Malware:
Viruses, Worms, Rootkits, Botnets

Spring 2015

Franziska (Franzi) Roesner
franzi@cs.washington.edu

Thanks to Dan Boneh, Dieter Gollmann, Dan Halperin, Yoshi Kohno, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials...
Looking Ahead

• Today: Malware
• Next time(s): Mobile platform security and privacy
• Lab 2 due Wednesday, May 20 (5pm)

• Two guest lectures next week
  – **Wednesday:** Emily McReynolds (UW Tech Policy Lab)
    • Bring your law-related questions!
  – **Friday:** Peter Ney and Ian Smith on hacking tools etc.

• The following Monday is a holiday 😊
Malware

• Malicious code often masquerades as good software or attaches itself to good software
• Some malicious programs need host programs
  – Trojan horses (malicious code hidden in useful program)
• Others can exist and propagate independently
  – Worms, automated viruses
• Many infection vectors and propagation methods
• Modern malware often combines techniques
Viruses

• Virus propagates by infecting other programs
  – Automatically creates copies of itself, but to propagate, a human has to run an infected program
  – Self-propagating viruses are often called worms

• Many propagation methods
  – Insert a copy into every executable (.COM, .EXE)
  – Insert a copy into boot sectors of disks
    • PC era: “Stoned” virus infected PCs booted from infected floppy disks, stayed in memory, infected every inserted floppy
  – Infect common OS routines, stay in memory
First Virus: Creeper

- Written in 1971 at BBN
- Infected DEC PDP-10 machines running TENEX OS
- Jumped from machine to machine over ARPANET
  - Copied its state over, tried to delete old copy
- Payload: displayed a message
  “I’m the creeper, catch me if you can!”
- Later, Reaper was written to delete Creeper

http://history-computer.com/Internet/Maturing/Thomas.html
Virus Detection

• Simple anti-virus scanners
  – Look for signatures (fragments of known virus code)
  – Heuristics for recognizing code associated with viruses
    • Example: polymorphic viruses often use decryption loops
  – Integrity checking to detect file modifications
    • Keep track of file sizes, checksums, keyed HMACs of contents
Arms Race: Polymorphic Viruses

- **Encrypted viruses**: constant decryptor followed by the encrypted virus body
- **Polymorphic viruses**: each copy creates a new random encryption of the same virus body
  - Decryptor code constant and can be detected
  - Historical note: “Crypto” virus decrypted its body by brute-force key search to avoid explicit decryptor code
Smarter Virus Detection?

• Generic decryption and emulation
  – Emulate CPU execution for a few hundred instructions, recognize known virus body after it has been decrypted
  – Does not work very well against viruses with mutating bodies and viruses not located near beginning of infected executable
To detect an unknown mutation of a known virus, emulate CPU execution of until the current sequence of instruction opcodes matches the known sequence for virus body.
Arms Race: Metamorphic Viruses

• Obvious next step: *mutate the virus body*, too
• Apparition: an early Win32 metamorphic virus
  – Carries its source code (contains useless junk)
  – Looks for compiler on infected machine
  – Changes junk in its source and recompiles itself
  – New binary copy looks different!
• Mutation is common in macro and script viruses
  – A macro is an executable program embedded in a word processing document (MS Word) or spreadsheet (Excel)
  – Macros and scripts are usually interpreted, not compiled
Mutation Techniques

- Real Permutating Engine/RPME, ADMutate, etc.
- Large arsenal of obfuscation techniques
  - Instructions reordered, branch conditions reversed, different register names, different subroutine order
  - Jumps and NOPs inserted in random places
  - Garbage opcodes inserted in unreachable code areas
  - Instruction sequences replaced with other instructions that have the same effect, but different opcodes
    - Mutate SUB EAX, EAX into XOR EAX, EAX or MOV EBP, ESP into PUSH ESP; POP EBP

- There is no constant, recognizable virus body
Example of Zperm Mutation

[From Szor and Ferrie, “Hunting for Metamorphic”]
Obfuscation and Anti-Debugging

• Common in all kinds of malware
• Goal: prevent code analysis and signature-based detection, foil reverse-engineering
• Code obfuscation and mutation
  – Packed binaries, hard-to-analyze code structures
  – Different code in each copy of the virus
    • Effect of code execution is the same, but this is difficult to detect by passive/static analysis (undecidable problem)
• Detect debuggers and virtual machines, terminate execution
Drive-By Downloads

• Websites “push” malicious executables to user’s browser with inline JavaScript or pop-up windows
  – Naïve user may click “Yes” in the dialog box

• Can install malicious software automatically by exploiting bugs in the user’s browser
  – 1.5% of URLs - Moshchuk et al. study
  – 5.3% of URLs - “Ghost Turns Zombie”
  – 1.3% of Google queries - “All Your IFRAMEs Point to Us”

• Many infectious sites exist only for a short time, behave non-deterministically, change often
Obfuscated JavaScript

document.write(unescape("%3CHEAD%3E%0D%0A%3CSCRIPT%20LANGUAGE%3D%22Javascript%22%3E%0D%0A%21--%0D%0A/*criptografado pelo Fal - Deboaca para seu site renda extra%0D...
%20gr%Eltis%20para%20seu%20site%20renda%20extra%0D
...
3C/SCRIPT%3E%0D%0A%3C/HEAD%3E%0D%0A%3CBODY%3E%0D%0A%3C/BODY%3E%0D%0A%3C/HTML%3E%0D%0A"));
//-->
</SCRIPT>
Viruses vs. Worms

VIRUS
• Propagates by infecting other programs
• Usually inserted into host code (not a standalone program)

WORM
• Propagates automatically by copying itself to target systems
• A standalone program
**1988 Morris Worm (Redux)**

- No malicious payload, but bogged down infected machines by uncontrolled spawning
  - Infected 10% of all Internet hosts at the time
- Multiple propagation vectors
  - Remote execution using rsh and cracked passwords
    - Tried to crack passwords using a small dictionary and publicly readable password file; targeted hosts from `/etc/hosts.equiv`
  - Buffer overflow in fingerd on VAX
    - Standard stack smashing exploit
  - DEBUG command in Sendmail
    - In early Sendmail, can execute a command on a remote machine by sending an SMTP (mail transfer) message
Slammer (Sapphire) Worm

• January 24/25, 2003: UDP worm exploiting buffer overflow in Microsoft’s SQL Server (port 1434)
  – Overflow was already known and patched by Microsoft... but not everybody installed the patch
• Entire code fits into a single 404-byte UDP packet
  – Worm binary followed by overflow pointer back to itself
• Classic stack smash combined with random scanning
  – Once control is passed to worm code, it randomly generates IP addresses and sends a copy of itself to port 1434
Slammer Propagation

• **Scan rate** of 55,000,000 addresses per second
  – Scan rate = the rate at which worm generates IP addresses of potential targets
  – Up to 30,000 single-packet worm copies per second

• Initial infection was doubling in 8.5 seconds (!!)
  – Doubling time of Code Red (2001) was 37 minutes

• Worm-generated packets **saturated carrying capacity** of the Internet in 10 minutes
  – 75,000 SQL servers compromised
  – ... in spite of the broken pseudo-random number generator used for IP address generation
05:29:00 UTC, January 25, 2003

[from Moore et al. “The Spread of the Sapphire/Slammer Worm”]
30 Minutes Later

[from Moore et al. “The Spread of the Sapphire/Slammer Worm”]

Size of circles is **logarithmic** in the number of infected machines
Impact of Slammer

• $1.25 Billion of damage
• Temporarily knocked out many elements of critical infrastructure
  – Bank of America ATM network
  – Entire cell phone network in South Korea
  – Five root DNS servers
  – Continental Airlines’ ticket processing software
• The worm did not even have malicious payload... simply bandwidth exhaustion on the network and CPU exhaustion on infected machines
Slammer Aftermath

• Slammer packets were ubiquitous in the Internet for many years after 2003
  – Could be used as a test for Internet connectivity 😊
  – Packets provided a map of vulnerable machines
• Vanished on March 10-11, 2011

Evidence of a clock-based shutoff trigger
Rootkits

- **Rootkit** is a set of trojan system binaries
- **Main characteristic: stealthiness**
  - Create a hidden directory
    - `/dev/.lib`, `/usr/src/.poop` and similar
    - Often use invisible characters in directory name
  - Install hacked binaries for system programs such as `netstat`, `ps`, `ls`, `du`, `login`
  - Modified binaries have same checksum as originals
    - Should use cryptographically secure hash function instead

Can’t detect attacker’s processes, files or network connections by running standard UNIX commands!
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
Example: Attack on RSA

• Successful attack on a big US security company
• Target: master keys for two-factor authentication
• Spear-phishing email messages
  – Subject line: “2011 Recruitment Plan”
  – Attachment: 2011 Recruitment plan.xls
• Spreadsheet exploits zero-day vulnerability in Adobe Flash to install Remote Administration Software (RAT)
  – Reverse-connect: pulls commands from C&C servers
  – Stolen data moved to compromised servers at a hosting provider, then pulled from there and traces erased

Detecting a Rootkit

• Sad way to find out
  – Run out of physical disk space because of sniffer logs
  – Logs are invisible because du and ls have been hacked

• Manual confirmation
  – Reinstall clean ps and see what processes are running

• Automatic detection
  – Host-based intrusion detection can find rootkit files
    • ...assuming an updated version of rootkit did not disable the intrusion detection system!
Botnets

• **Botnet** is a network of autonomous programs capable of acting on instructions
  – Typically a large (up to several hundred thousand) group of remotely controlled “zombie” systems
    • Machine owners are not aware they have been compromised
  – Controlled and upgraded from command-and-control (C&C) servers

• Used as a platform for various attacks
  – Distributed denial of service, Spam and click fraud
  – Launching pad for new exploits/worms
Bot History

- Eggdrop (1993): early IRC bot
- DDoS bots (late 90s): Trin00, TFN, Stacheldracht
- RATs / Remote Administration Trojans (late 90s):
  - Variants of Back Orifice, NetBus, SubSeven, Bionet
  - Include rootkit functionality
- IRC bots (mid-2000s)
  - Active spreading, multiple propagation vectors
  - Include worm and trojan functionality
  - Many mutations and morphs of the same codebase
- Stormbot and Conficker (2007-09)
Life Cycle of an IRC Bot

- Exploit a vulnerability to execute a short program (shellcode) on victim’s machine
  - Buffer overflows, email viruses, etc.
- Shellcode downloads and installs the actual bot
- Bot disables firewall and antivirus software
- Bot locates IRC server, connects, joins channel
  - Typically need DNS to find out server’s IP address
    - Especially if server’s original IP address has been blacklisted
  - Password-based and crypto authentication
- Botmaster issues authenticated commands
Command and Control

(12:59:27pm) -- A9-pcgbdv (A9-pcgbdv@140.134.36.124) has joined (#owned) Users : 1646

(12:59:27pm) (@Attacker) .ddos.synflood 216.209.82.62

(12:59:27pm) -- A6-bpxufrd (A6-bpxufrd@wp95-81.introweb.nl) has joined (#owned) Users : 1647

(12:59:27pm) -- A9-nzmpah (A9-nzmpah@140.122.200.221) has left IRC (Connection reset by peer)

(12:59:28pm) (@Attacker) .scan.enable DCOM

(12:59:28pm) -- A9-tzrkeasv (A9-tzrkeasv@220.89.66.93) has joined (#owned) Users : 1650
Detecting Botnet Activity

• Many bots are controlled via IRC and DNS
  – IRC used to issue commands to zombies
  – DNS used by zombies to find the master, and by the master to find if a zombie has been blacklisted

• IRC/DNS activity is very visible in the network
  – Look for hosts performing scans and for IRC channels with a high percentage of such hosts
  – Look for hosts who ask many DNS queries but receive few queries about themselves

• Easily evaded by using encryption and P2P 😞
What to Do With a Botnet?

• Denial of service (including cyber-warfare)
• Spam
• Fake antivirus sales, Ransomware
• Advertising clickfraud
• Bitcoin mining
  – According to Symantec, one compromised machine yields 41 US cents a year...
CryptoLocker

Your personal files are encrypted!

Your important files encryption produced on this computer: photos, videos, documents, etc. Here is a complete list of encrypted files, and you can personally verify this.

Encryption was produced using a unique public key RSA-2048 generated for this computer. To decrypt files you need to obtain the private key.

The single copy of the private key, which will allow you to decrypt the files, located on a secret server on the Internet; the server will destroy the key after a time specified in this window. After that, nobody and never will be able to restore files...

To obtain the private key for this computer, which will automatically decrypt files, you need to pay 300 USD / 300 EUR / similar amount in another currency.

Click «Next» to select the method of payment and the currency.

Any attempt to remove or damage this software will lead to the immediate destruction of the private key by server.
Denial of Service (DoS)

- **Goal:** overwhelm victim machine and deny service to its legitimate clients
- DoS often exploits networking protocols
  - **Smurf:** ICMP echo request to broadcast address with spoofed victim’s address as source
  - **SYN flood:** send lots of “open TCP connection” requests with spoofed source addresses
  - **UDP flood:** exhaust bandwidth by sending thousands of bogus UDP packets
  - **HTTP request flood:** flood server with legitimate-looking requests for Web content
Distributed Denial of Service (DDoS)

Attacker

Master machines

Zombie machines

Victim
How to Protect Yourself?

• Nothing is perfect but...
  – Keep your software updated
  – Be vigilant for phishing attacks
  – Anti-virus
  – Firewalls
  – Intrusion detection systems