March 7, 2016

Malware & Botnets

Computer Security 2 (GA02)

Lecturers: Emiliano De Cristofaro, Gianluca Stringhini

Malware

- Shortening for Malicious Code
  - Software that fulfills author's malicious intent
  - Intentionally written to cause adverse effects
  - Comes in many flavors but all share a common characteristic: perform some unwanted activity

- Term often used equivalent with virus (media…)
  - However, a virus is just a type of malware
  - Classic viruses account for only <3% of malware in the wild

Motivation for writing malware?

- Hacker old school (prior to ‘90s)
  - Fun
  - Show that something is possible
  - Prove the author “leetness”
- The virus golden age (‘80s-‘90s)
  - Vandalism
- The new “criminal” age (2000-)
  - Steal information
  - Control machines for, e.g., DDoS or spam
  - Make money!!!

An Underground Economy

- Moving from “hack for fun” to “hack for profit”
  - Underground economy
- Understanding the technical side is not enough
  - Monitor underground channels
  - Understanding the economic structure
  - Find and disrupt weak links
- Need solutions that make an attack vector economically not viable

History 1/5

- 1981: First widespread outbreak of a virus
  - Propagates on floppy disks for the Apple II platform
- 1983: The first documented experimental virus
  - Len Eidelmen coined the term virus in connection with self-replicating computer programs
  - Fred Cohen‘s pioneering work on the topic
- 1987: File infectors
  - “Christmas Tree” worm hit IBM Mainframes (500K replications/hour)
- 1987: Vienna virus is the first to intentionally destroy data
- 1988: First Worm, or the Internet worm
  - Internet worm created by Robert Morris
- 1989: IBM releases Viruscan for MS-Dos

History 2/5

- 1991: First polymorphic virus
- 1992: First virus to look for (and damage) an anti-virus
- 1995: First macro virus
- 1998: First Java virus
- 1999: Melissa
  - Word macro virus + first large scale email propagation
- 1999: First Distributed denial of service (DDOS)
- 1999: Kernel Rootkits
  - Knark (modification of system call table)

History 3/5

- 2000: Spyware:
- 2000: First virus for mobile phones
- 2001: Code Red
  - First large-scale, exploit-based worm
- 2003: SQL Slammer worm
  - Extremely fast propagation
- 2003: Sobig makes the first attempt to create a botnet
- 2007: Storm Worm+Botnet
- 2008: Rogue Antivirus
- 2008: Koobface targets Social Networks
- 2009: Conficker is the first pandemic malware infection
  - Microsoft is still offering $250K to find the authors

History 4/5

- 2010: Stuxnet
  - Several Zero day exploits, stolen root certificates
  - Pass “air gap”
- 2011: Duqu
  - Malware “Framework”
  - Zero day in MS Word
  - Linked to Stuxnet ?
- 2012: Flame
  - Fake Microsoft certificate, using a MD5 Collision
  - Used to impersonate update.microsoft.com !
- 2013: Red October
  - Framework, 1000's modules, diplomatic targets
  - In memory only, no disk persistence

History 5/5

- 2013: NSA
  - Catalogue of implants
  - Backdoors security products, standards...

Vulnerabilities

- Mixing data and code
  - Violates important design property of secure systems
  - Unfortunately very frequent (macros, ...)
- Homogeneous computing base
  - Windows is just a very tempting target
- Unprecedented connectivity
  - Easy to attack from safety of home
- Clueless user base
  - Many targets available
- Malicious code has become profitable
  - Compromised computers can be sold and used to make money

Taxonomy

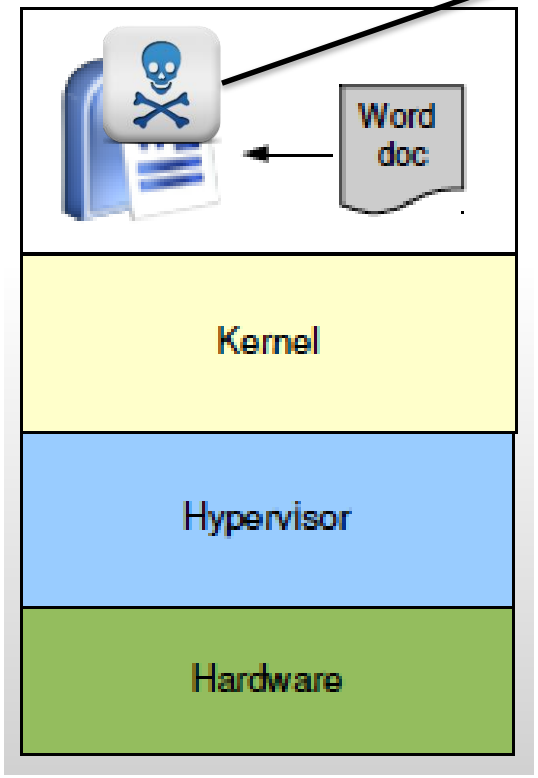
	NEED AN HOST PROGRAM	SELF-CONTAINED PROGRAM
SELF SPREADING	Computer Viruses	Worms
NON-SPREADING	Trojan Horses Rootkit	Keyloggers Spyware Dialers

Virus

- A program that can infect other programs by modifying them to include a, possibly evolved, version of itself
  - Fred Cohen (1983)

- Distinguishing features:
  - Reproduce its own code
  - Attach itself to other files (a virus cannot “survive” by itself)
  - Get executed when the infected executable file is executed

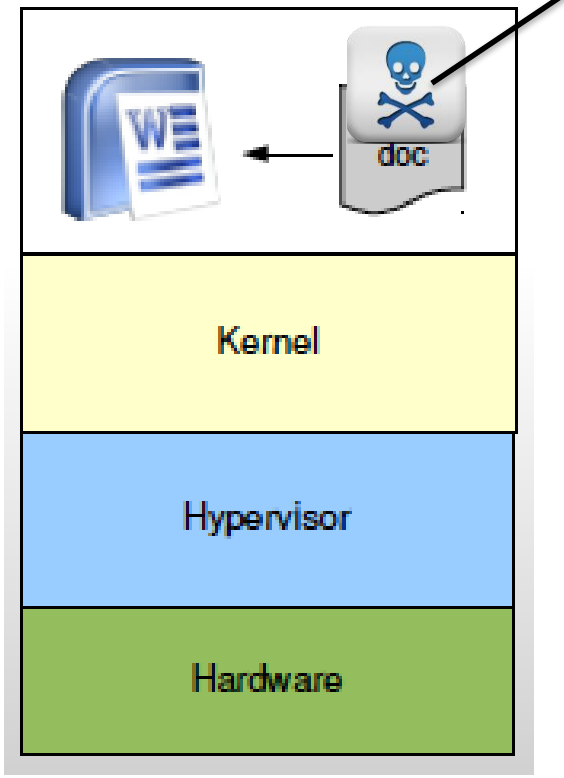
Where is it?



File infection

- Overwrite virus
  - Substitute the original program
- Parasitic virus
  - Append virus code before or after the program
  - Modify program entry point
- Cavity virus
  - Install inside of the file it is infecting
  - Usually in empty gaps inside the program binary

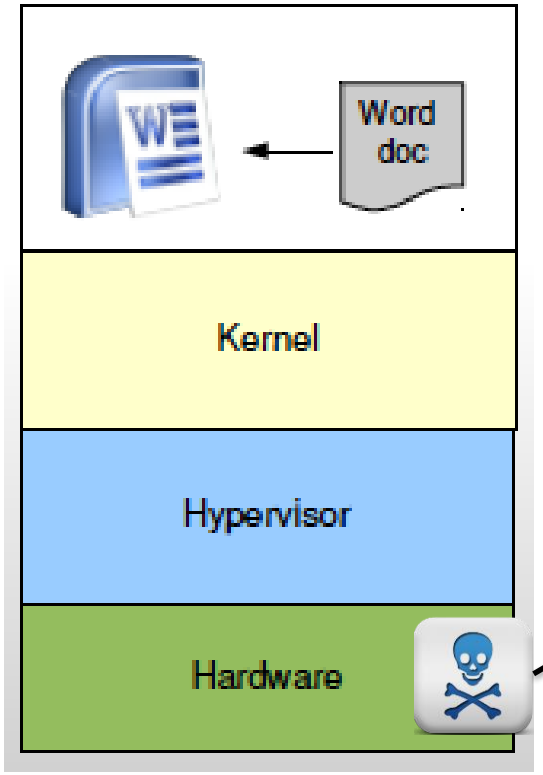
Where is it?



Macro infection

- Many modern applications support macro languages
  - Microsoft Word, Excel, Outlook
  - Macro languages are powerful
- Embedded macros automatically executed on load
- Infection through an exploit
- MS Word and Acrobat are common victims

Where is it?



Boot infection

- Master boot record (MBR) of hard disk (first sector on disk)
- Boot sector of partitions

Virus Defense

- Antivirus Software
  - Signature-based detection
    - Database of byte-level or instruction-level signatures that match virus
    - Wildcards and regular expression can be used
    - Hash of known malicious programs
  - Heuristics (check for signs of infection)
    - Code execution starts in last section
    - Incorrect header size in PE header
    - Suspicious code section name
    - Patched import address table
  - Behavioral signatures
- Sandboxing
  - Run untrusted applications in restricted environment

Fake Antivirus

Worms

- *“A self-replicating computer program that uses a network to send copies of itself to other nodes”*

- Distinguishing features:
  - Autonomous spread over network
  - It is self-contained (it does not need to be embedded in a file like a virus)
  - Speed of spreading may increase over time

Worm Propagation

- **Email harvesting**
  - Consult address books (W32/Melissa)
  - Files that might contain email addresses
    - Inbox of email client (W32/Mydoom)
    - Internet Explorer cache and personal directories (W32/Sircam)
- **Network share enumeration**
  - Windows discovers local computers, which can be attacked
  - Some worms attack everything, including network printers
- **Scanning**
  - Randomly generate IP addresses and send probes
  - Some worms use hit-list with known targets

Worms Mail Propagation

- Social engineering techniques to get executed
  - Fake *from* address
  - Promise interesting pictures or applications
  - Hide exec extension (.exe) behind harmless ones (.jpg)
- Attempt to hide from virus scanners
  - Zipped, sometimes even with password-encrypted (ask user to unpack)
- Exploit IE bugs when HTML content is rendered
- Speed of spread limited, humans are in the loop
  - Patterns correspond to time-of-day

Exploit-based Propagation

- Require no human interaction
  - Exploit well-known network services
  - Spread much faster
- Propagation speed limited either...
  - By network latency
    - Worm thread has to establish TCP connection (Code Red)
  - By bandwidth
    - Worm can send (UDP) packets as fast as possible (Slammer)
- Spread can be modeled using classic disease model
  - Worm starts slow (only few machines infected)
  - Enters phase of exponential growth
  - Final phase where only few uncompromised machines left

CodeRed (July 2001)

- Exploits known buffer overflow in IIS Idq.dll
- Propagates through an HTTP request (TCP)
- Malicious code stays in memory
- Target: DDoS www.whitehouse.gov
- Vulnerable population (360,000 servers) infected in 14 hours

Slammer (January 2003)

- Exploits a vulnerability in Microsoft SQL Server disclosed 7 months before
- Propagates through a single 380 byte UDP packet
- In 3 mins, the worm achieved its full scanning rate (more than 55 million scans per second)
- 75,000 victims infected within ten minutes
- Consequences
  - The worms packets saturated many Internet links
  - Routers collapsed
  - Problem in ATM and emergency numbers
  - 2 to 5 of the 13 root DNS shut down
  - Cancelled airline flights

Detecting and Measuring Propagation

- Observe and measure traffic targeting the dark (unused) address-space of the network
  - All incoming traffic in a darknet is suspicious by definition
- Manifestation of different events
  - Attackers probing blindly
  - Random scanning from worms
  - DoS

Worms Defense 1/2

- Virus scanners
  - Effective against email-based worms
- Host-level defense
  - Protecting software from remote exploitation
  - Stack protection techniques
  - Address Space Layout Randomization (ASLR)
  - Attempts to achieve diversity to increase protection
  - Not effective when users execute malware themselves

Worms Defense 2/2

- Network-level defense
  - Intrusion detection systems
    - Fast and automatic signature generation is active research and innovation area
  - Limit the number of outgoing connections
    - Helps to contain worms that perform scanning
  - Personal firewall
    - Block outgoing SMTP connections (from unknown applications)

Trojan Horses (TH)

- Class of malware that often appears to perform a desirable function but it also performs undisclosed malicious activities

- Distinguishing features:
  - Require the user to explicitly run the program
  - Unable to make copies of themselves or self-replicate
  - Can be used to perform any kind of malicious activity

Trojan Horses (TH)

- Two types of Trojan horses
  - Malicious functionality is included into useful program
    - Games, disk utility, screensaver..
    - Famous compiler that included backdoor into code
  - The malware is a stand-alone program
    - Possibly disguised file name
- Many different types and malicious functionalities
  - Spy on sensitive user data (spyware)
  - Disguise presence (rootkit)
  - Allow remote access (backdoor)
    - Base for further attacks, mail relay (for spammers)
  - Damage routines (corrupting files)

How do TH get in?

- Mainly through social engineering
- The user opens a suspicious mail attachment
- The user follows a link to an infected site
- The user downloads and executes an unknown binary

Spyware

- Software that secretly monitors the user's behavior and collects private information without the user knowledge

- Distinguishing features:
  - Threat to the user privacy
  - Hidden from the user and usually difficult to detect
  - Leak sensitive information (credit card numbers, visited webpages, passwords..) to someone else over the internet
  - Often implemented as a browser plugin/extension

Keyloggers

- Records key strokes typed by user
  - Runs as process in the background
- Typically used to capture interesting user data
  - Passwords, information for social engineering attacks (emails)
  - Successful independent of any application-level protection or network-level protection (e.g., encryption)
- Key loggers are quite easy to write
  - Easy to re-implement
  - Difficult to recognize by antivirus scanners

Rootkits

- Disguise the existence of a malicious program in a system and enable continued privilege access to it
- Distinguishing features:
  - Installed by an attacker after a system has been compromised
  - Hide the presence of the attacker and allows him to return on a later time
  - Difficult to detect

User-Space Rootkits

- Replace system programs with *trojaned* versions
  - System browsing (ls, find)
  - System logging (syslogd)
  - System monitoring (ps, top, netstat)
  - User authentication (login, sshd)

Real-Life Examples

[From “The Art of Intrusion”]

- Buffer overflow in BIND to get root on Lockheed Martin’s DNS server, install password sniffer
  - Sniffer logs stored in directory called /var/adm/ …
- Excite@Home employees connect via dialup; attacker installs remote access trojans on their machines via open network shares, sniffs IP addresses of promising targets
  - To bypass anti-virus scanners, use commercial remote-access software modified to make it invisible to user

Function Hooking

- Rootkit may “re-route” a legitimate system function to the address of malicious code
- Pointer hooking
  - Modify the pointer in OS’s Global Offset Table, where function addresses are stored
- “Detour” or “inline” hooking
  - Insert a jump in first few bytes of a legitimate function
  - This requires subverting memory protection
- Modifications may be detectable by a clever rootkit detector

Kernel Rootkits

- Modify kernel data structures to hide processes, files, and network activities

- Installed by
  - Kernel patch
  - Loadable Kernel Module
  - Kernel memory patching (/dev/kmem)

Rootkit Defense

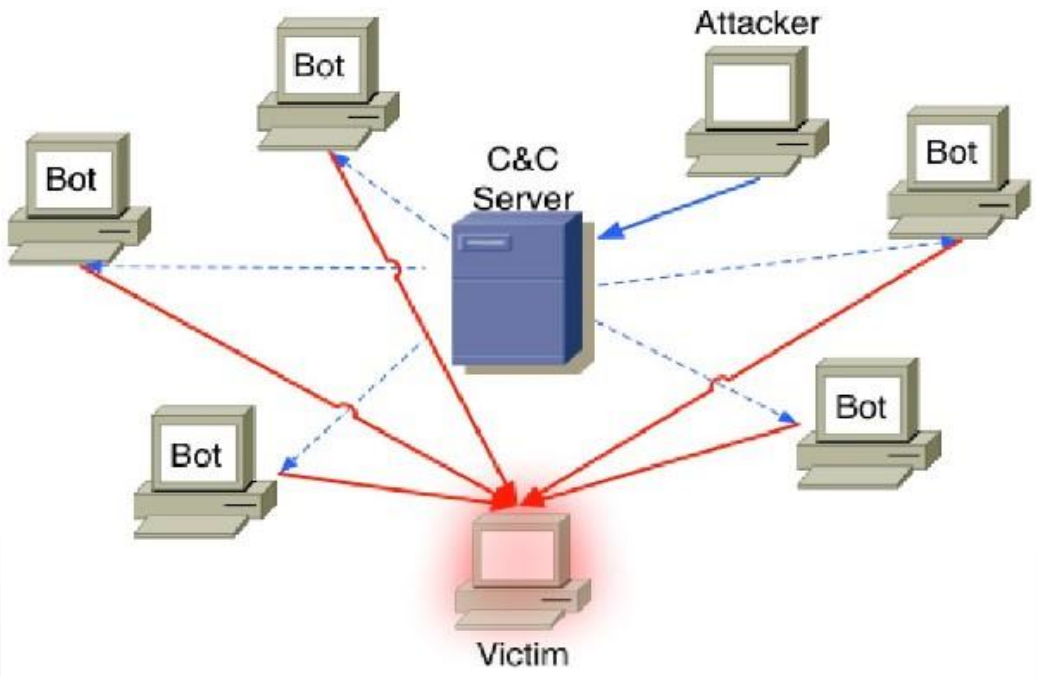
- User-space
  - Tripwire (user-space integrity checker)
  - Chkrootkit (user-space, signature-based detector)
- Kernel-space
  - KSTAT (Kernel Security Therapy Anti-Trolls)
    - Linux tool that uses several methods to check the integrity of the running kernel, e.g., *“sys_read 0xc2846868 Warning! Should be at 0x12699c”*
- Limitations
  - Kernel rootkits have complete control over operating system
  - All applications can be arbitrarily fooled (including rootkit detectors)

Bots and Botnets

- A bot (aka zombie or drone) is a compromised machine that can be controlled by an attacker remotely
  - Originally automated programs used to provide useful services on Internet Relay Chat (IRC) channels
- Distinguishing features:
  - Easy to control remotely
  - Implementation of different commands
  - Infected machines are incorporated in large networks (botnets) that are controlled by the botmaster
  - Loaded on a computer after compromise (e.g., by a trojan or a worm)

Botnets

- The attacker controls a Command and Control (C&C) server
- All the bots contact the C&C to report their status or receive commands



Common Commands

- Harvest email addresses from host
- Log all key presses
- Sniff network traffic
- Take screenshots
- Start an http server that allows to browse C:
- Kill a hard-coded list of processes (AV programs, rival malware)
- Steal windows CD keys
- Sets up a proxy to be used as a "stepping stone" for SPAM
- Download file at an url
- Run a shell command
- Update (allows to change the available commands!)

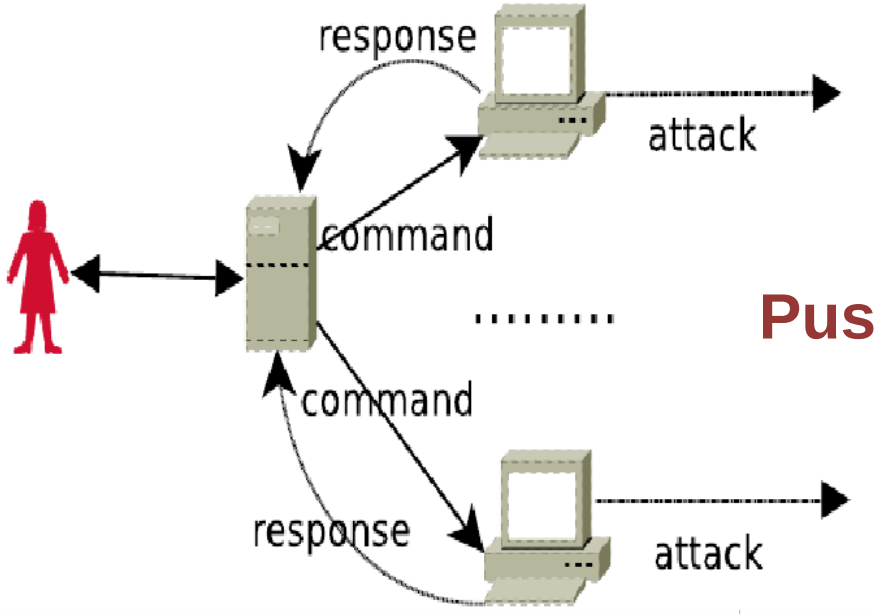
C&C Mechanisms

- The most distinguishing and powerful feature of botnets...
  - As long as there is an update command defined for the bots the botmaster can change the command set by updating her bots
  - C&C brings a great flexibility to the activities that can be performed by bots
- However, C&C is also the weakest link
  - Single point of “failure”
  - So real botnets don't have a single C&C server

Centralized C&C

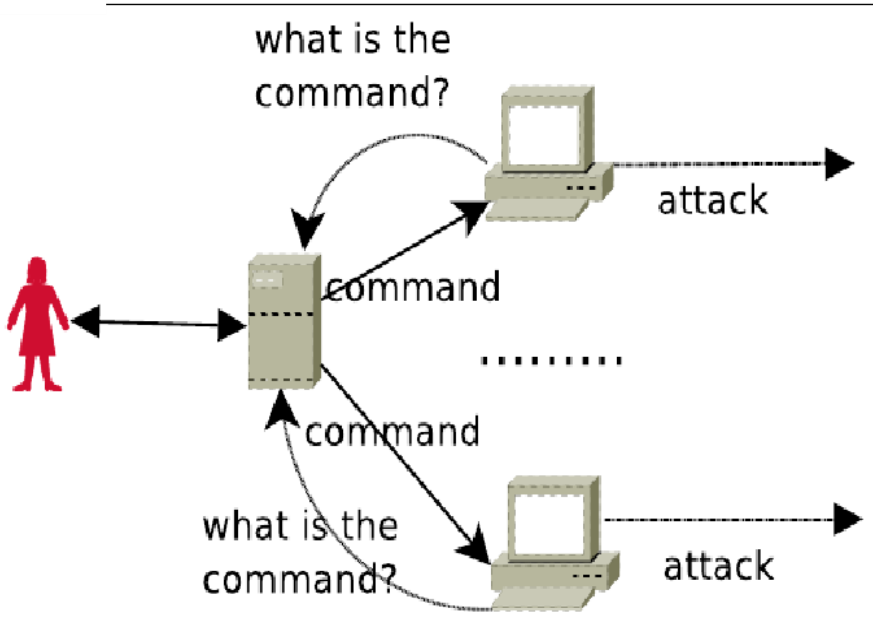
- Two approaches:
  - Push style (e.g. IRC)
  - Pull style (e.g. HTTP)
- Multiple C&C servers
  - Multiple layers of hosts between bot and botmaster
- Address of C&C server(s) must be available to each bot
  - In binary, config file, etc
  - Frequently updated
- Large botnets often partitioned into smaller ones
  - Each bot knows only a few C&C servers

Push vs. Pull



Push-style C&C

Pull-style C&C



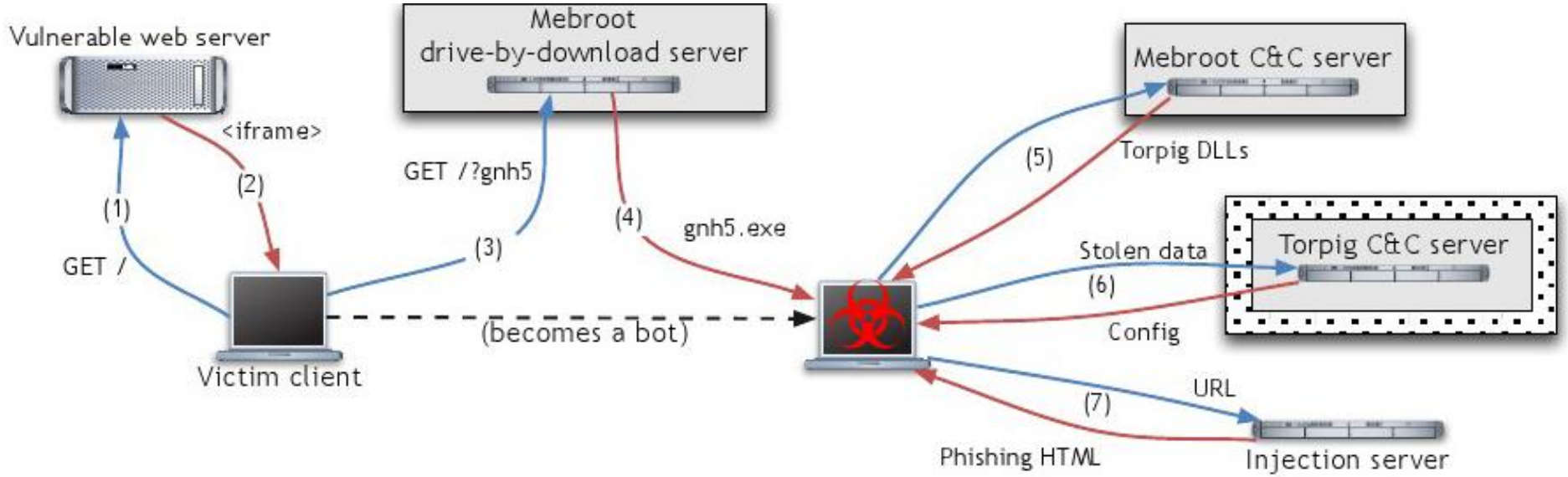
Peer-to-Peer C&C

- Bot commands propagate in a P2P network
- More robust
  - More difficult to catch botmaster
  - Even if some nodes are down, network continues its activities
- Custom protocol:
  - The new, unknown traffic can be “noisy”
  - A bot needs a way to find new neighbors
- Standard protocol:
  - Allows C&C traffic to blend in with legitimate P2P traffic
  - Legitimate P2P is blocked by many ISPs

Bullet-Proof Hosting

- Sometimes C&C is hosted by ISPs that are completely unresponsive to abuse complaints
  - Take down the entire ISP
- De-peering: other ISPs rescind their peering agreements with the malicious ISPs
  - Not easy, involves law enforcement
- Examples:
  - Russian Business Network (2007), Atrivio (2008), McColo (2008), 3FN (2009), Voze (2011), …

The Torpig Botnet



- Mebroot: rootkit that takes control of a machine by replacing the system's Master Boot Record (MBR)
  - Executed at boot time, undetected by most anti-virus
  - Has no malicious capability, but provides a generic platform to manage (install, uninstall, and activate) other malicious modules
    - E.g., Torpig botnet

Hijacking the Torpig Botnet

- Torpig uses "domain flux" to locate its C&C servers
  - Each bot uses an algorithm to compute a list of domain names
  - This list is computed independently by each bot and is regenerated periodically
  - Each bot attempts to contact hosts in the list until one succeeds
- Researchers at UCSB reverse-engineered the algorithm
  - They predict the C&C domain that was going to be used later
  - Registered the domain, took control of the botnet
- Data collected
  - 70GB from 52,540 different infected machines
  - 297,962 unique username/password
    - 8,310 for bank accounts
  - 1,235,122 windows password
  - 1,258,862 email accounts

Bots Defense

- Attack C&C infrastructure
  - Take IRC channel off-line
  - When dynamic DNS is used for central command server, route traffic to black hole
- Honeypots
  - Vulnerable computer that serves no purpose other than to attract attackers and study their behavior in controlled environments
  - When honeypot is compromised, bot logs into botnet
  - Allows defender to study actions of botnet owners

Stuxnet 1/2

- Malware targeting Iranian Nuclear centrifuges
- Spreads by
  - Copying itself to USB drives
  - Network shares
  - Using two known and four 0-day Microsoft vulnerabilities
- Windows rootkit to hide Windows binaries
  - Signed by one of 2 stolen certificates from ‘JMicron’ and ‘Realtek’

Stuxnet 2/2

- Injects STL code into Progammable Logic Controllers
  - Uses rootkit techniques to hide injected PLC code
  - Patches Siemens Step 7 software (used to view PLC code)
- Communicates with C&C servers using HTTP
  - www.mypremierfutbol.com
  - www.todaysfutbol.com
- Steals designs documents for industrial control systems
- Sabotages targeted industrial control systems

Other types of malicious code

- Rabbit: code that replicates itself w/o limit to exhaust resources
- Logic (time) bomb: code that triggers action when condition (time) occurs
- Dropper: code that drops other malicious code
- Trapdoor/backdoor: code feature that allows unauthorized access to machine or program
- Tool/toolkit: program used to assemble malicious code (not malicious itself)
- Scareware: false warning of malicious code attack

Summary of vulnerabilities

- Voluntary introduction
  - Malicious code introduced by user or admin
- Unlimited privilege
  - Users (or programs) have often root access
- Stealthy behavior
  - Very hard to detect/characterize

Summary of countermeasures

- “Hygiene”: avoiding points of contamination and blocking avenues of vulnerability
- Detection tools
- Basic security principles
  - Memory separation
  - Least privilege
- Trusted Computing