UNIVERSITÀ DEGLI STUDI
DI TRENTO

Network Security

AA 2015/2016

Malware

Dr. Luca Allodi

Malicious software

- Programs acting without the conscious or designed authorization of a user or system
  - May exploit system vulnerabilities
- known as malicious software or malware
  - Programs that need a host program to operate
    - Not executable per se
    - e.g. viruses, logic bombs, and backdoors
  - independent self-contained programs
    - e.g. worms, bots
  - replicating or not
- sophisticated threat to computer systems

Taxonomy

- Virus → modifies legitimate software
- Worm → self-replicates
- Trojan horse → allows remote control of machine
- Keyloggers → sends typed info to attacker
- Rootkit → hook to libraries or system files
- Zombie, bot → remote coordinated control of multiple machines

→ Malware can assume characteristics of more than one type

Viruses

- software that replicate and install themselves without user consent
- Copies can be installed into
  - Programs
    - modifying them to include a copy of the virus
    - so it executes secretly when host program is run
  - Data files
  - Boot sector

Virus structure

- components:
  - infection mechanism - enables replication
  - trigger - event that makes payload activate
  - payload - what it does, malicious or benign
- prepended / postpended / embedded into infected program
  - when infected program invoked, executes virus code
  - Virus payload may change size of executable
    - Embedded layout may avoid this (system dependent)
      - e.g. Portable executables headers often have "empty" allocated memory words

Types of viruses

- boot sector
- file infector
- macro virus

By infection target

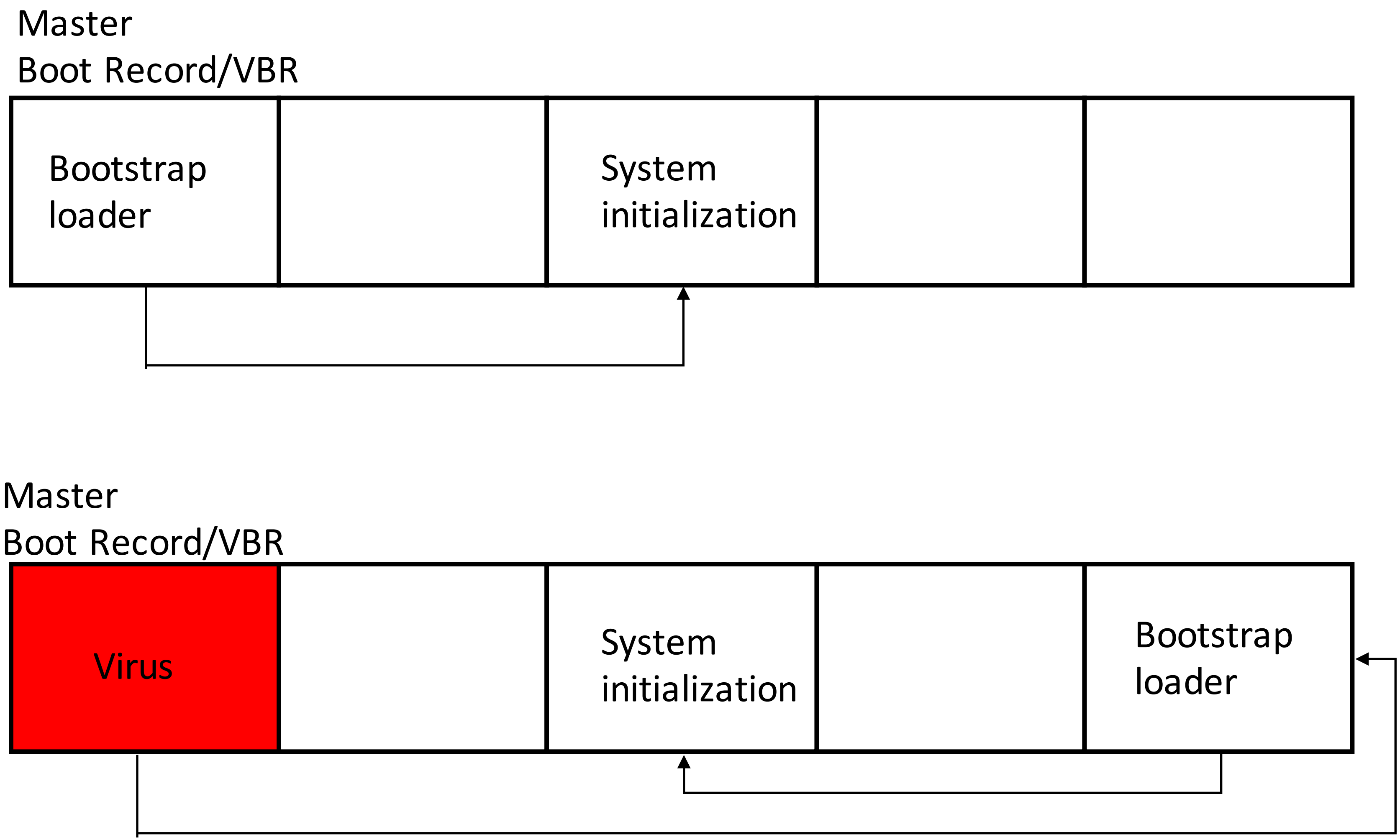
- encrypted virus
- polymorphic virus
- metamorphic virus

By concealment mechanism

Boot sector

- At boot time, the firmware checks for system components and tests them
- The operating system is then copied from the hard drive to the RAM
  - Master Boot Record contains code that ultimately leads to loading OS in memory
  - MBR typically small in size, points to **boot loader** (in Volume boot record, VBR)
    - “chain loading”
  - **Boot loader** actually loads OS

Boot sector infections - depiction



Rootkits

- Can take control of MBR
  - Can inject into kernel
  - Defeat disk encryption → Stone Bootkit
- set of programs installed for admin access
- subverting report mechanisms on processes, files, registry entries etc
- may be:
  - persistent or memory-based
  - user level → less powerful, may need additional vulns
  - kernel mode → hard to detect and remove
  - installed by user via trojan or intruder on system

Macro virus and file infectors

- became very common in mid-1990s
  - platform independent
  - infect documents
  - easily spread
- exploit macro capability of office apps
  - executable program embedded in office doc
  - often a form of Basic
- more recent releases include protection
- recognized by many anti-virus programs
- → evolved to email viruses
  - Exploit auto-execution bug in email-clients to infect system

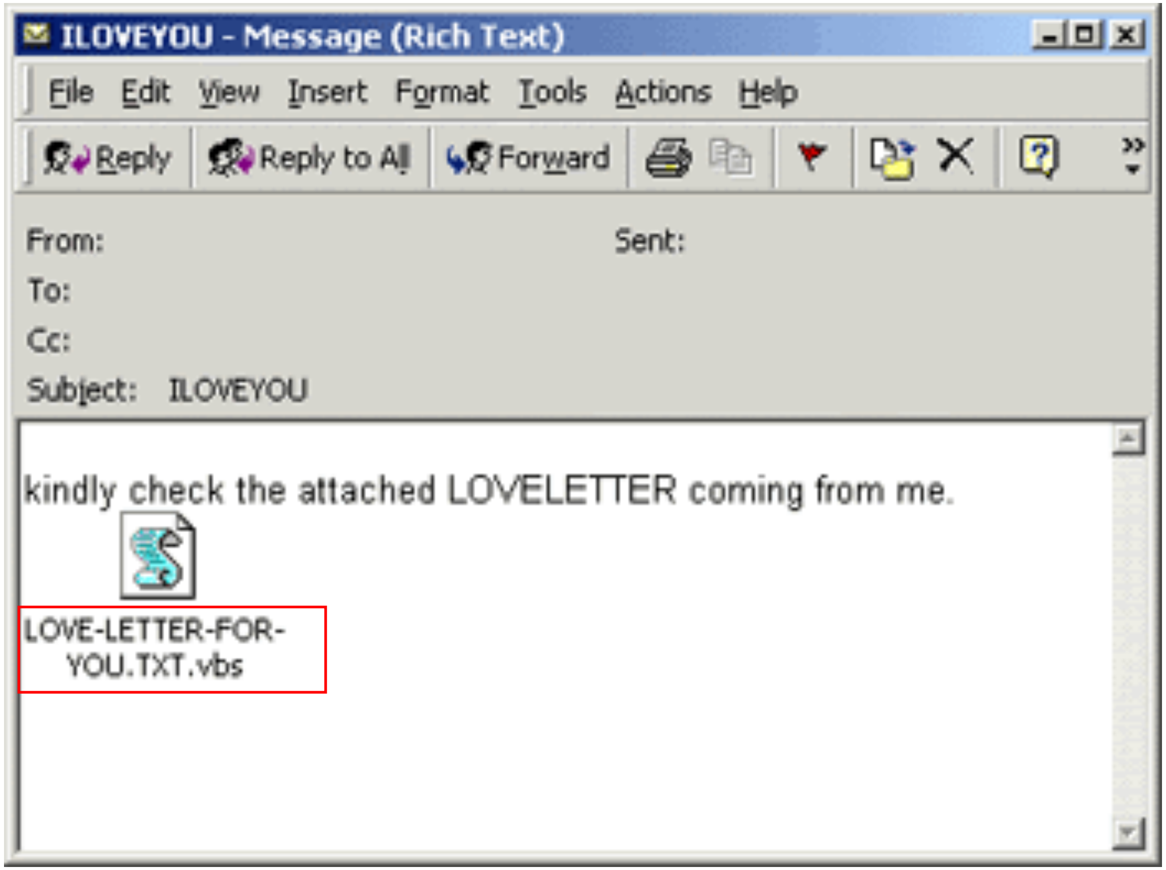
I Love You

User believes that's a txt file; It's actually VBS (Visual Basic Script).

Opening the attachment loads and executes script.

Impact → Disrupt system files
Replication → sends itself to the full contact list

Not relying on office, it still relies on an "interpreter" to execute → not native code



Virus countermeasures

- prevention - ideal solution but difficult
- realistically need:
    - detection
    - identification
    - removal
- if detect but can’t identify or remove, must discard and replace infected program

AV Defenses - evolution

- Virus & antivirus tech have both evolved
- Early viruses simple code, easily removed
- As become more complex, so must the countermeasures
- Generations
    1. signature scanners → looks for known traces of virus in memory
    2. heuristics → looks for features common in malware traces/strands
    3. identify actions → behavioral fingerprint of the malware execution
    4. Machine learning → classifiers trained to decide whether a file or program is acting maliciously

Defense 1 - Signature scanners

- Malware is analysed by security firm
- Footprint of malware in memory
  - Every time malware is loaded into memory, a pre-fixed series of bits will appear in ram
  - This footprint is the "signature" of the malware
  - Recognition happens through matching those sequence of bytes with all signatures known to a security product
- Purely "reactive" strategy → unknown malware does not yet have a signature
  - Detection can only happen after analysis

Defense 1 - Heuristics

- Partially addresses the polymorphism problem
- Viruses may evolve to different strains of the same virus family
    - Manual modifications
    - New malware versions
    - Genetic algorithms
- Different footprint but common characteristics
- Rather than having an exact match of the footprint in memory, detection happens by
    - Partial matching
    - Common characteristics of a virus strain

Evolution 1 - Polymorphic viruses

- Polymorphic:
  - the first technique that posed a serious threat to Antivirus
  - Uses encryption to obfuscate code
  - Decryption module is modified at each infection
    - → all samples will have a different footprint in memory
    - Fixed encryption per se would not suffice → Why?
- A well-written polymorphic virus has no parts which remain identical between infections
  - Signature checking is useless
  - Heuristics may work if encryption-decryption pair does not vary enough

Defense 2 - Generic Decryption

- Each polymorphic virus will look different on disk
- But at execution time code will always be the same
  - If detection happens when malware is executed, it's too late
- Generic Decryption → aka Sandboxing
  - Potential virus executed on an emulated environment
  - No actual access to system resources
  - the malware decrypts itself → signature checking will now work
- Modern malware can prevent execution in emulated or virtual environment
  - Via analysis of the execution environment
  - Prevent analysis by researchers

Evolution 2 - Metamorphic viruses

- Metamorphic:
  - To avoid being detected by emulation, some viruses rewrite themselves completely each time they are to infect new executables
  - After execution on emulated environment, signature won't match
- Metamorphic engine is needed to enable virus
  - Very Large and Complex
  - Ex. W32/Simile consisted of over 14,000 lines of assembly code

Defense 3 – behavioural detection

- Addresses issue with metamorphic malware and detection of previously unseen malware
- Based on set of actions that the malware performs
- Basic idea → malware behaves differently from legitimate software
  - System calls
  - Interaction with drivers (e.g. I/O)
  - System interrupts ..
- Very hard to enumerate all possible actions → exponential time
- Also hard to correctly identify set of actions that characterise malware
  - Risk of false positives higher than for heuristics and signatures (you need an hash collision for that)

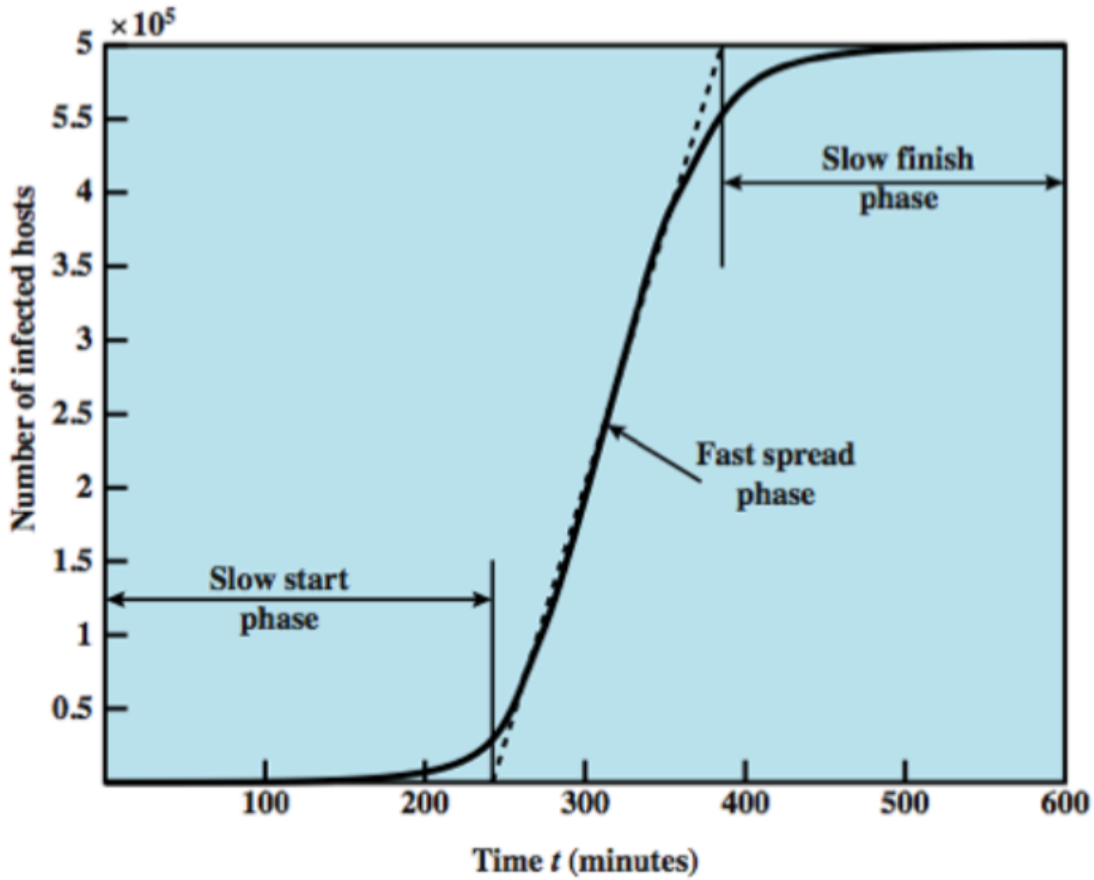
Defenses in practice

- Defense is only effective when it **prevents** malware execution
- Once the system is infected, system can not be trusted anymore
  - Malware removal can not be trusted
- Why?
  - Malware can affect the integrity of system procedures too
    - intercept antivirus' calls to OS disk drivers to analyse stored malware → returns "null" or benign file
    - Disable antivirus itself → e.g. Conficker
  - Run analysis from a clean drive on uninitialized infected OS

Worms

- replicating program that propagates over net
  - using email, remote exec, remote login
  - Exploitation of remote exploits
    - typically arbitrary code execution → buffer overflows
- has phases like a virus:
  - dormant, propagation, triggering, execution
  - propagation phase: searches for other systems, connects to it, copies self to it and runs; repeat.
- may disguise itself as a system process
- implemented by Xerox Palo Alto labs in 1980's

Worms propagation model

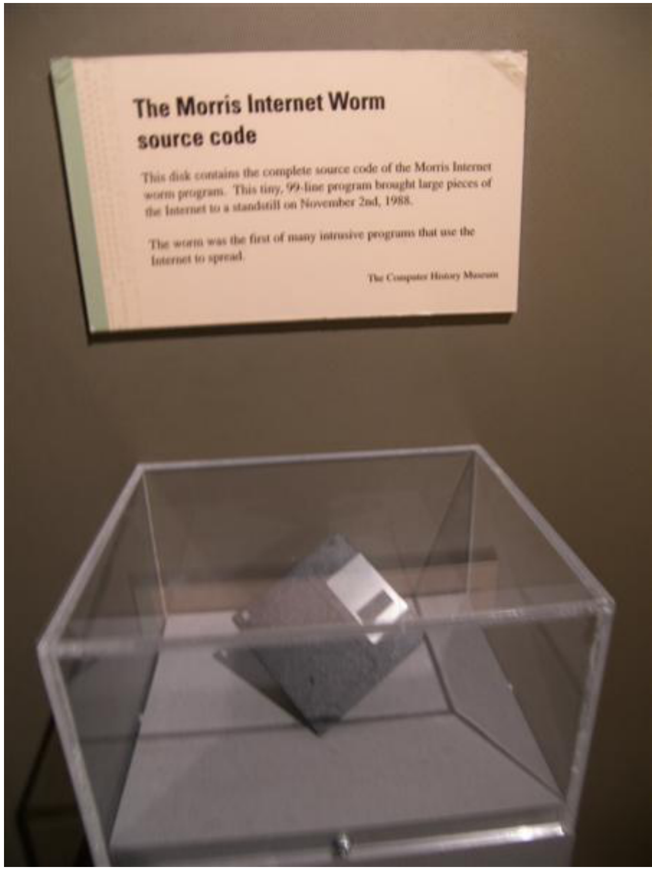


Historical internet worms

- Morris worm (1988): overflow in fingerd
  - 6,000 machines infected (10% of existing Internet)
- CodeRed (2001): overflow in MS-IIS server
  - 300,000 machines infected in 14 hours
- Blaster (2003): RPC overflow
- SQL Slammer (2003): overflow in MS-SQL server
  - 75,000 machines infected in **10 minutes**
- Sasser (2004): overflow in Windows LSASS
  - Around 500,000 machines infected

Morris worm

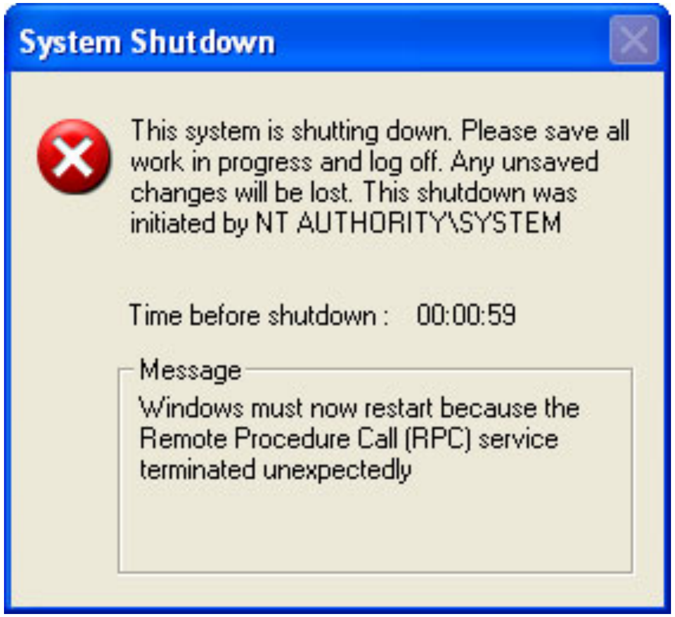
- 1988 by Robert Morris
  - Convicted under Computer Fraud and Abuse Act
  - 3 yrs probation
  - Now CS professor @ MIT
- Vulns:
  - Sendmail → could execute command via SMTP
  - Finger → BoF
  - weak passwords → dictionary attack
- No malicious payload but propagation too fast for the infrastructure to hold
  - Single computer could be infected multiple times → similar to a "fork bomb" issue
    - Malware needs testing too
  - Several million dollars in damage



The Welchia and Blaster worms

- Blaster → Appears in august 2003
  - Affects primarily Windows XP machines
  - SYN DoS against windowsupdate.com
  - Exploits a BoF in RPC (patch existed since May 2003)
  - Side effect → makes RPC unstable, XP unusable

- Welchia (anti-worm)
  - Removes Blaster infection, patches the vulnerability
  - Used the same Microsoft RPC bug as Blaster
  - Deletes itself after January 1, 2004
  - Was it a good idea ? (Why?)

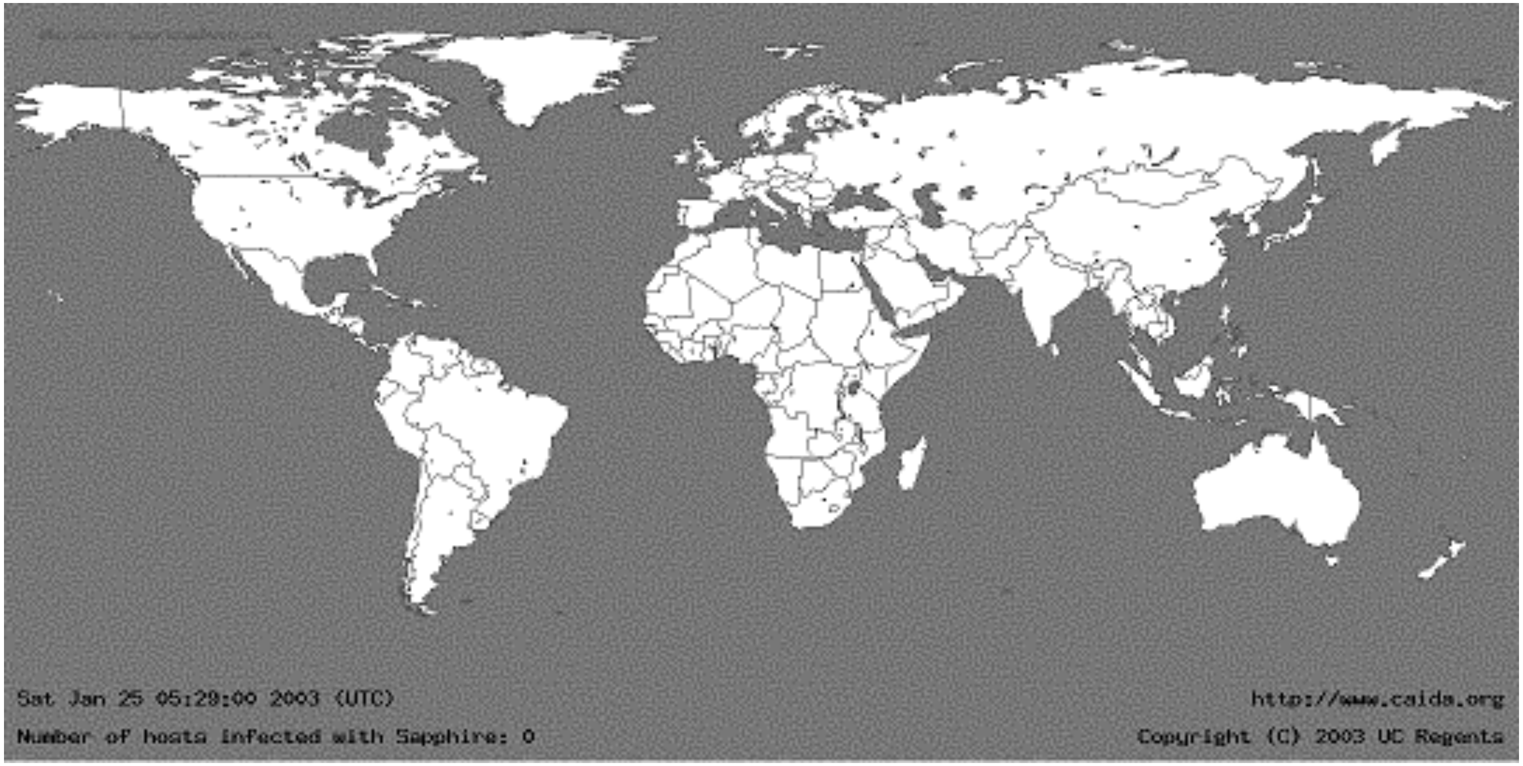


Slammer

- BoF in Microsoft's SQL server
  - Patch released 6 months earlier
- Single UDP packet to port 1434 infects the machine
  - Binary fits in the packet
  - Overwrite RET to point to malware in buffer
- Propagation by random generation of IP addresses
  - → Send copy of itself
- Works because IP space is populated, most MS systems
  - Do not care about false postives
  - 30k copies/second → UDP
  - Exponential growth
- So fast it saturated the bandwidth of the whole internet in 10 minutes
  - In combination with routers failing and subsequent generation of route table updates traffic
  - 75k SQL servers infected

Slammer – 5.29am UTC 25.01.03

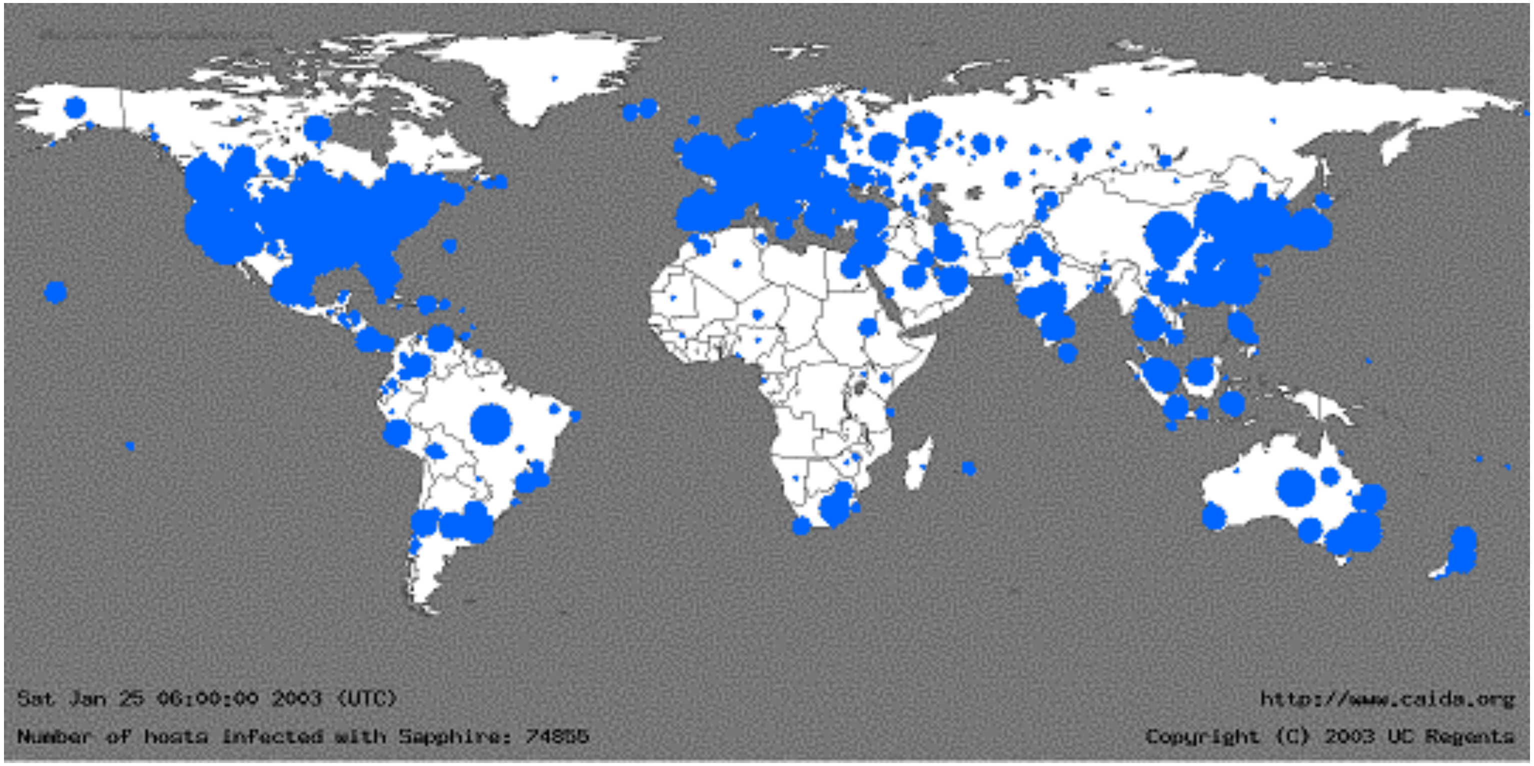
- http://www.caida.org/publications/papers/2003/sapphire/sapphire.html



UNIVERSITÀ DEGLI STUDI DI TRENTO

Slammer – 6am UTC 25.01.03

- Disc size is logarithmic in no. infected machines

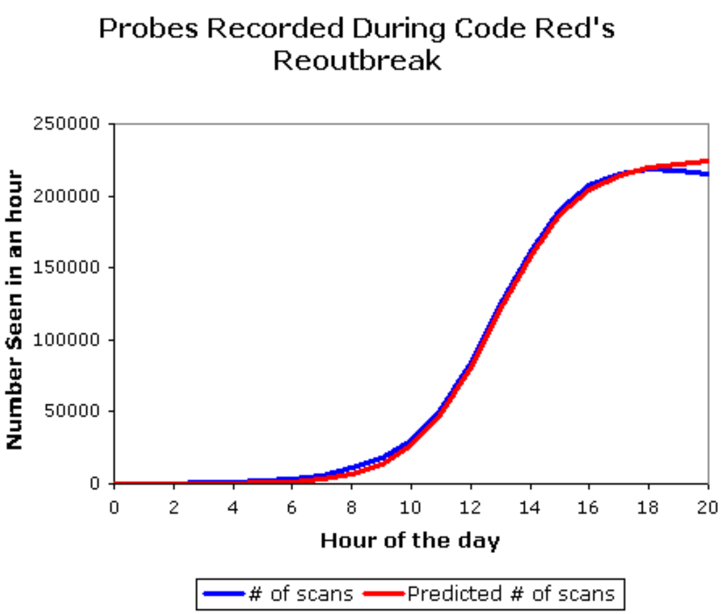


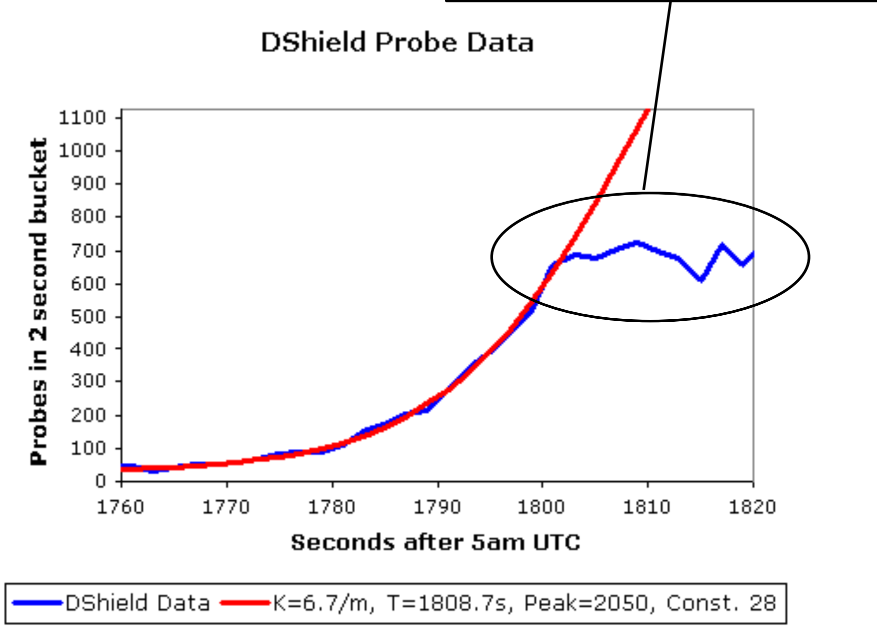
UNIVERSITÀ DEGLI STUDI DI TRENTO

Effects

- Killed several critical points of internet infrastructure
  - 5 DNS root servers
  - South Korea’s cell phone network (all of it)
  - Bank of America ATMs
- No malicious payload on infected systems
- Infection follows a logistic model in finite systems
  - Starts off exponentially, then levels out

Bandwidth saturation + Network failure





More recent worms

- Conficker (2008-09): overflow in Windows RPC
  - Around 10 million machines infected (estimates vary)
  - Introduces auto-updates, Domain Gen Algorithms,..
- Stuxnet (2009-10): several zero-day overflows + same Windows RPC overflow as Conficker
  - Windows print spooler service
    - Also exploited by Flame (announced in 2012)
  - Windows LNK shortcut display
  - Windows task scheduler
- Flame (2012) → MD5 collision, valid certificate for windows update

Conficker

- First detection in November 2008
  - Patch available in October 2008
- Uses a buffer overflow in Windows Server Service
  - MS08-067
  - Forged RPC request leads to shellcode execution
- Several versions of the worm
  - Conficker.A → B,C,D → Conficker.E
  - Shellcode connects to remote HTTP server
  - Attaches malicious DLL to svchost.exe or other processes
  - Variants B,C → introduced new infection drivers

Conficker - impacts

- Hard to estimate actual extension of infection
  - Different versions of malware have different propagation strategies
  - Anywhere from ~2 million hosts to 15 million hosts
- Stealing personal and sensitive information
  - Banking credentials
  - CCNs
  - Machines under the control of attacker → "botnet"
- Some very high-level targets were infected
  - French Navy systems shutdown → aircrafts grounded
  - Sheffield Hospital, UK → managers turned off security updates for 8000 systems
    - Bad decision? Some systems rebooted because of an update mid-surgery → shut it all off
    - 800+ systems infected

Conficker B → Infection drivers

- NetBIOS functionalities
  - Execute remotely by copying itself into admin share
  - If share is pwd protected, attempt dictionary attack
    - Attempts 240 passwords
- USB removable device
  - Malware copies itself as autorun.inf
  - Malware is run everytime a user mounts the driver

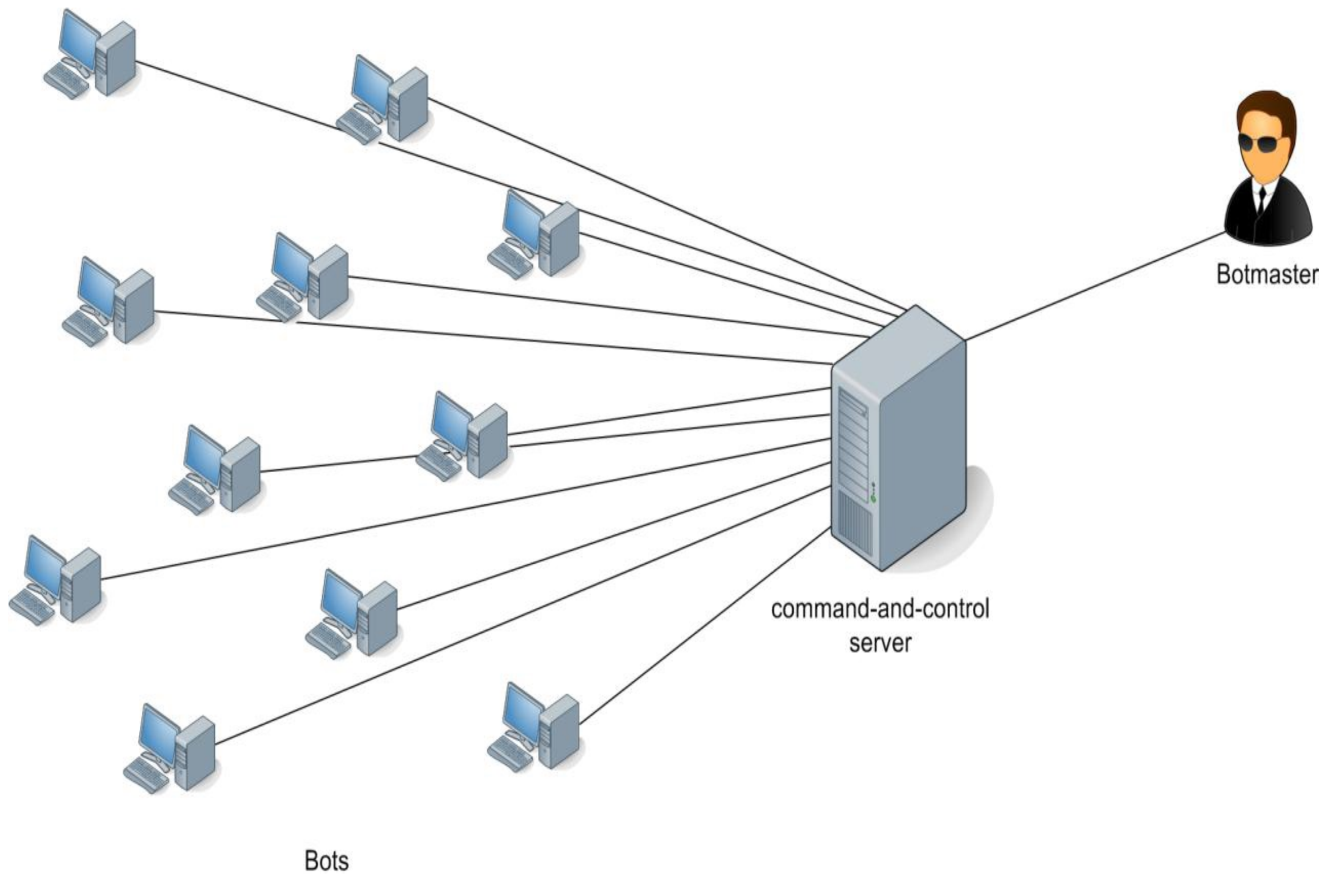
Conficker - defenses

- Conflicker patches MS08-067 after infection
  - This is to minimize infections from other malware
- Installed patch is custom
  - Allows for Conficker re-infections
  - Essentially a backdoor for the worm
    - Can be used to update malware on infected hosts
- Disables several system services
  - No autoupdate, Win Security service, ..
  - Blocks DNS requests for antivirus-relate domains & winupdate
- Conficker payloads are signed (SHA-1 hash + RSA w/ 1024 bit secret key) and encrypted (RC4)
  - Public key hardcoded in payload
  - Variants increase key size & hashing algorithm

Botnets

- Virtual Network of infected machines under the control of a "bot herder"
- Machines can perform any kind of action for the bot herder
- Managed through a **command & control** server under the control of an attacker
  - Pushes configuration files
  - Functionality updates
  - Bots must be able to communicate with C&C server
- Centralised vs peer-to-peer design

Botnets – centralised architecture

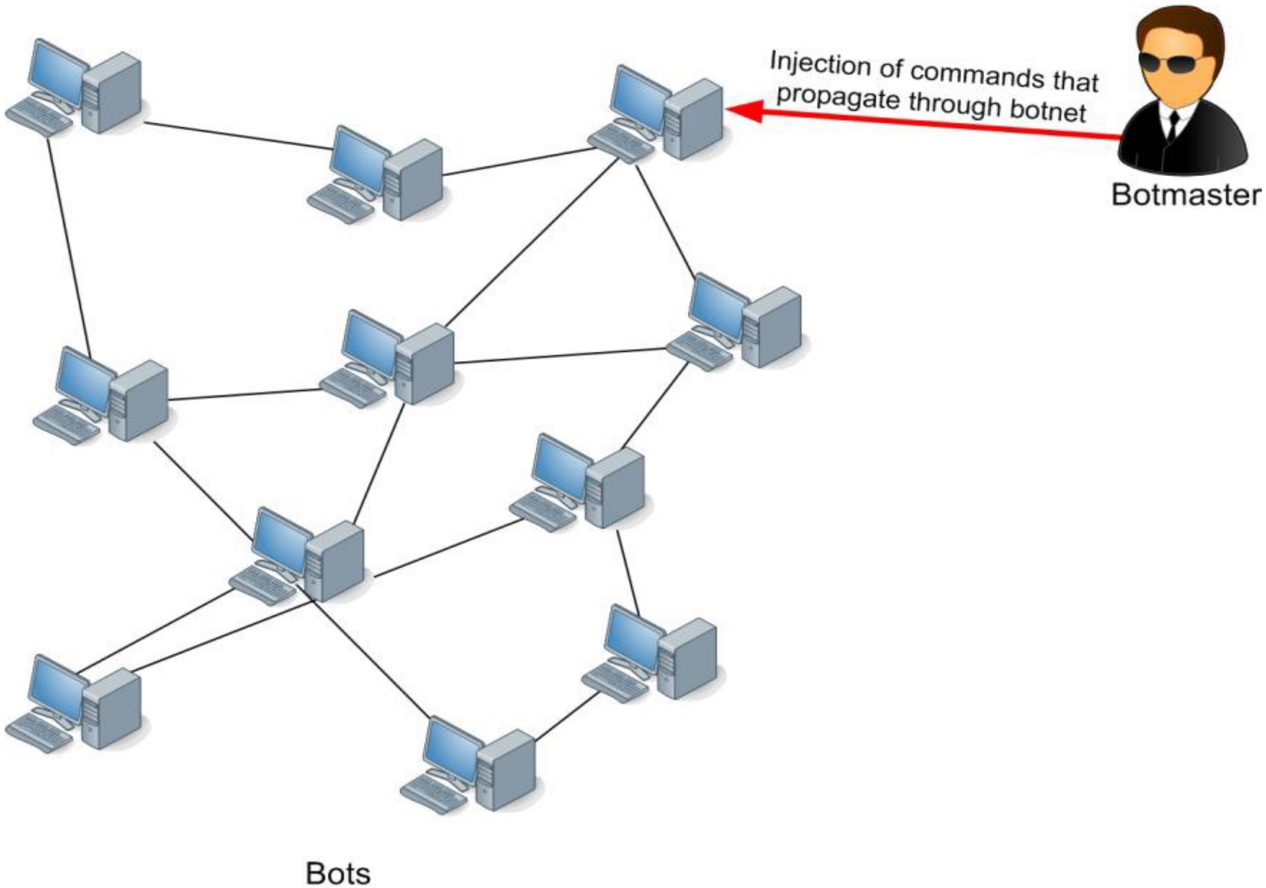


Source: Botnets: Detection, Measurement, Disinfection & Defence - ENISA

Types of centralised botnets

- Bots communicate with the bot herder via
  - IRC (Internet relay chat) server
    - First definition of “bot”
    - Served “human users” by providing automatised services
    - Essentially a program accepting commands in inputs and retrieving answers
  - HTTP
    - Connects to a remote HTTP server
    - Two approaches
      - Bot contacts fixed (set of) IP(s)
      - Bot resolves domain dynamically
    - Fast-flux vs domain-flux
- C&C server is single-point-of-failure
  - Who controls the C&C controls the botnet

Botnet – p2p architecture



Source: Botnets: Detection, Measurement, Disinfection & Defence - ENISA

p2p architecture

- More robust than centralised architecture
- Commands are spread through the network
- Bots can act as both slaves and masters dynamically
- When new machine is infected, bot joins the network
  - Hard-coded list of peers are contacted upon infection
    - Updates its neighboring peer list
  - Mixed p2p/centralised approach
    - Centralised web cache with list of peers
  - Infected bot inherits peer list from infector

Three types of p2p botnets [Silva 2012]

- Parasite:
  - all bots are selected from vulnerable hosts within an existing P2P network.
  - Number of vulnerable hosts in the existing P2P network limits the scale of a parasite botnet.
  - Not flexible and greatly reduces the number of potential bots under the botmaster's control.
- Leeching:
  - members join an existing P2P network and depend on this P2P network for C&C communication.
  - Bot candidates may be vulnerable hosts that were either inside or outside an existing P2P network.
- Bot-only:
  - builds its own network in which all members are bots

Botnets - usage

- Performing distributed denial of service attacks (DDoS)
  - Same techniques as normal DoS attacks, but amplified by a factor equal to size of botnet
- Spam → used to distribute spam emails
  - Can lead to further infections
  - Subscription to services / goods
- Computational power → use CPU/GPU time to find hash collisions, break ciphers, mine bitcoins ..
- Steal sensitive information from the infected machine
- Rental → bot herder can rent part of the bots to other criminals
  - Outsource computations / buy Credit card numbers (CCNs) ..

Centralised botnets - details

- Bots can not operate if they can not contact the C&C server
- Centralised Botnet take downs happen by "sinkholing"
  - Security researcher/firm takes control of C&C
- C&C server needs to be protected
  - Change IP address frequently → **fast-flux**
    - Makes it hard for an attacker to take it down
    - One domain mapped to several IP addresses
  - Change domain frequently → **domain-flux**
    - Each bot generates "valid domain names" periodically and resolves them

Domain flux

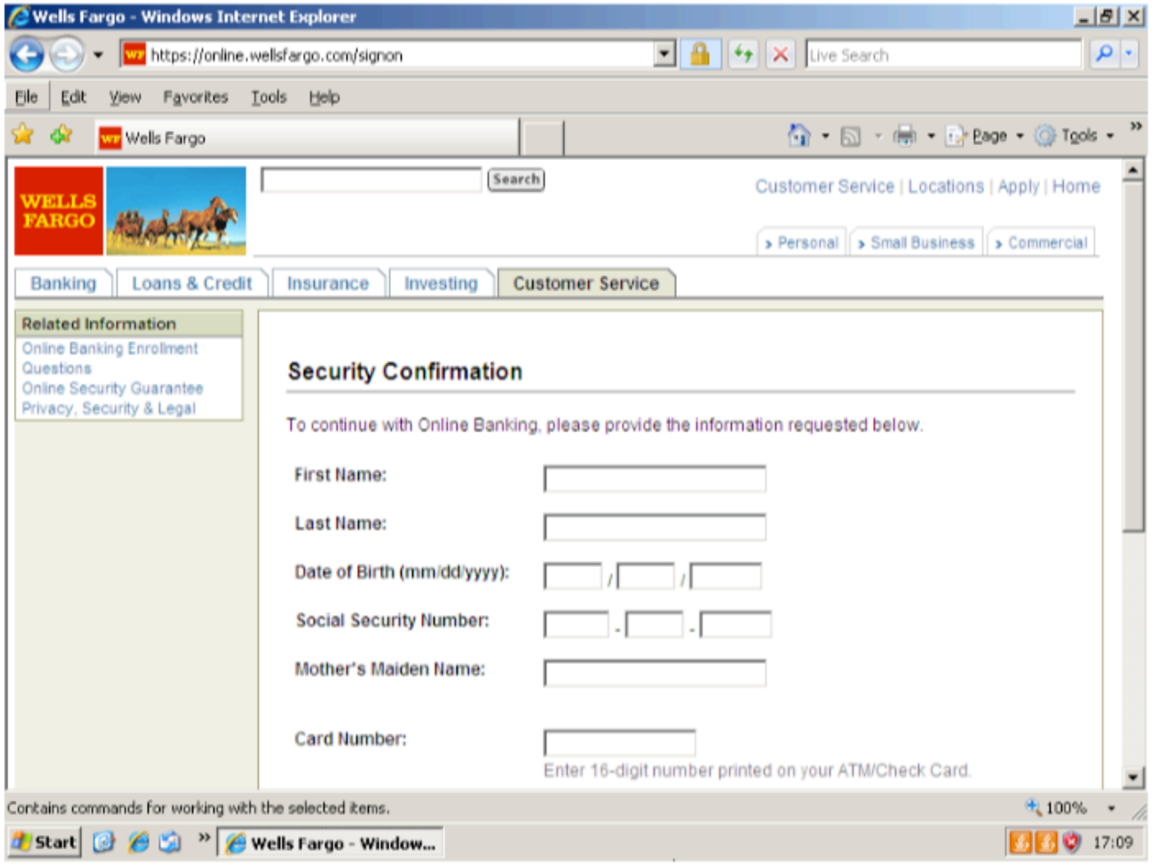
- Each bot uses a **Domain Generation Algorithm (DGA)** to generate a list of possible domains at a certain time
  - "rendezvous" domains
  - List is generated independently by each bot
- If bot gets no answer from a generated domain, it simply switches over to the next in list
- Conficker A → e.g. txkjngucnth.org
  - http://blogs.technet.com/b/msrc/archive/2009/02/12/conficker-domain-information.aspx
- Sometimes botnets perform accidental DoS attacks against "colliding" domain names
  - DGA generates a domain that already exists
  - All bots try to contact that domain (it happened)
    - jogli.com, praat.org, …

Putting it all together – a case study: Torpig [Stone-Gross 2009]

- Torpig was a botnet active in 2009
- Used Mebroot as a rootkit
- Mebroot substitutes the Master Boot Record of the machine → used to perform actions at boot time
  - Harder to detect malware
  - Executed in the context of *explorer.exe*
  - Operates directly on disk blocks (through disk drivers)
  - Upon reboot, downloads and activates malware
    - Torpig in this case
    - Encrypted communciation with Mebroot server
    - Malware stored locally, encrypted
- Mebroot provides functionalities to embed (malicious) modules to normal system boot

Torpig - functionalities

- Credential stealing
- Generation of phishing attacks for a set of pre-defined websites
- Torpig module injects phishing content to webpage presented to user
  - typically a login page



Sinkholing Torpig

- Team @ University of California reverse engineered the DGA
- Noticed that a set of domains that will be generated between 25th Jan and 15th Feb were not registered yet
- Researchers registered the domains and replicated "fake" C&C server
  - All it needed to do is to confirm itself as a valid server
  - Torpig uses HTTPS but accepts any certificate as valid
  - Passively listening to whatever the bots were sending
- 4th Feb Mebroot pushed update for Torpig → only about 10 days of data

UNIVERSITÀ DEGLI STUDI
DI TRENTO

Torpig size

- IPs change very frequently → counting unique IPs not a good proxy for botnet size
- Each bot has unique id + additional features
- About 180.000 hosts (1.2M IP addresses)

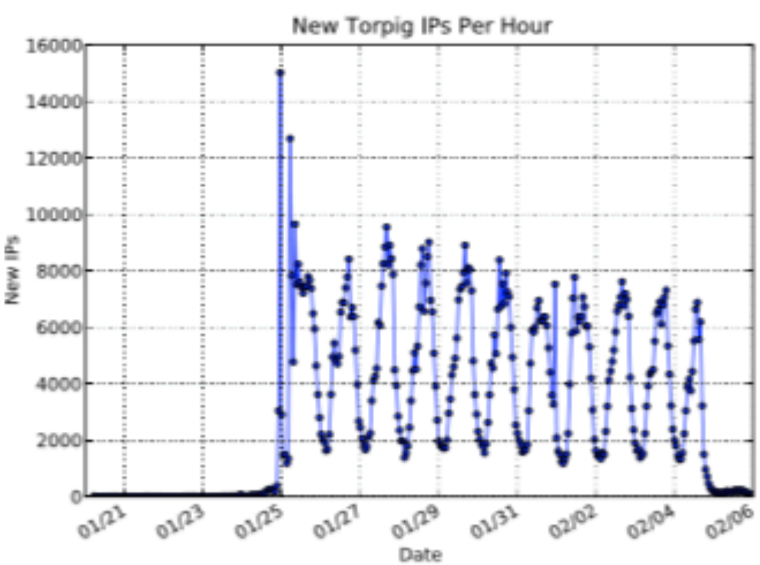


Figure 5: New unique IP addresses per hour.

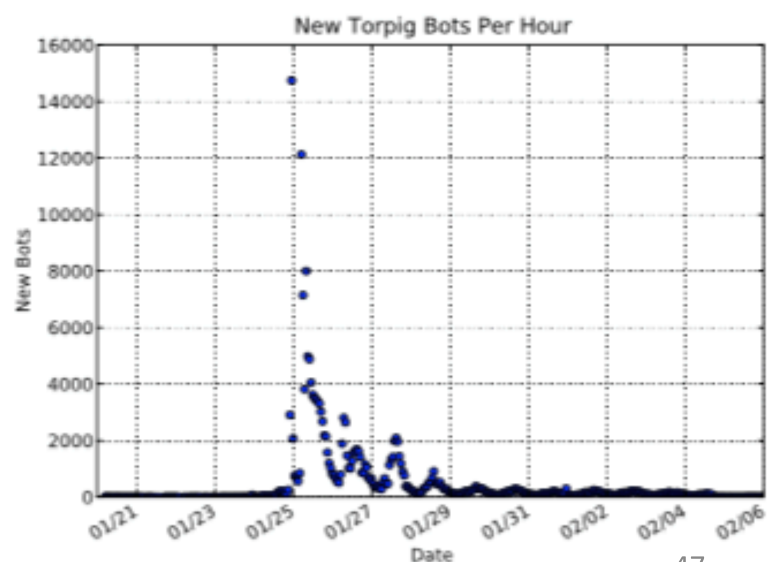
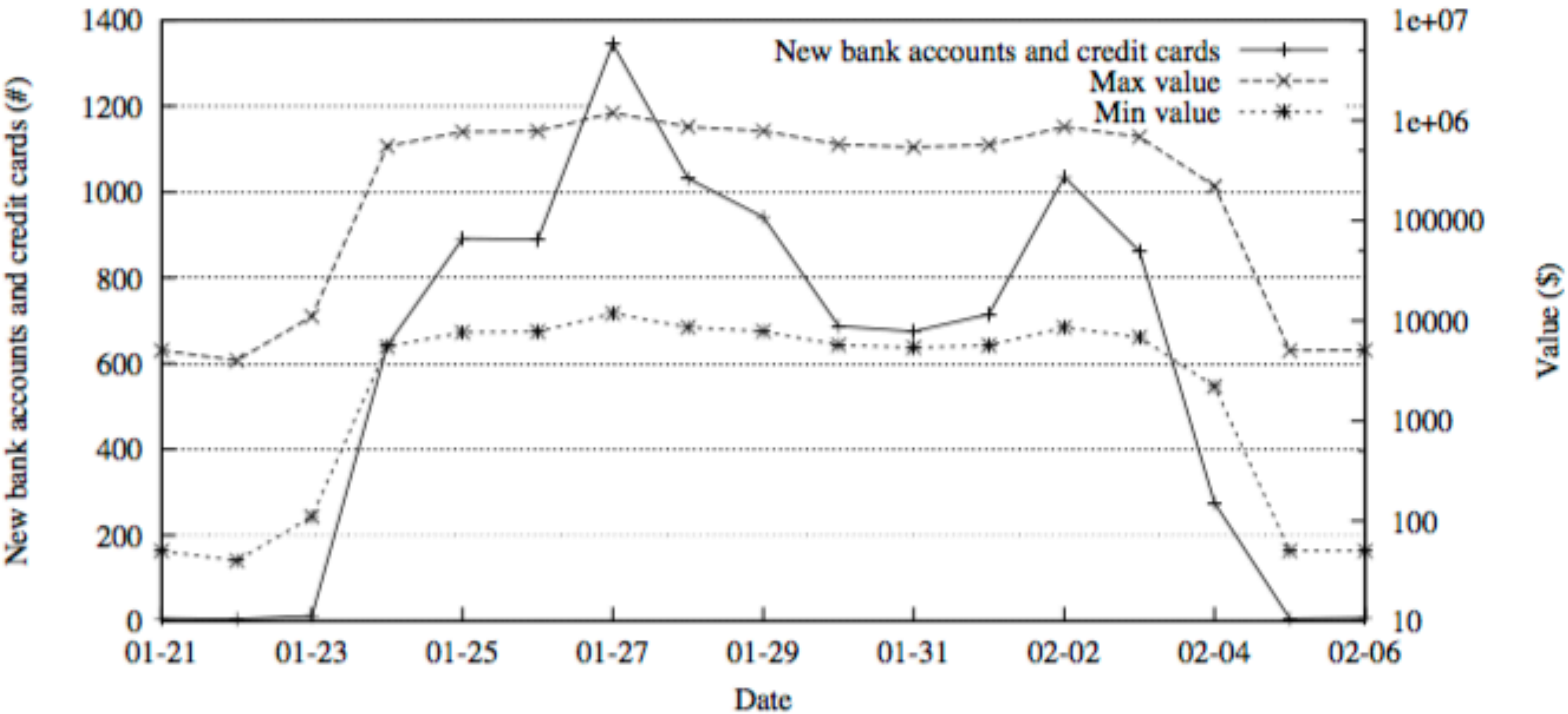


Figure 6: New bots per hour.

Torpig – collected data

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

Torpig – collected data



Reading list

- Silva, Sérgio SC, et al. "Botnets: A survey." *Computer Networks* 57.2 (2013): 378-403.
- Stone-Gross, Brett, et al. "Your botnet is my botnet: analysis of a botnet takeover." *Proceedings of the 16th ACM conference on Computer and communications security*. ACM, 2009.