CM2025: Computer Security

Summary

Arjun Muralidharan

19th June 2021

Contents

List of Figures

List of Tables

List of Listings

1 Introduction to computer security and malware

1.1 Security threats

Computer security is the protection of computer-related assets against an attacker, as well as the ensuring the secure usage of these assets by non-attackers.

Three goals of computer security:

1. **Prevention:** Safeguarding of assets from threats.
2. **Detection:** Having systems in place to tell you that an attack or malicious activity is or is about to take place.
3. **Reaction:** Defining the procedures that enable you to deal with an attack.

Three further aspects related to security:

1. **Policy:** Dealing with confidentiality, integrity and availability of data.
2. **Threat model:** The set of assumptions about the people involved in malicious activity.
3. **Mechanism:** The hardware and software designed to make sure the policies are enforced using assumptions which we made using the threat model.

Five key terms:

1. **Attack:** Activities harmful to computer systems.
2. **Risk:** Possibility of loss or damage to assets in case of an attack.
3. **Zero-Day Vulnerability:** A vulnerability used by an attacker before being discovered by anyone else.
4. **Exploit:** Software used to take advantage of a vulnerability.
5. **Hacker:** Actor exploiting a vulnerability, usually grouped into white hat (authorized) and black hat (unauthorized) hackers. Grey hats are somewhere in between and use varying combinations of good and bad intentions.

1.1.1 Types of Malicious Software

Malware is software designed to disrupt, damage and destroy the normal functionality of an information system.

Viruses Self-replicating software that inserts itself into other files and programs, and easily spread. Early viruses were Creepr (1971) and the Elk Cloner (1982), which were both benign.

Worms Self-replicating software that does not need a host program or file to attach itself to. Examples are Stuxnet, which was designed to target Iranian power plants.

Adware Displays ads on-screen with the purpose of collecting user data and usually downloaded to a computer without consent of the user. Examples include Fireball (2017) which secretely changed the default search engine to track user searches and was able to download and execute remote code.

Trojans Named after the trojan horse, these are programs that hide themselves inside an application or program data and spread based on a triggering event or user action. Examples include Zeus (2010) that was able to compromise the networks of Cisco, BofA and Amazon.

Spyware Designed to extract data from the user silently and used for selling collected data on the dark web. Examples include Dark Hotel, which used hotel Wi-Fi to target the personal systems of government officials and business tycoons.

Keyloggers Records every keystroke of the user and transmits this to an attacker. Examples include Olympic Vision.

Ransomware Software that encrypts a victims software, deletes backups and requires payment in order to release the data. Examples include WannaCry in 2017.

Botnets Programs that infect computers on the internet, where the attacker takes control of them without the user realizing. The taken over "bots" can then take orders from a central commander to launch a distributed denial-of-service (DDoS) attack by flooding a server with requests.

Rootkits Hidden software on the target computer that activitates in secret, allowing attackers to gain access to low-level system resources, use the computer for any of the other attack types. Examples include the Zaclino rootkit, which impersonates legitimate software by simulating user actions so it doesn't get detected by anti-virus software. It then continues to execute it's actual objective.

1.2 Malware Analysis

Malware Analysis is a set of processes and techniques that help a security address, understand the functionality, origin, impact, and intend to malicious software.

Indicator of Compromise Also known as **IOC**, this depicts the behabiour of the malicious software. Identifying the IOC is the goal of malware analysis.

1.2.1 Static Analysis

Static analysis involves analysing files without running them. This is usually a starting point for security analysis.

Two techniques are commonly used:

- **Anti-virus scanning**: Runs a file through software to determine if a file is clean or not. Example: virusTotal.
- **Hashing**: Run the file through software that produces a digital fingerprint (a hash) of a file. This can be used to analyze the structure of malware. Examples: MD5 and SHA1. A check against a known database of bad hashes will help to find if a file is malicious.

1.2.2 Dynamic Analysis

Dynamic analysis allows a security professional to understand the behaviour and functionality of software. This is often not possible through static analysis.

Sandboxes A way to analyse malware in a controlled environment. The virtualized environment contains a virtual network, services, drives, etc. to ensure that the malware behaves exactly as it would in a real environment. Two types of sandboxes are typically used:

1. **Agent-Based**: Requires software to be installed on every computer that needs to be monitored. Examples: Cuckoo, Threat Expert, BitBlaze and Comodo.
2. **Agent-Less**: Monitors computers from afar. Examples: VMRay, Analyzer and SNDBox.

Research suggests that agent-less sandboxes are more efficient. Another tool on Windows machines is **Process Monitor**, which allows monitoring of the registry, file system, network, running processes etc. in a Windows-based operating system.

1.3 Ethics

When a vulnerability is discovered, it might be sufficient to secure a local system, but there are usually wider **ethical ramifications**. It is common practice to identify a bug to a provider of software, and offer enough time to fix it before public disclosure. The biggest challenge in this context is **transparency**, as the provider may not fix the issue or even seek legal action - in which case there is a dilemma of disclosing the vulnerability for the sake of the wider public.

1.4 Passwords

It is becoming more common for sharing leaked data on the dark web. Reusing passwords is a common problem, and therefore these data are usually not unique to a single system. We can encourage use of stronger passwords, encourage two-factor authentication. We need to balance **accessibility** with **security** and **usability**.

1.5 Social Engineering

An example of social engineering is scattering USB drives with malicious software and having people just plug them into computers. The drives can be run as an input/output device, and can therefore act e.g. as a keyboard.

Other examples are e-mails that feel and look legitimate but a deeper look may show they are not authentic.

2 Network Security

2.1 Objectives of network security

Network security is a set of policies and practices to protect a network. A policy might be that nobody can access resources without being authenticated. The associated practice would be to implement the authentication system that enforces the policy. Objectives of network security are:

1. **Confidentiality:** Controlling the read access of users to resources.
2. **Integrity:** Controlling the write access, or ability to change, resources.
3. **Availability:** Maintaining functionality. A unavailable network would be a network that is hard to access, with numerous safeguards in place. A highly available network is easy to access.

2.2 Types of attacks

2.2.1 Denial of service

A **denial of service** (DoS) attack is an attack where a service is attacked so that users are denied usage of that service generally by overloading the network. The attack surface is comprised of a classical OSI model:

1. Hardware
2. Data link
3. Network
4. Transport
5. Session
6. Presentation
7. Application

Example 1: ARP flood Every machine on the network has an IP address. An ARP request ia a broadcast on the network that asks for the MAC address from the IP address. All machines receive

the request, and the matching device responds. By flooding a network with ARP requests for one machine, you can take the machine down. This operates at the **data link** layer.

Example 2: ICMP ping flood A host is flooded with internet control message protocol messages, or pings, and gets overwhelmed as the machine must respond to each ping request it receives. This operates at the **network** layer.

Example 3: TCP-SYN flood TCP involves opening a connection using the TCP-SYN request (part one of the three-part TCP handshake). Opening millions of these connections will overwhelm a machine. This operates at the **transport** layer.

2.2.2 Distributed denial of service, botnets and Mirai

The main difference of a **distributed denial of service attack (DDoS)** compared to a DoS attack is that there are multiple agents attacking a target simultaneously. Essentially it is a DoS attack performed with a network of agents instead of a single agent.

A **botnet** attack is a kind of DDoS attack where a control server instructs bots on the network (that have first been placed on host machines on the network) to take action. This is a risk with the internet of things, where items might get infected with bots that are then activated or controlled by a control server.

An example of a botnet is the **Mirai** attack, where 100.000 devices where used to attack the DNS provider Dyn, taking down a lot of large websites. It used code to remove its own binary code (residing only in memory) and hiding its process name.

The Mirai botnet placed **bots**, each comprising the following components:

1. **Attack**: The component that generates the network traffic, in this case 10 DoS attacks
2. **Kill**: Takes down other malware and servers so that it can do what it needs to document
3. **Scan**: Find other devices to infect

The bots communicate with the **server** that handles data (e.g. if a bot finds another devices, this gets stored on the server) and instructs the bots.

A **loader** is separate from the server and is responsible for creating new bots. It gets the list of new devices from the server, and then loads the attack payloads and creates new bots, and the server then sends these out.

2.2.3 Wireless attacks

WiFi standards have evolved, and WiFi security has seen numerous updates:

1. WEP (802.11a/b) - 1997

2. WPA (802.11g) - 2003
3. WPA2 (802.11i) - 2004
4. WPA3 - 2020

Each standard has increased the encryption and authentication standards. Wireless networks are much more easily accessed than physical networks. For example, WEP was subject to a **dictionary attack** where short keys allow brute force encryption breaking.

A **replay attack** replays a series of packets and inspects the response to potentially get valuable data.

A **PRGA attack** uses packet tampering to impersonate packets and gain access to systems.

2.3 Firewalls

2.3.1 Stateless Firewalls

Access control list An example of a stateless firewall that checks all traffic that's coming through, inspecting each packet, and applying a set of rules. For example, it can define that packets targeted at a certain IP address cannot come through.

2.3.2 Stateful Firewalls

A stateful firewall inspects the initial connection packet but then doesn't monitor the traffic anymore. It's slightly more efficient than a stateless firewall because it doesn't check every packet.

2.3.3 Proxy Firewalls

A proxy firewall serves as a single point where all traffic between an external and internal network (WAN and LAN) is routed through. This makes sure that any traffic to the internal network is checked. The proxy pretends to be the internal machine towards the outside world.

2.4 Intrusion Detection Systems

An **Intrusion Detection System (IDS)** is a system that runs on the network or a machine which aims to detect when an intruder is operating on the network or machine.

Most security experts agree that a completely secure system is impossible to achieve. So we must stay alert for attacks.

The model is based on the hypothesis that exploitation of a system's vulnerabilities involves abnormal use of the system; therefore, security violations could be detected from abnormal patterns of system usage.

Challenges

1. **Effectiveness** False positives, false negatives
2. **Performance**: Resource use, speed
3. **Scalability** Ability to operate over a whole network

New techniques involve using pattern recognition (AI, machine learning) to detect abnormal behaviour.

Commercial examples are the Cisco Firepower appliances (physical servers installed in your rack) and the Darktrace Enterprise Immune System. The latter uses machine learning and is software-based.

3 Operating System Security

OS security is needed more today due to most computers being connected to the internet. Key components of a secure OS are:

1. **Kernel:** Operates services at the lowest level
2. **Security Kernel:** Resposinbile for managing all the OS's security processes
3. **Reference Monitor:** Component of the security kernel that manages access to the device

On Windows, the components are typically:

1. **Security Reference Monitor:** Element of kernel mode which executes access checks, sets audit log entries and manipulates privileges. The SRMP performs every permission check.
2. **Local security authority:** Accountable for executing windows local security policies and publishes users authentication tokens as they sign into the network. It also includes authentication policy and privileged parameters.
3. **Security account manager:** A database that stores user credentials and related individual user and local community sensitive data. When a user signs in, the SAM server is checked.
4. **Active Directory:** Handling cloud workflows and monitoring in-house network identification and encryption control.
5. **WinLogon Android NetLogon** Manages local and network-wide input logins respectively.

Client-side vulnerabilities Every piece of installed software, such as a browser, a productivity suite or another program can present a client-side vulnerability.

Hardeing Reduced the number of features that unauthenticated users are exposed to, and removing unused software or files. Examples include disabling automatic login, requiring credentials to exit screensaver, disabling WiFi when not needed, disabling updates, turning authentication on, creating backups, establishing a root user and password, and so on.

3.1 File System and Directory Structure

A **file** is a set of linked data collected in a non-volatile memory location. Each file has a logical location for storage and retrieval.

A **file system** manages the stucture of where files are stored and how they are retrieved. On Windows, this is often NTFS or FAT32, on macOS it's usually HFS or APFS, and Ext4 on Linux.

There are three file structure categories on an OS.

1. Text files, describing bits arranged in lines
2. Object files, a collection of bits, structured in blocks
3. Source files, a file related to a variety of activities in the OS

A file has **file system attributes**, which include the *name*, *position*, *type*, the *taylor* (which helps calculate the file size), *protection*, and finally *time, date* and *safety.*

The **directory structure** keeps the structure of all the files involved in a directory. It's used to also enforce authentication policies.

1. Users should have the right to assign a file an appropriate title without worrying about other identical files on the network.
2. Users should be able to access the files they and others generate

A file structure consists of the *file name*, the *type* and the *location.* There's also protection information, which includes the flag (D for directories, L for links or M for miscellaneous installed file format).

3.2 Windows Security History

Windows 1 to Vista Use the FAT file system with very limited functionality on the file system level and limited user security (e.g. Win95 had only one user). Rudimentary and not super-secure networking stack.

Windows NT An enterprise class version of Windows with a different kernel with NTFS. More secure, multiple users and authentication, disk encryption.

Windows 2000-XP Merged the NT kernel into the consumer product, with a new interface in XP. Suffered the major onslaught of malware and security attacks, since it was also incredibly popular.

Windows 10 Introduced consumer windows security with Microsoft Defender, Windows Hello and Onedrive which significantly improved the security architecture for consumers. Enterprise windows security include features such as *identity and access*, *threat protection* and *information protection.*

3.3 Linux Security

Most online services run on Linux, and therefore need to be secure. The **Harbian** Linux distribution (based on Debian) provides a security-hardened version of Debian, which provides an auditing script to run static checks to identify security issues. Usage of the script is described at https://www.kitploit.com/2020/07/harbian-audit-hardened-debian-gnulinux.html and the Git repository is at https://github.com/hardenedlinux/harbian-audit.

The script executes around 272 tests and provides output for each; often tests fail if a piece of software is not installed, so not every error is necessarily a security issue.

3.3.1 Android Attack Surface

Radios An Android device will have **WiFi**, which exposes it to any local network attacks, **Bluetooth**, which might connect to other devices such as a heart rate monitor or a watch, and **cellular antennas** (incl. 5G, 4G, GSM, etc.) which connects to local cell towers, which in turn connect to the cell network and the internet.

Sensors Devices have **Cameras**, **Touchscreens**, **Microphones**, and a **Gyroscope**. These can also be attacked in various ways. **GPS** is another sensor that can be attacked.

Android Stack The software stack consists of the **Kernel**, the **base linux**, **vendor extensions** and then software provided by the **network provider**. The question is if all parties in the stack have practices for providing secure software and patching old software.

3.3.2 Container Security

A **virtual machine** spins up a complete operating system in a ringfenced fashion running on another system. It's slow to startup. A **container** shares the kernel, it's much more lightweight than a VM and can start up faster. Used for rapid scaling of web applications, e.g. with Docker. The four typical attack routes in a container environemtn are:

1. **Container → Host**
2. **Host → Container**
3. **Container → Container**
4. **Application → Container**

4 Cryptography

We are interested in the general setup of **Alice** sending a message to **Bob**, and **Eve** tries to intercept the message.

A basic concept to secure this message is to place it in a "box" and provide two keys, and Alice and Bob each have one key to lock and unlock the message.

Encryption involves transforming the plain text message into a *ciphertext* before transmitting it. The usual process involes *encrypt* $\rightarrow$ *send* $\rightarrow$ *decrypt*.

History of Cryptography The first documented scientific documents related to cryptography was *Al Kindi*, an arabic scientist. More modern documents are *Viginère* in the sixteenth century.

Modern **public key cryptography** was developed with the **RSA** algorithm. More recently, **elliptic curves**, used e.g. for Bitcoin.

Goals of Cryptography

1. Confidentiality (secrecy)
2. Integrity (anti-tampering)
3. Authentication
4. Non-repudiation (cannot be denied that it was sent)

4.1 Symmetric cryptography

Symmetric cryptography involves using the same key for encryption and decryption. Alice and Bob have the same key. An example of this is the *Caesar Cypher* which involves mapping the alphabet to an alternative alphabet. This is also called a *substitution cypher*. It can be broken by analysing the frequency of letters, and mapping those to the frequency of letters in the regular alphabet.

4.2 Asymmetric cryptography

In **public-private key cryptography**, both Alice and Bob each have their own set of *public* and *private* keys. Messages encrypted with the public key $F(M)$ can be decrypted with the corresponding private key $G(M) = F^{-1}(M)$.

Messages encrypted with a public key are hard to reverse, the public key encryption process is a **one-way function**.

To send a message from Alice to Bob:

1. Alice encrypt's a message using **Bob's public key**
2. Only Bob can read it, because he has his **private key**
3. Eve cannot read it, because she doesn't have **Bob's private key**
4. Alice can of course read it because she has access to the plaintext.

All that is needed to initiate a two-way communication is an exchange of the public keys.

Signing Messages To send a message that confirms that Alice sent a message, we use signing or authentication. This is done by encrypting a signing message with her **private key**, because this can be decrypted by anyone (using Alice's public key) but only created using Alice's private key.

SHA-1 collision In 2017, Google discovered that SHA-1, a hashing algorithm, could be broken, which means it was possible to produce two different PDFs with different content that hashed to the same value. See https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html and https://www.schneier.com/blog/archives/2005/02/cryptanalysis_o.html.

4.3 Advanced techniques

Transposition Cipher Rearranging the letters to a scrambled version is done using this kind of cipher. It's done using a *transposition grid*. This requires the length of the plaintext message, and a transposition number.

The letters of the plaintext are written in rows of a grid with as many columns as the transposition number (with rows to accomodate the entire message), and the cipher is generated by reading the columns.

Bob can decipher the message knowing the transposition number and the length of the message.

If spaces are left blank in the grid, we can fill them with a special character, such as `x`.

A more advanced variation is by using a keyword instead of a number. The keyword length determines the number of columns. We assign each letter in the keyword to a column. Then I arrange the letters in the keyword by alphabet, and also rearrange the columns accordingly. This is the cipher grid.

4.3.1 Playfair Cipher

This cipher was used by the CIA. It uses a 25-space grid for the alphabet, treating I and J as one letter. It uses a keyword.

1. The keyword is entered into the first spaces of the grid. The rest of the spaces are filled up with all letters that don't appear in the keyword. The keyword needs to therefore consist of unique letters, with no repeated letters.
2. We then break our plain text into pairs of letters ("This" becomes TH IS) Double letters are separated with an X.
3. If the letters are not even, add an X at the end.
4. The first pair is encoded by looking at where the letters appear on the cipher grid. The two positions span a rectangle. The letters are then replaced with the letters in the opposite corners

of the spanned rectangle. The convention is that the first letter is the one that shares the row with the first encrypted letter.

5. If the letters are in the same column, they get replaced with the letter below them in the column.

A weakness of the Playfair cipher is that pairs remain the same, so given a large body of text, the pairs can provide a way to revers-engineer the cipher grid.

4.3.2 Viginère Cipher

The Viginère cipher is similar to the Caesar cipher but varies the amount by which one moves, using a keyword.

1. The keyword's letters define by how much a letter is offset. For example, an E in the keyword indicates an offset by 5 (because E is the fifth letter).
2. The keyword is mapped to the plaintext, by repeating it as many times as needed.
3. Each letter in the plaintext is then ciphered by the offset given in the keyword.

This cipher results in the same letter being ciphered to different result letters, so it's a clear improvement over Caesar. This technique is best done using a Viginère table.