CSC 482/582: Computer Security

Malware
Topics

1. Malware detection
2. Theory of malware
3. Viruses
4. Worms
5. Rootkits
6. Malware handling process
7. Malware factories
8. Botnets
Malware, short for malicious software, is software designed to gain access to confidential information, disrupt computer operations, and/or gain access to private computer systems. Malware can be classified by how it infects systems:

- Trojan Horses
- Viruses
- Worms

Or by what assets it targets:

- Ransomware
- Information stealers
- Spyware and adware
- Backdoors
- Rootkits
- Botnets
How much malware is out there?

New unique samples added to AV-TEST's malware repository (2000-2012)
Static Malware Detection

Signature-based
- Look for known patterns of bytes in malicious code.
- Defeated by polymorphic/metamorphic viruses.

Decryption
- Brute-forces simple XOR-based encryption.
- Checks decrypted text against small virus signature set to decide whether has plaintext or not.

Heuristics
- Set of signatures that can match in any order, with stretches of metamorphic or legitimate code between.
- List of suspicious coding techniques (a type of signature).
- Software scored on number of signatures matched, weighted by the importance of each one.
Sandbox

Isolated VMs for dynamic malware detection.

- Execute potential malware on VM.
- Scan VM after a certain amount of time.
- Examine memory, filesystem, network.
### VirusTotal Analysis

**SHA256:** 4a1e0d954831d6f2b65713bcbdb6afa22e9b0c9b102b6e6b516088f400415e114ac5

**File name:** 02052

**Detection ratio:** 42 / 47

**Analysis date:** 2013-06-16 20:35:42 UTC (3 months, 4 weeks ago)

---

<table>
<thead>
<tr>
<th>Antivirus</th>
<th>Result</th>
<th>Update</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agnitum</td>
<td>Trojan.Yakas!CJkhhv$Jlko</td>
<td>20130616</td>
</tr>
<tr>
<td>AhnLab-V3</td>
<td>Win32/Hamweq.worm.18344.B</td>
<td>20130616</td>
</tr>
<tr>
<td>AntiVir</td>
<td>Worm/Hamweq.A.326</td>
<td>20130616</td>
</tr>
<tr>
<td>Antiy-AVL</td>
<td>Trojan/Win32.Yakas.gen</td>
<td>20130616</td>
</tr>
<tr>
<td>Avast</td>
<td>Win32.Kryptik-kt [Tr]</td>
<td>20130616</td>
</tr>
<tr>
<td>AVG</td>
<td></td>
<td>20130616</td>
</tr>
<tr>
<td>BitDefender</td>
<td>Trojan.Generic.KDV.602208</td>
<td>20130616</td>
</tr>
<tr>
<td>ByteHero</td>
<td>Trojan.Malware.Obscu.Gen.002</td>
<td>20130613</td>
</tr>
<tr>
<td>CAT-QuickHeal</td>
<td>Worm.Dorkbot.A</td>
<td>20130616</td>
</tr>
<tr>
<td>ClamAV</td>
<td>Win.Trojan.Yakes-763</td>
<td>20130616</td>
</tr>
<tr>
<td>Commtouch</td>
<td>W32/Trojan.YEPD-1711</td>
<td>20130616</td>
</tr>
</tbody>
</table>
Theorem 1: It is undecidable whether an arbitrary program contains a computer virus.

Proof:
Define virus $v$ as TM program that copies $v$ to other parts of the tape, while not overwriting any part of $v$.
Reduce to Halting Problem: $T'$ running code $V'$ reproduces $V$ iff running $T$ on $V$ halts.

Theorem 2: It is undecidable whether an arbitrary program contains malicious logic.
A computer virus is a type of malware that, when executed, replicates by inserting copies of itself (possibly modified) into other files. This process is called infecting.
Types of Viruses

1. Boot Sector
   - When system boots, code in boot sector executed.
   - Propagate by altering boot disk creation.
   - Uncommon today because of low use of boot floppies, but some Vista laptops shipped with one.

2. Executable
   - Infects executable programs (e.g., COM, EXE).
   - Executes when infected program is run.
   - Virus usually runs first, then runs original code.

3. Dynamic Library
   - Infected dynamically linked libraries (DLLs.)
   - Executed when any program uses infected DLL.
Types of Viruses

4. Device Driver
   - Infects loadable device driver.
   - Executes in kernel mode.

5. Virtual Machine (.NET)
   - Infects .NET MSIL binaries.
   - Portable: compiled to native code by CLR.

6. Archive Infectors
   - Inserts Trojan Horse into ZIP files.
   - Uses social engineering techniques to get user to run.
Types of Viruses

7. **Macro Virus**
   - Infects embedded interpreted code.
   - Needs interpreter like sh, MS Word macro.
   - Can infect executables or data files
     - Executables must invoke appropriate interpreter.
   - Most modern data formats support some type of scripting, including
     - Microsoft Office
     - Windows Help files
     - HTML: VBScript, JScript
Infection Methods

1. Overwriting
   - Overwrites section of program file with virus.
   - May break infected program.

2. Companion
   - Infects COM file of same name as EXE file.
   - Given a name, Windows runs COM before EXE.
   - Infects alternate data stream of Win32 file.

---

**Legitimate application**  
File: file.exe

**Virus:**
- `infect()`
- `if trigger`
- `payload()`
- `run file.exe`

**File:**
- file.exe
- file.com
3. Prepending

- Insert virus code at beginning of executable.
- Shift original code to follow virus.
4. Appending

- Append virus code to executable.
- Insert JMP at beginning of executable.
Infection Methods: Fragmenting

5. Fragmenting
   - Append virus code to executable.
   - Insert JMP at beginning of executable.
Worms

Copies self from one computer to another
Self-replicating: No user action required unlike virus or Trojan horse programs.
Spreads via network protocols
ex: SMTP (email), fingerd, MS SQL
## Timeline of Notable Worms

<table>
<thead>
<tr>
<th>Year</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1988</td>
<td>Morris Worm</td>
<td>Spreads across entire Internet, causing service disruptions. Vectors: buffer overflow, backdoors, passwords, and other authentication vulnerabilities.</td>
</tr>
<tr>
<td>1999</td>
<td>Melissa</td>
<td>Macro virus that spread via e-mail to all people in address book, shutting down many e-mail servers.</td>
</tr>
<tr>
<td>2000</td>
<td>ILOVEYOU</td>
<td>VBScript Trojan spread via e-mail to tens of millions of Windows PCs.</td>
</tr>
<tr>
<td>2001</td>
<td>Code Red</td>
<td>Spread to millions of hosts, defaced web sites, and launched DDoS. Vector: buffer overflow in IIS.</td>
</tr>
<tr>
<td>2001</td>
<td>Nimda</td>
<td>Vectors: multiple, including backdoors left by Sadmind and Code Red II worms.</td>
</tr>
<tr>
<td>2003</td>
<td>Slammer</td>
<td>Fastest spreading worm, infecting all vulnerable MS SQL Server installations in &lt;1hr. Vector: buffer overflow</td>
</tr>
<tr>
<td>2003</td>
<td>Nachi/Welchia</td>
<td>Exploited vulnerability in MS RPC service to download MS security patches and remove Blaster worm.</td>
</tr>
</tbody>
</table>
## Timeline of Notable Worms

<table>
<thead>
<tr>
<th>Year</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2004</td>
<td>Cabir</td>
<td>First worm to target mobile phones (Symbian OS), spreading via Bluetooth to nearby phones.</td>
</tr>
<tr>
<td>2004</td>
<td>Santy</td>
<td>First worm to target web applications, exploiting a vulnerability in phpBB and using Google searches to find targets to infect.</td>
</tr>
<tr>
<td>2007</td>
<td>Storm</td>
<td>E-mailed Trojan horse that recruited machines into the Storm botnet, infecting millions of hosts.</td>
</tr>
<tr>
<td>2008</td>
<td>Conficker</td>
<td>Exploited buffer overflows in MS RPC services and poor passwords to recruit ~10 million hosts into botnet.</td>
</tr>
<tr>
<td>2010</td>
<td>Stuxnet</td>
<td>Multi-vector malware designed to target SCADA facilities. Designed by US to target Iranian nuclear facilities.</td>
</tr>
<tr>
<td>2012</td>
<td>Flame</td>
<td>Multi-vector malware designed for cyberespionage that targets Middle Eastern countries.</td>
</tr>
</tbody>
</table>
Worm Components

1. Vector
2. Propagation Engine
3. Target Selection
4. Scanning Engine
5. Payload
Exploit code to gain access to target host.

Common vectors:

- Buffer overflow exploits.
- Network file sharing, both NFS/SMB and P2P.
- Social-engineering via email or IM.
- Weak passwords.
- Parasitism: target backdoors and worm flaws.
Propagation Engine

Transfers worm to host exploited by vector.
- Small worms like Slammer included in vector.

Worm Propagation Methods:
- FTP
- HTTP
- SMB
- TFTP
Remote Control Interface

RCI allows creator to control infected hosts.
Many worms do not have a RCI.
May be a well-known backdoor program.

Common remote control features:
Start/stop infecting new targets.
Download new vectors.
Download new target selectors.
Download new payloads.
Target Selection

Selecting targets for potential infection.

E-mail address harvesting
- Address books.
- Parse disk files.
- Search news groups.

Network share enumeration
- Check for filesystems shared with other systems.

Network scanning
- Target hosts on current network and connected nets.
- Randomized scanning of Internet space.

Web searching
- Search Google for addresses or vulnerable software.
Scanning Engine

Check targets for vulnerabilities.
- If vector small, scanning can be skipped.

Scan for vulnerable services.
- Like targeted nmap port scan.

OS Check
- Check for correct OS for vector to work.

Version checking.
- Check version of target software.
- May customize vector based on information.
Morris Worm

- First Internet Worm: November 1988
- Claimed purpose: Mapping the Internet
- Multi-architecture: Sun, VAX
- Multi-vector
  - sendmail (debug backdoor)
  - fingerd (buffer overflow)
  - rsh (open .rhosts; password cracking)
Morris Worm

Spreading algorithm
- Local network topology: gateways, neighbors.
- Used users’ `.rhosts`, `.forward` files.
- Limited reinfection rate to $1/7$.

Detection Avoidance
- Forged process listing as `(sh)`.
- Removed created files quickly after use.
Morris Worm

Resource Requirements

- Disk Space.
- C compiler and linker.
- Network connection to parent computer.

Problems

- Didn’t limit re-infections.
- Saturated CPU, network resources.
Fast Worms

Slammer Worm Characteristics

- Attacked MS SQL servers.
- Worm is single 404-byte UDP packet.
- Random-scan (PRNG bugs limited.)
- Limited by network bandwidth, not latency.
- Observed scan rate of 26,000 hosts/second.
- Infected 90% of vulnerable hosts in 10 min.
- Too fast for humans to react.
- Shutdown 13,000 Bank of America ATMs due to compromising db servers, heavy traffic.
Counter-worms

Worm that removes other worms from net.

Nachi/Welchia

- Multi-vector W32 worm
- Nachi.A removes W32/Blaster worm
- Nachi.B removes W32/MyDoom worm
- Installed MSRPC DCOM patch to prevent future infections from Blaster.

Side-effects

- Infected Diebold ATMs
- Worm traffic DOSed Internet, esp Microsoft.
Unintentional Offline Impact

- Slammer infected nuclear power plant
  - Reached Plant Process Computer and Safety Parameter Display System via contractor network that supposedly not connected to internal plant network.
- Slammer also disabled Seattle’s 911 system
- Blaster infected First Energy systems
  - Contributing cause to 2003 blackout of northeast US.
- Sasser worm disabled many systems
  - Lund University Hospital X-ray department disabled,
  - Delta Air Lines cancelled flights,
  - British Coastguard lost access to maps, etc.
A rootkit is a type of software used to hide files, processes, network connections.

Much modern malware uses rootkit techniques to hide.

Rootkits can be detected by examining filesystem when OS is not running.

Rootkit detectors can find some rootkits while running by looking for modifications of OS data structures.
Malware Handling Process

1. Static Analysis (for signatures)
2. Dynamic Analysis (for signatures)
3. Reverse Engineering (if needed)
4. Signature Creation
5. Quality Assurance
6. Signature Deployment
**Static Analysis**

**Static analysis** includes any approach to analyzing a program without running it.

- File type
- Strings found in file
- Assembly code for program

Safest form of analysis, as malware is never executed.
Dynamic analysis includes any approach to analyzing a program while running it.

- Malware sandbox (to run malware safely).
- Host analysis tools (local changes).
- Network analysis tools (network traffic).
Reverse engineering is the use of static analysis (decompilers, disassemblers) and dynamic analysis (debuggers) techniques to determine precisely how a program works.
Static Malware Protection

1. Encryption.
   - Virus starts with small decryption algorithm.
   - Remainder of virus is encrypted, typically XOR with key.
   - Often encrypts different parts of virus with different encryption techniques or keys.

2. Polymorphism
   - Changes encryption key or technique each infection.
   - Dynamic scan can wait till malware decrypts self in RAM.

3. Metamorphism
   - Rewrites decryption code using different machine instructions each infection too.
   - Too many variants for signature detection
1. **Anti-sandboxing techniques.**
   - Researchers use VMs to analyze malware safely, so
   - If detect VM, then disable or delete self.

2. **Anti-debugging tricks.**
   - Researchers use debuggers to analyze software activity.
   - Detect some debuggers and break debugging session.
   - Write code in such a way to make it difficult to debug.

3. **Anti-detection techniques.**
   - Disable AV software on system.
   - Obfuscate code.
Malware factories are software and processes to automate the production and protection of malware, allowing threats to produce thousands of different malware from an original program, all of which are well protected.
Malware Factory Process

1. Mutate malware code (metamorphism).
2. Encrypt malware (polymorphism).
4. Bind malware to Trojan cover file.
5. Armor malware (anti-sandbox, debug, detect)
6. Quality assurance

Upload to vscan.novirusthanks.org to verify no current AV tool can detect it.
**Runtime Packers**

*Runtime packers* compress the code of programs better than general purpose compression tools and automatically decompress the program upon execution. Today they are primarily used to prevent analysis of programs.

- Thousands of packers exist.
- Can pack code multiple times with different packers.
- AV can unpack some formats, but not all.
EXE Binding

An **EXE binder** takes multiple programs and merges them into a single executable file that performs the functions of all of them. It is primarily used to bind malware into legitimate programs to make Trojan horses.
A botnet is

- A network of compromised machines
- that can be controlled remotely
- used for malicious activities.
Botnet Components

Host Component
- Malware running on victim’s computer.
- Receives commands from botmaster.
- Executes attacks.
- Sends data to botmaster.

Network Components
- Command & Control (C&C) servers.
- Malware distribution servers.
- Drop zone (for data exfiltration).
C&C Structure

Centralized
- All bots connect to a single C&C server.
- Single point of failure that can take down botnet.

Decentralized (peer-to-peer)
- Every bot is also a C&C server.
- Botmaster can administrate from any bot.

Hybrid
- Every bot is a C&C server.
- Also have separate C&C server.
Botnet Applications

1. Distributed Denial of Server (DDoS)
2. Click fraud
3. Spam replay
4. Pay-per-install agent
5. Large-scale information harvesting
6. Information processing
Stopping Botnets

1. Identify C&C server(s) and take down.
2. Block C&C server DNS name.
3. Block C&C server IP address.
4. Update anti-virus to identify host component and remove (difficult to reach every infected host.)
5. Identify botmaster(s) and work with law enforcement to stop them.
Protecting C&C

1. Bulletproof Hosting
   1. ISP that permits criminal activity, spamming, &c.

2. Dynamic DNS
   1. C&C changes IP addresses every few minutes.
   2. Uses DDNS service to point to changing IPs.

3. Fast Fluxing
   1. DNS requests resolve to set of flux agent IPs.
   2. Flux agents redirect traffic to C&C server.
Double Flux Networks

1. Single flux change DNS name to IP mapping.
2. Double flux change DNS servers for domain too.
Domain Generation Algorithms

Generate domain names for C&C servers
- Algorithmic, but hard to predict.
- Generate hundreds to thousands per day.

Botmaster knows DGA domains
- Sets up C&C at one of the domains.
- Waits for bots to contact & request commands.

Defends against C&C takedown
- Too many domains to blacklist.
- Generated domains are expendable, so takedowns don’t matter as botmaster has moved on to new ones.
- Botmaster can register C&C domains in future, so they’re not available to takedown on infection.
DGA Disadvantages

1. DGA bots produce much network traffic to nonexistent domains, returning errors.
2. If DGA is reverse engineered, someone can take control of botnet by registering domain name of a future C&C server.
Key Points

1. Malware Handling Process
   Static Analysis (for signatures), Dynamic Analysis (for signatures), Reverse Engineering (if needed), Signature Creation, QA, Signature Deployment
2. Rootkit techniques hide malware from running software.
3. Malware Factory Process
   Mutate malware code (metamorphism), Encrypt malware (polymorphism), Pack malware, Bind malware to Trojan cover file, Armor malware (anti-sandbox, debug, detect), QA
4. Botnets
   1. Host and network components.
   2. Command & Control servers are weak point.
   3. C&C protection: bulletproof hosting, dynamic DNS, fast flux, DGA
5. Provably impossible to construct perfect AV software.
6. Malware has real world impacts: theft, but also DoS.
References