Recap: Encryption for Confidentiality

- **Symmetric-key scheme (e.g., AES)**
  - The keys for encryption and decryption are the same.
  - Communicating parties must have the same key before communication

- **Public key scheme (e.g., RSA)**
  - Public key is published for anyone to encrypt a message
  - Only authorized parties have the private key to decrypt the message

- **Trade-offs**
  - Symmetric-key → quick, low cost, but needs to set the same key
  - Public-key → computationally heavy, but no need to exchange the key
Recap: Encryption for Confidentiality

- **Symmetric-key scheme (e.g., AES)**
  - The keys for encryption and decryption are the same.
  - Communicating parties must have the same key before communication.

- **Public key scheme (e.g., RSA)**
  - Public key is published for anyone to encrypt a message.
  - Only authorized parties have the **private key** to decrypt the message.

- **Trade-offs**
  - Symmetric key scheme: quick, low cost, but needs to set the same key.
  - Public key scheme: computationally heavy, but no need to exchange key.

**What about a Combined Approach?**
- First use *public-key* scheme to exchange the “key”
- Then, use *symmetric-key* to encrypt large messages.
Recap: Cryptographic Hash Functions

Input: arbitrary length data
Output: fixed-size digest (n bits)

No key, fixed function, “hard” to reverse

- Examples: MD5, SHA-1, SHA-256, SHA-512, SHA-3
- Desired notions of security for a cryptographic hash function $H$:
  - collision resistance: find any $m_1 \neq m_2$ s.t. $h(m_1) = h(m_2)$
  - second-preimage resistance: given $m_1$, find $m_2$ s.t. $h(m_1) = h(m_2)$
  - preimage resistance: given $h(m)$, find $m$
Malicious Code
Malicious Software
**Evolving Landscape Of Attacks**

[1980’s – early 1990’s]

**curiosity fueled hacking**: capability demonstration of hackers

[late 1990’s – present]

**Financial driven attacks**: spam, stealing credit cards, phishing, large-scale botnets

[late 2000 – present]

**Targeted attacks**: stealing proprietary information, information warfare, political manipulation

Challenges caused by:

Scale, complexity, anonymity
“Malicious Software”: Definition and Goals

- Software (more generally, a set of instructions) that runs on a computer it doesn’t have access to and typically does something nefarious

- Goals:
  - Steal private data
  - Display ads, send spam
  - Damage local machine
  - Congest a network
  - Attack other systems on the network
  - Commit online fraud
  - Gain, then grant, unauthorized access
  - Up to the attacker(s) really…
The Problem of Malware

- How does this software get on victim computers?

Buffer overflow in a vulnerable & network-accessible device
The Problem of Malware

- How does this software get on victim computers?

Vulnerable client connects to remote system that sends an attack over (Drive-by download, Email attachment)
The Problem of Malware

- How does this software get on victim computers?

Social engineering attacks

- https://www.youtube.com/watch?v=PWVN3Rq4gzw&feature=youtu.be&t=35s
- https://www.youtube.com/watch?v=lc7scxvKQOo
The Problem of Malware

- How does this software get on victim computers?

**Insider threats** (e.g., manufacturer installs something; MITM; software provider, logic bomb)
The Problem of Malware

- How does this software get on victim computers?

“Autorun” functionality (e.g., code executes as soon as a USB device is plugged in)
From Kaspersky Lab 2015 report:

- Exploits at different stages
- “Browsers” category includes landing pages
- Attackers “have mastered” non-Windows platforms
- Attackers using Tor and Bitcoins
Computer Virus

- The concept first mentioned in “Westworld”?
  - The 1973 film: the first mention of the concept in a movie: “the spread of malfunctions”

- Virus: replicates itself, infects (modifies) other programs

Westworld 2016 (HBO)
Computer Virus

- In 1982, high-school student Rich Skrenta wrote first virus released in the wild: Elk Cloner, a boot sector virus infecting the boot-sector of a floppy disk
  - Virus copies itself from disk to disk
  - boot sector was a popular target as it is executed automatically

- Brain (another boot-sector virus), by Basit and Amjad Iqbal in 1986, credited with being the first virus to infect PCs (MS-DOS)
  - https://youtu.be/InedOWfPPT0?t=129
Computer Worms

- **Worm vs. Virus**
  - Virus: self-replicating, needs to infect a host program
  - Worm: self-replicating, does not need a host program (spreads through a network)

- **Morris Worm**, the first Internet worm
  - by Cornell student Robert T. Morris Jr. in 1988
  - ~10% of computers (6k machines); $10M in damages
Computer Worms

- **Love letter worm** in 2000
  - A Visual Basic program disguised as a love letter
  - love-letter-for-you.txt.vbs

- **Code Red** worm spreaded in 2001
  - >500k servers; $2.6B in damages
  - [http://www.youtube.com/watch?v=v6GnX3ZhuAg](http://www.youtube.com/watch?v=v6GnX3ZhuAg)
Trojan Horse

- Trojan horse is a tale about the Trojan War in Troy
- A huge wooden horse with Greek soldiers hidden in it
- Malicious software disguised as legitimate software
Trojan Example

For the second time in five months, the Transmission BitTorrent client for Mac has been infected with malware.

The malware, dubbed OSX/Keynap, is pretty nasty. It’s designed to steal the contents of the OS X system keychain and maintain a permanent backdoor. And for a few hours, that malware found its way into the popular Mac BitTorrent.
Ransomware: Trojan Horse? Worm?
Mariposa botnets

Mariposa botnet 12 million IPs;
Stolen data belonging to 800K users;
Malware changes every 48 hours;
Attacker uses real name in DNS

<table>
<thead>
<tr>
<th>#</th>
<th>Cities</th>
<th>%</th>
<th>IPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Seoul</td>
<td>5.36%</td>
<td>761,444</td>
</tr>
<tr>
<td>2</td>
<td>Bombay</td>
<td>4.45%</td>
<td>631,927</td>
</tr>
<tr>
<td>3</td>
<td>New Delhi</td>
<td>4.27%</td>
<td>605,518</td>
</tr>
<tr>
<td>4</td>
<td>Mexico</td>
<td>3.89%</td>
<td>551,705</td>
</tr>
<tr>
<td>5</td>
<td>Bogotá</td>
<td>2.68%</td>
<td>380,487</td>
</tr>
</tbody>
</table>

http://pandalabs.pandasecurity.com/mariposa-stats/
Bots and Botnets

- Command and control (C&C)
  - **Centralized**: Single server directs bots (i.e., botmaster)
    - Simple, but easy to detect/disable
  - **Distributed**: Bots exchange control messages via P2P network
    - Complex, but hard to detect/disable
What Botnet is usually used for?

- Steal sensitive information from infected hosts
  - Credit card info, bank account info, etc.
- Spam as a service
  - Spreading email spam
- Bitcoin mining
- Deny of service (DOS) as a service
- “Anti”-Deny of Service as a service?
Rootkit

- **Rootkit**: malicious code attempt to operate as root and hide its presence
- It interferes with the normal interaction between a user and OS
  - E.g., rootkit deletes itself from files, processes listing (`ls`, `ps` commands in Linux)
- Detection is usually based on comparison
  - Compare data obtained at different OS levels to find discrepancies
- Sony XCP rootkit (2005) prevents a user from copying a music CD
  - Automatically install itself from the music CD (called autorun.exe)
  - It blocks the display of any program beginning with `$sys$`

What Sony did wrong:
1. distributed code that may open a system to possible infection by other attackers
2. install code with user’s knowledge, much less consent, and hide itself
**Major Malware Types**

- **Trojans**: appears to perform desirable function, but does something malicious behind the scenes

- **Virus**: self-replicating software that infects other programs and can mutate itself to avoid detection

- **Worm**: Self-replicating software that spreads over the network

- **Botnet**: “Zombie” computers that do a botmaster’s bidding

- **Rootkit**: malware that uses stealth to achieve persistent presence on a machine
Virus Infection Types

- Overwriting
  - Destroys original code

- Pre-pending
  - Keeps original code, possibly compressed

- Infection of libraries
  - Allows virus to be memory resident
  - E.g., kernel32.dll

- Macro viruses
  - Infects MS Office documents
  - E.g., Melissa virus
Degrees of Complication

- Viruses insert themselves in computer code in different ways
## Virus Effects and Causes

<table>
<thead>
<tr>
<th>Virus effect</th>
<th>How it is caused</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attach to executables</td>
<td>Modify file directory,Write to executables</td>
</tr>
<tr>
<td>Attach to data/control file</td>
<td>Modify directory, data, append to data</td>
</tr>
<tr>
<td>Remain in memory</td>
<td>Intercept interrupt by modifying interrupt handler address</td>
</tr>
<tr>
<td>Infect disks</td>
<td>Intercept interrupt, syscalls, Modify system file</td>
</tr>
<tr>
<td>Conceal self</td>
<td>Intercept syscalls that would reveal self and falsify result, classify self as hidden file</td>
</tr>
<tr>
<td>Spread infection</td>
<td>Infect boot sector, system program</td>
</tr>
<tr>
<td>Prevent deactivation</td>
<td>Store copy to reinfect, activate before anti-virus program runs</td>
</tr>
</tbody>
</table>
An Idea for Detection

- Build **signatures** based on the virus code
Virus signatures (characteristic patterns)

- **Virus signature**
  - Hexdecimal opcode: 88 16 00 80 88 26 00 0d cd 13 cd 19
  - With many NOPs: 88 16 00 80 90 88 26 00 0d 90 90 cd 13 90 90 90 cd 19

- **How to extract virus signature**
  - Signature is found in every infected object by the virus, but not otherwise
  - Statistical methods on a large corpus of programs

- **How to do fast scan**
  - Boyer-Moore fast string search algorithm
Encrypted Viruses

- If a user launches an infected program, the virus decryption routine will
  - First gain control of the computer
  - Then decrypt the virus body
  - Then transfer the control of the computer to the decrypted virus

The decryption routing stays the same, but key is changed from one infection to another.

Virus Decryption Routine

1. Count = VirusBytes
2. Temp = FetchNextByte
3. Temp = Decrypt(Temp)
4. StoreNextByte(Temp)
5. Decrement Count
6. If Count>0, GOTO 2
7. #$^@^#^!^!#^!^#^!
8. !#@%$!@%!@%!@#
9. $#&!&%!#&#!%^!!

First Decrypted Byte

1. Count = VirusBytes
2. Temp = FetchNextByte
3. Temp = Decrypt(Temp)
4. StoreNextByte(Temp)
5. Decrement Count
6. If Count>0, GOTO 2
7. $S$^@^#^!^!#^!^#^!
8. !#@%$!@%!@%!@#
9. $#&!&%!#&#!%^!!

If a user launches an infected program, the virus decryption routine will
- First gain control of the computer
- Then decrypt the virus body
- Then transfer the control of the computer to the decrypted virus

The decryption routing stays the same, but key is changed from one infection to another.

Virus Decryption Routine

1. Count = VirusBytes
2. Temp = FetchNextByte
3. Temp = Decrypt(Temp)
4. StoreNextByte(Temp)
5. Decrement Count
6. If Count>0, GOTO 2
7. #$^@^#^!^!#^!^#^!
8. !#@%$!@%!@%!@#
9. $#&!&%!#&#!%^!!

First Decrypted Byte

1. Count = VirusBytes
2. Temp = FetchNextByte
3. Temp = Decrypt(Temp)
4. StoreNextByte(Temp)
5. Decrement Count
6. If Count>0, GOTO 2
7. $S$^@^#^!^!#^!^#^!
8. !#@%$!@%!@%!@#
9. $#&!&%!#&#!%^!!
What is the weakness of this Virus?

Clear patterns of the Decryptor → virus signature
Polymorphic Viruses

- Polymorphic virus
  - Mutate encryption/decryption routine with each copy of the code

- When user runs infected program
  - decryption routine takes control, decrypts virus and mutation engine.
  - decryption routine transfers control to virus code
  - when infecting new program, mutation engine randomly generates a new decryption routine for use in the new copy of virus
  - virus encrypts body and mutation engine, append new decryption engine
Polymorphic Virus Detection

To load encrypted virus into a virtual machine and let it decrypt itself; Then, scan for known signatures on the decrypted memory region

Read file from disk into VM memory

From Symantec.com
Polymorphic Virus Detection

Load the virus (decryptor, encrypted body) and host program into memory;
Virus decrypts itself;
We record how the memory changes as the virus decrypts itself
After the virus is decrypted, we scan the memory for virus signatures
Metamorphic Virus

- Decrypted **program body** mutates as well (along with the decryption routine)
  - Metamorphics are body-polymorphic.

1. Use different registers
2. Insert NOP instructions
3. Insert dummy instructions
4. Insert JUMPS
5. Reorder/permutation of instructions (that are independent of each other)
6. Code the instruction differently
   - sub eax, eax $\rightarrow$ xor eax, eax
Other Threats: Side Channel Attack

- Timing attack, side channel attack
  - E.g., infer cryptographic key based on observing computation time

- Covert channels
  - Transfer information between processes that are not supposed to be allowed to communicate
Covert Channels

Malicious Program

It gives away info about protected data by manipulating the lock of a file;

the lock is visible by an attacker.
Example: The service program sends out Signal 100 to the attacker via the lock of the file.
Another example: Covert Timing Channel.
Summary of Detection Approaches

- Use **anti-virus (AV) software** (defense against trojans, viruses, bots, slow worms)

- **Signature-based detection**
  - Find a string that can identify the virus (like a fingerprint)
  - Difficult against mutating viruses

- **Heuristic-based detection**
  - Analyze program behavior to identify unusual patterns
  - E.g. network access, file open or delete, modify boot sector
Comparing different program analysis methods

- Useful to identify new and “zero day” malware

- **Static programming analysis**
  - Based on the instructions to determine whether the program is malicious

- **Dynamic analysis**
  - Run code in isolated emulation environment (e.g., VM)
  - Monitor actions that target file takes
  - If the actions are harmful, mark as virus

- May trigger false alarms!
Other Detection Approaches

- **Tripwire**
  - Store hash of known-good binaries and config files
  - Later, compare to detect changes
  - Need to boot from external device to avoid rootkits

- **Defending against fast-spreading worms?**
  - Too quick to use a signature -- detect in the *network* instead
  - Infer worm signature (< 1 second), suppress traffic spreading the worm
Virus Detection is Undecidable!

- Theoretical result by Fred Cohen
- Virus abstractly modeled as program that eventually executes `infect`
- Code for `infect` may be generated at runtime
- Proof by contradiction
No Perfect Solution to Virus Detection?

1. Suppose program \texttt{isVirus}(P) determines whether program P is a virus

2. Define new program Q as follows:
   
   \begin{verbatim}
   if (not \texttt{isVirus}(Q))
       infect
   stop
   \end{verbatim}

3. Running \texttt{isVirus} on Q achieves a contradiction
Challenges in Practice

- Today’s viruses try to detect whether they are in a VM / sandbox
  - if there are clear signals of VM → does not run
  - If the system looks like a real machine → infect

- Signals of VM
  - **OS artifacts**: names of drivers and processes, the presence of specific files, the configuration of the OS.
  - **Hardware artifacts**: instruction execution time and variance, abnormal CPU information reporting, hardware identifiers
  - **User-space artifacts**: age of the files, browsing history

- Debugging and monitoring tools