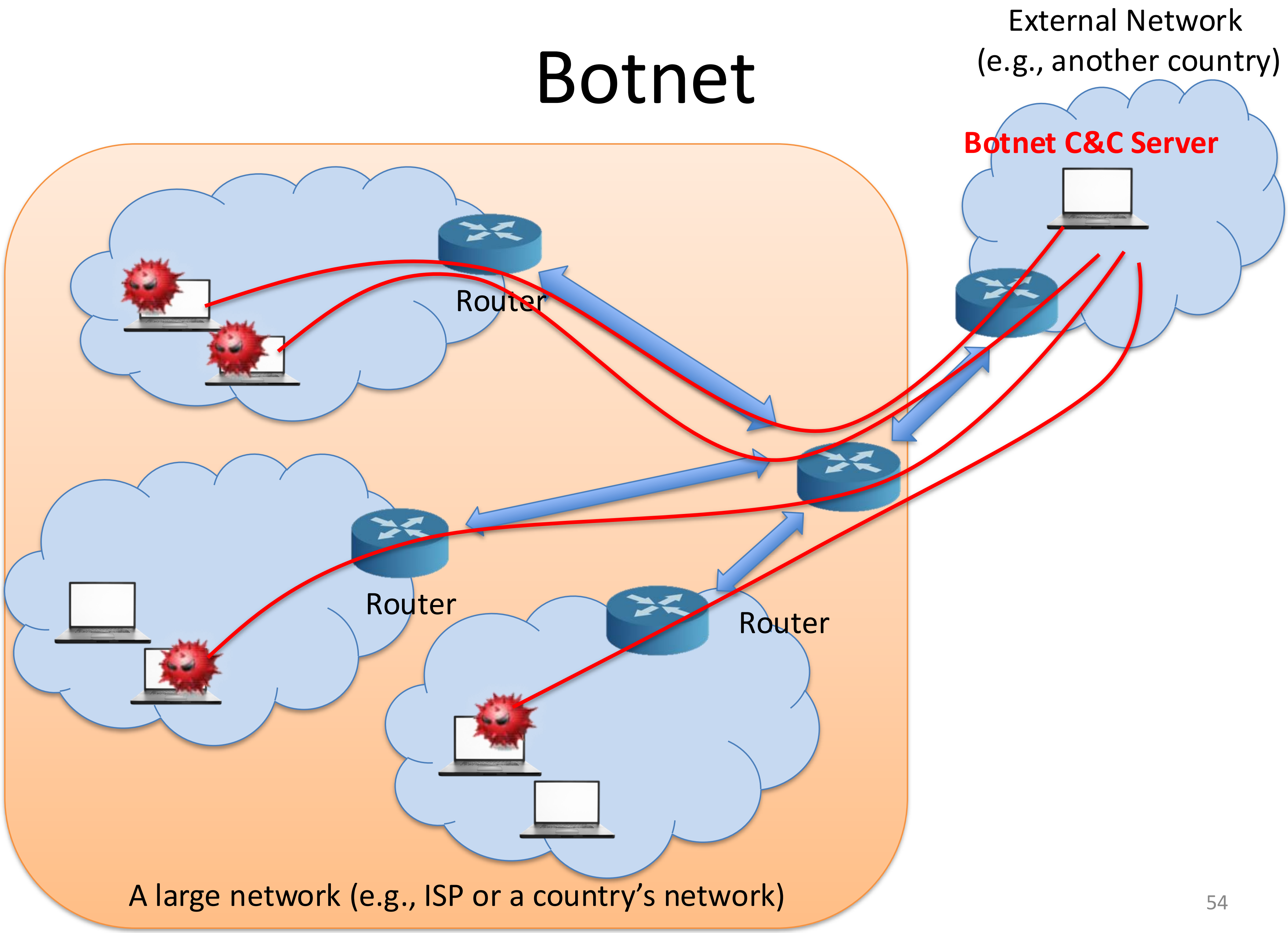
- A botnet is a collection of ***bot-compromised hosts (bots)*** that are coordinated via ***a command and control (C&C) channel*** by an attacker to commit a variety of attacks



Botnet


External Network
(e.g., another country)
Router
Router
Router
A large network (e.g., ISP or a country's network)

Botnet

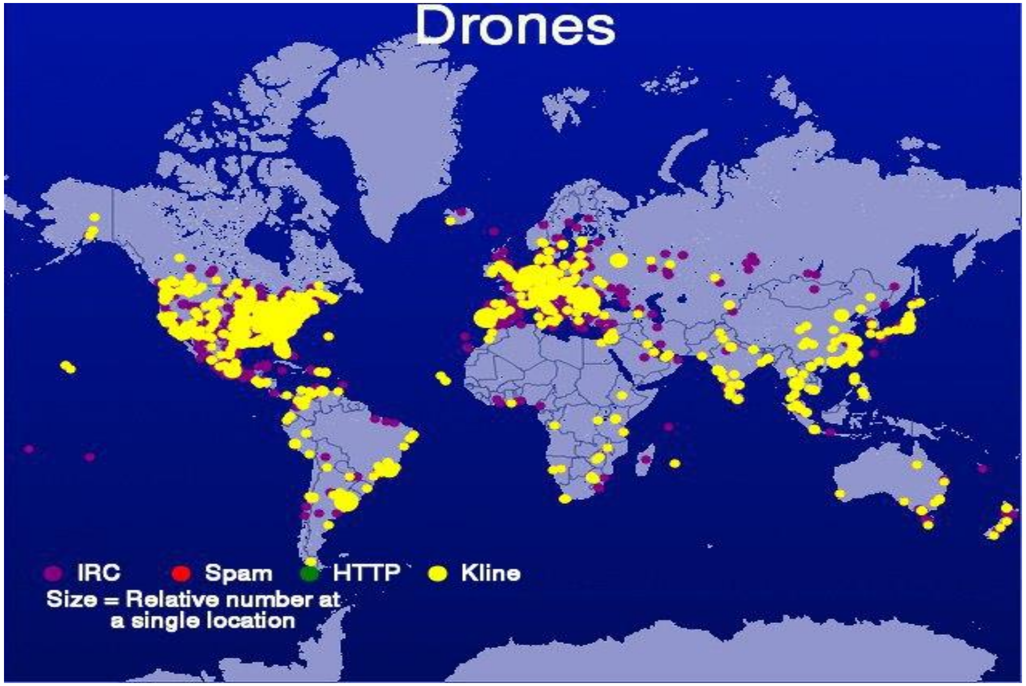


External Network
(e.g., another country)
Botnet C&C Server
Router
Router
Router
A large network (e.g., ISP or a country's network)

Botnet

- The infrastructure responsible for a variety of cyber attacks
  - Distributed Denial of Service
  - Spamming
  - Click fraud
  - Phishing
  - ……

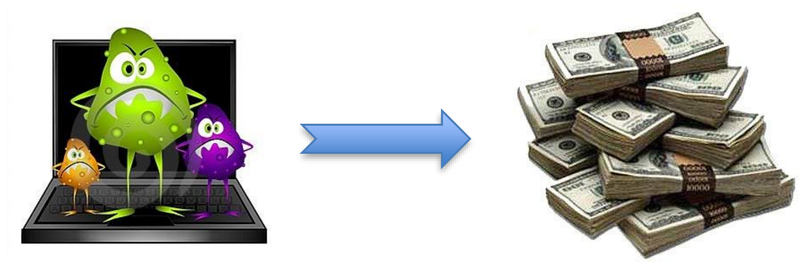
Botnet – A Global Problem

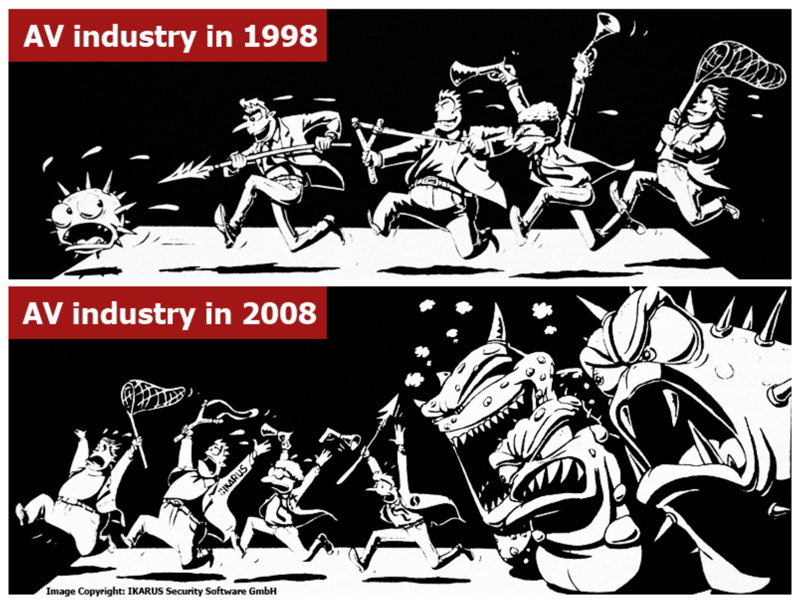


One quarter of the Internet is infected by malware

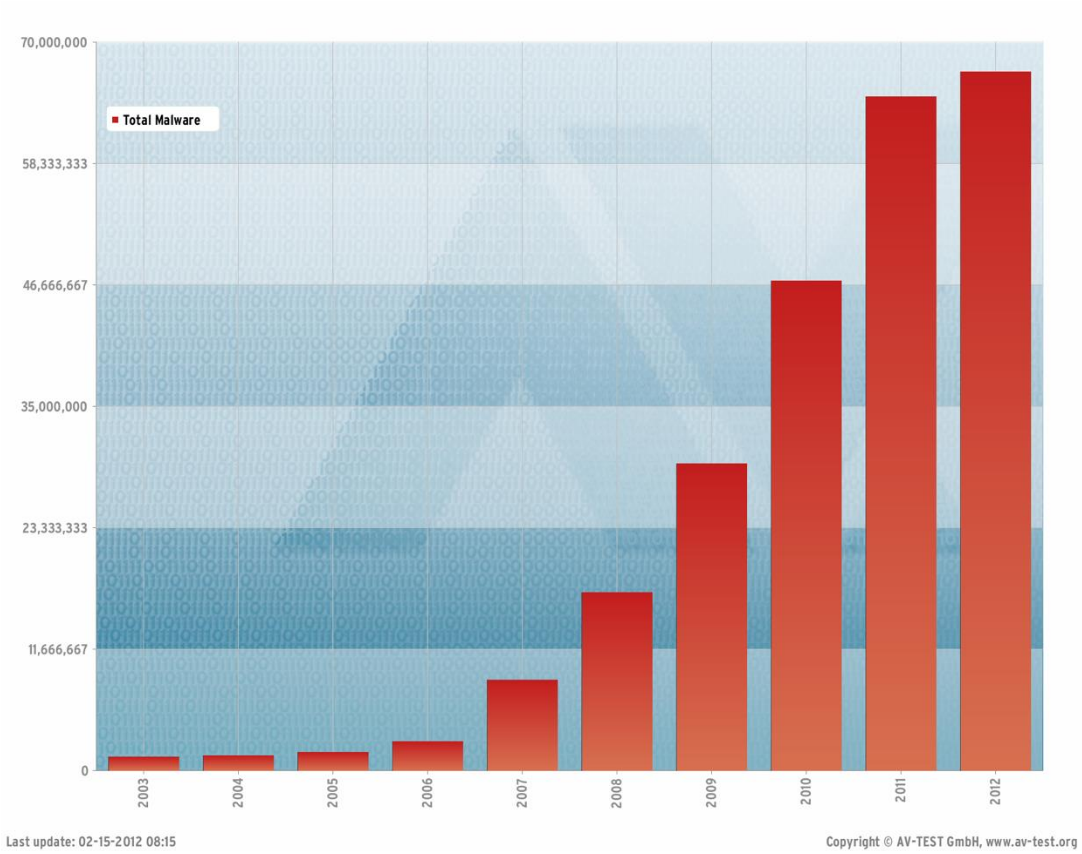
Source: Vint Cerf, "father of the Internet"

Botnet - A Persistent Threat





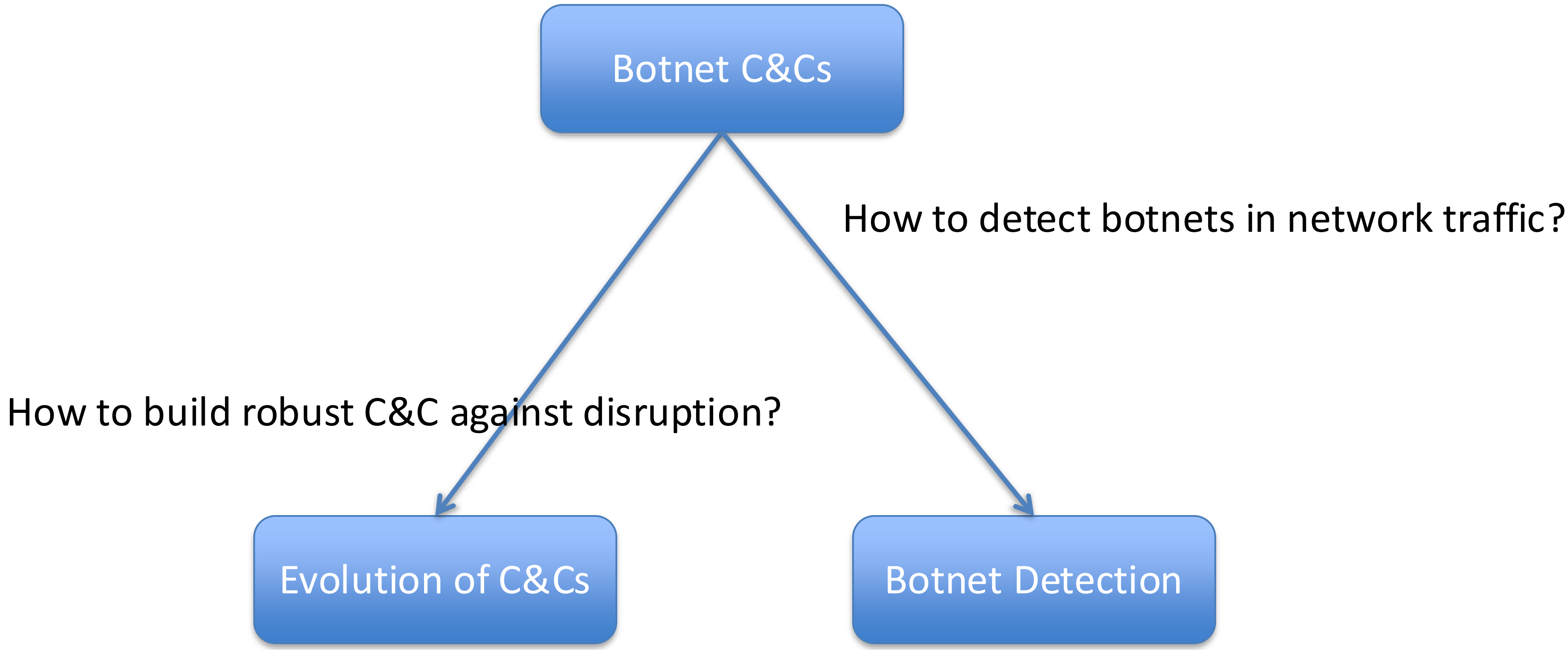
Botnet - A Persistent Threat



Botnet - A Persistent Threat

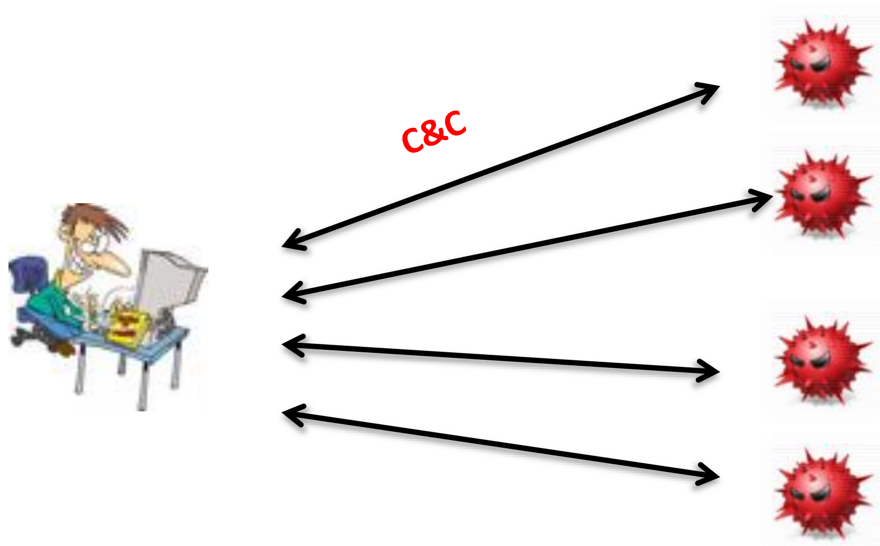
Supervisory control and data acquisition (SCADA) systems

Roadmap


Botnet C&Cs
How to detect botnets in network traffic?
How to build robust C&C against disruption?
Evolution of C&Cs
Botnet Detection

Botnet C&Cs

- C&C channels are essential to botnets
  - Without C&C, a botnet will be degraded into individual bots



The War Field

Botmasters

- Objective: keep most of their bots operational against disruption efforts
- Resources: access public network resources such as registering domains, setup servers, and etc.

Network Operators

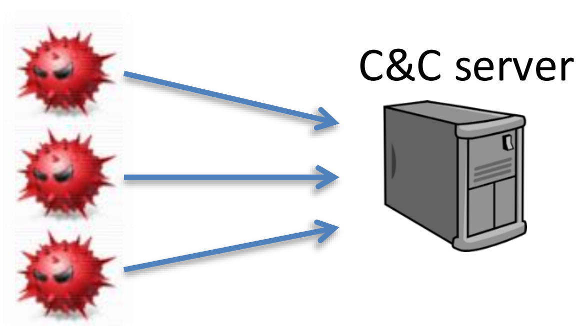
- Objective: disrupt botnets at scale using affordable costs
- Resources
  - Bot binary
  - Execute it for a short time (efficient)
  - Capability of reverse engineering (labor-intensive)
  - Access public network resources such as DNS
  - Limited financial resources

Build Robust C&Cs – Round 1

A Bot

- Directly connect to one C&C server or a small number of C&C servers based on IP(s) (i.e., botnet with Centralized C&Cs)

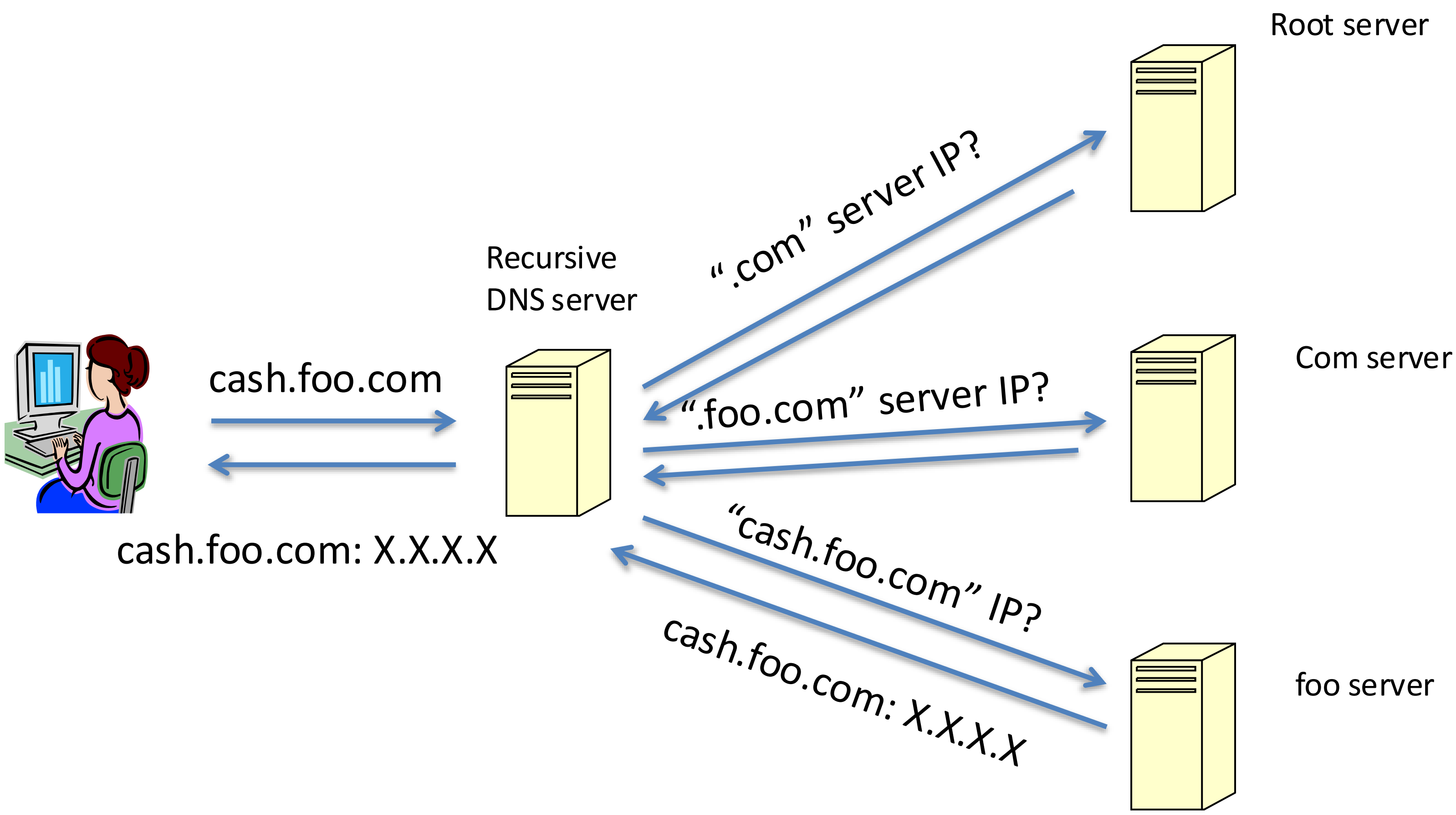
```
main(){
    cnc = connect("111.111.111.111");
    while(every 5 minutes){
        msg = cnc.read();
        ......
    }
}
```



Disruption Strategy

- Block all connections to the IP address(es)
  - Network-level
  - Efficient
- Take down the server if you are capable of doing it
  - You may not be able to do it if the server is out of your control (e.g., in foreign countries)

How does DNS work?



Build Robust C&Cs – Round 2

A Bot

- Use DNS to add more agility to the C&C server
  - E.g., give new IP every time when a bot issues the DNS query
- Centralized C&C

```
main(){
	cnc = connect("www.malicious.com");
	while(every 5 minutes){
		msg = cnc.read();
		……
	}
}
```

Disruption Strategy

- IP + firewall may not work well
- Execute the binary
- Hijack the malicious domain in your network
- Collaborate with DNS service to disrupt the DNS record
  - You may not be able to do it
- Take down the C&C server(s) if you can

Build Robust C&Cs – Round 3

A Bot

- User automatically generated DNS
  - E.g., automatically generate a new domain name every day;
- Attacker randomly pick one domain for each day and register the domain

Disruption Strategy

- Reverse engineering the bot executable and discover those automatically generated domains
- Proactively register all of them (***N*** domain names)
  - Cost: ***N * R***
  - ***R:*** the cost to register one domain

Build Robust C&Cs – Round 4.1

Conficker D variant: 50,000 domains across 110 TLDs per day!

A Bot

- User automatically generated DNS
  - E.g., automatically generate ***a lot*** domain names every day;
  - Query all of them;
  - Use the IP that is successfully resolved;
- Attacker randomly pick one domain for each day and register the domain

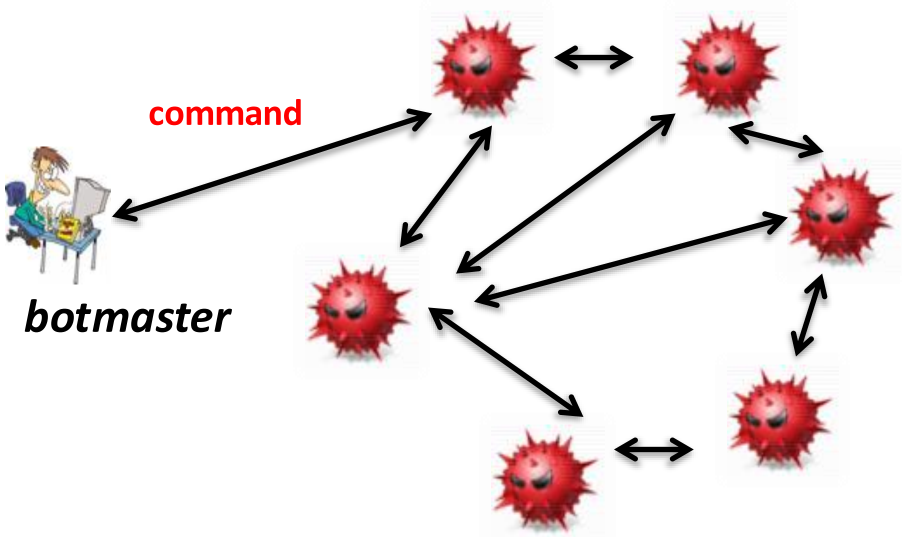
Disruption Strategy

- Reverse engineering the bot executable and discover those automatically generated domains
- Proactively register all of them (***N*** domain names)
  - Cost: ***N * R***
  - ***R:*** the cost to register one domain

Build Robust C&Cs – Round 4.2

A Bot

- Eliminate the centralized C&Cs
- Build a C&C with peer-to-peer structure



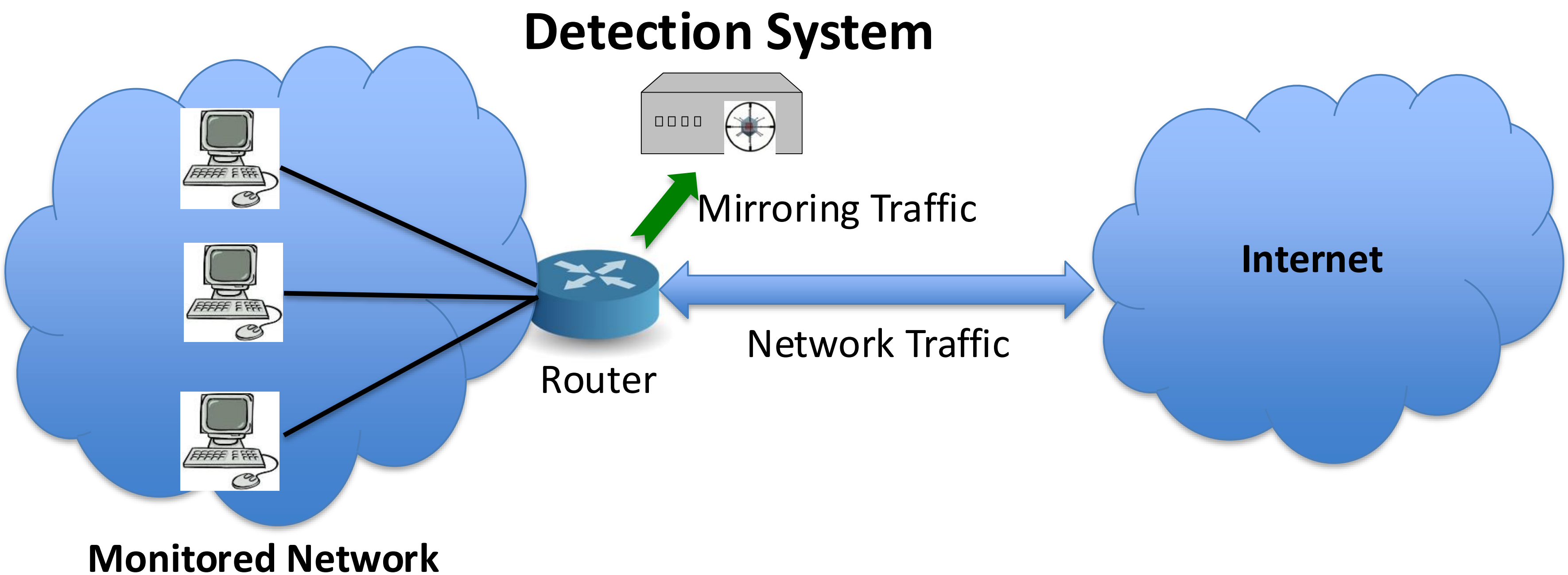
Disruption Strategy

- Identify bots and take them down. But the whole botnet is still functional unless a significantly large portion of peer-to-peer bots are taken down

Examples: Storm, Waledac, Conficker

P2P Botnets

Detecting Centralized IRC-Based Botnet



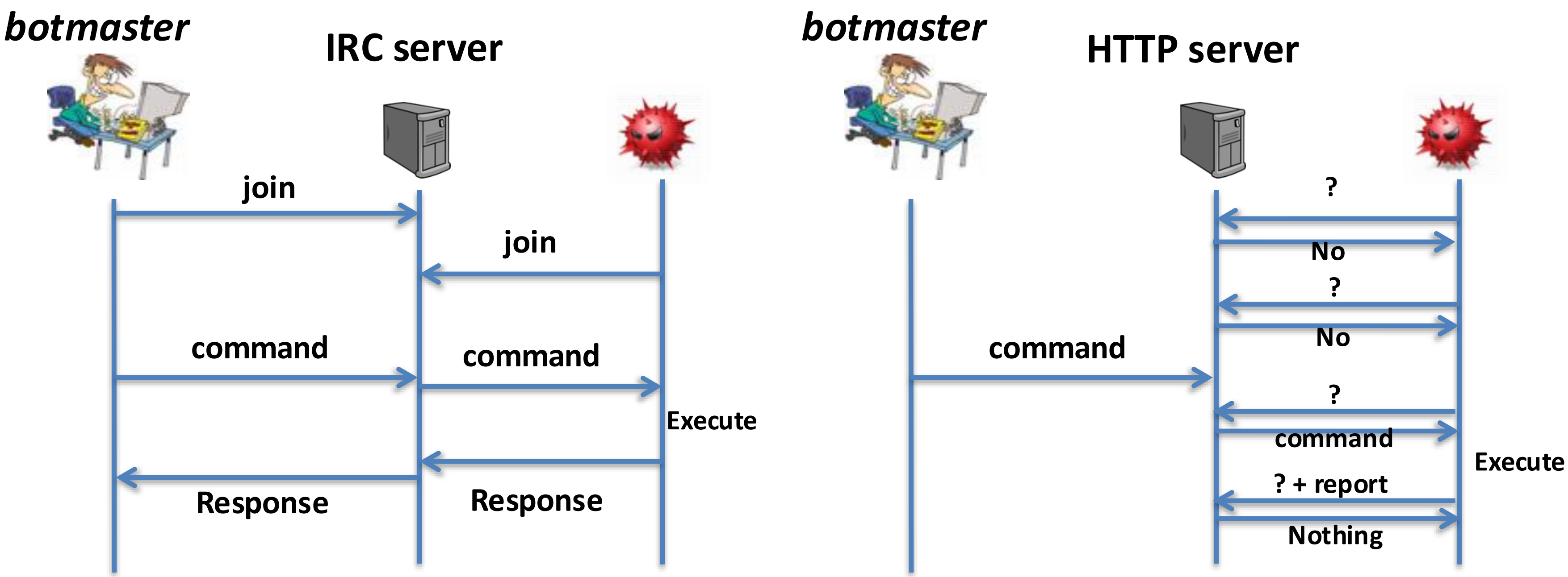
- G. Gu, J. Zhang, and W. Lee, "*BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic*", NDSS 2008

Detecting C&C Channels

- C&C is essential to a botnet
  - Without C&C, a botnet will be degraded into individual infections, incapable of launching large-scale and coordinated attacks
- Detecting C&C is important
  - Reveal both C&C servers and bots
  - An effective way to mitigate botnet threats

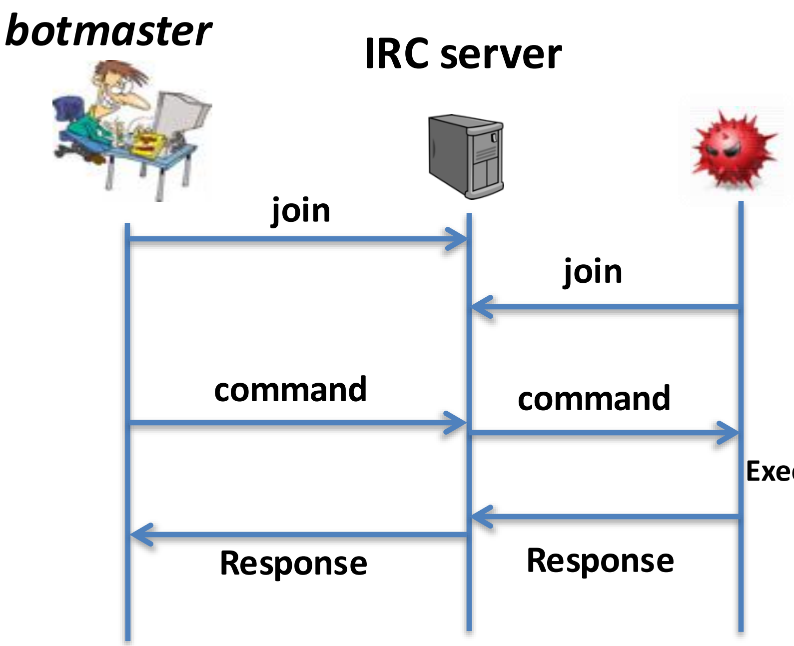
Bot Examples

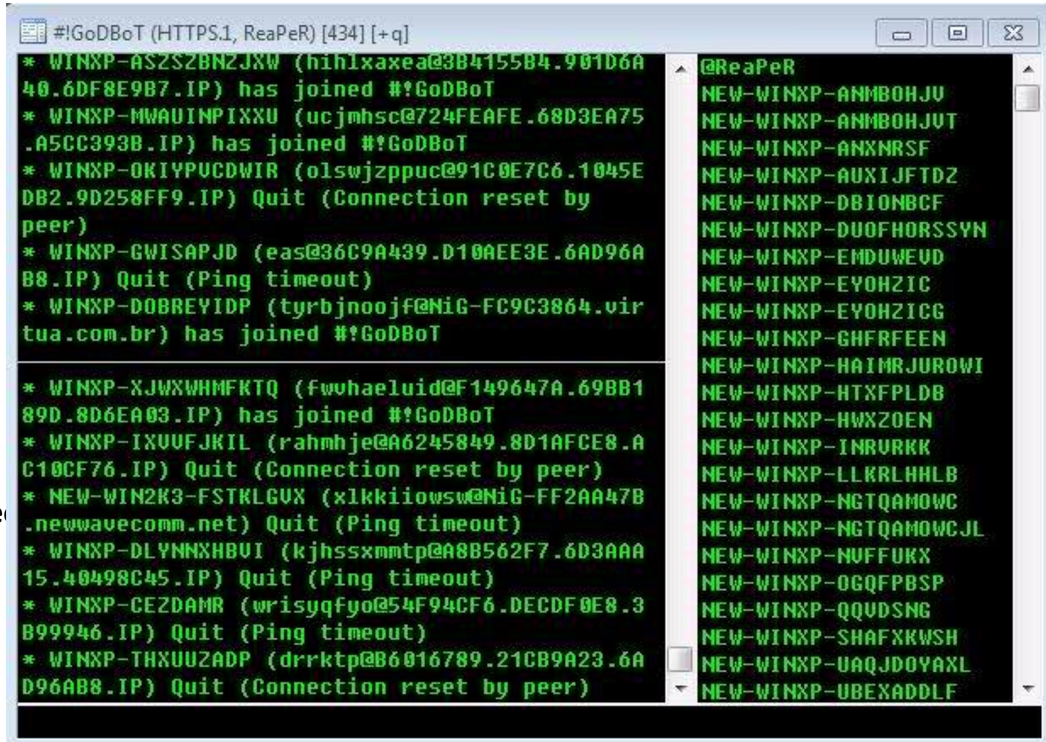
- IRC/HTTP-based C&C



Bot Examples

- IRC-based C&C



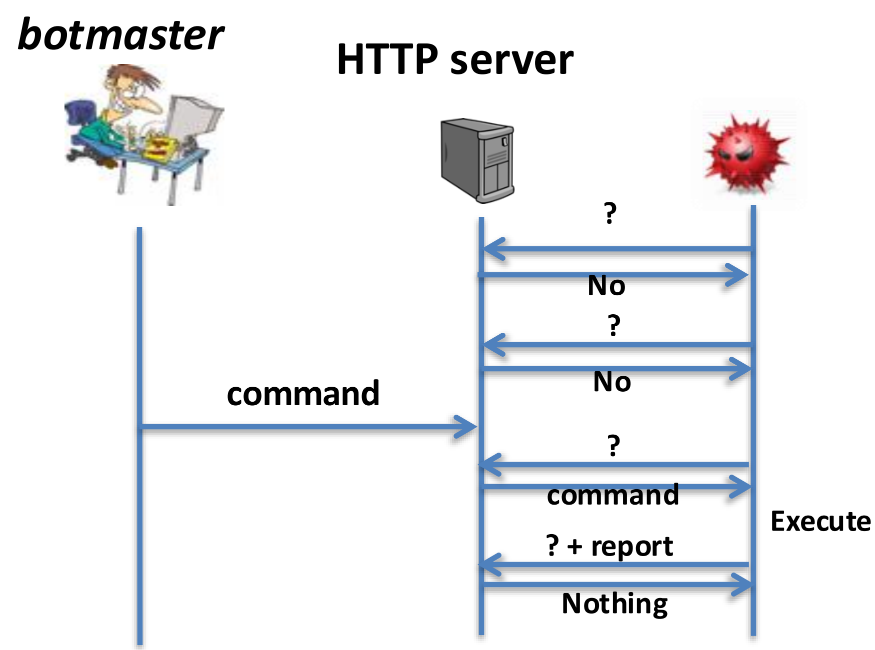


Bot Examples

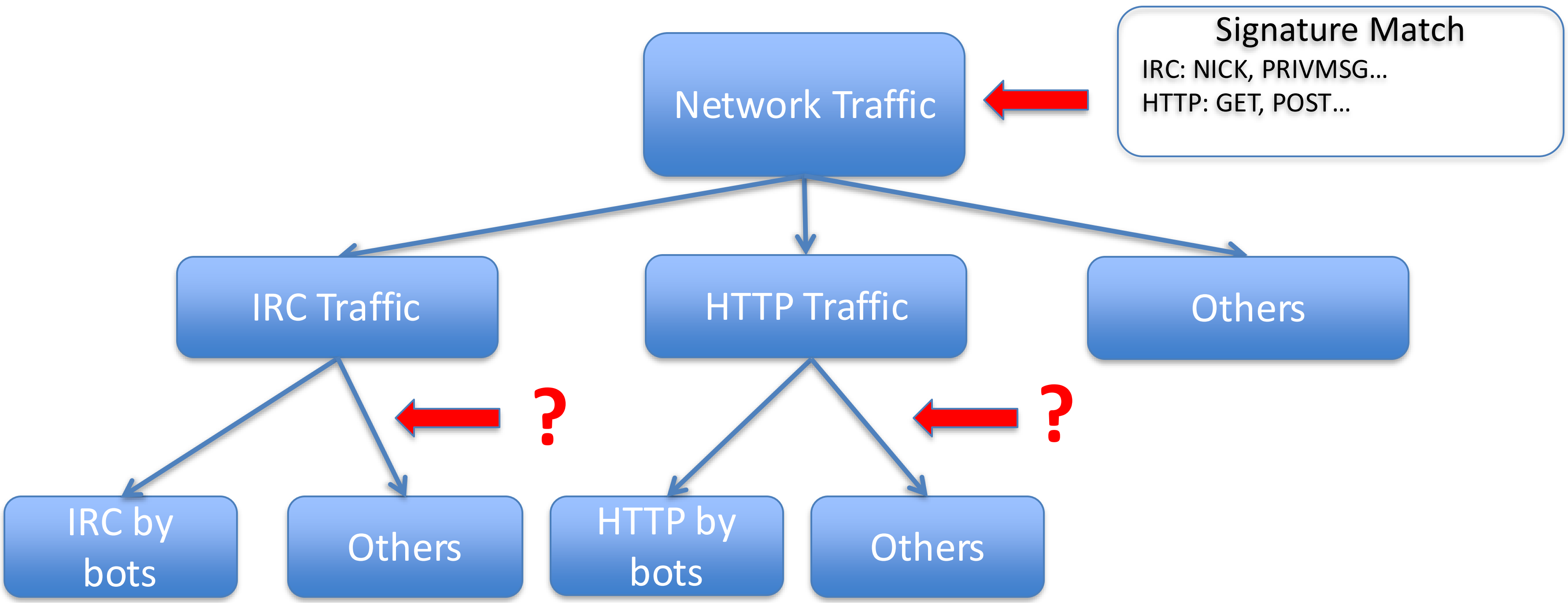
- HTTP-based C&C

IP	Country	Time	Clicks	Version	Manage
		03:30:05	0	v0.007	Block
		03:30:04	Holded	v0.007	Allow
		03:30:04	8	v0.007	Block
		03:30:04	0	v0.007	Block
		03:30:04	0	v0.007	Block
		03:30:04	3	v0.007	Block
		03:30:03	Holded	v0.007	Allow
		03:30:03	Holded	v0.007	Allow
		03:30:03	Holded	v0.007	Allow
		03:30:03	14	v0.007	Block

Source: "The Anatomy of Clickbot.A"



Roadmap


Network Traffic
Signature Match
IRC: NICK, PRIVMSG…
HTTP: GET, POST…
IRC Traffic
HTTP Traffic
Others
?
?
IRC by bots
Others
HTTP by bots
Others

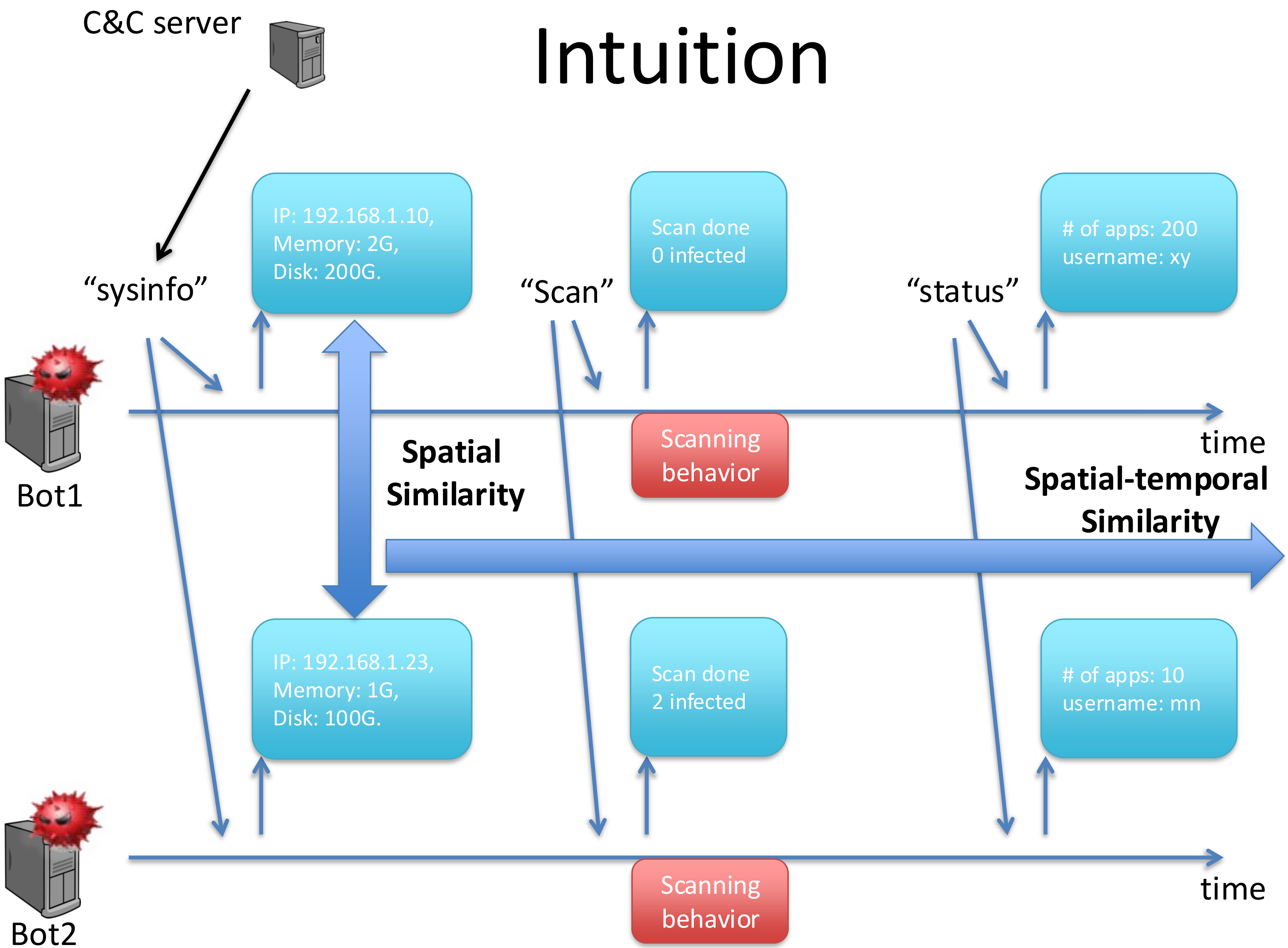
Intuition

C&C server

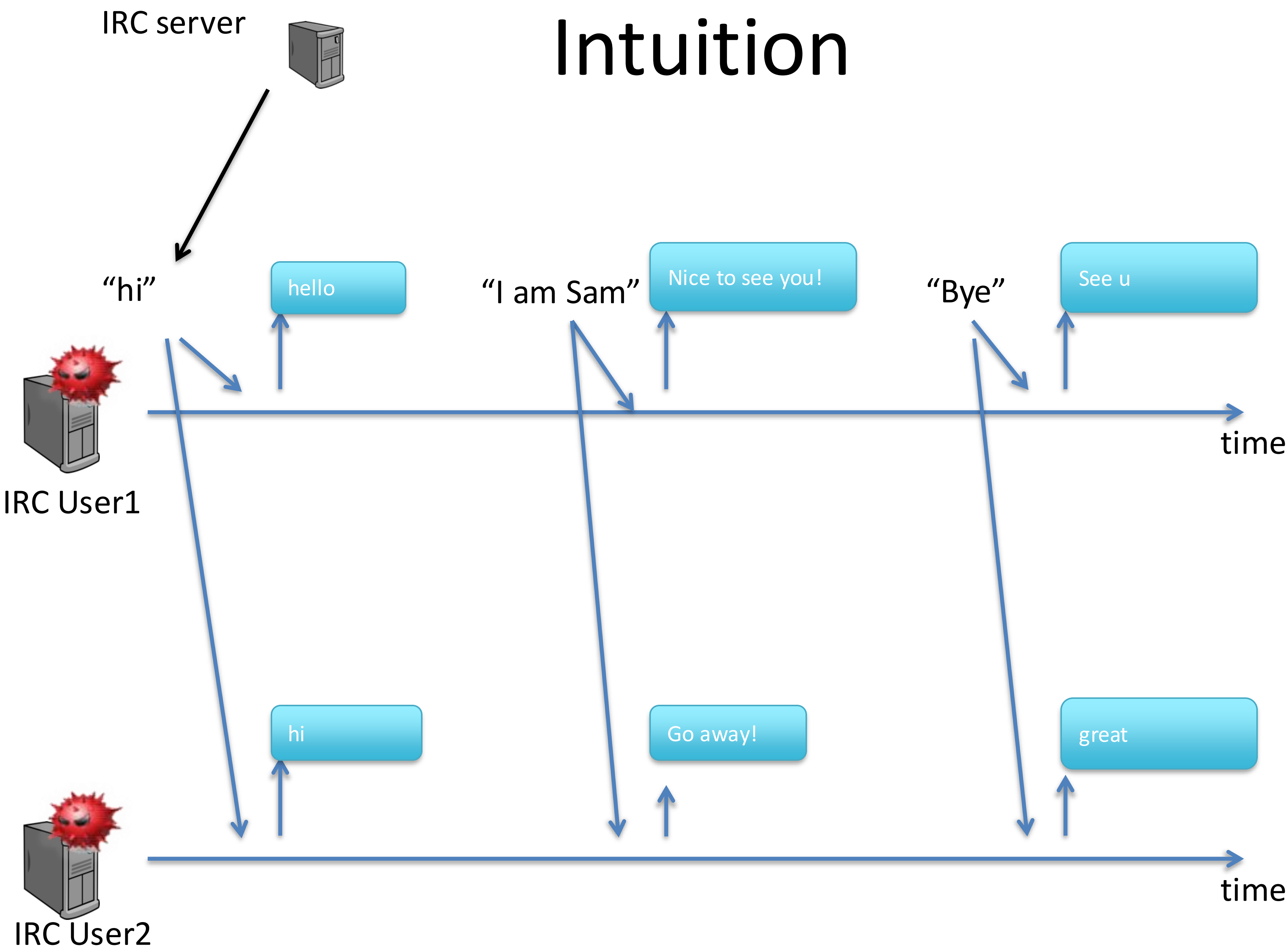
Bot

```
Establish the C&C connection;
while(cmd=receive(server))
{
        If(cmd.match("sys.info"))
        {
                mem = OS.getMemSize();
                disk = OS.getDiskSize();
                IP = OS.getIP();
                send(" IP: " + IP + " Mem: " + mem + " Disk" + disk);
        }
        If(cmd.match("scan"))
        {
                n = Attack.scan();
                send("scan done " + n + " infected");
        }
}
```

Intuition


C&C server
"sysinfo"
IP: 192.168.1.10,
Memory: 2G,
Disk: 200G.
"Scan"
Scan done
0 infected
"status"
# of apps: 200
username: xy
Bot1
Scanning behavior
time
Spatial Similarity
Spatial-temporal Similarity
IP: 192.168.1.23,
Memory: 1G,
Disk: 100G.
Scan done
2 infected
# of apps: 10
username: mn
Bot2
Scanning behavior
time

Intuition



IRC server
"hi"
hello
"I am Sam"
Nice to see you!
"Bye"
See u
time
IRC User1
hi
Go away!
great
time
IRC User2

Intuition

- Detection algorithm in a nutshell
  - For an IRC server, if the majority of its clients in our monitored network keep responding similar responses to the server, we will label the clients as bots and the server as C&C server

How can we expand our intuition to a system?

Intuition

- Detection algorithm in a nutshell
  - For an ***IRC server***, if the ***majority*** of its clients in our monitored network ***keep responding similar responses*** to the server, we will label the clients as bots and the server as C&C server

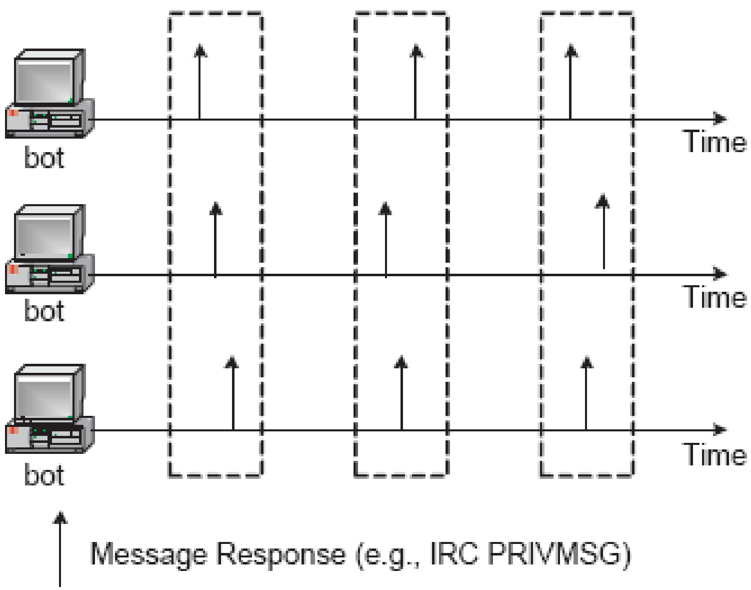
Edit Distance: Use DICE coefficient to evaluate the similarity between two texts

$$Dice(X,Y) = \frac{2|ngrams(X) \cap ngrams(Y)|}{|ngrams(X)| + |ngrams(Y)|}$$

E.g., "abcde" and "bcdef", common 2-grams: "bc,cd,de", DICE distance is 2*3/(4+4)=6/8=0.75

SPRT(Sequential Probability Ratio Testing): calculate an anomaly score by observing a sequence of certain actions (e.g., similar/dissimilar behaviors)

Detection



Experiments

189 days' of IRC traffic

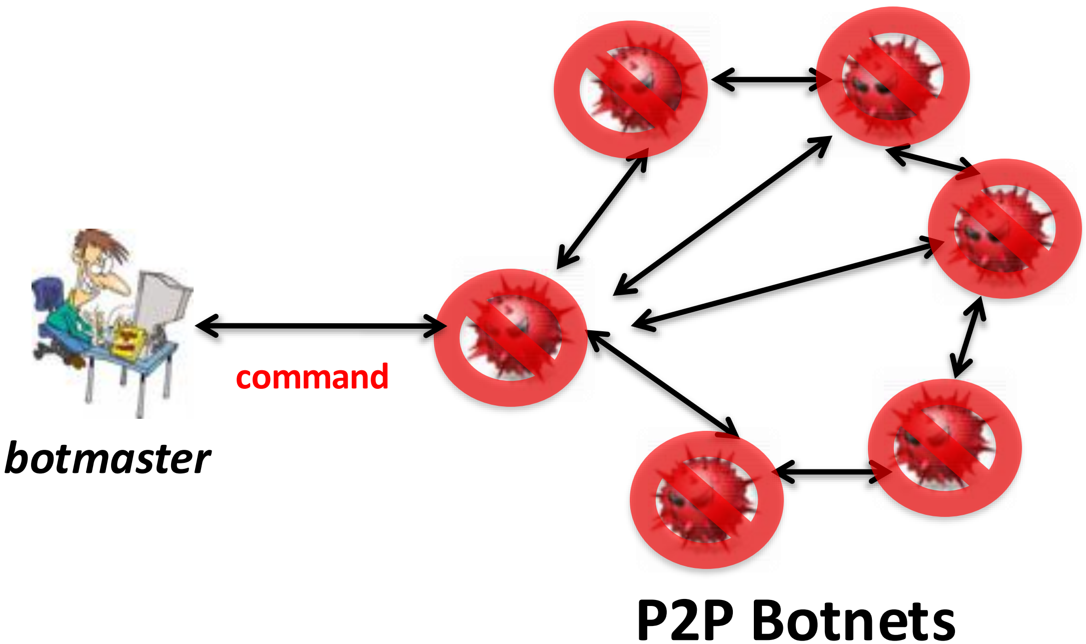
Trace	trace size	duration	Pkt	TCP flows	(IRC/Web) servers	FP
IRC-1	54MB	171h	189,421	10,530	2,957	0
IRC-2	14MB	433h	33,320	4,061	335	0
IRC-3	516MB	1,626h	2,073,587	4,577	563	6
IRC-4	620MB	673h	4,071,707	24,837	228	3
IRC-5	3MB	30h	19,190	24	17	0
IRC-6	155MB	168h	1,033,318	6,981	85	1
IRC-7	60MB	429h	393,185	717	209	0
IRC-8	707MB	1,010h	2,818,315	28,366	2,454	1
All-1	4.2GB	10m	4,706,803	14,475	1,625	0
All-2	6.2GB	10m	6,769,915	28,359	1,576	0
All-3	7.6GB	1h	16,523,826	331,706	1,717	0
All-4	15GB	1.4h	21,312,841	110,852	2,140	0
All-5	24.5GB	5h	43,625,604	406,112	2,601	0

Experiments

BotTrace	trace size	duration	Pkt	TCP flow	Detected
B-IRC-G	950k	8h	4,447	189	Yes
B-IRC-J-1	-	-	143,431	-	Yes
B-IRC-J-2	-	-	262,878	-	Yes
V-Rbot	26MB	1,267s	347,153	103,425	Yes
V-Spybot	15MB	1,931s	180,822	147,921	Yes
V-Sdbot	66KB	533s	474	14	Yes
B-HTTP-I	6MB	3.6h	65,695	237	Yes
B-HTTP-II	37MB	19h	395,990	790	Yes

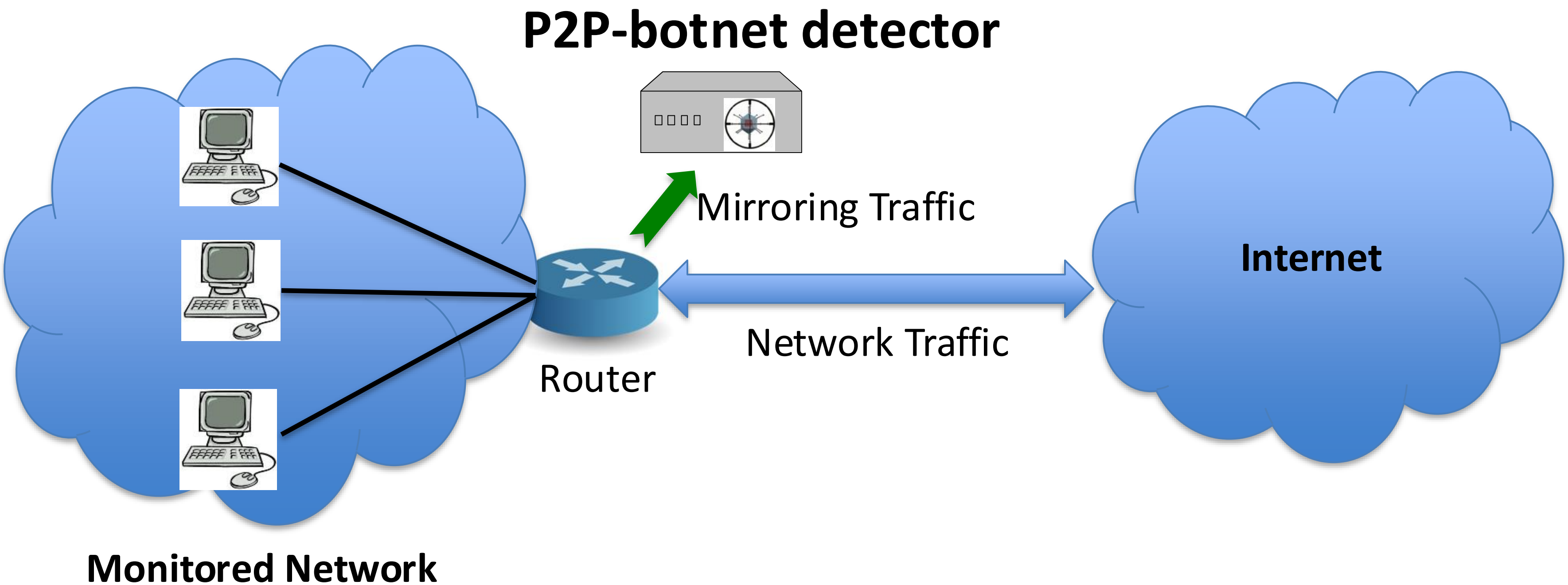
Detecting P2P Botnets

- **P2P Botnet**: A botnet with a peer-to-peer C&C structure

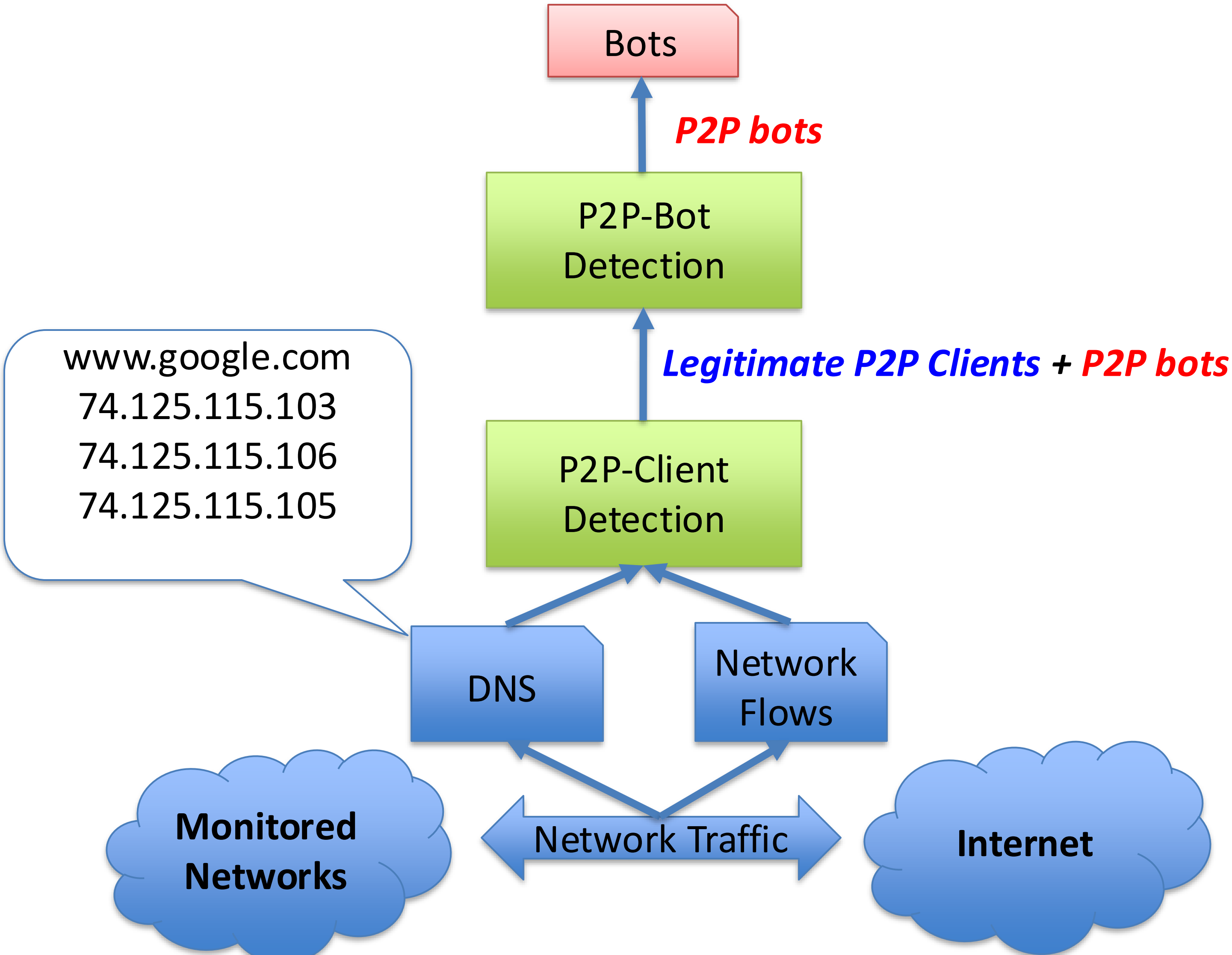


Goal

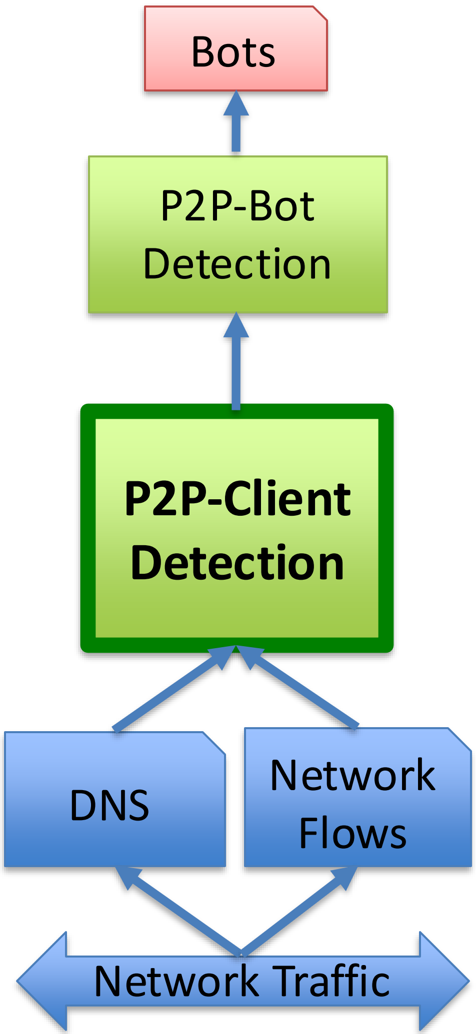
- A network-based P2P-botnet detection system

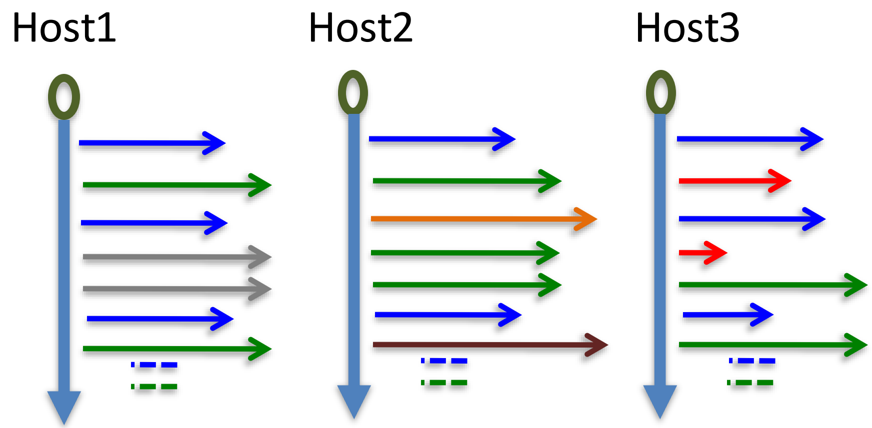


System Architecture

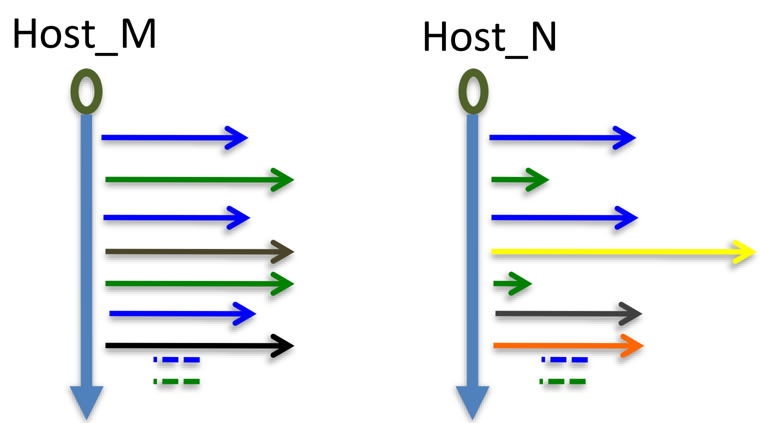


P2P Client Detection

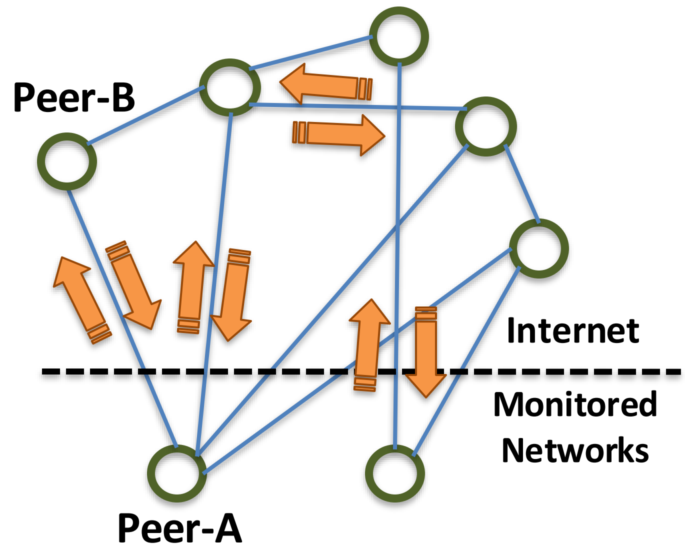




Which hosts are P2P Clients?



P2P Client Detection

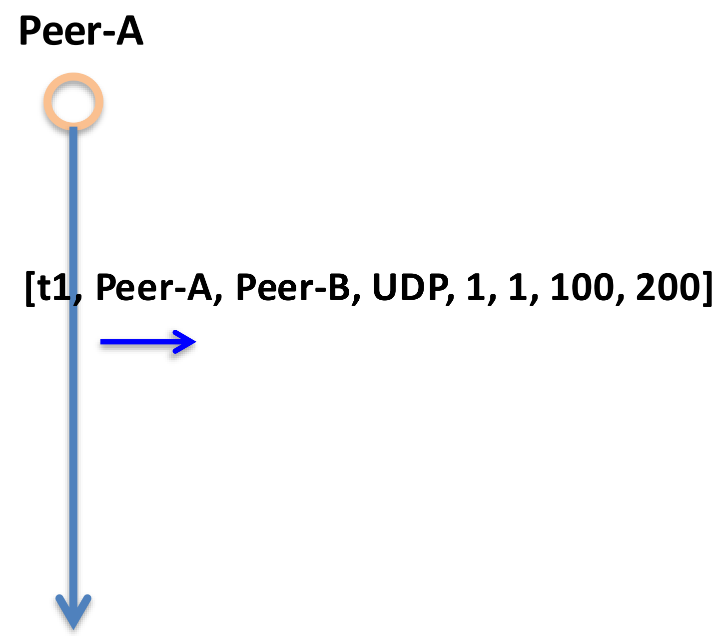


- P2P Control Messages
  - Used to maintain the functionality of P2P networks
  - Essential to any P2P network
  - Examples: PING/PONG, Routing Update, Content Publish
- ***A P2P Control Flow:*** A network flow corresponding to a network session of control messages

An Example of A P2P Control Flow

PING/PONG

```
while(every 3 miniutes)
{
    foreach(p in PeerList)
    {
        var s = new Connection(p);
        s.send("PING" + self.time() + self.id());
        var data = s.receive();
        s.close();

        //process data
    }
}
```



Is this client a P2P client?

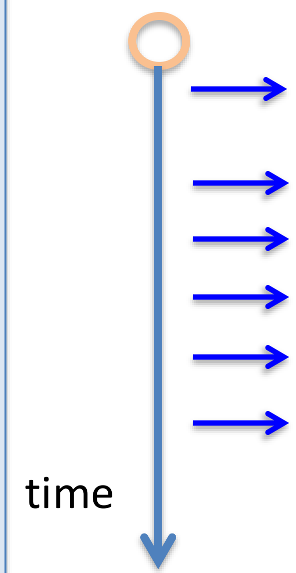
=

Does this client generate P2P control flows?

Characteristics of P2P Control Flows

PING/PONG

```
while(every 3 miniutes)
{
    foreach(p in PeerList)
    {
        var s = new Connection(p);
        s.send("PING" + self.time() + self.id());
        var data = s.receive();
        s.close();

        //process data
    }
}
```

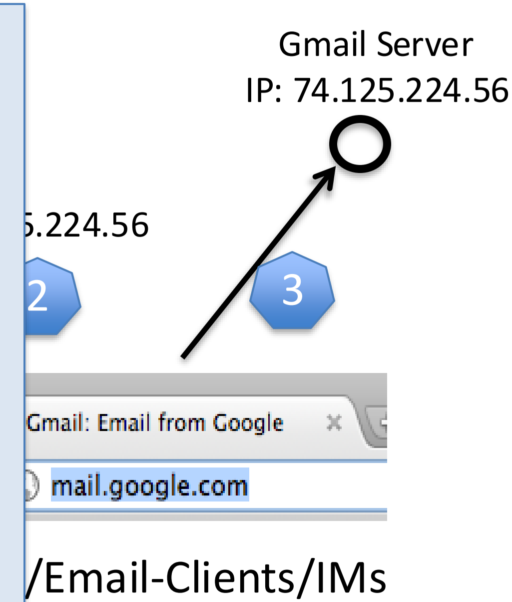


1. **Automatically generated**
2. ……
3. ……
4. ……

Characteristics of P2P Control Flows

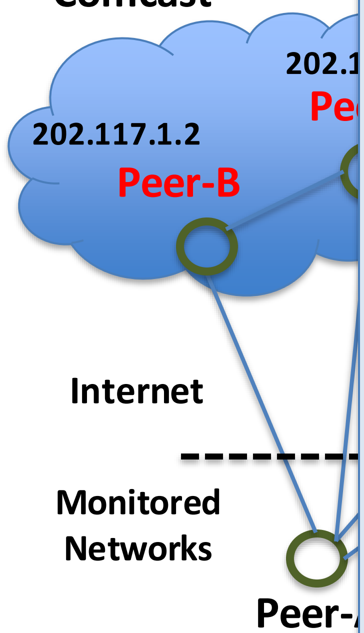
PING/PONG

```
while(every 3 miniutes)
{
    foreach(p in PeerList)
    {
        var s = new Connection(p);
        s.send("PING" + self.time() + self.id());
        var data = s.receive();
        s.close();

        //process data
    }
}
```



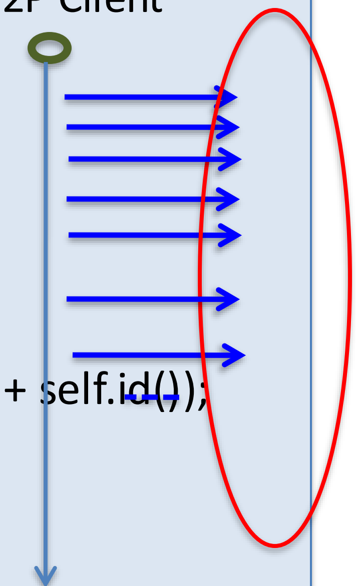
2. **DNS-Free, different from popular network applications**
3. ……
4. ……

Characteristics of P2P Control Flows



PING/PONG

```
while(every 3 miniutes)
{
    foreach(p in PeerList)
    {
        var s = new Connection(p);
        s.send("PING" + self.time() + self.id());
        var data = s.receive();
        s.close();

        //process data
    }
}
```



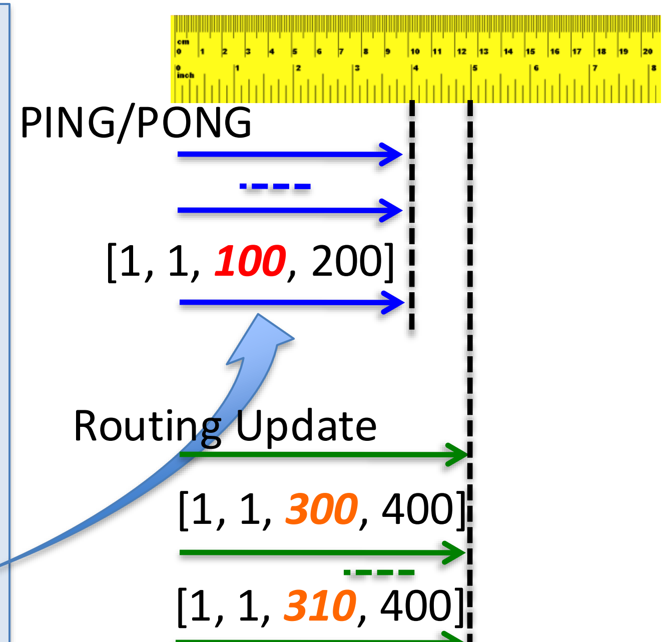
Comcast
AT&T
Chinacom
Eurocom
……

1. Automat
2. DNS-Free, different from popular network applications
3. **Sent to a large number of organizations (represented by network prefixes)**
4. ……

Characteristics of P2P Control Flows

PING/PONG

```
while(every 3 miniutes)
{
    foreach(p in PeerList)
    {
        var s = new Connection(p);
        s.send("PING" + self.time() + self.id());

        var data = s.receive();
        s.close();

        //process data
    }
}
```

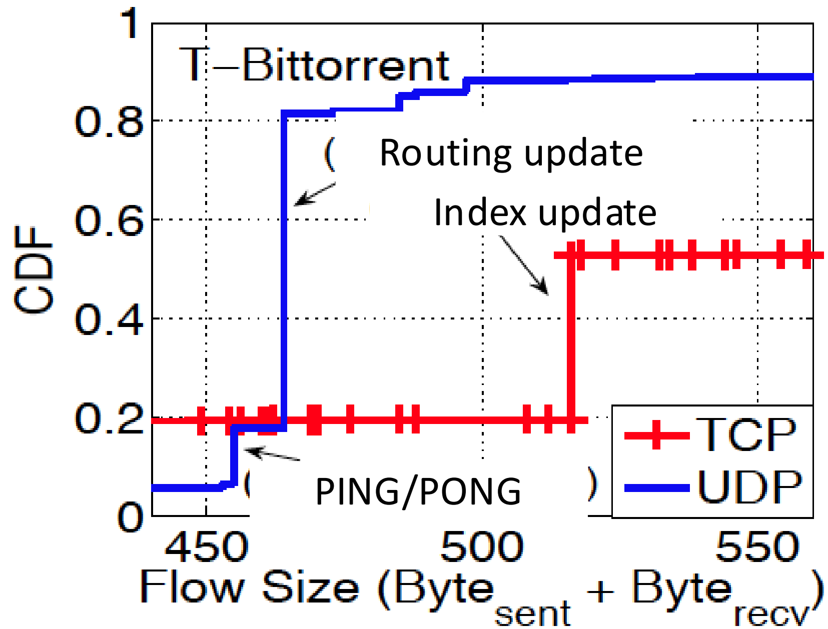


Routing Update

s.send(p, *"Routing " + self.RoutingTable()*);

4. P2P control flows used for the same purpose will share similar flow size regarding the same P2P application

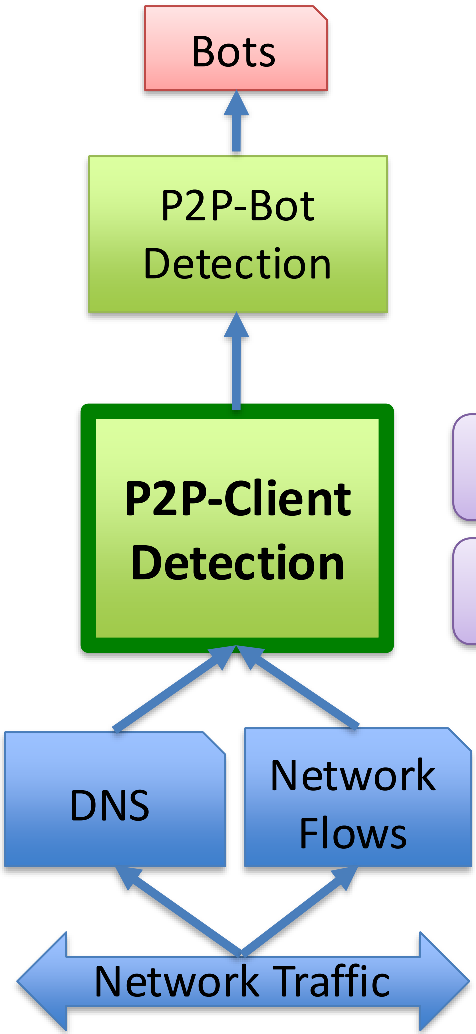
Characteristics of P2P Control Flows



- P2P Control flows for the same purpose (e.g., PING/PONG) share similar flow size

- P2P Control flows for different purposes (e.g., PING/PONG vs. Routing Update) have different flow sizes

P2P Client Detection



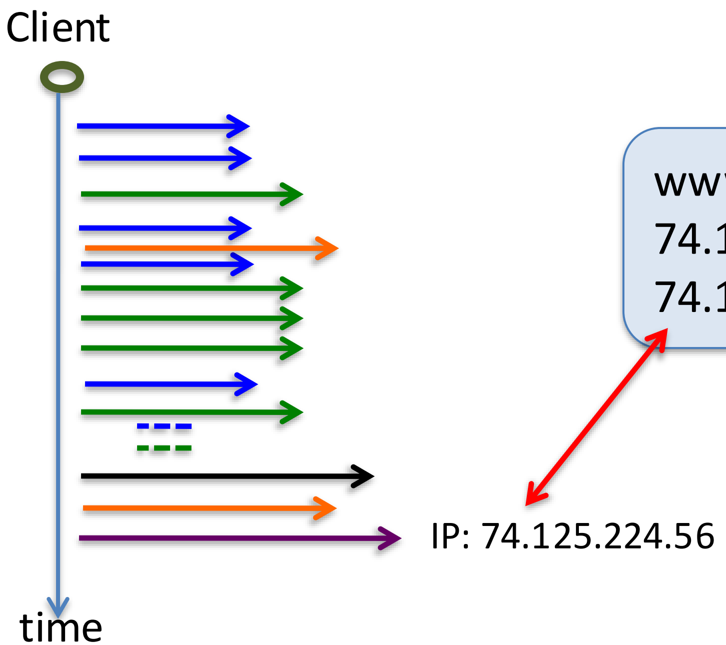
Clustering-Based Flow Analysis

DNS-Based Traffic Filtering

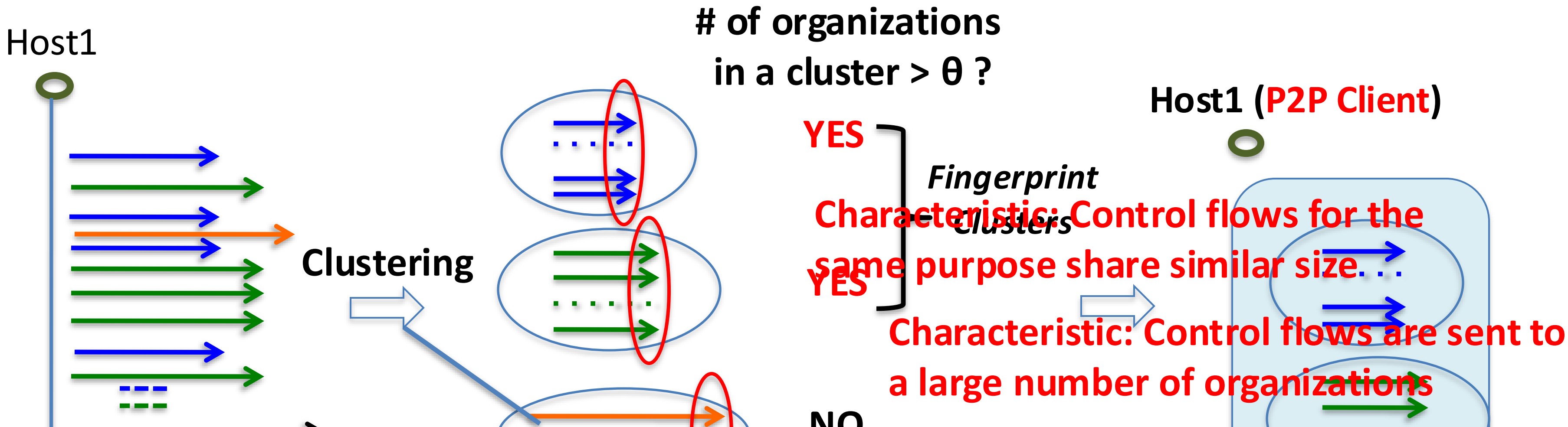
DNS-Based Traffic Filtering

Characteristic: Control flows are DNS-free

- Discard flows whose destination IPs are resolved from DNS queries



Clustering-Based Flow Analysis



Scalable Clustering Analysis

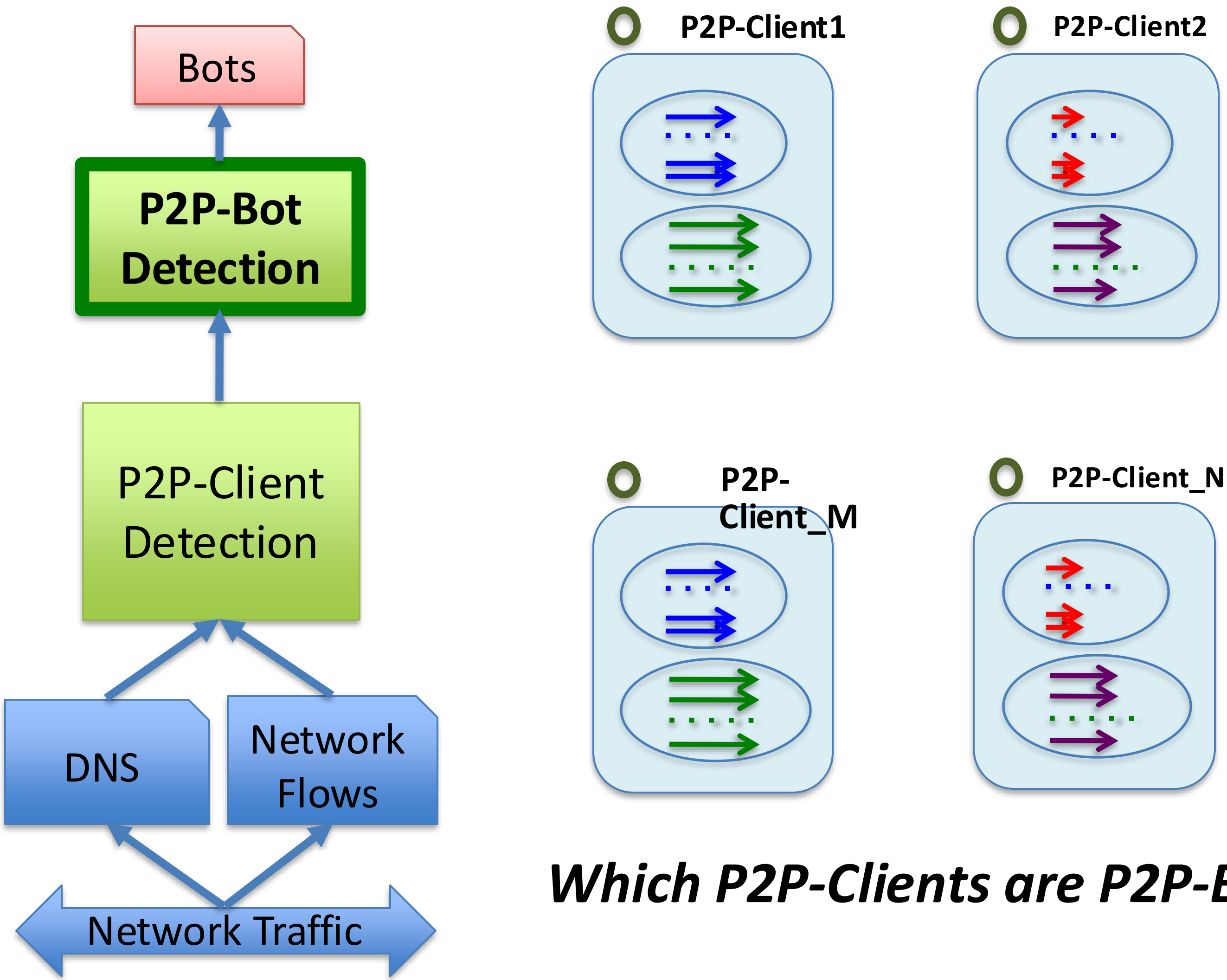
1. Two-level clustering scheme: Streaming Clustering (Birch) + Hierarchical Clustering
2. Distribute the clustering analysis workload

1. Aggregate flows with similar size into one cluster

$$dist(f_i, f_j) = \sqrt{(pkt_{s,i} - pkt_{s,j})^2 + (pkt_{r,i} - pkt_{r,j})^2 + (byte_{s,i} - byte_{s,j})^2 + (byte_{r,i} - byte_{r,j})^2}$$

1. Define a cluster as a ***fingerprint cluster*** if # of unique organizations >= θ (=50)
2. Claim a host as a P2P client if it has at least one ***fingerprint cluster***

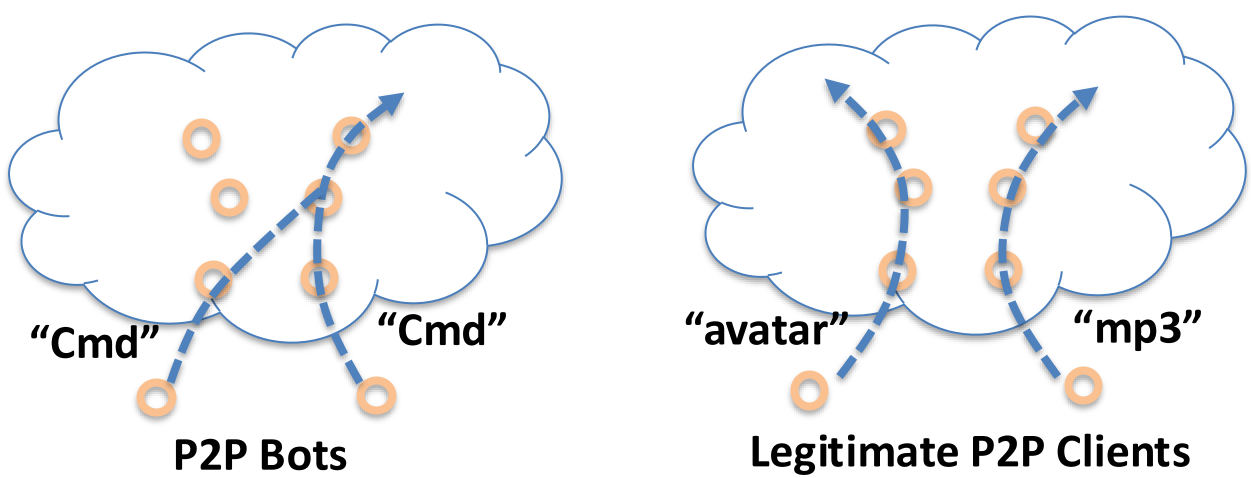
P2P-Botnet Detection



Which P2P-Clients are P2P-Bots?

P2P Bots v.s. Legitimate P2P Clients

	P2P Bots	Legitimate P2P Clients
Temporal Feature	Persistent	Transient
Spatial Feature	Same content (commands) => Large overlap of peers	Different contents => Small overlap of peers

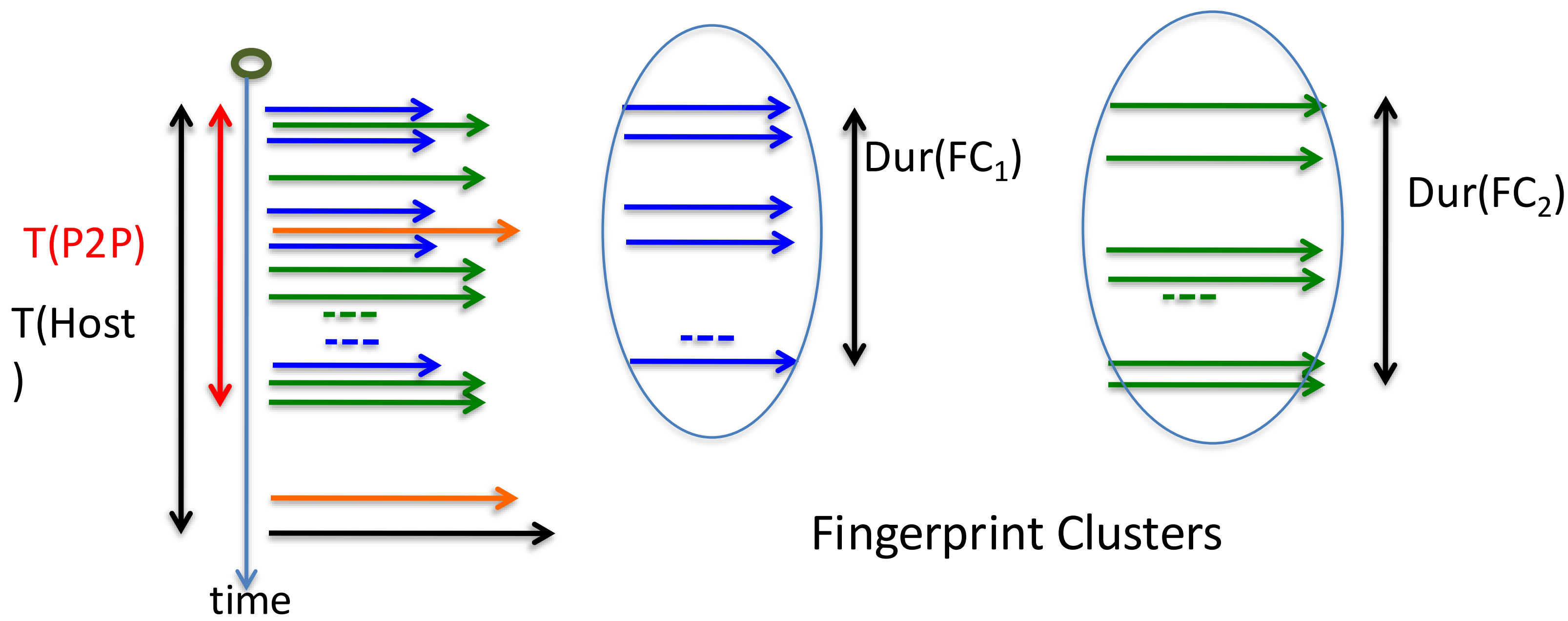


Temporal Feature Based Detection

- A P2P client is persistent if

$$T(P2P) / T(Host) > 0.5$$

Characteristic: Control flows are automatically generated



$$T(P2P) \cong MAX(Dur(FC_1), Dur(FC_2)......Dur(FC_n))$$

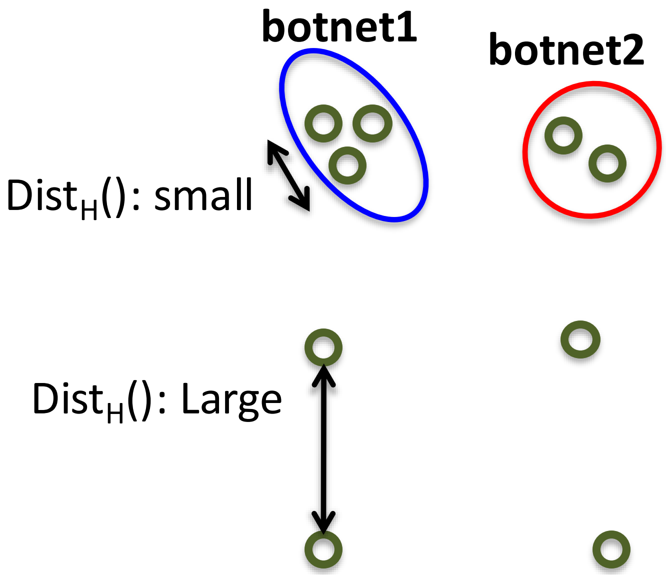
Spatial Feature Based Detection

$$Dist_H(H_m, H_k) \propto (1 - \frac{|IPSet_m \cap IPSet_k|}{|IPSet_m \cup IPSet_k|})$$

H_m	H_k	$Dist_H(H_m, H_k)$
Bot1	***Bot2***	***Small***
Bot1	***Bot2 + Emule1***	***Small***
Emule1	Emule2	Large
Bot1	Emule1	Large

Spatial Feature Based Detection

- A pair of persistent P2P clients belong to a **botnet** if they have a small $Dist_H()$, using hierarchical clustering algorithm

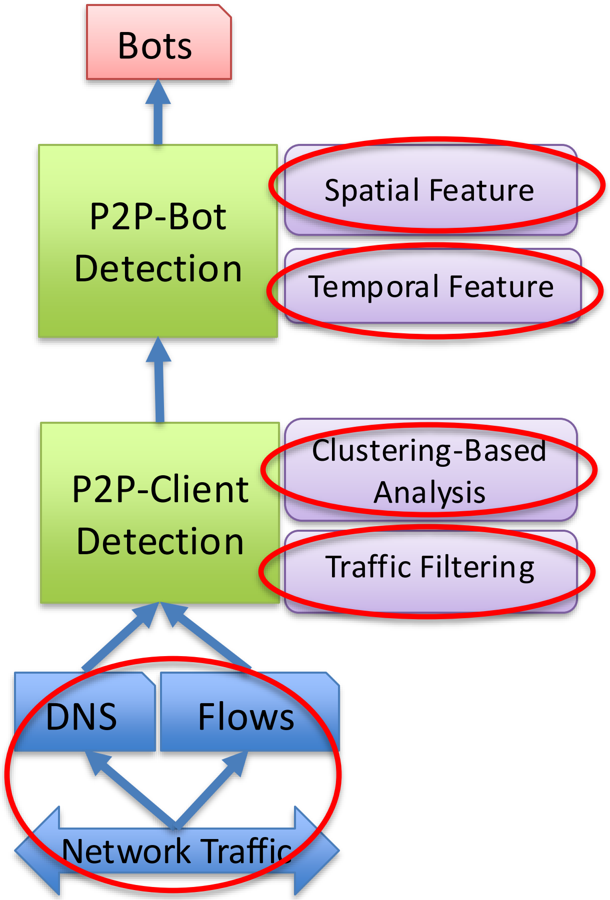


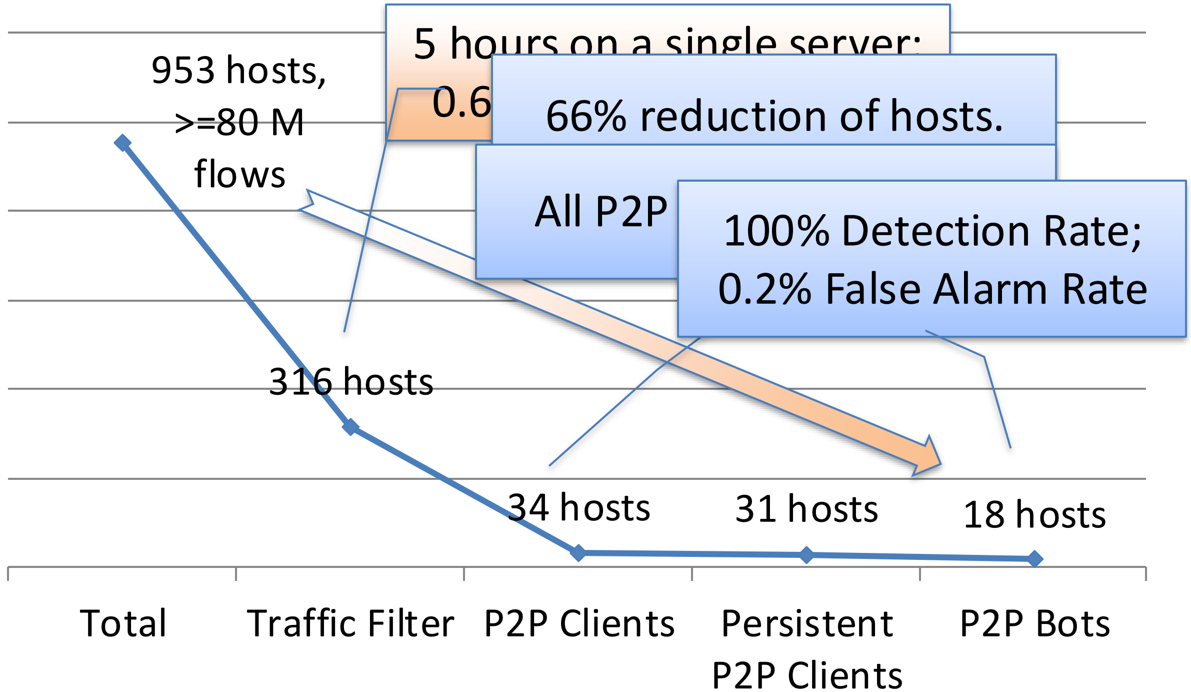
Evaluation

- Data

Data	# of hosts and clients	Summary
College networks	1K hosts 3 Bittorrent clients 5 Skype clients	**24 hours** **1K hosts totally** **18 legitimate P2P apps** ***16 Real P2P bots*** **80 Million flows**
Legitimate P2P applications	2 Bittorrent clients 2 Emule clients 2 Ares clients 2 Skype clients 2 Limewire clients	
P2P botnets	13 Storm bots 3 Waledac bots	

Evaluation


Bots
P2P-Bot Detection
Spatial Feature
Temporal Feature
P2P-Client Detection
Clustering-Based Analysis
Traffic Filtering
DNS
Flows
Network Traffic


5 hours on a single server;
0.6
66% reduction of hosts.
All P2P
100% Detection Rate;
0.2% False Alarm Rate
953 hosts, >=80 M flows
316 hosts
34 hosts
31 hosts
18 hosts
Total
Traffic Filter
P2P Clients
Persistent P2P Clients
P2P Bots

Conclusion of P2P Botnet Detection

- A novel method to ***detect*** and ***profile*** P2P applications
  - A novel DNS-based traffic filter
  - A Clustering-based flow analysis approach
- A novel P2P botnet detection method
  - Detect bots even if their underlying operating systems are running legitimate P2P applications
  - Detect bots even if their malicious activities are not observed
  - 100% detection rate
  - 0.2% false alarm rate
- A scalable system
  - Process 80 million flows in 0.69 hours

Limitations and Future Work

- Evasion
  - Randomize the communication
  - Randomize the command
- Backbone network
  - Network flows are heavily sampled
- Online Detection