Advanced Network Security 2019

Botnets and botnet detection

Harald Vranken

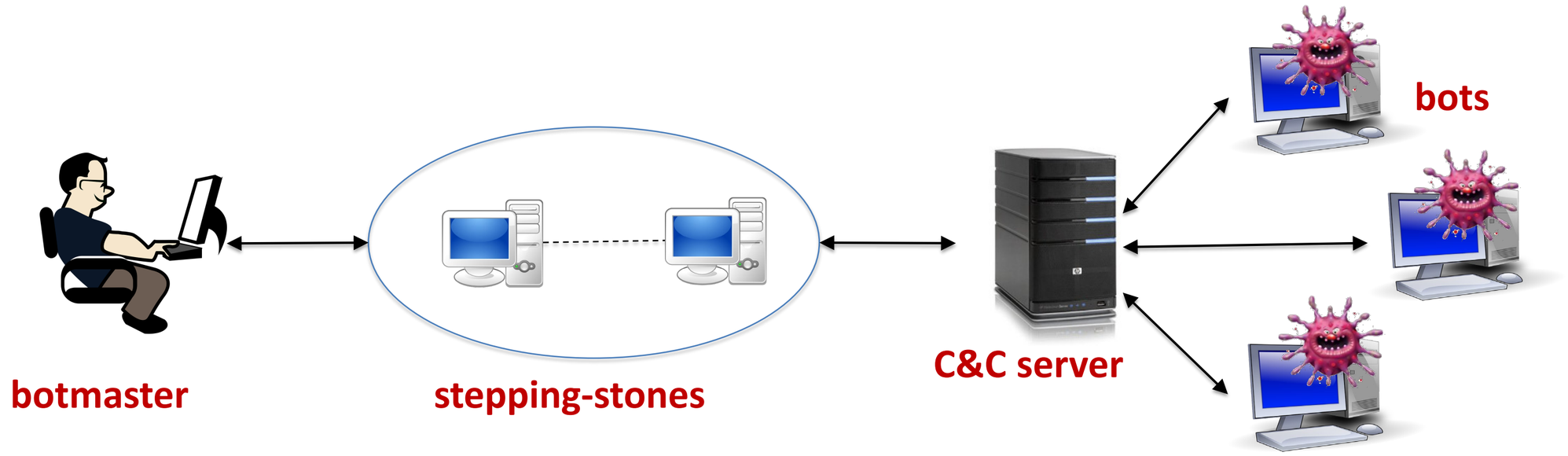
harald.vranken@ou.nl

Agenda

- Botnets
- Botnet detection
- Defense against botnets
- Our research on botnet detection

Botnet

- A botnet is a network of compromised machines (*bots*) that are infected by malware (*bot binary*)
- Bots receive and respond to commands from a server (*Command & Control server*)
- C&C server acts as a rendezvous mechanism for commands from a human controller (*botmaster*)
- Botmaster can employ a number of proxy machines (*stepping-stones*) to evade detection
- Ultimate goal of a botnet is to carry out malicious activities or attacks on behalf of botmaster



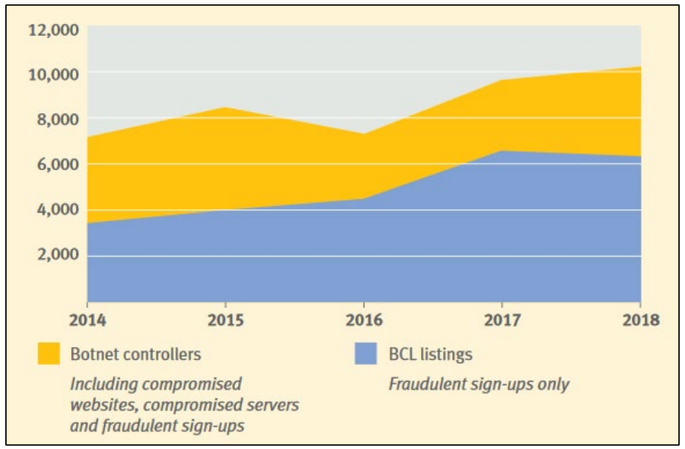
Botnet threat

- Botnets form a major threat for users of the Internet
- How big is the problem?
  - "*One quarter of all computers* connected to the Internet are part of a botnet"
    (Vint Cerf, World Economic Forum, 2006)
  - "Computers of at least *5-10% of all Dutch broadband subscribers* in 2009 were part of a botnet"
    [TU Delft, 2011]
  - [NCSC Cyber Security Assessment Netherlands]
    - 2013: Botnets affect citizens, businesses, and government
    - 2014: Botnets are hired easily via *black markets*
    - 2015: *6% of most visited websites* in world spread malware/spam or are part of a botnet
    - 2017: *Botnets of (consumer) electronics* (Mirai) caused *large-scale DDoS attacks*
    - 2018: *Cryptomining malware* (Smominru botnet)

Botnet threat

[Spamhaus Botnet Threat Report 2019]

- Rise in CoinMiners and CoinStealers
- Decentralized TLDs (such as .bit)
- Increased use of anonymization services (such as Tor)
- Inadequate verification processes and slow reaction of (cloud) hosting providers



Geolocation of botnet C&Cs in 2018:
(1) US, (2) Russia, (3) Netherlands)

Malware families associated with botnets

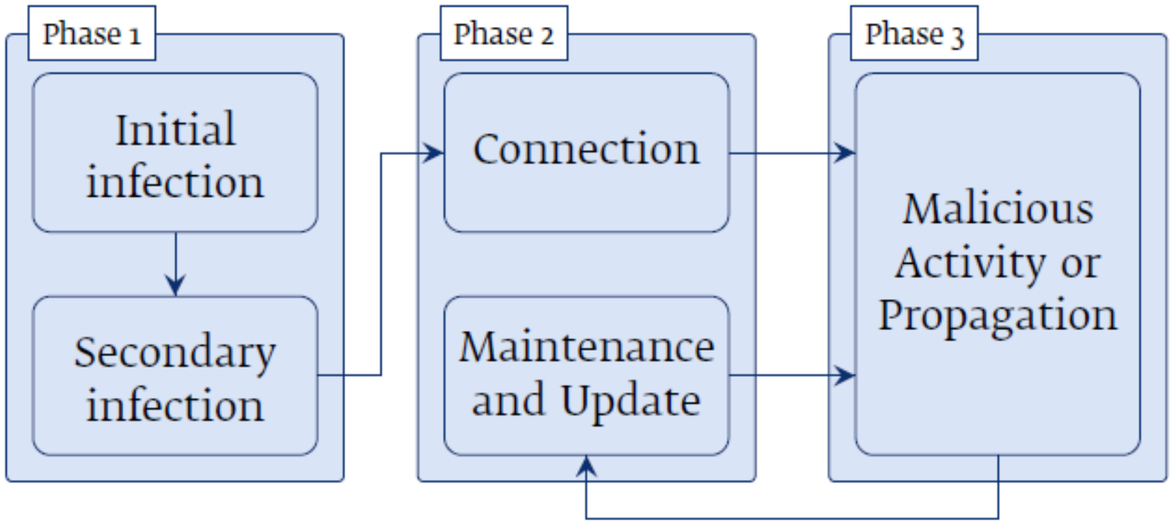
Rank	C&Cs	Malware	Note
1	2,347	Lokibot	Credential Stealer
2	1,300	JBifrost	Java based Remote Access Tool (RAT)
3	955	Pony	Dropper/Credential Stealer
4	915	AZORult	Credential Stealer
5	686	Heodo/Emotet	Dropper/Backdoor
6	413	Gozi ISFB	e-banking Trojan
7	322	NanoCore	Remote Access Tool (RAT)
8	269	Smoke Loader	Dropper/Backdoor
9	241	TrickBot	e-banking Trojan
10	203	RemcosRAT	Remote Access Tool (RAT)
11	157	RedAlert	Android Trojan
12	122	NetWire	Remote Access Tool (RAT)
13	117	AgentTesla	KeyLogger/Remote Access Tool (RAT)
14	107	Chthonic	e-banking Trojan
15	106	PandaZeuS	e-banking Trojan
16	98	ImminentRat	Remote Access Tool (RAT)
17	96	Neurevt	e-banking Trojan
18	82	ISRStealer	Credential Stealer
19	70	ArkeiStealer	Credential Stealer
20	51	NjRAT	Remote Access Tool (RAT)
–	89	CoinMiners malware	Various crypto currency miners
–	46	IoT malware	Various IoT malware
–	456	Generic	*
–	1,015	Others	Other malware families

Fight against botnets

- *Legal countermeasures* have succeeded in disbanding a number of botnets
  - some notable takedowns
    - *Avalanche* (December 2016): cooperation between 30 countries;
      5 people arrested, 221 servers taken offline, 37 servers seized,
      more than 800,000 domains seized/blocked/disrupted
    - *GameOverZeus* (June 2014), *Zeus* (March 2012): FBI, Microsoft and industry
  - taking down C&C servers yields promising results in short-term,
    but botnets often find a way to resurrect and resume their malicious activities
- Pressing need to address botnets *closer to the source*
  - more emphasis on preventive detection and defense at home computers and routers
  - ISPs can play a significant role in this context

Bot(net) life cycle

- *Infection*: bot receives and executes bot binary
- *Rallying*: bot contacts C&C server and announce its presence
  - establishes *C&C channel* through which the bot receives updates and commands
- *Passive*: bot waits for commands
- *Active*: serve the actual *purpose* of the botnet
  - optionally spread infection to other hosts using *propagation* mechanisms

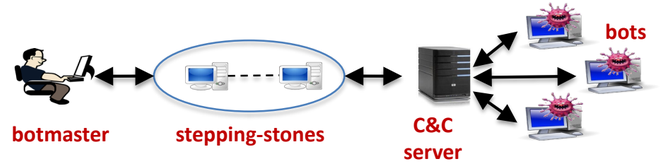


Propagation of botnet binary

- *Active:* botnet can locate and infect other hosts without any (human) user intervention
  - *scan* other hosts in network to look for vulnerabilities that can be exploited
  - if found, gain administrative privilege on victim machine and install bot binary
  - some botnets borrow their propagation tactics from worms

- *Passive*: requires some level of user intervention, such as:
  - *drive-by download*: automated installation of bot binary when visiting a website (eg through JavaScript)
  - *infected media*: sharing eg USB sticks (example: Stuxnet)
  - *social engineering*: entice user to willingly download the bot binary (example: Storm sent spam emails with malicious links to install bot binary)

Rallying (bots establishing contact with C&C server)



- *IP address:* IP address of C&C server is provided along with bot binary
  - *static hardcoded:* IP address of C&C server is hardcoded into bot binary
    - eliminates use of DNS (stealthy), but reverse engineering of bot binary may reveal C&C server (leading to hijacking of C&C server, or blacklisting of C&C server IP address)
  - *static seeding*: (p2p) bot is provided with an initial list of peers
    - list may not be in bot binary, eg hidden in Windows registry
- *Domain name:* domain name(s) of potential C&C server(s) or stepping-stone(s) is provided
  - *DNS*: look up IP address (possibly through DDNS and rogue DNS server)
  - *hardcoded:* domain name of C&C servers is hardcoded into the bot binary
    - but may be resolved to different IP addresses
  - *generated:* dynamically generate domain names by DGA (Domain Generation Algorithm)
    - known to bot and botmaster
    - *bot-herding*: taking down a domain is complicated; by the time a domain is taken down, the C&C server has typically already moved to a new one

C&C communication

- *Existing protocol:* tried and tested, mix with regular traffic making detection difficult
  - Initially: *IRC* (Internet Relay Chat)
    - simple text-based command syntax, almost real-time communication
    - not common anymore, easily distinguishable from other traffic
  - Next step: *HTTP*
    - bots contact C&C server periodically to fetch commands
    - most common protocol used on the Internet, blocking HTTP traffic is not a viable option
    - in larger botnets, some strategy must be adopted to keep the C&C server from being overwhelmed if all the bots contact it simultaneously
    - taking down a (C&C) web server is not easy (requires third-parties and legal intervention)
  - Recent: *peer-to-peer* (BitTorrent, Gnutella, …, even Skype and VOIP)
    - commands can be dispersed using any node in p2p network
    - p2p traffic classification is more difficult task

C&C communication (2)

- *Neoteric protocol (usage)*
  - proprietary application-level protocols
  - use existing applications for C&C in a way in which they were not intended to be used
  - examples
    - DNS channels
    - fake profiles on social networks (Facebook, Twitter, …) to publish commands as ‘feed’ or ‘status’, but can be taken down easily

Botnet purpose

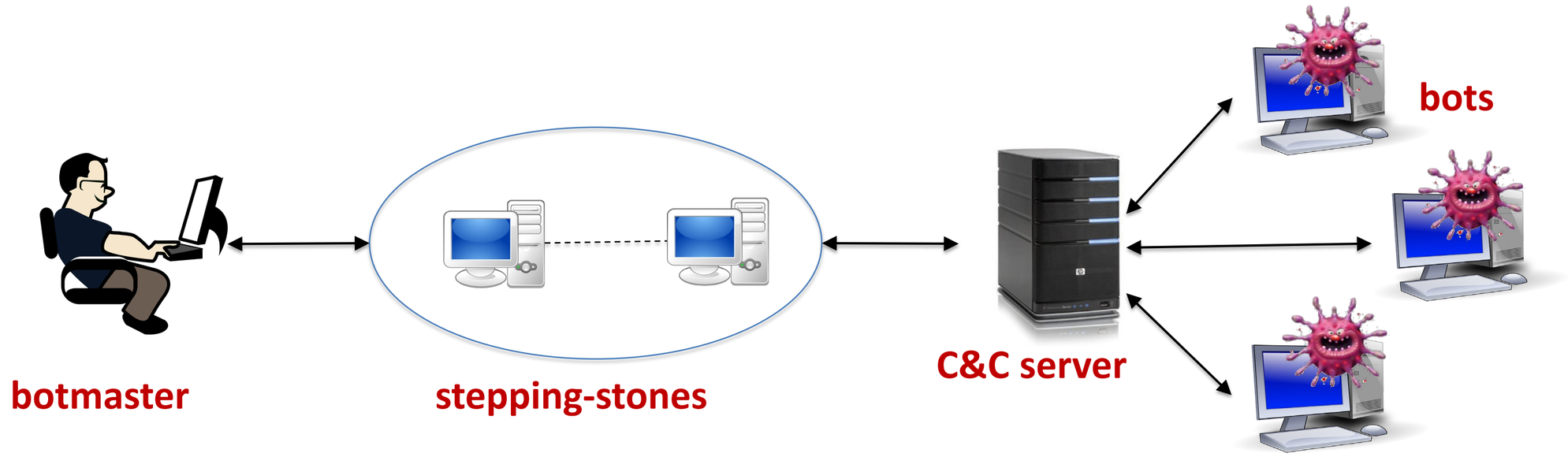
- Use combined power of (thousands of) bots to carry out malicious activities
- *Swiss army knife* of cyber criminals: versatile
- *Information gathering:* for financial gains (credit card numbers, bank account numbers) or reconnaissance purposes (cyber espionage)
- *Distributed computing:* use storage and processing power of bots to host and share files, perform distributed password cracking, mine cryptocurrencies
- *Cyberfraud:* web-phishing on fake web server, rig results of online games and polls, clickfraud by directing bots to click on pay-per-click advertisements, manipulate search engine
- *Spreading malware:* launching other malware
- *Unsolicited marketing:* unsolicited advertisements in the form of spam emails, pop up ads
- *Cyberwarfare:* DDoS attacks against Estonian websites in 2007, Stuxnet in 2010
- *Network service disruption:* DDoS attacks; enterprises are willing to pay extortion money to botmasters rather than losing sales and credibility

Evasion tactics of C&C server

- Botnets want to stay *hidden*, since detection may
  - help defenders understand how botnet operates and exploit this to damage the botnet
  - enumerate bots which can be subsequently disinfected
  - expose C&C server(s), possibly take-over of botnet
  - may reveal botmaster

Botnet tactics to evade detection

- Botnets employ various mechanisms to *evade detection* of
  - bot or bot binary
  - C&C communication
  - C&C server(s)
  - botmaster



Evasion tactics of bots

- *Binary obfuscation*
  - polymorphism: ability of bot binary to exist in several forms,
    eg by encryption or compression
  - metamorphism: bot binary rewritten into different, but semantically equivalent code
- *Anti-analysis*
  - bot binary can check the environment in which it is being executed
  - detect if it is being analyzed when running in virtual machine, sandbox or honeypot,
    and if so, then bot binary refuses to run or modifies its functionality (‘sjoemelsoftware’)
  - not widely applied: check may be incorrect, virtual machines are popular
- *Security suppression*: bot binary may disable security software on victim machine
- *Rootkit technology*: bot binary may subvert OS to remain persistent and undetectable

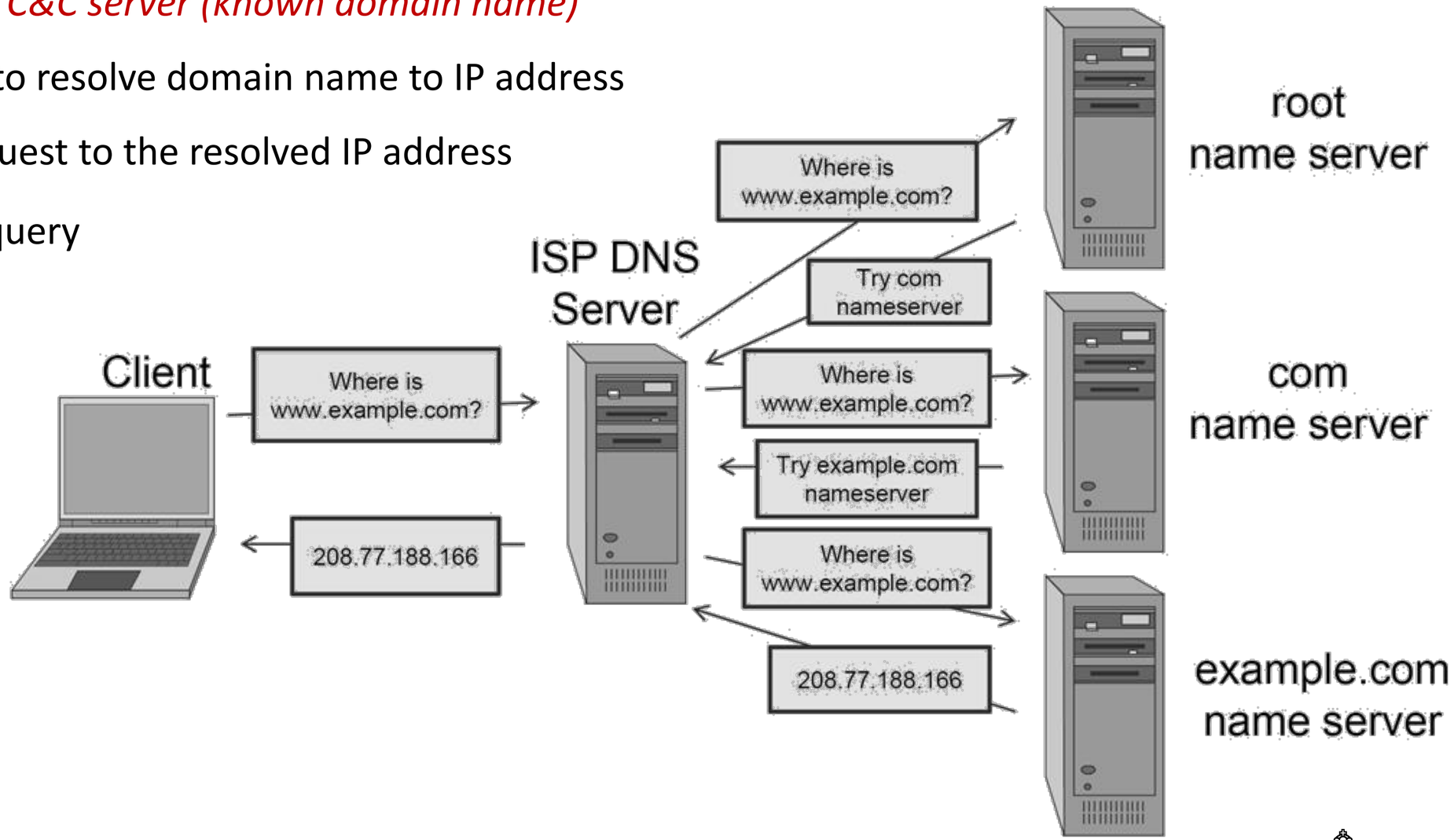
Evasion tactics of C&C server

- *IP flux (fast flux)*
  - frequently change IP address associated with domain name of C&C server, while keeping the same domain name
  - helps in evading blacklisting and blocking of IP addresses
  - real-time update of DNS facilitated by Dynamic DNS (DDNS) services

- *DNS* maps domain names to *static IP addresses*
- *DDNS* maps domain names to *dynamic IP addresses*
  - IP address handed out by ISP to customers may change from time to time
  - problem if customer wants to provide a web service
  - DDNS services provide automatic reconfiguration of DNS
  - hence, web service stays accessible through DNS

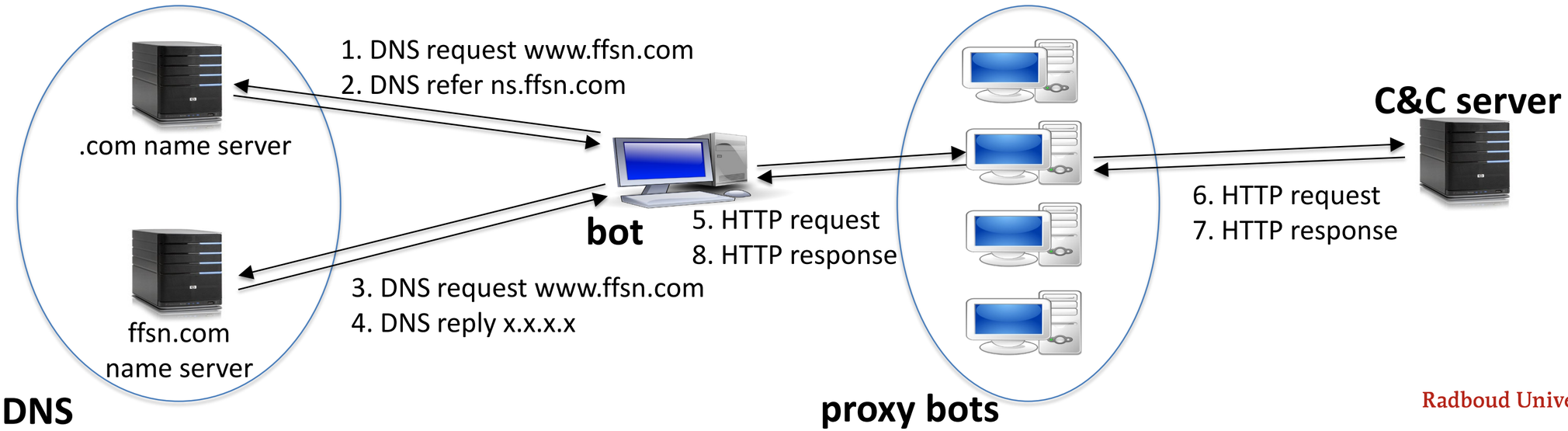
Evasion tactics of C&C server

- *How bot contacts C&C server (known domain name)*
  - first, use DNS to resolve domain name to IP address
  - next, send request to the resolved IP address
- *Resolving* a DNS query



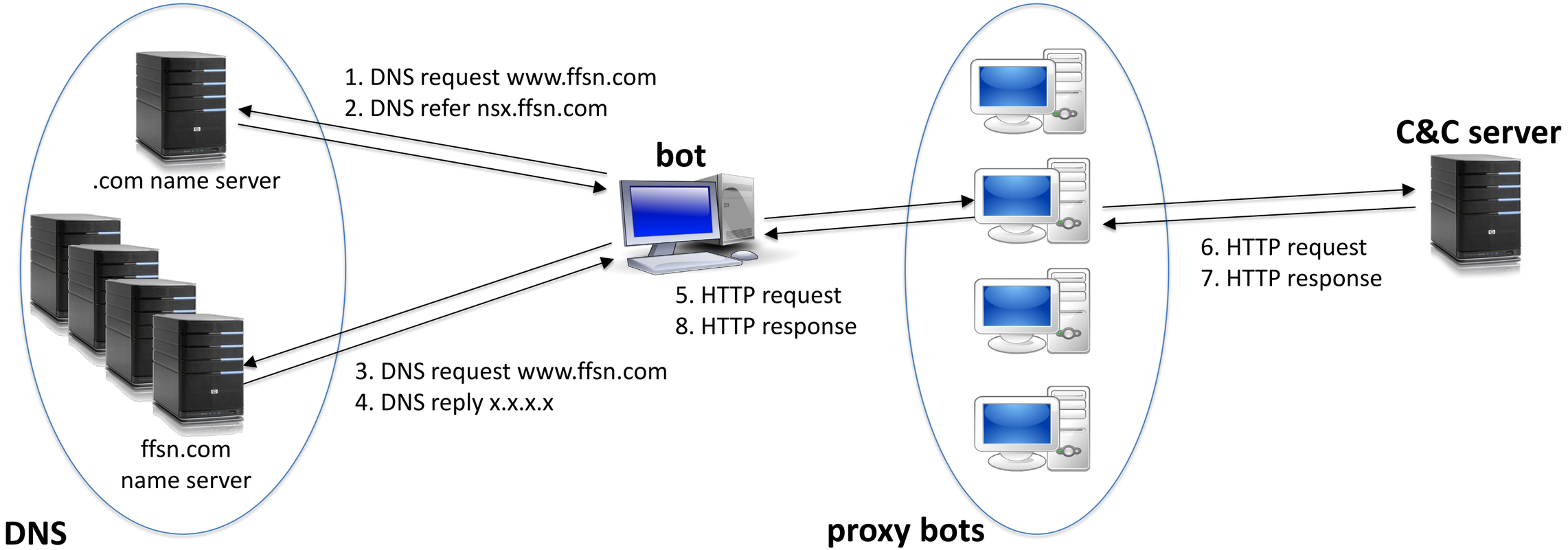
Evasion tactics of C&C server

- *Single flux*
  - bots do not communicate directly with C&C server, but through intermediate layer of *proxy bots* (machines compromised by botnet, configured as proxies, that relay communication)
  - DNS provides IP address of proxy bot with very short TTL (few seconds)
  - botnet uses proxy bots in round robin fashion, in short intervals of time
  - botnets also use domain names only for a short period of time, associating a new domain name with same set of proxy bots
- *Example*: bot knows domain name of C&C server (www.ffsn.com)



Evasion tactics of C&C server

- *Double flux*
  - extend concept of flux to name server responsible for resolving C&C domain name
  - IP address of name server changes frequently



Evasion tactics of C&C server

- *Domain Flux*
  - associates multiple domain names with same IP address
  - helps evade URL-based detection
- Achieved by
  - *domain wildcarding* (DNS service)
    - domain name prefixed with random string (eg *.cs.ru.nl), associated with same IP address
  - *DGA* (domain name generation algorithm)

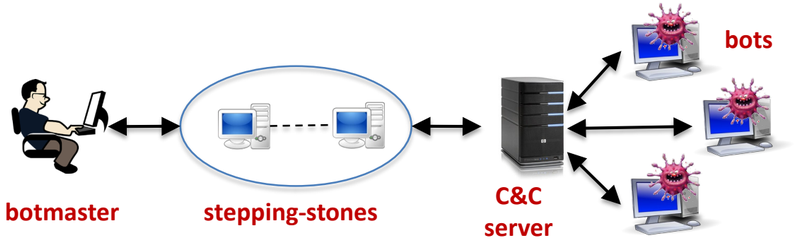
Evasion tactics of C&C server

- *DGA* (domain name generation algorithm)
  - bot applies DGA to periodically generate a (large) number of domain names
  - not all generated domain names active at a given time (only one registered by botmaster)
  - bot uses DNS, trying to resolve all these domain names one by one
  - successfully resolved domain name is C&C server
  - other domain names result in Non-Existent Domain (NXDomain) responses
- Example: *Conficker* generates 50,000 domain names per day,
  of which 500 randomly selected are tried
- *Re-engineering DGA* by analysis of botnet binary
  - predict what domain names a bot will try
  - but unfeasible to register all those domains by law enforcement or check which ones are malicious

Evasion tactics of C&C server

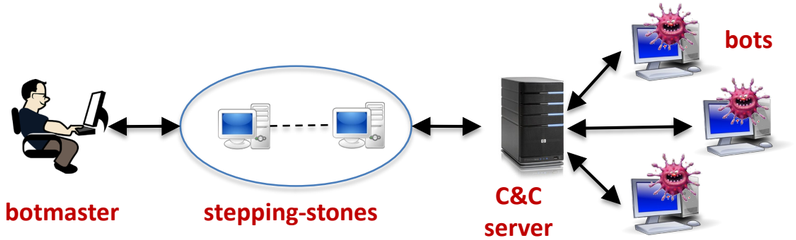
- *Rogue DNS server*
  - used by bots to resolve IP address of C&C server
  - can also redirect a website's traffic to malicious website (pharming), eg for stealing sensitive user information (phishing)
- *Anonymization*
  - message cannot be traced back to sender
  - eg bots using Tor to contact C&C server
  - C&C servers can be hosted as Tor hidden services

Evasion tactics of C&C communication



- *Encryption*
  - causes content-based analysis to fail
  - forces researchers to rely on other traffic characteristics (eg packet arrival times, packet length)
- *Protocol manipulation*
  - botnets use protocol tunneling to disguise C&C communication (eg HTTP tunnels, recently IPv6 tunneling)
- *Traffic manipulation*
  - botnets might purposely create low volume C&C traffic, spread over relatively large periods of time, to defeat statistical and volume-based detection
- *Novel communication techniques*
  - botnets use novel communication techniques for C&C which cannot be trivially detected (eg social networks, metadata in image files, least significant bit encoding in image files)

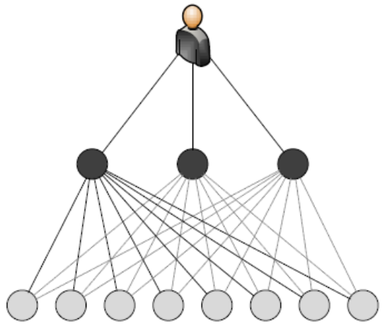
Evasion tactics of botmaster

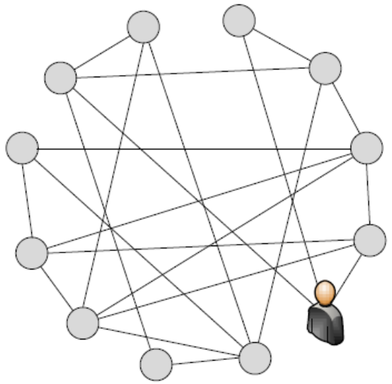


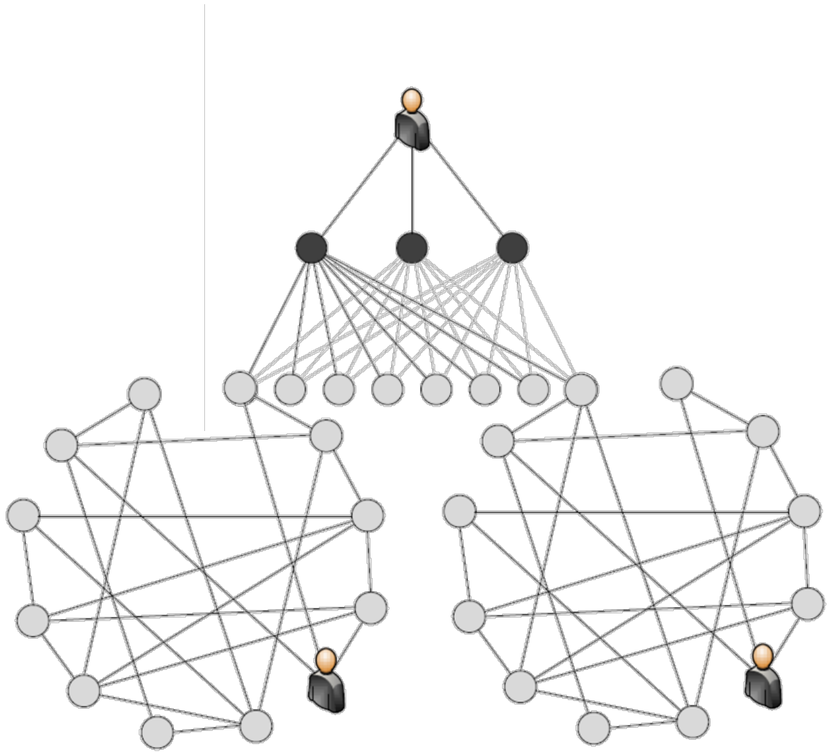
- Botmaster is the most protected component of a botnet
- *Stepping-stones*
  - botmasters generally hide their true identity by setting up a number of intermediate hosts between the C&C server and themselves
  - prefer to set up stepping-stones in countries with lax cybercrime legislature
  - cripples trace-back mechanisms because network redirection services operate at the application level and discard all lower layer information
  - botmaster can use an anonymization network as a stepping-stone

Topology

- Botnet topology
  - Centralized
  - Decentralized
  - Hybrid

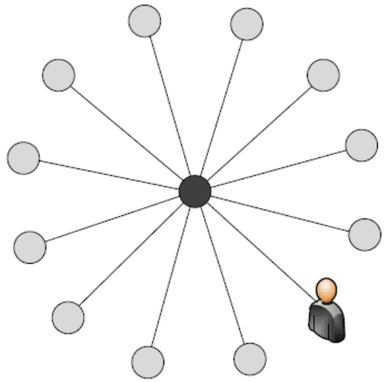


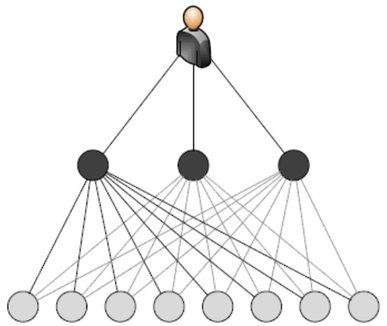




Topology

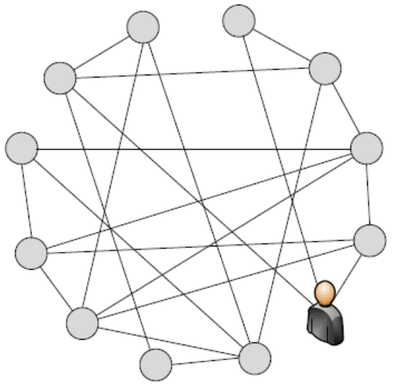
- *Centralized*
  - all bots report to and receive commands from a single C&C server (by IRC or HTTP)
  - easy implementation with minimum C&C overhead
  - *Star*
    - high-speed communication between bots and botmaster
    - however, C&C server is single point-of-failure
  - *Hierarchical*
    - one or more layers of proxies between botmaster and bots
    - proxies themselves are compromised machines
    - resilient to take down of some proxies
    - allows portions of botnet to be separately rented out to third parties
    - introduces some latency





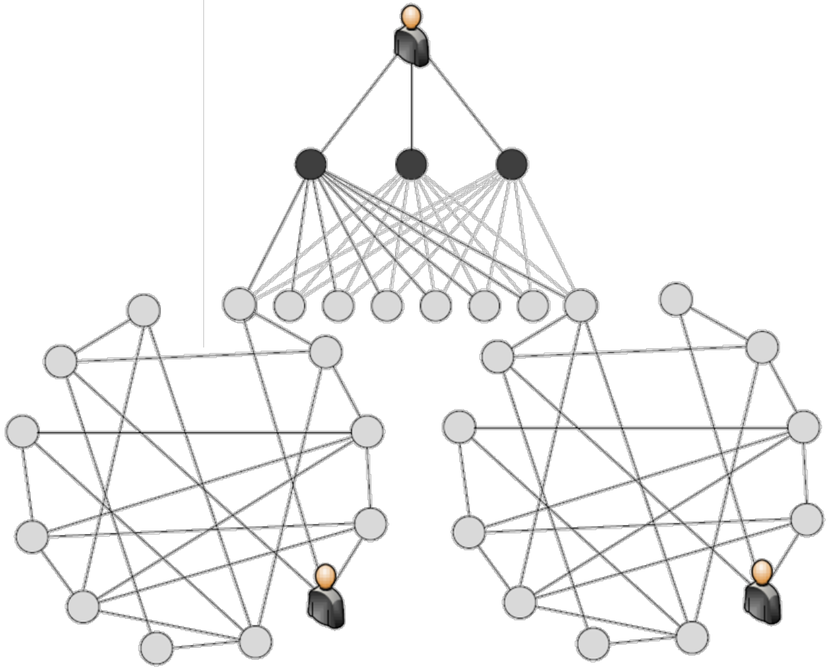
Topology

- *Decentralized*
  - no single entity is responsible for providing C&C
  - *Distributed*
    - bot management is distributed among multiple C&C servers, each controlling subsets of bots
    - servers are able to communicate directly with each other
    - typically, servers are spread over different geographical locations
    - allows fast communication, load distribution, availability, and resilience
  - *Random*
    - no clear master-slave relationship; any bot can issue commands to other bots
    - in p2p networks, communication between bots and botmaster forms unpredictable routes
    - eliminating a single bot has no effect as alternate routes are available, capture of a single bot may reveal several other bots in peer-list
    - C&C communication will experience unpredictable delays



Topology

- *Hybrid*
  - combination of centralized and decentralized topologies
  - eg (future) botnet using centralized structure between C&C server and front-end proxy bots, but p2p as C&C for the bots under control of individual proxy bots

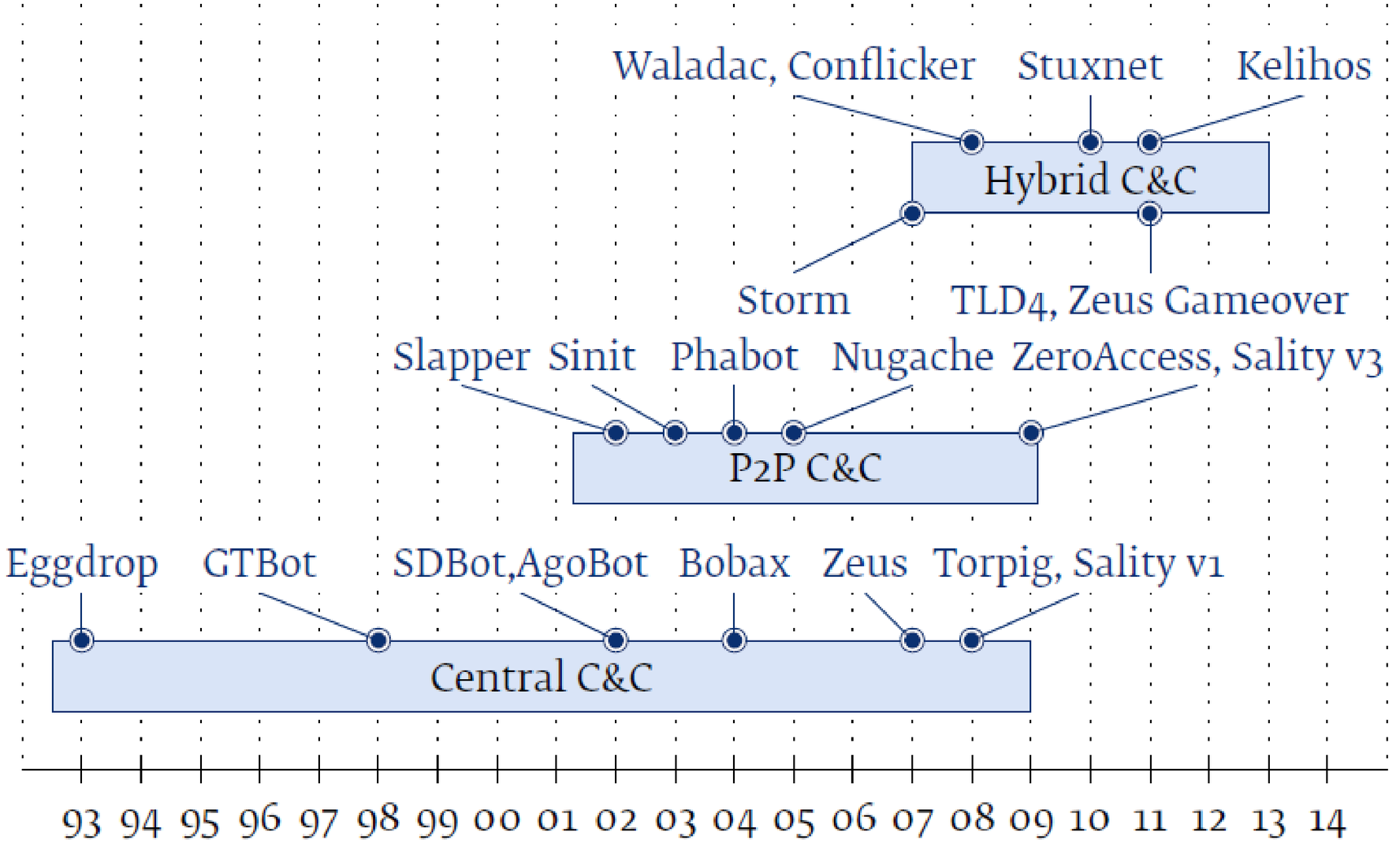


Topology

- C&C-topology characteristics

Topology	Design Complexity	Detectability	Message Latency	Survivability
Central	Low	Medium	Low	Low
Decentral/P2P	Medium	Low	Medium	Medium

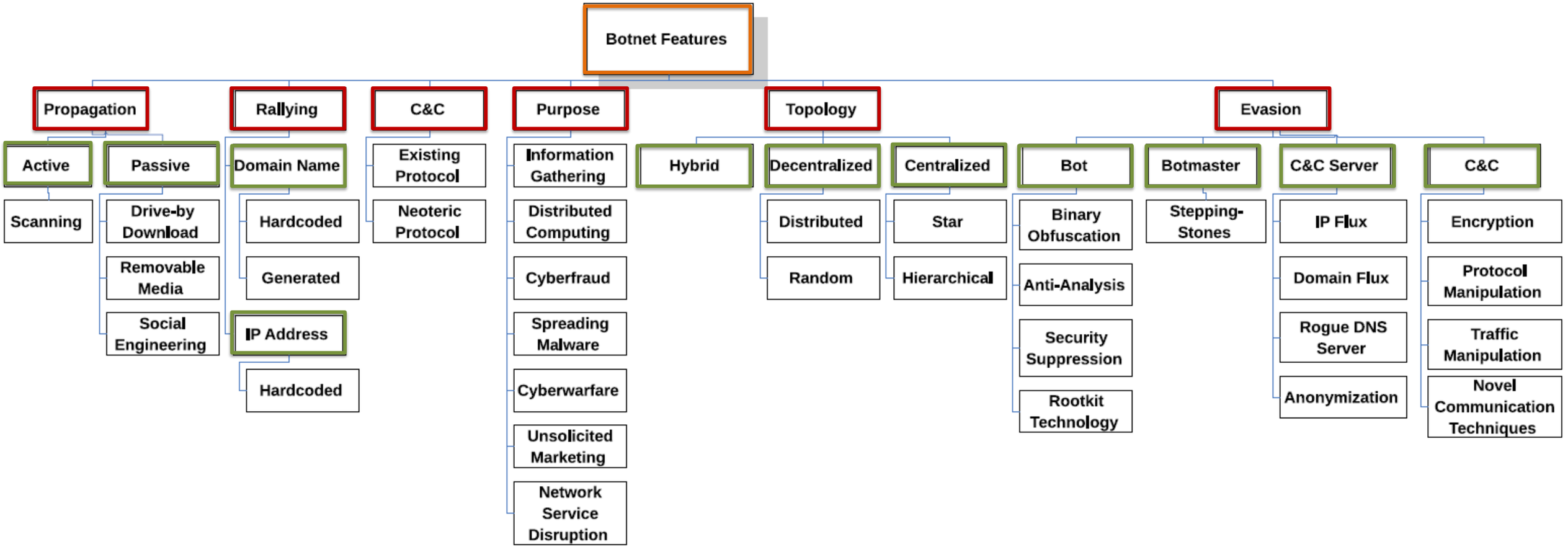
Topology



[Rossow, C., Andriesse, D., Werner, T., Stone-Gross, B., Plohmann, D., Dietrich, C. J. and Bos, H. *Modeling and Evaluating the Resilience of Peer-to-Peer Botnets.* IEEE Symposium on Security and Privacy, 2013, 97–111]

[Rodriguez-Gomez, R. A., Macia-Fernandez, G. and Garcia-Teodoro, P. *Survey and taxonomy of botnet research through life-cycle.* ACM Computing Surveys, 45(4): 1–33, 2013]

Summary


Botnet Features
Propagation
Active
Scanning
Passive
Drive-by Download
Removable Media
Social Engineering
Rallying
Domain Name
Hardcoded
Generated
IP Address
Hardcoded
C&C
Existing Protocol
Neoteric Protocol
Purpose
Information Gathering
Distributed Computing
Cyberfraud
Spreading Malware
Cyberwarfare
Unsolicited Marketing
Network Service Disruption
Topology
Hybrid
Decentralized
Distributed
Random
Centralized
Star
Hierarchical
Evasion
Bot
Binary Obfuscation
Anti-Analysis
Security Suppression
Rootkit Technology
Botmaster
Stepping-Stones
C&C Server
IP Flux
Domain Flux
Rogue DNS Server
Anonymization
C&C
Encryption
Protocol Manipulation
Traffic Manipulation
Novel Communication Techniques

Agenda

- Botnets
- *Botnet detection*
- Defense against botnets
- Our research on botnet detection

Botnet detection

- Many mechanisms for botnet detection have emerged over time
  - none unveils all botnet components at once
  - each identifies a part of the botnet

- Classification
  - bot detection
  - C&C communication detection
  - botmaster detection
  - (C&C server detection often is implicit)

Bot detection: active detection

- Detection by participating in botnet operation (impersonating as a botnet component)
- *Infiltration*
  - masquerade as an actual bot, probe C&C server or other peers
- *C&C server hijack*
  - taking control of C&C server will reveal all bots that contact it
  - by exploiting botnet rallying mechanism
  - *physical hijack*
    - law enforcement agencies physically seize the C&C server
    - or take over C&C server by mutual cooperation
  - *virtual hijack*
    - defenders redirect C&C communication to a machine under their control
    - DNS sinkholing (returns a non-routable or false IP address)
    - BGP blackholing (null-route)

Bot detection: passive detection

- Detection by silently observing and analyzing botnet activities
- *Syntactic* or *signature-based*
  - identify botnets by comparison with pre-determined patterns of botnet activity
  - strong evasion mechanisms degrade detection
  - no signatures yet for new botnets

Bot detection: passive detection (2)

- *Semantic* or *behavior-based*
  - use context of events and protocol information to detect malicious behavior
  - *correlation:* cluster hosts that perform similar activities or communication (propagation activities and attacks)
    - *horizontal correlation* by observing similarities in host behavior and/or communication
    - *vertical correlation* by observing single machine and comparing with model of bot behavior
  - *behavioral analysis* by observing deviations of machine/traffic behavior from normal pattern or its similarity with known botnet behavior
    - *host-based* methods look for signs of bot-like behavior on a host
    - *network-based* methods use information derived from network traffic and services to detect bots

C&C communication detection: active detection

- Detection by participating in botnet operation
- *Injection*
  - injecting packets (blindly) into suspicious network flows, observe response
- *Suppression*
  - suppress incoming/outgoing packets in suspicious network flows, observe response
- Difficult due to policy restraints and greater penalty in case of false positives

C&C communication detection: passive detection

- Detection by silently observing and analyzing botnet activities
- *Syntactic*
  - creating signature-based models of C&C traffic
- *Semantic*
  - use heuristics to associate certain behavior with C&C traffic
  - *statistical approaches*
    - machine learning for network traffic classification (particularly supervised learning)
  - *correlation-based methods*
    - identify similar communication patterns in network traffic (eg in router)
  - *behavior-based methods*
    - identify deviation from normal traffic or similarity with established behavioral model of C&C traffic (typically on individual hosts)

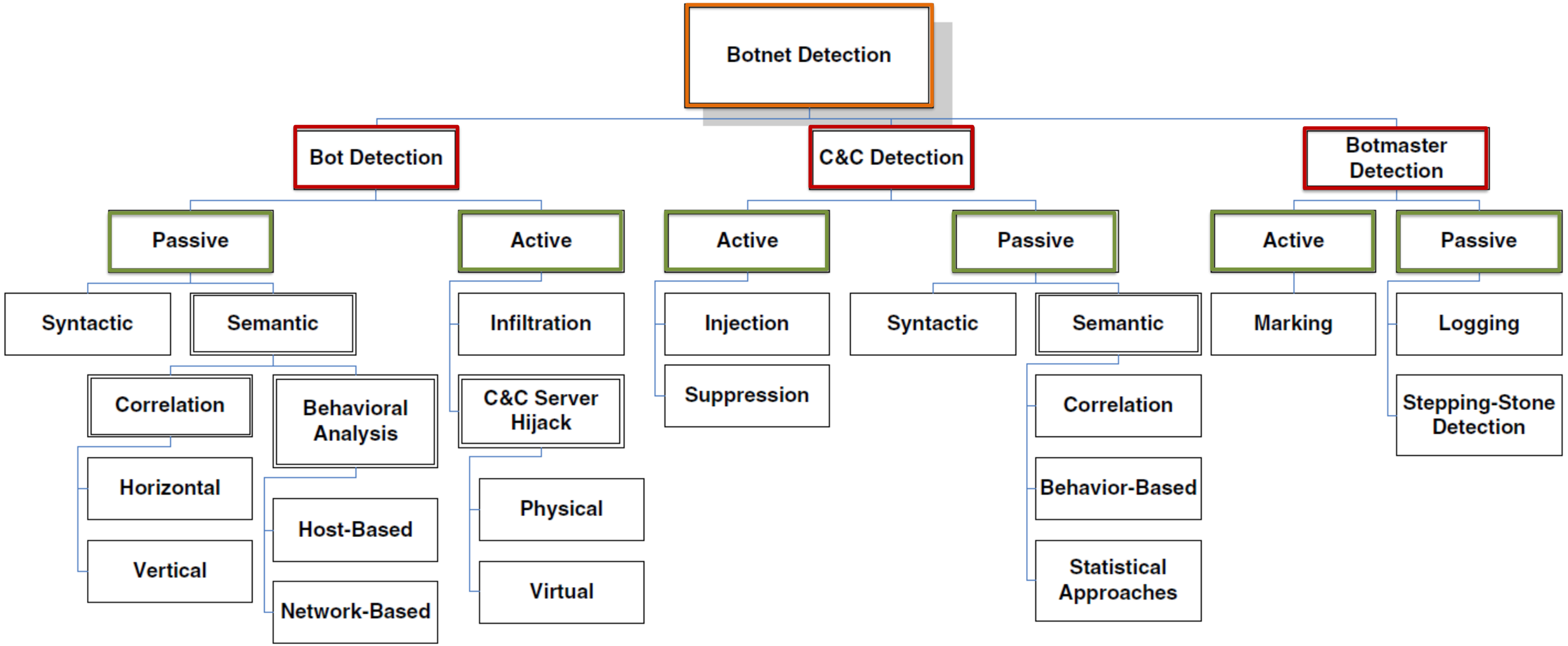
Botmaster detection: active detection

- Detection by manipulation of botnet activity
- *Packet marking*
  - information is written into packets by victim machine or intermediate routers, to help locate the attacker
  - used extensively to traceback culprits responsible for malicious activities over the Internet
  - probabilistic, ICMP traceback, deterministic

Botmaster detection: passive detection

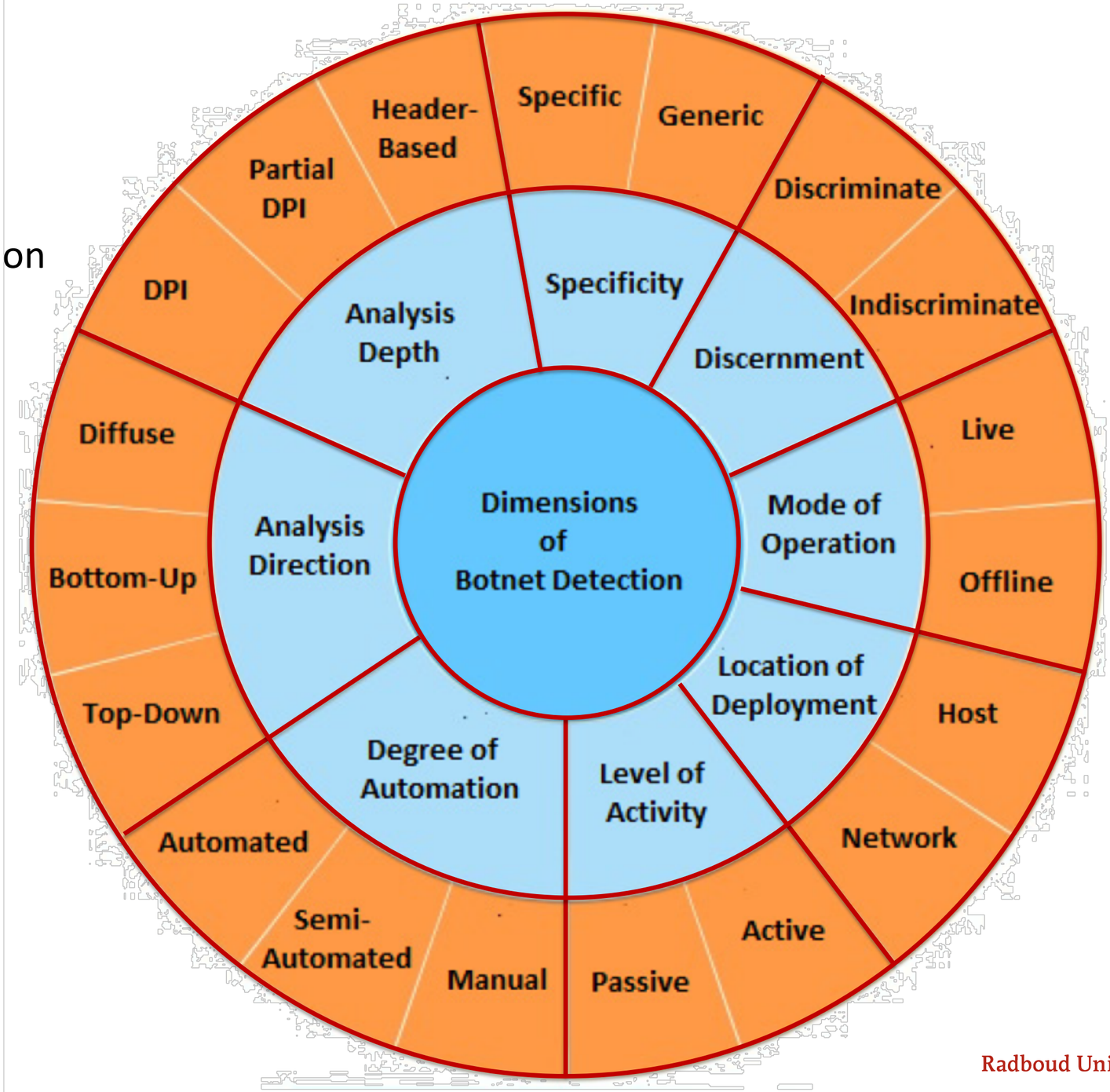
- Detection by analysis of network traffic/other data without manipulating/modifying botnet operation
- *Logging*
  - information logged by routers about packets passing through
  - heavy computational complexity and scalability issues; not used so far for botmaster detection
- *Stepping-stone detection*
  - recursively identify botmaster, based on correlation with content, host activity or packet timing
  - difficulties
    - botmaster may manipulate and forge log information
    - random delays in arrival times of packets (induced by network or by botmaster)
    - botmaster may add additional packets to confuse the detection process
    - encryption for obscuring C&C traffic

Summary


Botnet Detection
Bot Detection
C&C Detection
Botmaster Detection
Passive
Active
Active
Passive
Active
Passive
Syntactic
Semantic
Infiltration
Injection
Syntactic
Semantic
Marking
Logging
Correlation
Behavioral Analysis
C&C Server Hijack
Suppression
Correlation
Stepping-Stone Detection
Horizontal
Host-Based
Physical
Behavior-Based
Vertical
Network-Based
Virtual
Statistical Approaches

Botnet detection

- Other dimensions for classifying botnet detection



Agenda

- Botnets
- Botnet detection
- *Defense against botnets*
- Our research on botnet detection

Defense mechanisms against botnets

- *Preventive*: proactive measures to avoid botnet infection
- *Remedial*
  - *Defensive:* reactive measures to recover from botnet infection
    - bot disinfection is important from user point-of-view, but does not kill botnet
  - *Offensive*: measures to damage botnet infrastructure
    - should ideally incapacitate the whole botnet or significantly damage it

Preventive defense mechanisms: technical measures

- *Host cleanliness*
  - avoid exploitation of vulnerabilities (OS and applications) on victim machine
  - apply patch management system
  - raise user awareness (avoiding opening emails from unknown sources, avoid clicking links on untrusted websites, enable security settings of web browsers, recognize social engineering)
  - install security software (anti-virus, anti-spyware, firewall, IDS/IPS)
  - close ports favored by botnets for C&C communication (eg IRC, FTP)
- *Network cleanliness*
  - secure external connections, servers, email gateways
  - apply Network Anomaly Detection System, firewall, honeypot
  - block access to known malicious domains
  - use web proxy to scan content (inbound for malware, outbound for data leaks)

Preventive defense mechanisms: non-technical measures

- *Attacker dissuasion*
  - ‘follow the money’; target the business model employed by botnets
    (eg measures to combat click fraud, filtering spam, mitigate DDoS)
- *Legal accountability*
  - solid legal framework
  - USA, EU, China and Japan have strict penalties for cyber crimes
    (however, USA, Germany and France were top three countries hosting C&C servers)
  - strong need for collaborative legislation and cross-border cooperation to fight botnets
- *User education*
  - users are in general not concerned about computer security
  - some countries legally bind users to act responsibly
  - users should be educated

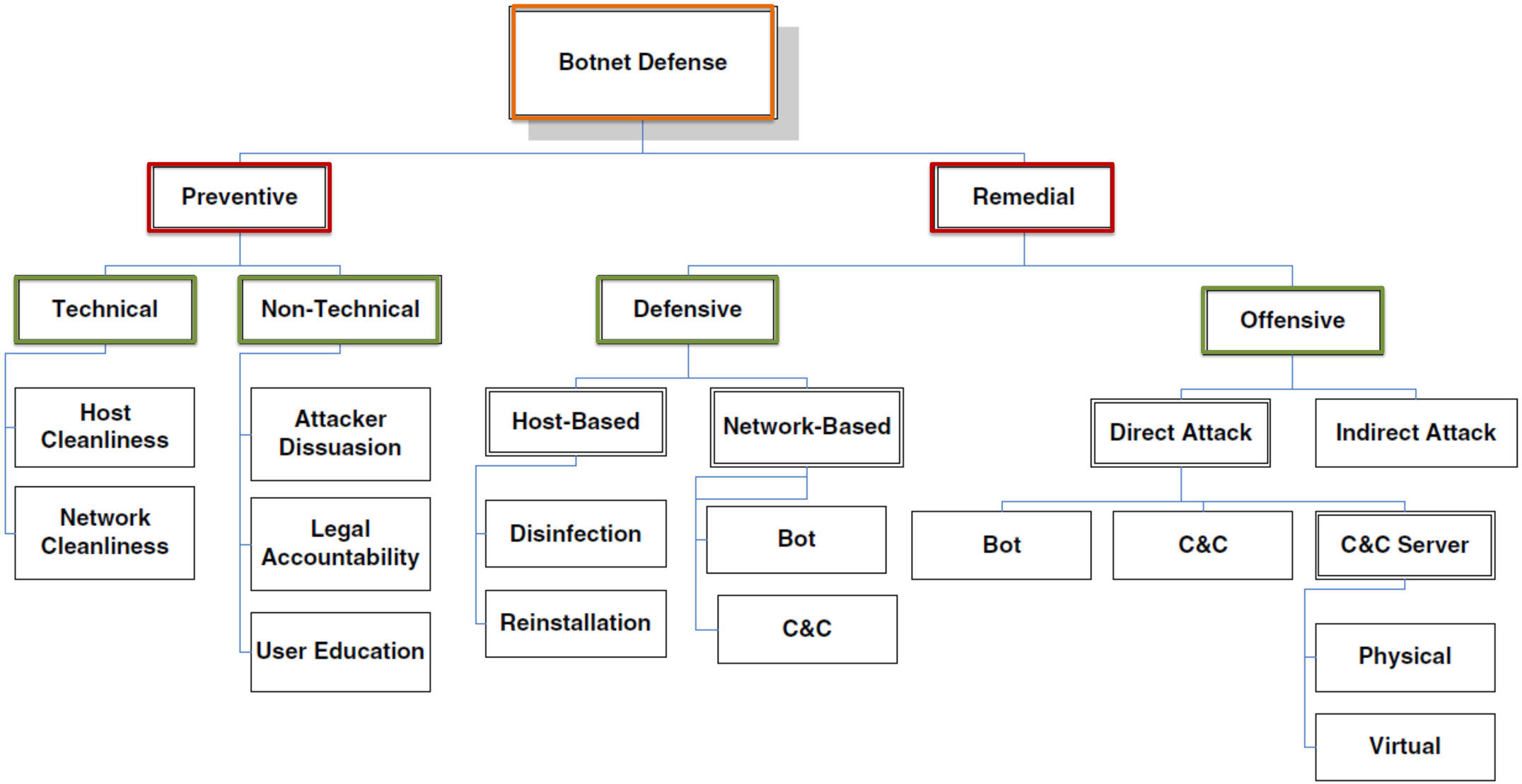
Remedial defense mechanisms: defensive strategies

- *S*elf-recovery in the event of botnet infection
- *Host-based*
  - restore individual bots to their clean state by *disinfection* or *reinstallation*
- *Network-based*
  - *block bots*
    - quarantine infected machines in administered network / walled-garden by ISP
    - infected machine is denied all network communication except whitelist of domain names helpful for remediation of the infection
  - *block C&C* communication
    - use URL and IP blacklists

Remedial defense mechanisms: offensive strategies

- Intimidate or completely paralyze a botnet
- *Indirect attack*
  - reduces usability of botnet to the botmaster
    (attacking underlying business model to significantly reduce profits)
  - eg inject fake information into the dropzone of information stolen by botnet
- *Direct attack*
  - attack one or more botnet components (bots, C&C communication, C&C servers)
  - eg exploit bugs in bot binary to take control of bot and remotely disinfect it,
    or trigger C&C command related to removal of the bot from the botnet
    (legal and ethical issues due to privacy invasion!)
  - poison C&C communication by introducing bogus commands
  - physical or virtual removal of C&C servers
  - carry out DoS attack against known C&C server

Summary


Botnet Defense
Preventive
Remedial
Technical
Non-Technical
Defensive
Offensive
Host Cleanliness
Network Cleanliness
Attacker Dissuasion
Legal Accountability
User Education
Host-Based
Network-Based
Disinfection
Reinstallation
Bot
C&C
Direct Attack
Indirect Attack
Bot
C&C
C&C Server
Physical
Virtual

Recent/future trends

- Botnets turning to *cloud computing*
  - host C&C server(s)
  - create bots in the cloud (botcloud) instead of infecting user machines
- Botnets based on *smartphones*
  - multiplies possibilities for C&C and malware propagation (3/4/5G, WiFi, SMS)
- Botnets based on *IoT devices*
  - already used for DDoS
- Botnets based on *social network*
  - as spam targets (victim) or malware propagation (facilitator)
  - already used for spreading fake news

Literature

- S. Khattak et al. (2014). *A Taxonomy of Botnet Behavior, Detection, and Defense*. IEEE Communications Surveys & Tutorials, vol. 16, no. 2, pp. 898-924

Agenda

- Botnets
- Botnet detection
- Defense against botnets
- *Our research on botnet detection*

Our research

- Detect botnets in Internet traffic
  - apply deep learning methods
  - on dataset of captured Internet traffic
  - classify traffic as botnet/non-botnet traffic

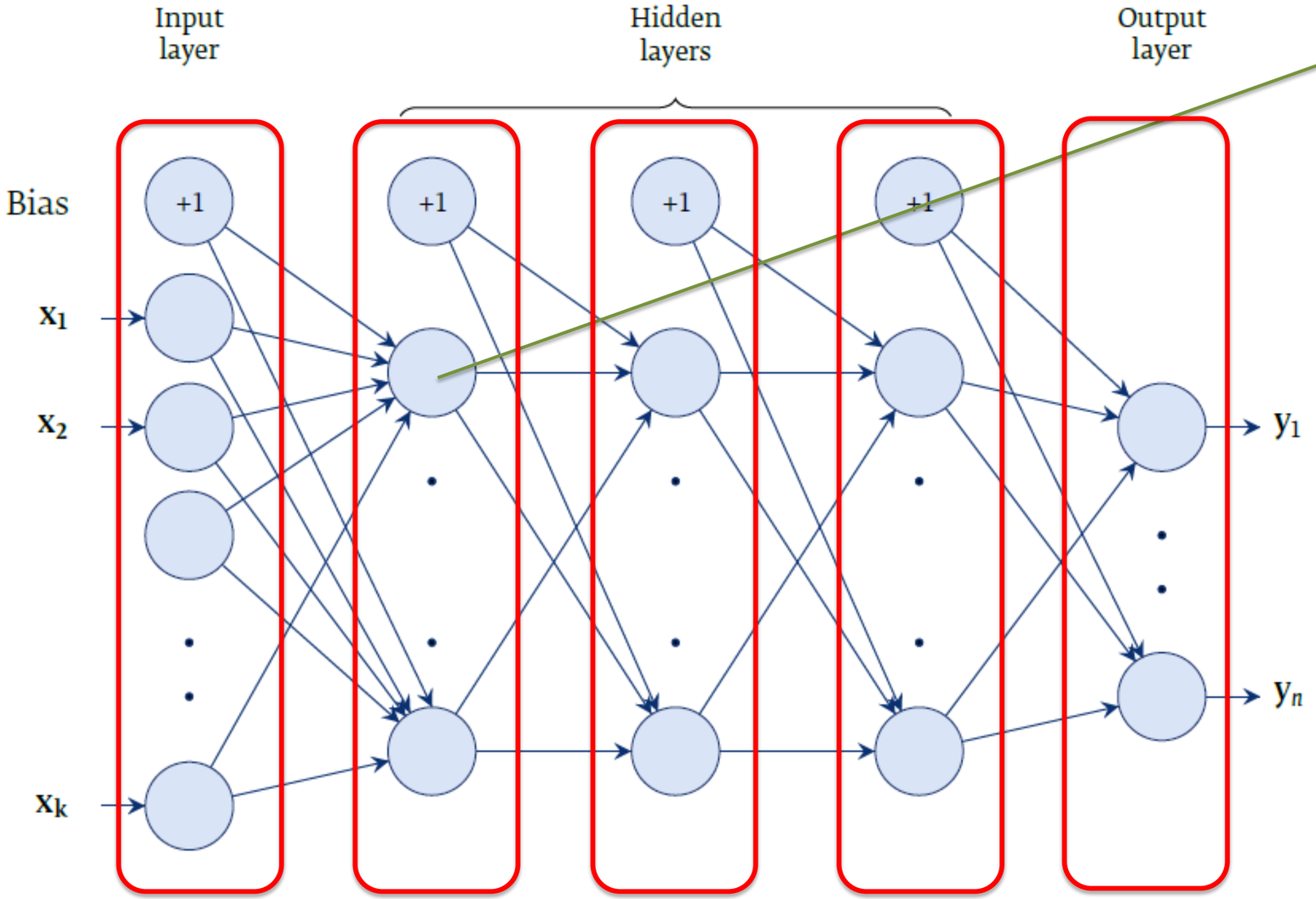
- Paper:
  Jos van Roosmalen, Harald Vranken, and Marko van Eekelen.
  Applying Deep Learning on Packet Flows for Botnet Detection.
  ACM Symposium on Applied Computing (SAC).
  April 9–13, 2018, Pau, France.

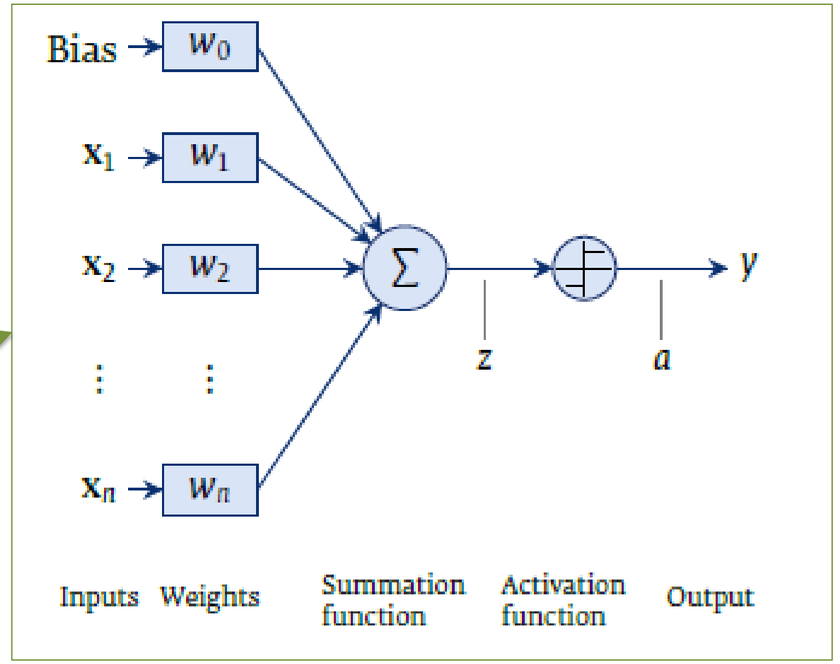
Research method

- Executed *650 experiments* with deep neural networks (DNNs) and ladder networks
  - *DNNs*: *supervised training*, and *unsupervised pretraining* with stacked denoising auto-encoders
  - *ladder networks*: *semi-supervised learning* (combining unsupervised and supervised training)
  - explored numerous network configurations, algorithms, and parameter settings

- Used a large *dataset* containing balanced mix of normal traffic and botnet traffic

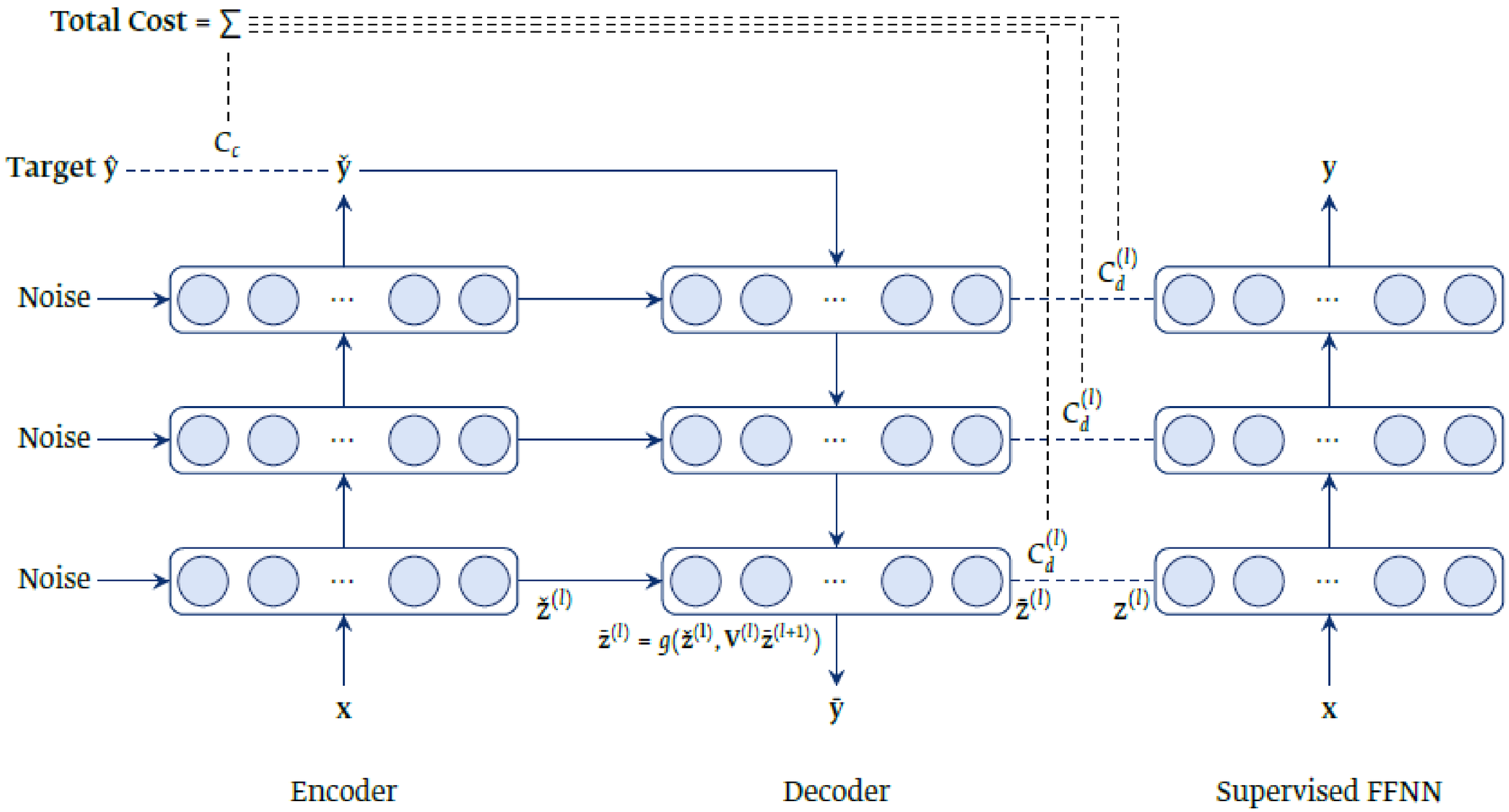
Deep neural network





$$y = \phi\left(\sum_{i=0}^{k} \mathbf{w}_i \mathbf{x}_i\right)$$

Ladder network

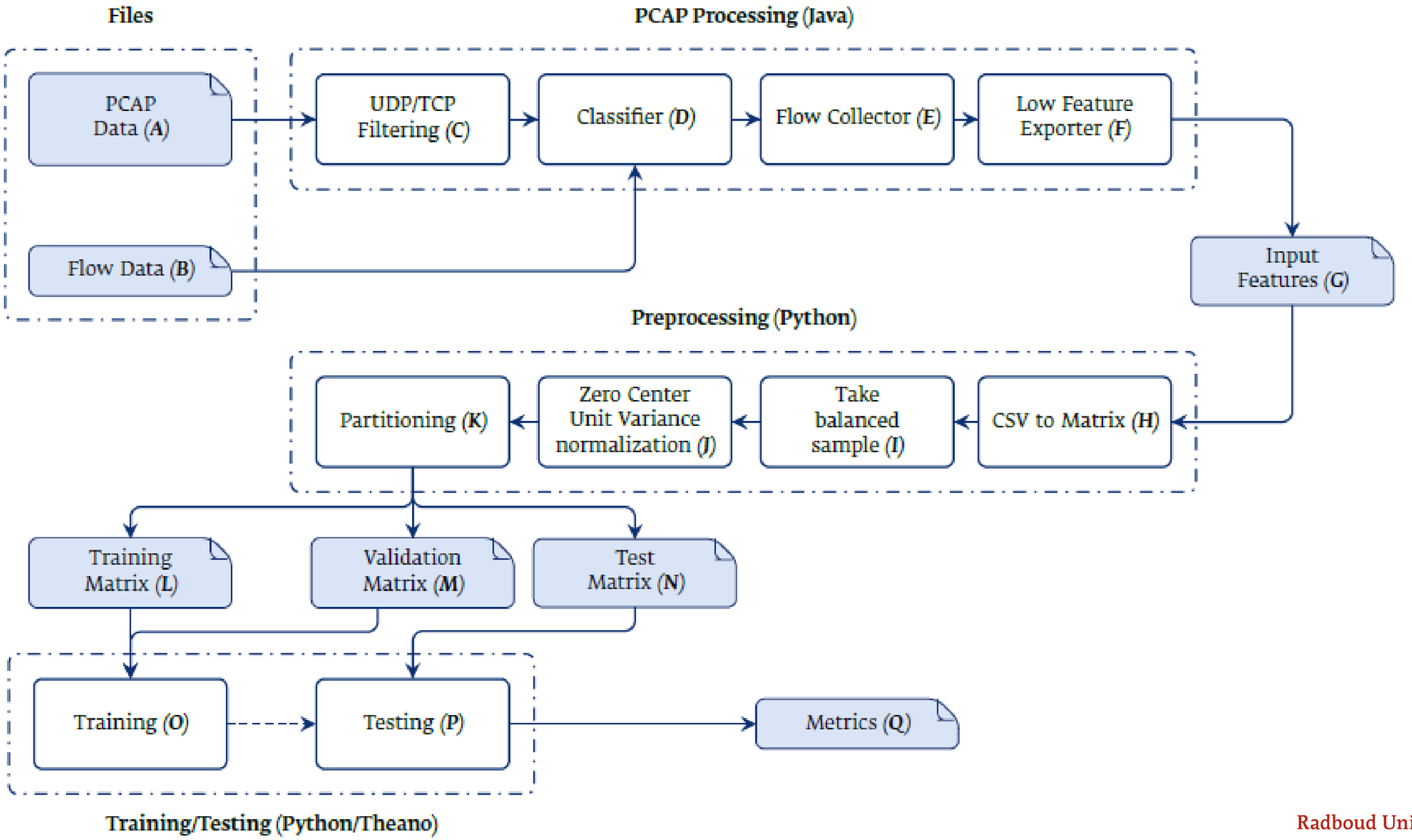


Datasets

- We performed our experiments on a *83 GB dataset* that we assembled from 5 research datasets
- Identified *flows*: streams of packets, each packet follows previous one within a certain time interval, that share $\langle source\text{-}address, source\text{-}port, destination\text{-}address, destination\text{-}port \rangle$
- From each flow, we used *4,158 features*
  - from each flow, we took the first *320 packets*
  - from each packet, we extracted *13 features* (Send/Receive indicator, Interval, DSCP, TTL, Protocol, Source port, Destination port, ACK, RST, SYN, FIN, PSH, payload size)

Dataset	Start date	End date	#flows	#used
UNB ISCX IDS	2010-06-11	2010-06-19	3,261,409	1,485,775
Ericsson Research	2007-10-08	2007-10-08	88,369	40,104
Lawrence Berkeley	2004-10-04	2005-01-08	2,144,178	976,168
Peerrush Storm	2007-10-27	2007-10-28	7,521,576	1,321,374
Peerrush Zeus	2011-11-07	2011-11-25	89,571	89,571
Peerrush Waledac	2009-02-17	2009-02-18	1,082,130	1,082,130
ISCX Botnet	1970-01-01	2013-02-04	4,878	4,878
Total			14,192,111	5,000,000

Data processing flow



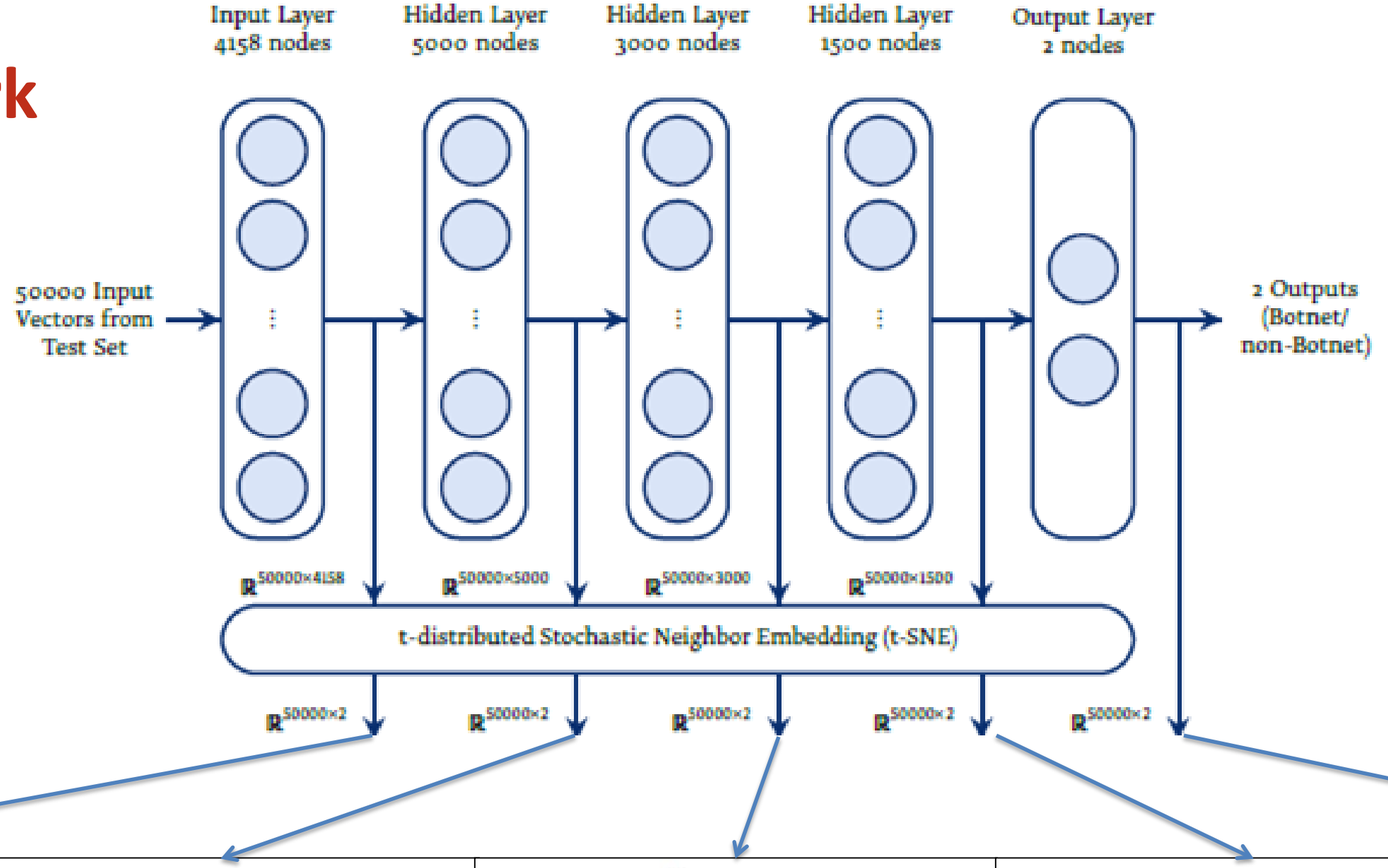
Experimental results

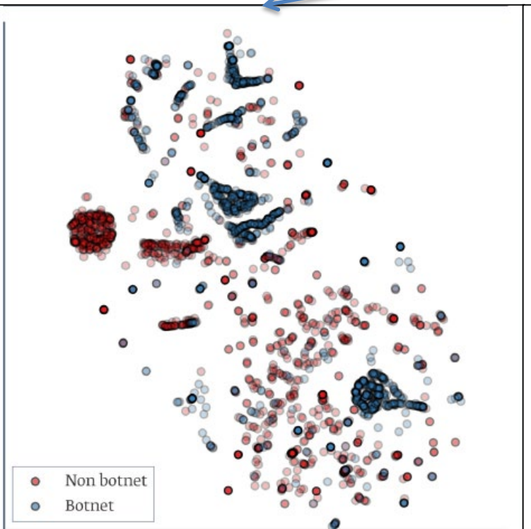
[Narang, P., Ray, S., Hota, C. and Venkatakrishnan, V. (2014). *PeerShark: Detecting Peer-to-Peer Botnets by Tracking Conversations*. IEEE Security and Privacy Workshops, pp. 108-115]
[Rahbarinia, B., Perdisci, R., Lanzi, A. and Li, K. (2014). *PeerRush: Mining for unwanted P2P traffic*. Journal of Information Security and Applications 19(3): 194 –208]

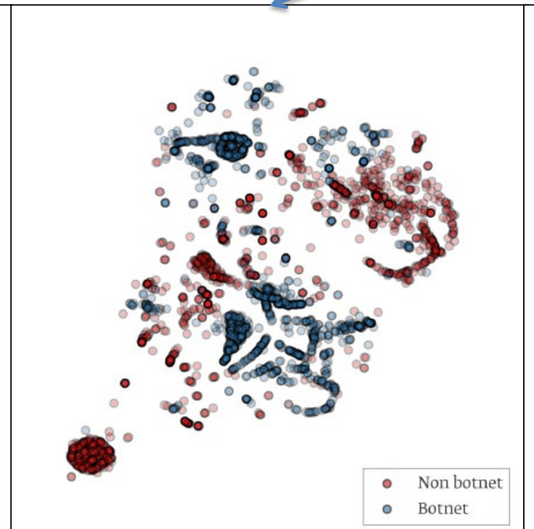
Botnet	Narang et al. [28]	Rahbarinia et al. [32]	Our results
Storm	0.988	1.000	0.998
Waledac	0.989	1.000	0.995
Zeus	-	0.925	0.938

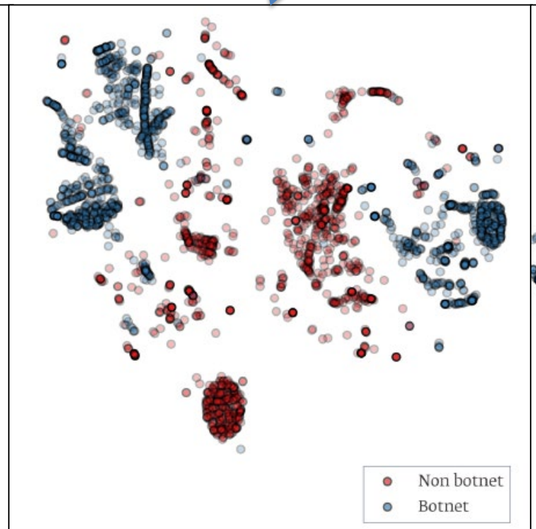
	Unbalanced		Balanced dataset						
						TPR			
Hyper-parameters	AUC	ER (%)	AUC	ER (%)	TNR	Storm	Waledac	Zero Access	Zeus
tanh; dropout 10%,40%	0.996	0.363	0.997	0.313	0.997	1.000	0.998	0.946	0.943
tanh; dropout 10%,50%	0.995	0.343	0.996	0.389	0.997	0.997	0.995	0.955	0.937
relu; dropout 10%,40%	0.996	0.333	0.996	0.357	0.997	0.999	0.996	0.920	0.942
relu; dropout 10%,50%	0.996	0.367	0.996	0.372	0.996	1.000	0.997	0.916	0.941
pretrain 25% noise; tanh; dropout 10%,50%	0.996	0.337	0.997	0.346	0.997	1.000	0.997	0.950	0.943
pretrain 25% noise; tanh; dropout 10%,50%	0.995	0.387	0.996	0.429	0.996	1.000	0.994	0.940	0.942
pretrain 10% noise; tanh; dropout 10%,40%	0.995	0.346	0.996	0.372	0.997	0.998	0.995	0.945	0.943
pretrain 10% noise; tanh; dropout 10%,40%	0.996	0.373	0.996	0.385	0.996	1.000	0.996	0.950	0.942
maxout poolsize 5; dropout 10%,40%	0.995	0.365	0.996	0.432	0.996	0.999	0.994	0.959	0.943
maxout poolsize 5; dropout 10%,50%	0.995	0.424	0.996	0.432	0.996	0.999	0.995	0.957	0.941
maxout poolsize 3; dropout 10%,40%	0.995	0.423	0.996	0.435	0.996	0.997	0.995	0.950	0.943
maxout poolsize 3; dropout 10%,50%	0.995	0.416	0.996	0.447	0.996	0.998	0.995	0.953	0.936
ladder; 0.01 noise; tanh	0.992	0.744	0.992	0.756	0.993	0.993	0.992	0.980	0.981
ladder; 0.01 noise; relu	0.992	0.821	0.992	0.830	0.993	0.993	0.992	0.981	0.980
ladder; 0.001 noise; tanh	0.992	0.825	0.991	0.837	0.993	0.993	0.991	0.975	0.993
ladder; 0.001 noise; relu	0.992	0.922	0.991	0.925	0.991	0.993	0.991	0.981	0.981

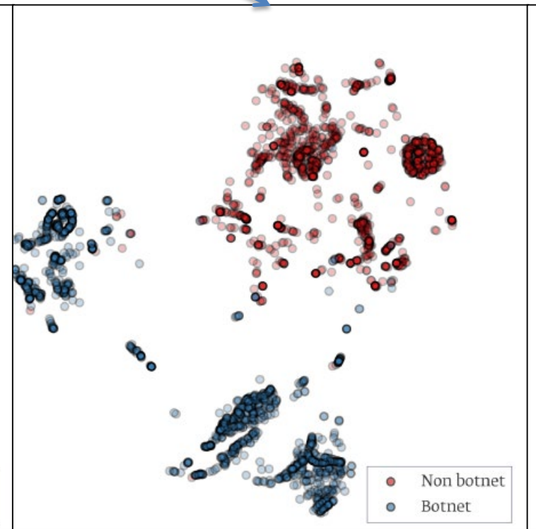
Best network

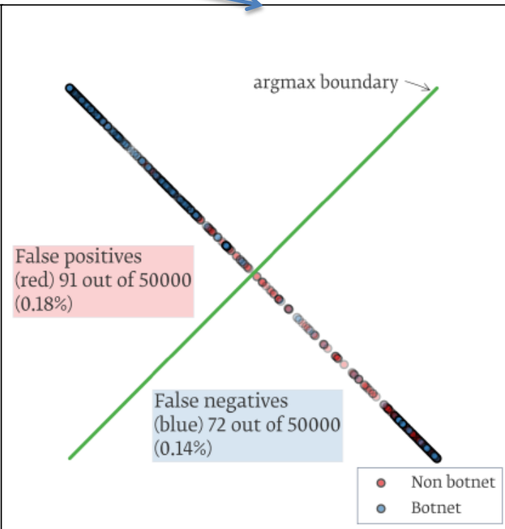










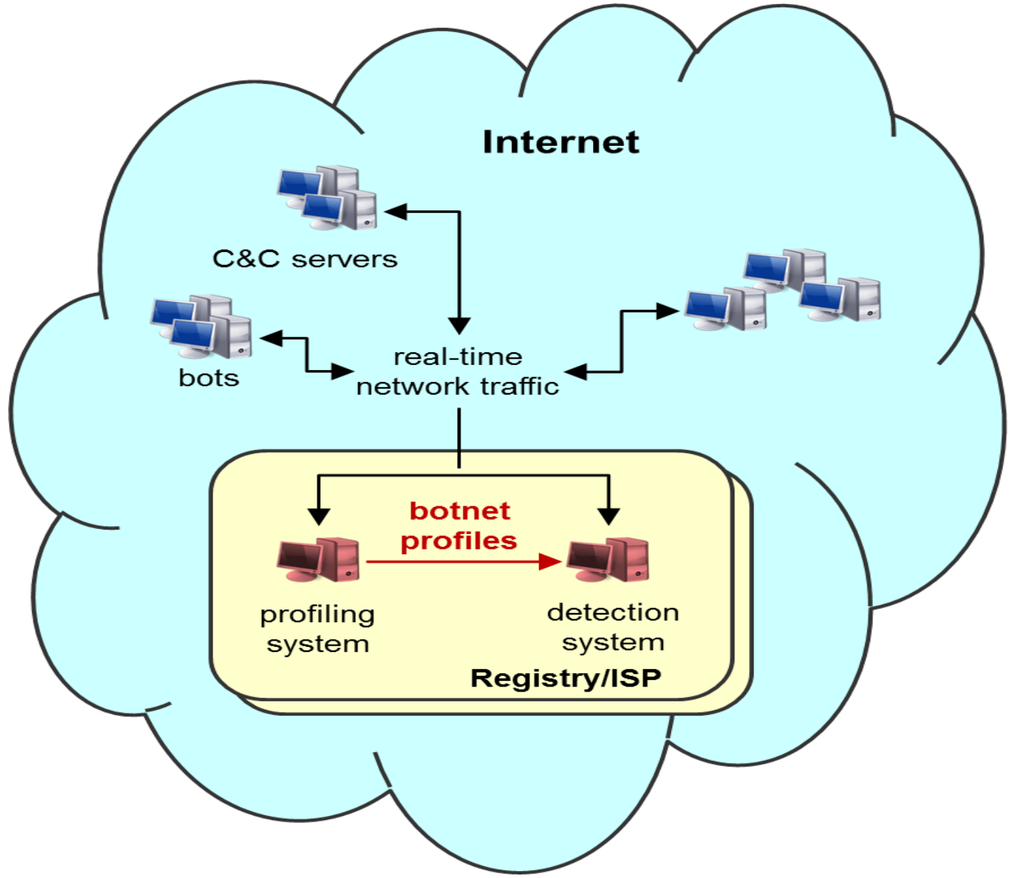


Follow-up research

- NN is still a black box; what is the feature hierarchy?
  - Pruning the DNN, from 4158-5000-3000-1500-2 to
    - 100-30-87-8-2 for 10% increase in mismatch
    - 505-60-295-14-2 for 5% increase in mismatch
  - Importance of features: PAYLOADSIZE from most of packets, SENDRECEIVE and TTL only in first packet(s), other features less significant
- Can similar results be achieved with other NNs (recurrent NN)?

Dagobert-project

- Design, application and governance of a botnet detection and profiling system (‘Dagobert’)
- Project goal
  - develop and evaluate an automated system
  - to accurately profile and detect botnets
  - at a national scale
  - by real-time analysis of real-life DNS traffic
  - at internet infrastructure providers



Graduation projects

- Application of machine learning to detect vulnerabilities/anomalies in
  - network traffic (eg botnet detection)
  - software (eg PHP web applications)

- In particular: *Detecting security vulnerabilities in C-code using machine learning* (Riscure)
  - http://www.open.ou.nl/hvr/RU_graduation_assignments.htm