IS U3 NOTES

Malware

Malware, also known as malicious code, refers to a program that is:

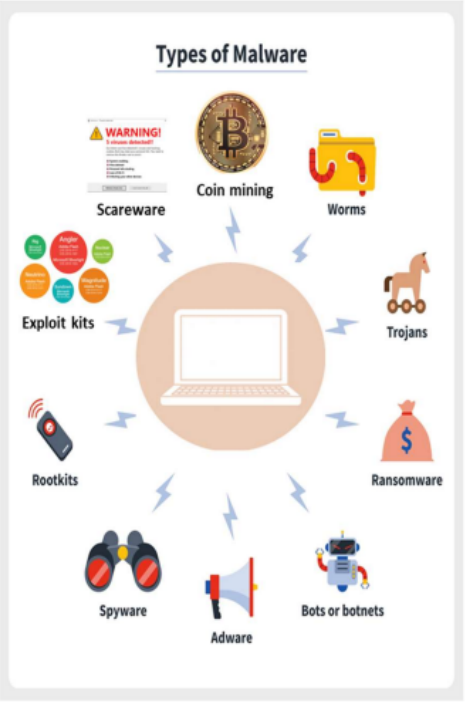
- Covertly inserted into another program with the intent to:
  - Destroy data
  - Run destructive or intrusive programs
  - Otherwise compromise the confidentiality, integrity, or availability of the victim’s data, applications, or operating system.

Malware is the most common external threat to most hosts, causing widespread damage and disruption and necessitating extensive recovery efforts within most organizations.

1. **Coin miners**
   - generates crypto coins and requires significant computing resources
   - XMRig mining Monero cryptocurrency.
2. **Scareware**
   - tricks users into believing their computer is infected with a virus.
3. **Exploit Kits**
   - a set of malware intended to exploit known vulnerabilities
   - Browsers, Adobe Flash Player, Oracle Java
   - Magnitude, Redline, Neutrino, Angler, Rig

1. **Rootkits**
   - remote access of device to hackers
   - bootloaders, firmware, kernel mode

1. **Spyware/keylogger**
   - monitor user activities or collect info
   - darkhotel, fireball, snake
2. **Adware**
   - Advertising software that serve fake advertisements
3. **Wipers**
   - Delete data from computers
   - Shamoon, NotPetya



8. **Virus**
   - require a host program to trigger the infection.
   - The virus writes into a host program/file/boot/usb to infect or propagate.
   - Upon execution of infected program/file/… there is propagation/damage.
9. **Worms**
   - spreads through a network by replicating itself
   - Stuxnet, Trickbot
10. **Trojans**
    - a Trojan horse is any malware that misleads users of its true intent.
    - The term is derived from the Ancient Greek story that led to the fall of the city of Troy

10. **Ransomware**
    - denies access to your data till ransom is paid
    - Ryuk, Conti, REvil, Petya, WannaCry

8

10. **Bots/Botnets**

Malware can be classified into two broad categories based on the actions or payloads it performs once a target is reached:

1. **Based on actions or payloads:**
   - Those that need a host program (parasitic code such as viruses)
   - Those that are independent, self-contained programs (worms, Trojans, and bots)
2. **Based on replication:**
   - Malware that does not replicate (Trojans and spam email)
   - Malware that does replicate (viruses and worms)

Propagation mechanisms include:

- Infection of existing content by viruses that is subsequently spread to other systems
- Exploit of software vulnerabilities by worms or drive-by-downloads to allow the malware to replicate
- Social engineering attacks that convince users to bypass security mechanisms to install Trojans or to respond to phishing attacks

Payload actions performed by malware once it reaches a target system can include:

- Corruption of system or data files
- Theft of service/make the system a zombie agent of attack as part of a botnet
- Theft of information from the system/keylogging
- Stealthing/hiding its presence on the system

Signs of Malware infection may include:

- Contacts receiving strange messages from you
- Slow computer performance
- Very frequent ads and pop-ups
- Rapid battery drainage
- System crashes
- Sudden loss of disk space
- Increased system's internet activity
- Browser redirects occurring on their own
- Antivirus software being shut down or uninstalled

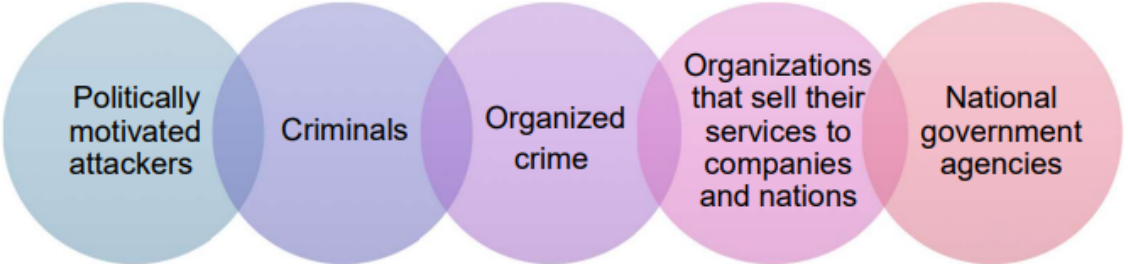
Attack Kits:

- Initially, the development and deployment of malware required considerable technical skill by software authors.
- The development of virus-creation toolkits in the early 1990s and then more general attack kits in the 2000s greatly assisted in the development and deployment of malware.
- Toolkits are often known as “crimeware” and include a variety of propagation mechanisms and payload modules that even novices can deploy.
- Variants that can be generated by attackers using these toolkits create a significant problem for those defending systems against them.

Widely used toolkits include:

- Zeus
- Blackhole
- Sakura
- Phoenix
- PeStudio
- Process Hacker
- Process Monitor (ProcMon)
- ProcDot
- Autoruns
- Fiddler
- Wireshark
- x64dbg
- Ghidra
- Radare2/Cutter
- Cuckoo Sandbox

Attack Sources:



- Another significant malware development is the change from attackers being individuals often motivated to demonstrate their technical competence to their peers to more organized and dangerous attack sources such as:
    - Organized crime groups
    - State-sponsored actors
    - Hacktivist collectives

- This shift has significantly changed the resources available and motivation behind the rise of malware and has led to the development of a large underground economy involving the sale of attack kits, access to compromised hosts, and stolen information.

Advanced Persistent Threats (APTs):

- APTs are well-resourced, persistent applications of a wide variety of intrusion technologies and malware to selected targets, usually businesses or political entities.
- They are typically attributed to state-sponsored organizations and criminal enterprises.
- APTs differ from other types of attacks by their careful target selection and stealthy intrusion efforts over extended periods.
- High-profile APT attacks include Aurora, RSA, APT1, and Stuxnet.

Characteristics of APTs:

- **Advanced:**
  - Attackers use a wide variety of intrusion technologies and malware, including the development of custom malware if required.
  - The individual components may not necessarily be technically advanced but are carefully selected to suit the chosen target.
- **Persistent:**
  - APTs involve the determined application of attacks over an extended period against the chosen target to maximize the chance of success.
  - A variety of attacks may be progressively applied until the target is compromised.
- **Threats:**
  - APTs pose threats to selected targets as a result of organized, capable, and well-funded attackers' intent to compromise specifically chosen targets.
  - The active involvement of people in the process greatly raises the threat level from that due to automated attack tools and also the likelihood of successful attacks.

APT Attacks:

- **Aim:**
  - Varies from theft of intellectual property or security and infrastructure-related data to the physical disruption of infrastructure.
- **Techniques used:**
  - Social engineering
  - Spear-phishing emails
  - Drive-by-downloads from selected compromised websites likely to be visited by personnel in the target organization.
- **Intent:**
  - To infect the target with sophisticated malware with multiple propagation mechanisms and payloads.
  - Once initial access to systems in the target organization is gained, a further range of attack tools are used to maintain and extend their access.

Viruses:

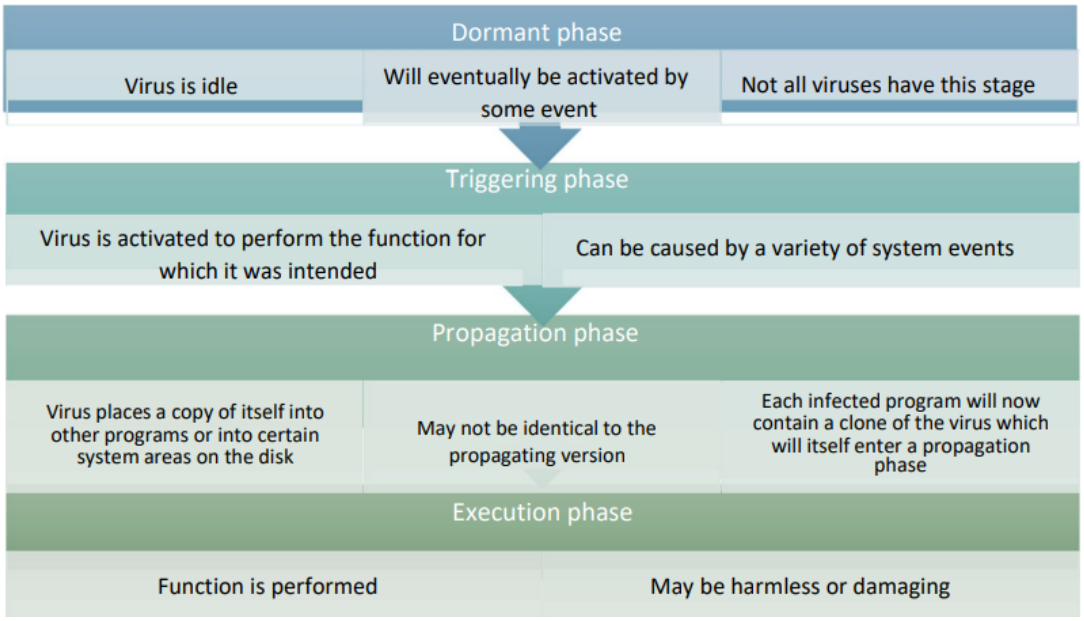
- A virus is a piece of software that infects programs by modifying them to include a copy of the virus. It then replicates and goes on to infect other content.
- Viruses are easily spread through network environments.
- When attached to an executable program, a virus can do anything that the program is permitted to do and executes secretly when the host program is run.
- Viruses are specific to operating systems and hardware and take advantage of their details and weaknesses.

Virus Components:

- **Infected Mechanism:**
  - The means by which a virus spreads or propagates, also referred to as the infection vector.
- **Trigger:**

- An event or condition that determines when the payload is activated or delivered, sometimes known as a logic bomb.

- **Payload:**
  - What the virus does besides spreading. This may involve damage or benign but noticeable activity.



Virus Classifications:

Classification by target:

- **Boot Sector Infector:**
  - Infects a master boot record or boot record and spreads when a system is booted from the disk containing the virus.
- **File Infector:**
  - Infects files that the operating system or shell considers to be executable.
- **Macro Virus:**
  - Infects files with macro or scripting code that is interpreted by an application.
- **Multipartite Virus:**
  - Infects files in multiple ways.

Classification by concealment strategy:

- **Encrypted Virus:**
  - A portion of the virus creates a random encryption key and encrypts the remainder of the virus.
- **Stealth Virus:**
  - A form of virus explicitly designed to hide itself from detection by anti-virus software.
- **Polymorphic Virus:**
  - A virus that mutates with every infection.
- **Metamorphic Virus:**
  - A virus that mutates and rewrites itself completely at each iteration and may change behavior as well as appearance.

Macro and Scripting Viruses:

- Macro and scripting viruses were very common in the mid-1990s.
  - They are platform-independent and infect documents rather than executable portions of code.
  - They are easily spread.
- They exploit the macro capability of MS Office applications.
  - More recent releases of products include protection against them.
- Various anti-virus programs have been developed, so these are no longer the predominant virus threat.

Create two folders:

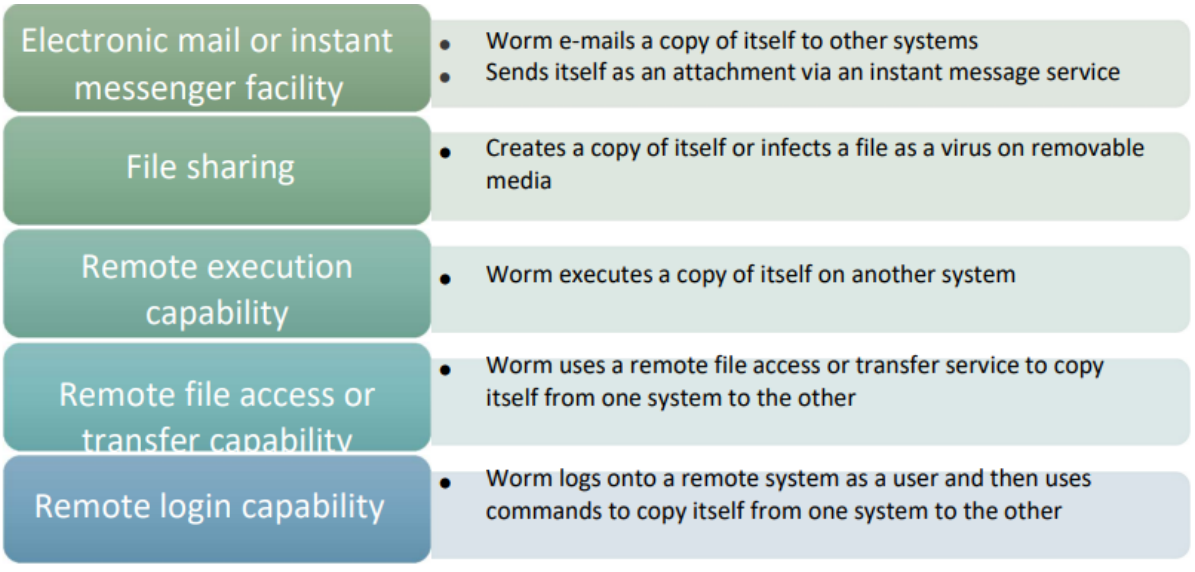
Folder with Virus
• A.foo
• B.foo
• C.fo
• D.fxx
• Foovirus.py

Folder without virus
• A1.foo

- E.foo
- F.fxs

Worms:

- A worm is a program that actively seeks out more machines to infect.
  - Each infected machine serves as an automated launching pad for attacks on other machines.
  - Worms exploit software vulnerabilities in client or server programs.
  - They can use network connections to spread from system to system.
  - They can also spread through shared media such as USB drives, CD, and DVD data disks.
- Email worms spread through macro or script code included in attachments and instant messenger file transfers.
  - Upon activation, the worm may replicate and propagate again.
  - Usually carries some form of payload.
- The first known implementation was done in Xerox Palo Alto Labs in the early 1980s.



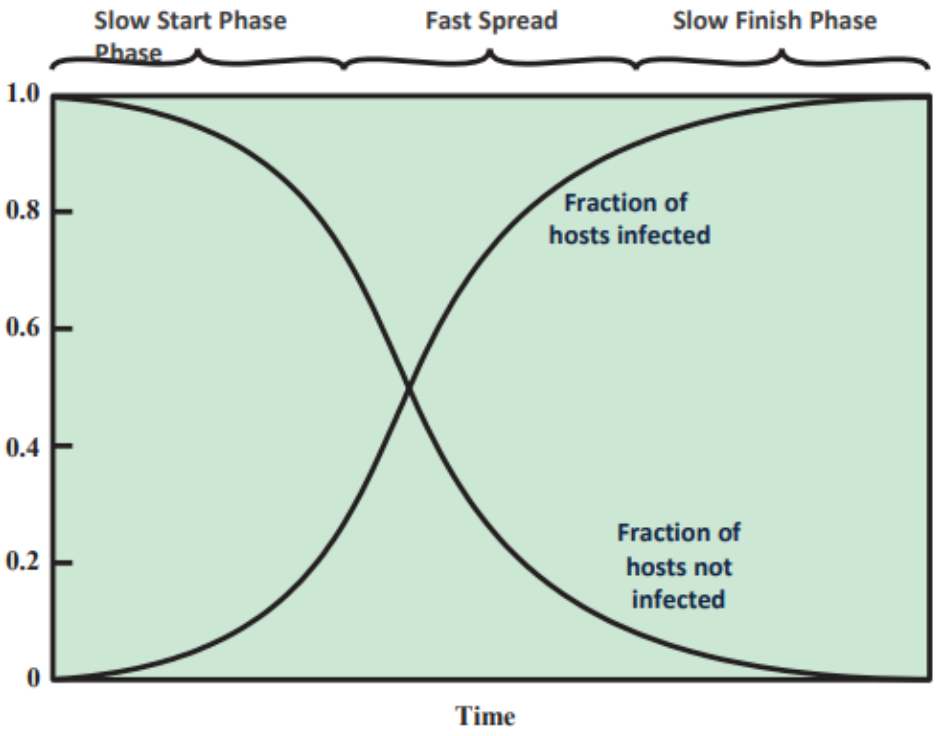
Target Discovery:

- **Scanning (or fingerprinting):**

- First function in the propagation phase for a network worm.
- Searches for other systems to infect.

Scanning strategies that a worm can use:

- **Random:**
  - Each compromised host probes random addresses in the IP address space using a different seed.
  - This produces a high volume of Internet traffic, which may cause generalized disruption even before the actual attack is launched.
- **Hit-list:**
  - The attacker first compiles a long list of potential vulnerable machines.
  - Once the list is compiled, the attacker begins infecting machines on the list.
  - Each infected machine is provided with a portion of the list to scan.
  - This results in a very short scanning period, which may make it difficult to detect that infection is taking place.
- **Topological:**
  - This method uses information contained on an infected victim machine to find more hosts to scan.
- **Local subnet:**
  - If a host can be infected behind a firewall, that host then looks for targets in its own local network.
  - The host uses the subnet address structure to find other hosts that would otherwise be protected by the firewall.



Morris Worm:

- The Morris Worm was the earliest significant worm infection.
- It was released by Robert Morris in 1988.
- Designed to spread on UNIX systems:
  - Attempted to crack the local password file to use login/password to log on to other systems.
  - Exploited a bug in the finger protocol, which reports the whereabouts of a remote user.
  - Exploited a trapdoor in the debug option of the remote process that receives and sends mail.
- Successful attacks achieved communication with the operating system command interpreter:
  - Sent the interpreter a bootstrap program to copy the worm over.

Melissa	1998	e-mail worm first to include virus, worm and Trojan in one package
Code Red	July 2001	exploited Microsoft IIS bug probes random IP addresses consumes significant Internet capacity when active
Code Red II	August 2001	also targeted Microsoft IIS installs a backdoor for access
Nimda	September 2001	had worm, virus and mobile code characteristics spread using e-mail, Windows shares, Web servers, Web clients, backdoors
SQL Slammer	Early 2003	exploited a buffer overflow vulnerability in SQL server compact and spread rapidly
Sobig.F	Late 2003	exploited open proxy servers to turn infected machines into spam engines
Mydoom	2004	mass-mailing e-mail worm installed a backdoor in infected machines
Warezov	2006	creates executables in system directories sends itself as an e-mail attachment can disable security related products
Conficker (Downadup)	November 2008	exploits a Windows buffer overflow vulnerability most widespread infection since SQL Slammer
Stuxnet	2010	restricted rate of spread to reduce chance of detection targeted industrial control systems

Mobile Code:

- Mobile code refers to programs that can be shipped unchanged to a variety of platforms.
- They are transmitted from a remote system to a local system and then executed on the local system.
- Often acts as a mechanism for a virus, worm, or Trojan horse.
- Takes advantage of vulnerabilities to perform its own exploits.
- Popular vehicles include Java applets, ActiveX, JavaScript, and VBScript.

Mobile Phone Worms:

- The first discovery was the Cabir worm in 2004.
- Followed by Lasco and CommWarrior in 2005.
- Communicate through Bluetooth wireless connections or MMS.
- Target smartphones.
- Can completely disable the phone, delete data on the phone, or force the device to send costly messages.
- CommWarrior replicates by means of Bluetooth to other phones, sends itself as an MMS file to contacts, and as an auto-reply to incoming text messages.

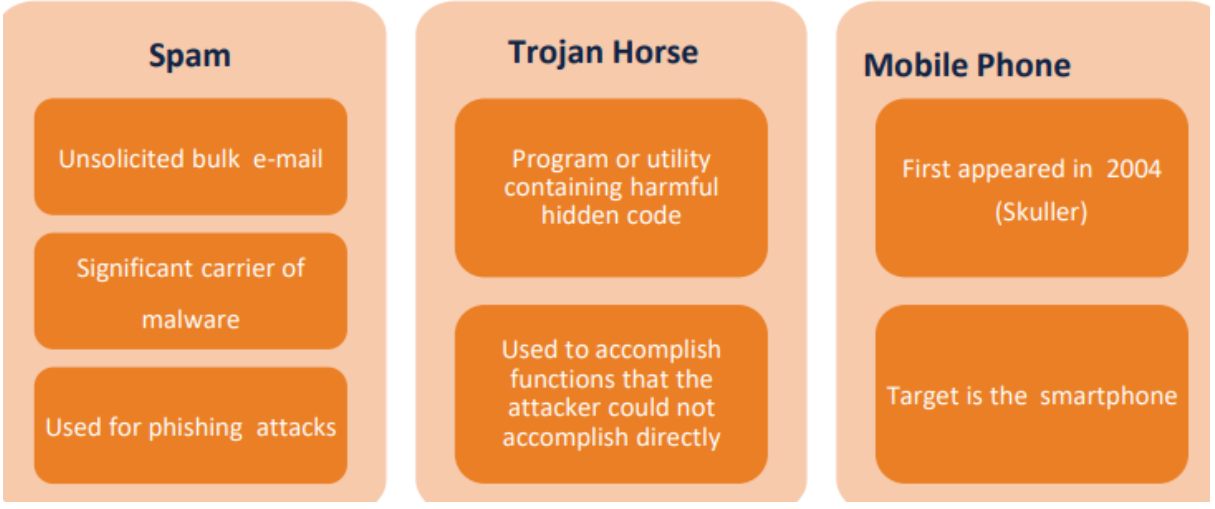
Drive-By-Downloads:

- Exploits browser vulnerabilities to download and install malware on the system when the user views a Web page controlled by the attacker.
- In most cases, does not actively propagate.
- Spreads when users visit the malicious Web page.

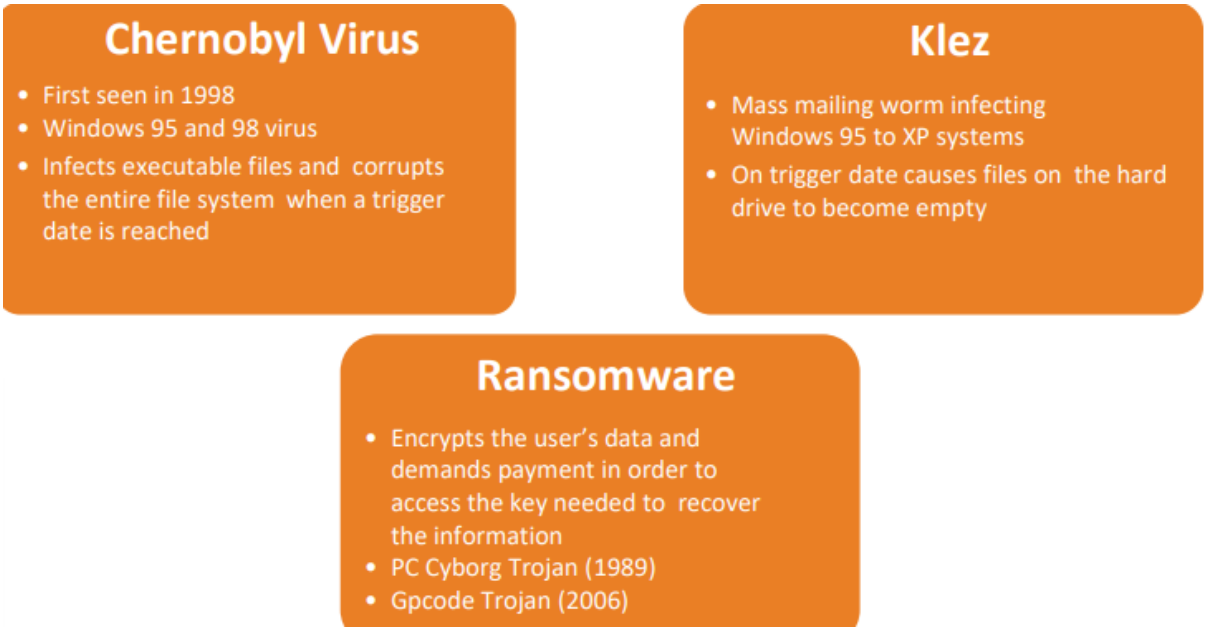
Clickjacking:

- Also known as a user-interface (UI) redress attack.
- Using a similar technique, keystrokes can also be hijacked.
- Vulnerability used by an attacker to collect an infected user's clicks.
- The attacker can force the user to do a variety of things from adjusting the user's computer settings to unwittingly sending the user to Web sites that might have malicious code.
- By taking advantage of Adobe Flash or JavaScript, an attacker could even place a button under or over a legitimate button making it difficult for users to detect.
- A typical attack uses multiple transparent or opaque layers to trick a user into clicking on a button or link on another page when they were intending to click on the top-level page.
- The attacker is hijacking clicks meant for one page and routing them to another page.

Social Engineering



Payload System Corruption



- Real-world damage:
  - Causes damage to physical equipment.
  - Examples:
    - **Chernobyl virus: Rewrites BIOS code.**
    - **Stuxnet worm: Targets specific industrial control system software.**
  - Concerns about using sophisticated targeted malware for industrial sabotage.
- Logic bomb:
  - Code embedded in the malware that is set to "explode" when certain conditions are met.

Payload – Attack Agents Bots:

- Takes over another Internet-attached computer and uses that computer to launch or manage attacks.
- Botnet: Collection of bots capable of acting in a coordinated manner.
- Uses:
  - Distributed denial-of-service (DDoS) attacks.

- Spamming.
- Sniffing traffic.
- Keylogging.
- Spreading new malware.
- Installing advertisement add-ons and browser helper objects (BHOs).
- Attacking IRC chat networks.
- Manipulating online polls/games.

Remote Control Facility:

- Distinguishes a bot from a worm:
  - Worm propagates itself and activates itself.
  - Bot is initially controlled from some central facility.
- Typical means of implementing the remote control facility is on an IRC server:
  - Bots join a specific channel on this server and treat incoming messages as commands.
  - More recent botnets use covert communication channels via protocols such as HTTP.
  - Distributed control mechanisms use peer-to-peer protocols to avoid a single point of failure.

Payload – Information Theft, Keyloggers, and Spyware:

- **Keylogger:**
  - Captures keystrokes to allow the attacker to monitor sensitive information.
  - Typically uses some form of filtering mechanism that only returns information close to keywords (“login”, “password”).
- **Spyware:**
  - Subverts the compromised machine to allow monitoring of a wide range of activity on the system.
  - Examples:

        - Monitoring history and content of browsing activity.
        - Redirecting certain Web page requests to fake sites.
        - Dynamically modifying data exchanged between the browser and certain Web sites of interest.

Payload – Information Theft Phishing:

- Exploits social engineering to leverage the user's trust by masquerading as communication from a trusted source.
    - Includes a URL in a spam email that links to a fake website that mimics the login page of a banking, gaming, or similar site.
    - Suggests that urgent action is required by the user to authenticate their account.
- **Spear-phishing:**
    - Recipients are carefully researched by the attacker.
    - Email is crafted to specifically suit its recipient, often quoting a range of information to convince them of its authenticity.

Worm Countermeasures:

- There is considerable overlap in techniques for dealing with viruses and worms.
- Once a worm is resident on a machine, anti-virus software can be used to detect and possibly remove it.
- Perimeter network activity and usage monitoring can form the basis of a worm defense.

Worm defense approaches include:

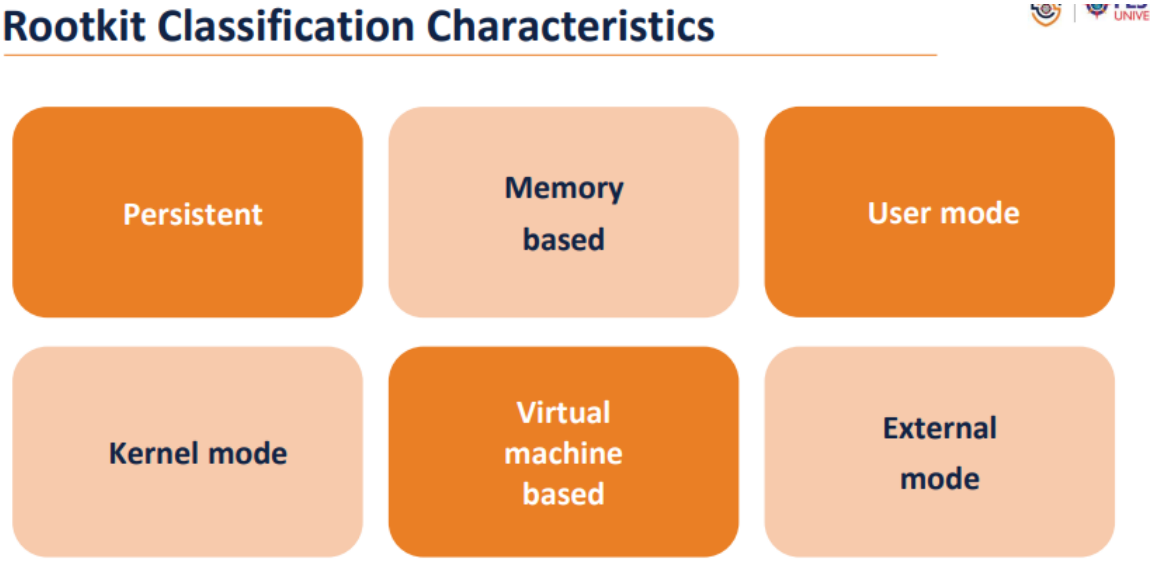
- **Signature-based worm scan filtering**
- **Filter-based worm containment**
- **Payload-classification-based worm containment**
- **Threshold random walk (TRW) scan detection**
- **Rate limiting**
- **Rate halting**

Payload – Stealthing Backdoor:

- Also known as a trapdoor.
- Secret entry point into a program allowing the attacker to gain access and bypass the security access procedures.
- Maintenance hook is a backdoor used by programmers to debug and test programs.
- Difficult to implement operating system controls for backdoors in applications.

Payload – Stealthing Rootkit:

- Set of hidden programs installed on a system to maintain covert access to that system.
- Hides by subverting the mechanisms that monitor and report on the processes, files, and registries on a computer.
- Gives administrator (or root) privileges to the attacker.
    - Can add or change programs and files, monitor processes, send and receive network traffic, and get backdoor access on demand.



Malware Countermeasure Approaches

- Detection
- Identification
- Removal

Prevention Elements - Policy , Awareness , Threat Mitigation

Generations of Anti-Virus Software:

- **First generation: simple scanners**
  - Requires a malware signature to identify the malware.
  - Limited to the detection of known malware.
- **Second generation: heuristic scanners**
  - Uses heuristic rules to search for probable malware instances.
  - Another approach is integrity checking.
- **Third generation: activity traps**
  - Memory-resident programs that identify malware by its actions rather than its structure in an infected program.
- **Fourth generation: full-featured protection**
  - Packages consisting of a variety of anti-virus techniques used in conjunction.
  - Include scanning and activity trap components and access control capability.

Generic Decryption (GD):

- Enables the anti-virus program to easily detect complex polymorphic viruses and other malware while maintaining fast scanning speeds.
- Executable files are run through a GD scanner which contains the following elements:
  - CPU emulator
  - Virus signature scanner
  - Emulation control module.
- The most difficult design issue with a GD scanner is to determine how long to run each interpretation.

Host-Based Behavior-Blocking Software:

- Integrates with the operating system of a host computer and monitors program behavior in real-time for malicious action.
    - Blocks potentially malicious actions before they have a chance to affect the system.
    - Blocks software in real-time so it has an advantage over antivirus detection techniques such as fingerprinting or heuristics.

Perimeter Scanning Approaches:

- Anti-virus software typically included in e-mail and Web proxy services running on an organization's firewall and IDS.
- May also be included in the traffic analysis component of an IDS.
- May include intrusion prevention measures, blocking the flow of any suspicious traffic.
- Approach is limited to scanning malware.

Ingress Monitors:

- Located at the border between the enterprise network and the Internet.
- One technique is to look for incoming traffic to unused local IP addresses.

Egress Monitors:

- Located at the egress point of individual LANs as well as at the border between the enterprise network and the Internet.
- Monitors outgoing traffic for signs of scanning or other suspicious behavior.

Ransomware

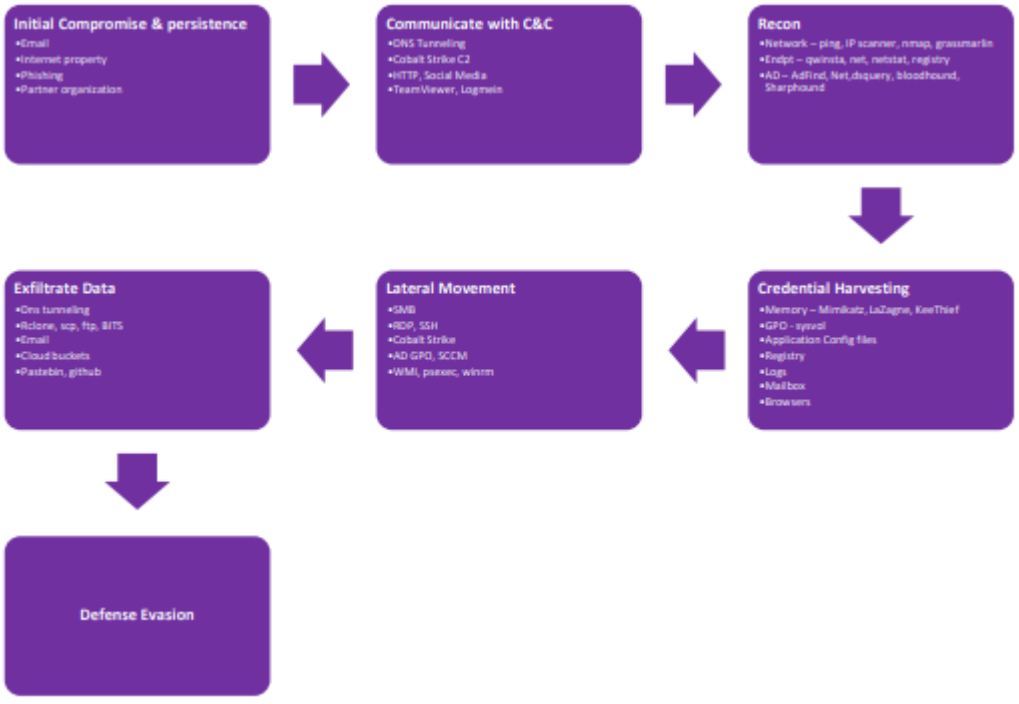
Double Extortion:

1. Pay to get access to your encrypted data.
2. Pay to prevent leaking sensitive data on the Internet.

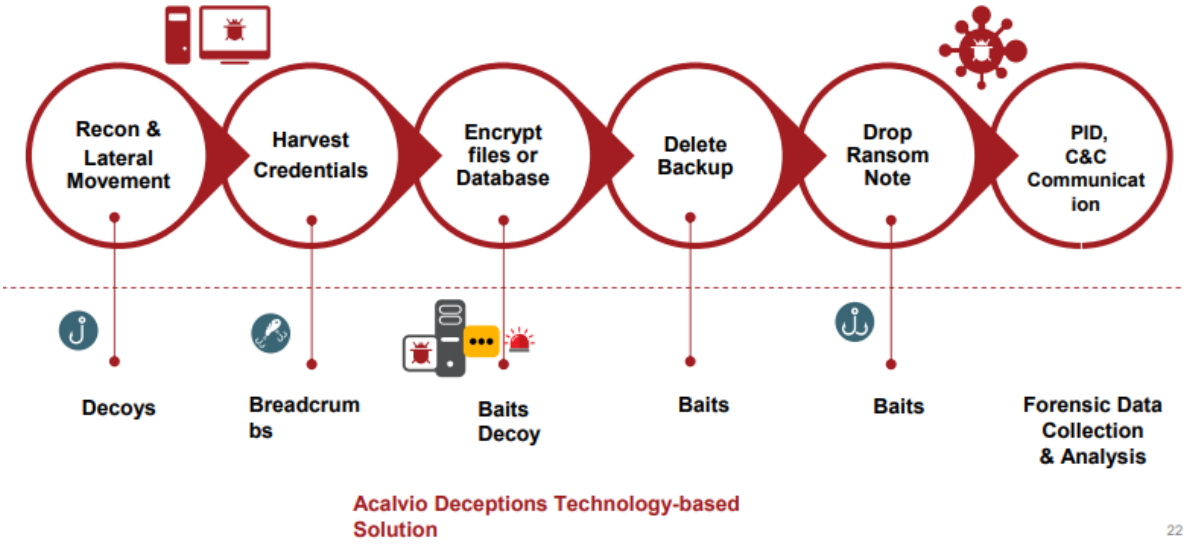
RaaS – Ransomware-as-a-Service:

- A SaaS service for ransomware to lower the skill requirements.
- Operators provide:
  - Infrastructure.
  - Service uptime.
  - Malware development.
  - Customer service.
  - Payment handling.
- Affiliates are responsible for spreading the ransomware.
- Operators also partner with peddlers like Emotet, Trickbot, etc.
- Operators gain scale.
- Affiliates focus on infiltrating network penetration (Enterprise).

Attack Flow



Ransomware Kill Chain



Four major approaches for Defense Evasion/Bypass:

1. **Disable Security Software:**
    - Shutdown or uninstall agent.
    - Examples:
        - REvil, Snatch.
        - Netwalker & DoppelPaymer disable AV.
    - Safe boot.
2. **Interrupt Agent Communication:**
    - Change firewall.
    - Example:
        - EKANS ransomware in ICS world.
3. **Hide from Security Software:**
    - Fileless/Memory Resident.
    - Bring your own VM (example: Ragnar Locker, Maze).
    - Obfuscation (example: PowerShell).
    - Deception.
    - Disable/Delete Monitoring.

- Proxy Execution.
- Living-off-the-Land tools.

4. **Lock out Incident Response (IR) and Defenders:**
    - Change passwords.

The Conficker Worm

- According to the Microsoft Security Bulletin MS08-067:
    - At worst, an infected machine could be taken over by the attacker, meaning by the human handlers of the worm.
    - More commonly, the worm disabled the Automatic Updates feature of the Windows platform.
    - The worm also made it impossible for the infected machine to carry out DNS lookups for the hostnames that correspond to antivirus software vendors.
    - The worm could also lock out certain user accounts. This was made possible by the modifications the worm made to the Windows registry.
- The worm was supposed to cause a major breakdown of the internet on April 1, 2009, but nothing happened.

- Speculation is that the worm was let loose by government organizations to test its power to propagate using what is now known as the “ms08-67 vulnerability” of the Windows machines of that era.
- This speculation is reinforced by the fact that the Stuxnet worm was let loose in 2010 shortly after Conficker; both shared many similarities.
- Stuxnet was used successfully to sabotage the nuclear program of a country.

Role of svchost.exe:

- Research determined that the worm infection spread by exploiting a vulnerability in the executable svchost.exe on a Windows machine.
- Svchost.exe is fundamental to the functioning of the Windows platform:
  - The always-running process that executes the svchost.exe file facilitates the execution of dynamically-linkable libraries (DLLs) that different applications reside in.
  - A program stored as a DLL cannot run on a stand-alone basis and must be loaded by another program.
  - Svchost.exe replicates itself for each DLL that needs to be executed.

How did the Conficker worm get to a computer?:

- A machine running a pre-patched version of the Windows server service svchost.exe could be infected because of a vulnerability with regard to how it handled remote code execution needed by the RPC (remote procedure call) requests coming in through port 445.
- Port 445 is assigned to the resource-sharing SMB protocol that is used by clients to access networked disk drives on other machines and other remote resources in a network.
- If a machine allowed for remote code execution in a network, perhaps because it made some resources available to clients, it would be open to infection through this mechanism.
- When such a machine received a specially crafted string on its port 445, the machine would:
  - Download a copy of the worm using the HTTP protocol from another previously infected machine and store it as a DLL file.

    - Execute a command to get a new instance of the svchost process to host the worm DLL.
    - Enter appropriate entries in the registry so that the worm DLL was executed when the machine was rebooted.
    - Give a randomly constructed name to the worm file on the disk.
- Once a machine was infected, the worm could drop a copy of itself (usually under a different randomly constructed name) in the hard disks on the other machines mapped in the previously infected machine ("network shares").
- The worm could also drop a copy of itself as the autorun.inf file in USB-based removable media such as memory sticks. This allowed the worm copy to execute when the drive was accessed (if Autorun was enabled).

Additional Information:

- What was the probability that a Windows machine at a particular IP address would be targeted by an unrelated infected machine?
    - Based on the reports on the frequency with which honeypots were infected, it would seem that a random machine connected to the internet was highly likely to be infected.
- It was suspected that the human handlers of the worm could communicate with it. This raised the question of how these humans managed to do so without leaving a trace as to who they were and where they were located.
- Microsoft had offered a $250,000 bounty for apprehending the culprits.

Challenges and Remediation:

- Could one imagine that several of the infected peers working in concert could cause internet disruptions that could be beyond the capabilities of the individual hosts?
- The worm cleverly prevented an automatic download of the latest virus signatures from the antivirus software vendors by altering the DNS software on the infected machine.
- Restoring an infected machine to good health was a challenge due to the worm's capability of resetting system restore points, rendering the system recovery impossible.

STUXNET WORM

Worms have generally been programmed to attack

- personal computers, particularly the computers running the Windows operating systems,
- for such nefarious purposes as stealing credit-card or bank information,
- sending out spam,
- mounting coordinated denial-of-service attacks on enterprise machines, etc

Stuxnet, on the other hand, was designed specifically to attack a particular piece of industrial software known as SCADA

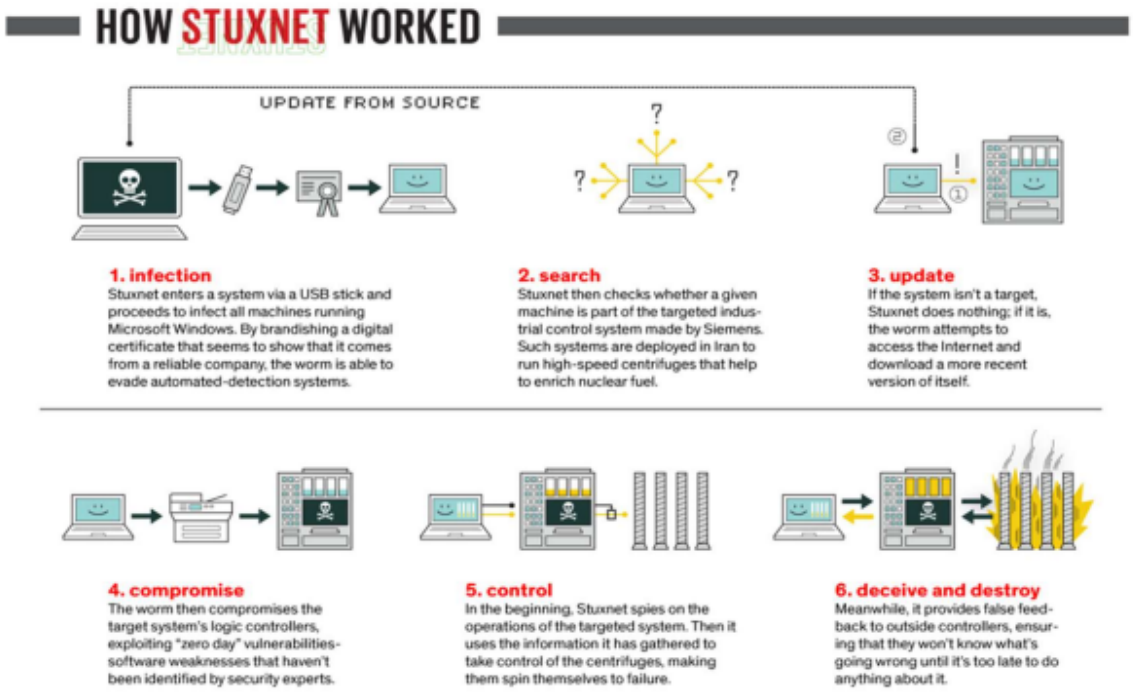
SCADA - Supervisory Control and Data Acquisition - A pipeline used for factory process control automation.

It is used in nuclear power plants and with web based SCADA you can control from WAN

The STUXNET worm, discovered in 2010, was a highly sophisticated cyberweapon designed to sabotage Iran's nuclear program by targeting its uranium enrichment facilities. Here are some key points about the STUXNET worm:

- **Designed to Target Siemens SCADA Systems:** The worm was designed to target the Siemens supervisory control and data acquisition (SCADA) systems, which were used for process control in Iran's nuclear facilities.
- **Manipulation of Sensor Data:** Once the worm infiltrated the SCADA systems, it would manipulate the data sent by the sensors to the central monitors. This manipulation was done to hide the fact that the centrifuges were operating at abnormal levels.
- **Sabotage of Centrifuges:** STUXNET specifically targeted the frequency converters used to control the centrifuge speeds. It would manipulate these converters to increase the centrifuge speeds to levels that would cause them to self-destruct eventually.
- **Designed for Precision:** The worm was designed with remarkable precision to target only specific systems. It was engineered to spread quietly and go undetected until it had achieved its destructive goal.

STUXNET represented a new era in cyber warfare, showcasing the potential for highly sophisticated cyber weapons to cause physical damage to critical infrastructure.



The STUXNET worm exploited three main vulnerabilities in the Windows operating system:

1. **MS10-061 Vulnerability:** This vulnerability was related to the print spooler service in Windows platforms. It allowed the worm to spread within a network of computers that shared printer services.
2. **MS08-067 Vulnerability:** This vulnerability was related to remote code execution. If a machine was running a pre-patched version of the Windows Server Service `svchost.exe` and received a specially crafted string on its port 445, it would download a copy of malicious code using the HTTP protocol from another previously infected machine and store it as a DLL, etc.
3. **MS10-046 Vulnerability:** This vulnerability in the Windows shell allowed the worm to propagate via removable disk drives. If a user clicked on the icon of a specially crafted shortcut displayed on the screen, remote code execution could occur. This vulnerability was also known as the Windows shortcut vulnerability and related to the .LNK suffixed link files that served as pointers to actual .exe files.

Ransomware Worm

The worm crippled several hospitals and banks in countries that included U.K, Canada, Colombia, and others. It also afected large coperations like FedEx and Nissan.

The WannaCry ransomware worm, which wreaked havoc in 2017, exploited two main methods of propagation:

1. **Exploiting SMB Vulnerability (MS17-010):**
    - WannaCry directly exploited a vulnerability in version 1.0 implementation of the Microsoft Server Message Block (SMB) protocol. This vulnerability was identified in Microsoft Security Bulletin MS17-010.
    - EternalBlue is an exploit of Microsoft's implementation of their Server Message Block (SMB) protocol
    - This vulnerability, originally discovered by the United States' National Security Agency (NSA), allowed WannaCry to spread rapidly across networks.
2. **Utilizing the DoublePulsar Backdoor:**
    - WannaCry could also propagate through the DoublePulsar backdoor if it was already installed on the host being attacked.
    - DoublePulsar is used to provide support for cross-platform (Microsoft Windows, macOS, and other Unix systems) sharing of files and printers.
    - Ports 139 and 445 are assigned to the SMB protocol, and WannaCry would exploit these ports for propagation.

Given these vulnerabilities, protecting against WannaCry and similar SMB-related malware involves:

- Closing SMB ports (139 and 445) in routers or using firewalls on individual computers to prevent malware propagation.
- Considering the need for file and printer sharing within a LAN, but recognizing that the need for such sharing across the broader internet is highly unlikely.
- For small office/home office (SOHO) networks, closing SMB ports in routers is crucial, as these ports are highly likely to attract malware and are unnecessary for communication with hosts outside the LAN.

Additionally, it's important to note that Windows SMBv1 implementation is vulnerable to a buffer overflow through the processing of File Extended Attributes (FEAs) in the kernel function `srv!SrvOs2FeaListToNt()`, which calls `srv!SrvOs2FeaListSizeToNt()` to calculate the size of the received FEA LIST before converting it to an NTFEA (Windows NT FEA) list.

WannaCry's rapid spread in 2017 demonstrated how vulnerable hosts can be quickly located on the internet through random scans of IP blocks. The worm would hop from one host to another through these scans, looking for hosts with open port 445. Before installing itself on a targeted host, WannaCry would ensure that the host had not previously been infected by the same worm.

Double Pulsar

DoublePulsar is believed to be one of several implants created by the NSA for penetrating foreign networks. Here are some key points about DoublePulsar:

- **Nature of Implants:** DoublePulsar is an implant, which is a polymorphic virus that may exhibit one behavior when first installed on a host and a different behavior when the same host is subsequently rebooted.
- **Behavior:** For example, upon its first installation, an implant may simply write a piece of malicious code in the boot sector of a disk. After a subsequent reboot, this code could execute automatically, potentially causing the host to freeze up or exhibit other malicious behavior.
- **Role in WannaCry Propagation:** When WannaCry targets a host, it first checks whether DoublePulsar is already installed on the target host. If DoublePulsar is present, WannaCry instructs DoublePulsar to pull it in. If DoublePulsar is not present, WannaCry pushes itself into the victim machine directly through the SMB port 443 while simultaneously creating the DoublePulsar backdoor on the victim host.
- **Persistence:** Even after WannaCry is removed from the victim host, DoublePulsar remains installed.

To detect DoublePulsar, Luke Jennings has developed a detection tool in Python that can be executed either as a stand-alone script or through the Nmap port scanner. Here are the links to the two versions of the tool:

1. DoublePulsar Detection Script
1. Nmap DoublePulsar Detection Script

Researcher Marcus Hutchins[55][56] discovered the kill switch domain hardcoded in the malware.[57][58][59] Registering a domain name for a DNS sinkhole stopped the attack spreading as a worm, because the ransomware only encrypted the computer's files if it was unable to connect to that domain, which all computers infected with WannaCry before the website's registration had been unable to do. While this did not help already infected systems, it severely slowed the spread of the initial infection and gave time for defensive measures to be deployed worldwide, particularly in North America and Asia, which had not been attacked to the same extent as elsewhere.[60][61][62][63][64] On 14 May, a first variant of WannaCry appeared with a new and second[65] kill-switch registered by Matt Suiche on the same day. This was followed by a second variant with the third and last kill-switch on 15 May, which was registered by Check Point threat intelligence analysts.[66][67] A few days later, a new version of WannaCry was detected that lacked the kill switch altogether.

It was discovered that Windows encryption APIs used by WannaCry may not completely clear the prime numbers used to generate the payload's private keys from the memory, making it potentially possible to retrieve the required key if they had not yet been overwritten or cleared from resident memory. The key is kept in the memory if the WannaCry process has not been killed and the computer has not been rebooted after being infected.[76] This behaviour was used by a French researcher to develop a tool known as WannaKey, which automates this process on Windows XP systems.[77][78][79] This approach was iterated upon by a second tool known as Wanakiwi, which was tested to work on Windows 7 and Server 2008 R2 as well.[80]

Within four days of the initial outbreak, new infections had slowed to a trickle due to these responses

Threat Modeling

Threat modeling is a structured approach to identifying and prioritizing potential security threats and vulnerabilities in a system or application. Here are some ways to find security issues, including threat modeling:

1. **Static Analysis of Code:** Analyzing the source code without executing it to identify security vulnerabilities, such as coding errors, insecure configurations, and potential weaknesses.

2. **Fuzzing or Other Dynamic Testing:** Dynamic testing techniques involve executing the software with various inputs to find vulnerabilities, including fuzzing, where the software is bombarded with invalid, unexpected, or random data.
3. **Penetration Testing (Pen Test) / Red Team Exercises:** Simulating real-world attacks on the system to identify security weaknesses and vulnerabilities from an attacker's perspective.
4. **Bug Reports After Release:** Waiting for reports from users or security researchers after the software has been released can help identify vulnerabilities and security issues that were missed during development.
5. **Threat Modeling:** Identifying and prioritizing potential security threats and vulnerabilities in a system or application. It involves thinking about security issues early in the development process, understanding requirements better, and creating a structured model to represent the system from a security perspective.

What is a Threat?

- A threat is a potential event that could exploit vulnerabilities in a system and cause harm.
- It differs from a vulnerability, which is a weakness in the system that could be exploited, and a risk, which is the likelihood and impact of a threat exploiting a vulnerability.
- A model is a simplified representation of a software process. Each model represents a process from a specific perspective, and some methodologies are known as the Software Development Life Cycle (SDLC).

Problems Associated with "Think Like an Attacker" Mentality:

- It can lead to a focus on only the most obvious and well-known attack vectors, overlooking less conventional but equally dangerous threats.
- It may result in overlooking critical aspects of the system's functionality or its intended use.

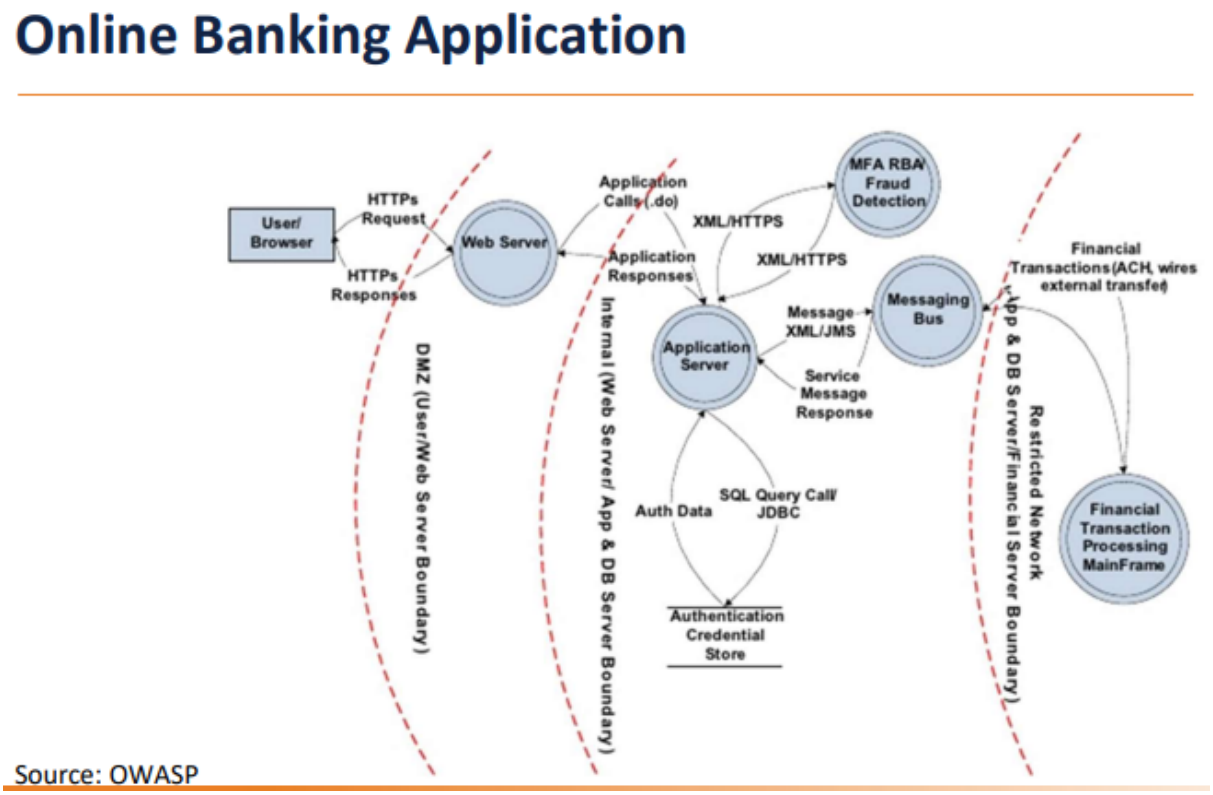
Problems Associated with Starting from Assets as an Approach to Threat Modeling:

- Assets-focused threat modeling may overlook potential threats that don't directly target identified assets.

- It may lead to a narrow focus, neglecting broader system vulnerabilities that could be exploited.

What You Learn by Making an Asset List:

- An engineering approach is needed for threat modeling to be predictable, reliable, and scalable.
- Threat modeling shouldn't rely on the expertise of one person but should be a collaborative effort.
- By creating an asset list, you gain a deeper understanding of the system, its components, and their interdependencies, allowing for more concrete and testable threat modeling.



Threat modeling begins by focusing on four key questions:

1. **What are you building?**
    - Define the scope of the Threat Model.
    - To understand the application you are building, consider techniques such as:

    - Architecture diagrams
    - Dataflow transitions
    - Data classifications
2. **What can go wrong?**
    - Identify potential threats and vulnerabilities within the defined scope.
    - Consider different attack vectors, potential weaknesses, and threats to the system.
3. **What should you do about those things that can go wrong?**
    - Develop mitigations and countermeasures to address identified threats and vulnerabilities.
    - Implement security controls to reduce the likelihood and impact of security incidents.
4. **Did you do a decent job of analysis?**
    - Evaluate the effectiveness of the threat modeling process.
    - Assess whether potential threats and vulnerabilities have been adequately addressed.

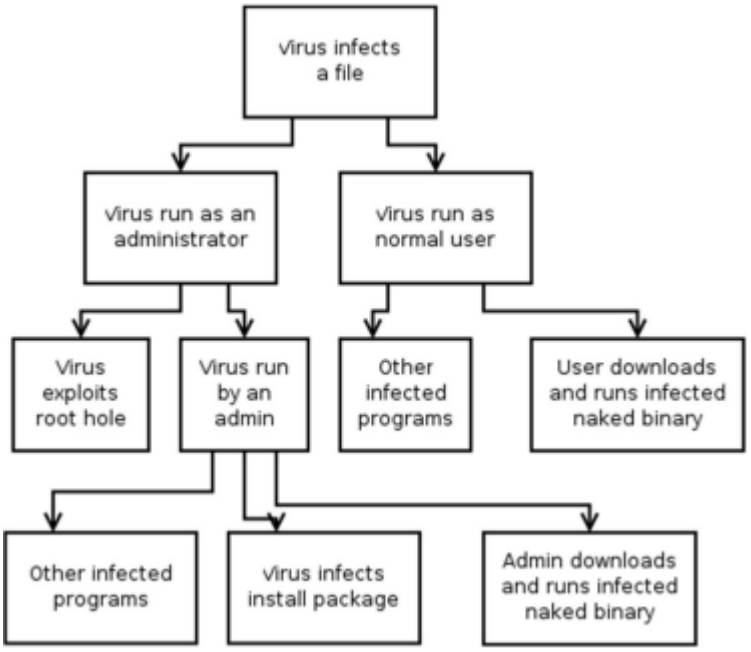
What are you building?

- Define the scope of the Threat Model by understanding the application you are building.
- Helpful techniques include:
    - Architecture diagrams: Visual representations of the system's architecture, including components and their interactions.
    - Dataflow transitions: Identifying how data moves through the system, from input to processing to output.
    - Data classifications: Identifying and classifying different types of data based on sensitivity and importance.

Trust Boundaries

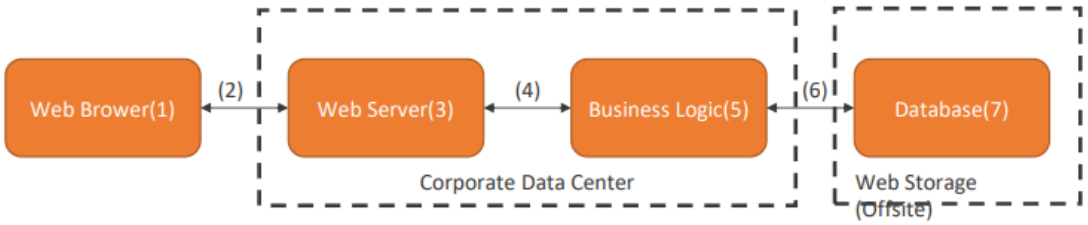
- A trust boundary is any place where various principals come together, where entities with different privileges interact.

- Adding boundaries to show "who controls what" is a simple way to add Trust Boundaries.
- Effective threat modeling requires making boundaries explicit, as they are sometimes implicit in development.



The definitive 'threat model' often sought by developers may take the form of a document or spreadsheet detailing the entire system alongside an exhaustive list of threats, vulnerabilities, and attack trees. However, there are some problems associated with this approach:

1. **System Changes:** The first problem is that systems change over time, and it's nearly impossible to identify all threats exhaustively.
2. **Separate Artifact:** Producing a sepaarate artifact detailing threats and vulnerabilities doesn't necessarily help in figuring out the necessary steps to combat those threats.



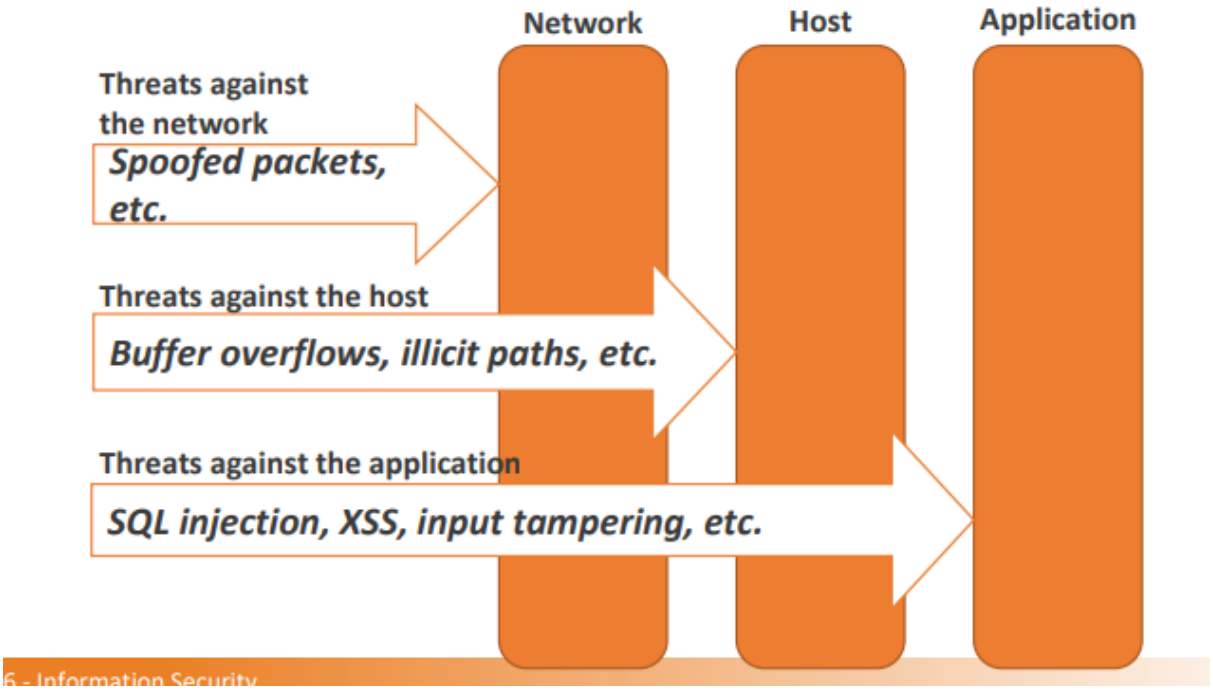
Instead of focusing on building an elaborate and formal 'threat model', it's recommended to follow a structured approach to threat modeling. Here are some steps you can take:

1. **Identify Scope:** Understand what you are building by defining the scope of the threat model. Techniques such as architecture diagrams, dataflow transitions, and data classifications can be helpful.
2. **Trust Boundary vs. Attack Surface:**
    - **Trust Boundary:** Any place where various principals come together, representing interaction between entities with different privileges.
    - **Attack Surface:** The areas of a system that are accessible to potential attackers. A system that exposes many interfaces has a larger attack surface than one with few APIs or interfaces.
3. **Addressing Threats:**
    - Go through the list of identified threats and address each one.
    - There are four types of actions you can take against each threat: **META**
        - Mitigate it
        - Eliminate it
        - Transfer it
        - Accept it
4. **Focus on Assets:**
    - Understand what assets are valuable to the business and need protection.
    - Consider whether an asset is something an attacker wants, something you want to protect, or a stepping stone for attackers.
5. **Engineering Approach:**
    - Need an engineering approach that is predictable, reliable, and scalable to large products.
    - Avoid dependency on one brilliant person.

By following these steps, you can effectively identify and address security threats and vulnerabilities in your system while also ensuring that your approach is practical and scalable.

Here are some ways to find security issues:

1. **Static Analysis of Code:**
    - Static analysis is a method of debugging by automatically examining source code before a program is run. It involves analyzing a set of code against a set (or multiple sets) of coding rules.
    - Find Security Bugs is a SpotBugs plugin for security audits of Java web applications and Android applications. It can detect 128 different vulnerability types, including Command Injection, XPath Injection, SQL/HQL Injection, XXE, and Cryptography weaknesses. SpotBugs is a static analysis tool that targets Java but also works with Groovy, Scala, and Kotlin projects.
2. **Pen Test/Red Team:**
    - A penetration test, colloquially known as a pen test, is an authorized simulated cyber attack on a computer system, performed to evaluate the security of the system.
3. **Wait for Bug Reports After Release:**
    - A software bug is an error, flaw, failure, or fault in a computer program or system that causes it to produce an incorrect or unexpected result or to behave in unintended ways. Waiting for bug reports after the release allows for the identification and resolution of security issues reported by users.
4. **Threat Modeling:**
    - Threat modeling involves thinking about security issues early in the development process and understanding requirements better to prevent bugs in the code.



Threat	Examples
Information gathering	Port scanning
	Using trace routing to detect network topologies
	Using broadcast requests to enumerate subnet hosts
Eavesdropping	Using packet sniffers to steal passwords
Denial of service (DoS)	SYN floods
	ICMP echo request floods
	Malformed packets
Spoofing	Packets with spoofed source addresses

Threat	Examples
Arbitrary code execution	Buffer overflows in ISAPI DLLs (e.g., MS01-033)
	Directory traversal attacks (MS00-078)
File disclosure	Malformed HTR requests (MS01-031)
	Virtualized UNC share vulnerability (MS00-019)
Denial of service (DoS)	Malformed SMTP requests (MS02-012)
	Malformed WebDAV requests (MS01-016)
	Malformed URLs (MS01-012)
	Brute-force file uploads
Unauthorized access	Resources with insufficiently restrictive ACLs
	Spoofing with stolen login credentials
Exploitation of open ports and protocols	Using NetBIOS and SMB to enumerate hosts
	Connecting remotely to SQL Server

Threat	Examples
SQL injection	Including a DROP TABLE command in text typed into an input field
Cross-site scripting	Using malicious client-side script to steal cookies
Hidden-field tampering	Maliciously changing the value of a hidden field
Eavesdropping	Using a packet sniffer to steal passwords and cookies from traffic on unencrypted connections
Session hijacking	Using a stolen session ID cookie to access someone else's session state
Identity spoofing	Using a stolen forms authentication cookie to pose as another user
Information disclosure	Allowing client to see a stack trace when an unhandled exception occurs

What are you building

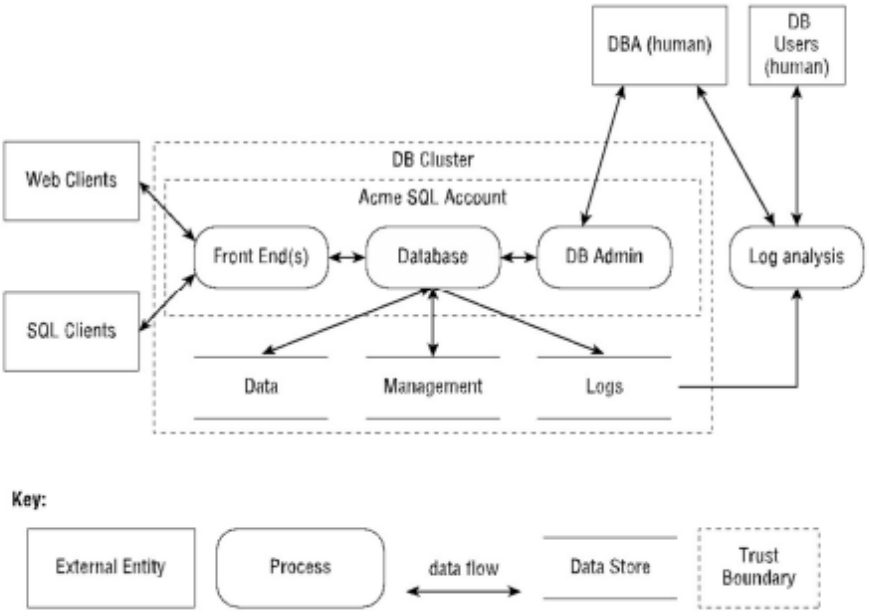
To understand what you are building and create a model of the software, system, or technology, consider the following:

1. **Create a Model:**

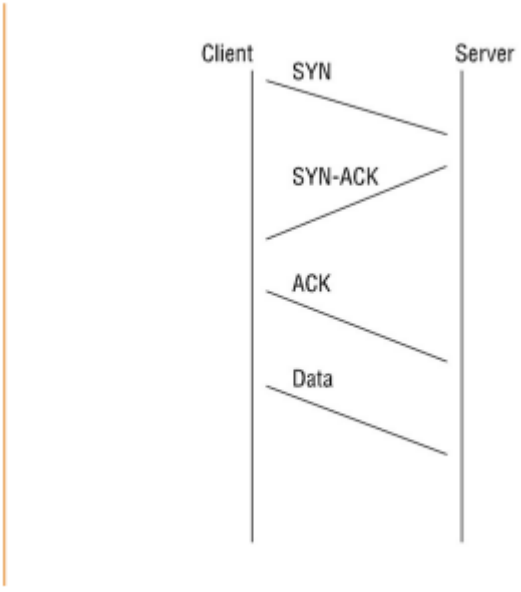
- A model abstracts away the details so you can look at the whole picture.
- Diagrams are a key approach to creating a model.
- Whiteboard diagrams are a great way to start.
- Software models for threat modeling usually focus on data flows and boundaries.
- Useful modeling methods include Data Flow Diagrams (DFDs), swim lanes, state machines, and others.

2. **Trust Boundaries:**
    - Trust boundaries are everywhere two or more principals interact.
    - Sometimes left implicit in development, effective threat modeling requires making boundaries explicit.
    - All interesting boundaries are semi-permeable and need to be enforced in some way.
    - Formal methods such as isolation, type safety, policy languages, and reference monitors/kernels help build boundaries.
3. **Brainstorming Your Threats:**
    - Brainstorming is a traditional way to enumerate threats.
    - Bring together a set of experienced experts, give them a way to take notes, and let them brainstorm.
    - The quality of the brainstorm is bounded by the experience of the brainstormers and the amount of time spent brainstorming.
    - Brainstorming involves a period of idea generation, followed by a period of analyzing and selecting the ideas.
    - Bring together a diverse group of experts with a broader set of experience for better results.
4. **Modeling Methods:**
    - **DFD (Data Flow Diagram):**
        - A DFD is a way of representing the flow of data in a process or system.

- It abstracts programs into processes, data stores, data flows, and external entities.

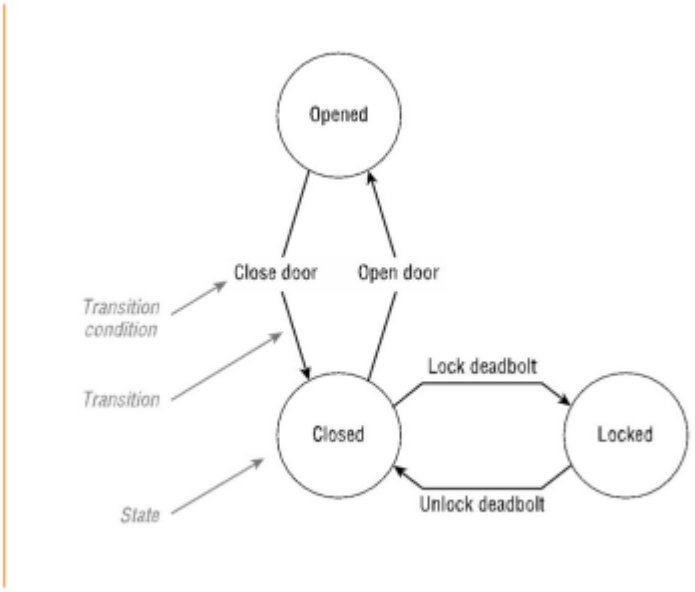


- **Swim Lane Diagrams:**
  - Show two or more entities communicating, each "in a lane."
  - Useful for network communication and showing implicit boundaries between lanes.



- **State Machines:**

- Helpful for considering what changes security state, such as from unauthenticated to authenticated, or from user to root/admin.



5. **What to Include in a Diagram:**
   - Show the events that drive the system.
   - Show the processes that are driven.
   - Determine what responses each process will generate and send.
   - Identify data sources for each request and response.
   - Identify the recipient of each response.
   - Focus on scope and ignore the inner workings.
   - Ask if something will help you think about what goes wrong or help you find threats.
   - Labels in diagrams should be short, descriptive, and meaningful.
   - Colors can be helpful, such as green for trusted, red for untrusted, and blue for what's being modeled.

Threat Models

Various Threat modeling techniques

- STRIDE

- DREAD
- Attack Trees
- Attack Libraries

STRIDE

Introduction:

- Invented in 1999 and adopted by Microsoft in 2002, STRIDE is currently the most mature threat modelling method.
- STRIDE is a mnemonic that helps identify different types of security threats.

Mnemonic:

- **S**poofing
- **T**ampering
- **R**epudiation
- **I**nformation Disclosure
- **D**enial of Service
- **E**levation of Privilege

Evolution:

- STRIDE has evolved over time to include new threat-specific tables and the variants STRIDE-per-Element and STRIDE-per-Interaction.

STRIDE (Review)

Threat	Property Violated	Definition	Example
Spoofing	Authentication	Pretending to be something or someone else.	Pretending to be any of Bill Gates, Paypal.com, or ntdll.dll; Packets with spoofed source addresses
Tampering	Integrity	Modifying data or code	Modifying a DLL on disk or DVD, or a packet as it traverses the network
Repudiation	Non-repudiation	Claiming to have not performed an	“I didn’t send that email,” “I didn’t modify that file,” “I

		action.	certainly didn’t visit that web site, dear!”
Information Disclosure	Confidentiality	Exposing information to someone not authorized to see it	Allowing someone to read the Windows source code; publishing a list of customers to a website
Denial of Service	Availability	Deny or degrade service to users	Crashing Windows or a website, sending a packet and absorbing seconds of CPU time, or routing packets into a black hole.
Elevation of Privilege	Authorization	Gain capabilities without proper authorization	Allowing a remote Internet user to run commands is the classic example, but going from a limited user to admin is also EoP.

Spoofing

Definition:

- Spoofing involves pretending to be something or someone else.

Examples:

1. **Identity Spoofing:**
    - Illegally accessing and using another user’s authentication information, such as username and password.
    - Example: Phishing attack to fool the user into sending credentials to a fake site.
2. **Packet Spoofing:**
    - Sending packets with spoofed source addresses.

Spoofing on the Local Machine:

Threat Example	What the Attacker Does	Notes/Examples
Spoofing a process	Creates a file before the real process	Then your process relies on it
	Abuses names	Create a version of “sudo” and alter PATH
Spoofing a filename	Creates a file in the local directory	Library, executable or config file
	Creates a link, changes it	Also called ‘race condition’ or TOCTOU
	Creates many files in a target directory	Code can easily create all possible /tmp/foo.random

Spoofing Over a Network:

Threat Example	What the Attacker Does	Notes/Examples
Spoofing a machine	ARP spoofing	
	IP spoofing	
	DNS spoofing	
	DNS compromise	Can be at the TLD, registrar or DNS server
	IP redirection	
Spoofing a person	Take over account	“Stranded in London”
	Set the display name	
Spoofing a role	Declares themselves to be that role	Sometimes opening a special account, setting up a domain/website, other “verifiers”

Understanding Authentication: Addressing Spoofing:

- Authentication is the process of proving or showing something to be true or genuine.
- Applies to all sorts of things such as programs or libraries on disk, remote machines, and people.

Tactics for Authentication

Local:

- Leverage the OS/program (database, web server, etc.).
- Note: Defaults are not always secure.

Remote Machines:

- Use cryptographic methods (more reliable).
- Perform consistency checking of DNS, IP, route (less reliable).

Cryptographic Key Exchange:

- Use technologies like DNSSec, PKI, etc., which involve trust delegation.
- Manual key exchange can be expensive but sometimes worthwhile for existing business relationships.

Developer Ways to Address Spoofing

- **Leverage the OS:**
  - Use full pathnames (`open("foo.txt")` should find the correct file).
  - Make pathnames canonical.
    - Resolve links including `../` or symlinks.
    - Remove `%20` or other encoding.
  - Check permissions.
  - Note: Shared directories are usually troublesome.
- **Cryptographic Identifiers & Validation:**
  - Use cryptographic identifiers and validation techniques.

Operational Ways to Address Spoofing

- Improving local (on-system) name resolution when the code is done can be difficult.
- Use SSH, IPSec, or other crypto tunneling to reduce spoofing issues over the network.

Technologies for Addressing Spoofing

Authenticating Computers:

- Use technologies like IPSec, DNSSec, SSH Host keys, Kerberos, Windows Domain authentication, PKI with SSL/TLS.

Authenticating Bits (Files, Messages, etc.):

- Use digital signatures.
- Use hashes (appropriately managed).

Multi-factor Authentication:

1. Something you know, like a password.
2. Something you have, like an access card.
3. Something you are (or are measured to be):
    - Biometrics (fingerprints, vein patterns, photographs).
4. Someone you know who can authenticate you.

Note:

- Traditional multi-factor authentication usually involves more than one factor from the list above.
- Some people consider channels as a factor.
- All these methods should be thoroughly threat modeled.

Tampering with Data

Data tampering involves the malicious modification of data.

Examples include:

- Unauthorized changes made to persistent data, such as that held in a database.
- Alteration of data as it flows between two computers over an open network, such as the Internet.

Example: Integrity of a message compromised to change parameters or values.

Tampering with a File

Threat Example	What the Attacker Does	Notes/Examples
Modifying a file...	... which you own and you rely on	
	... which they own and you rely on	
Modifying a file on a server...	...you own	
	...they own (or take over)	
Modifies links or redirects		Redirects are super-common on the web, and often rot away

Tampering with Memory

Threat Example	What the Attacker Does	Notes/Examples
Modifying code	Changes your code to suit themselves	Hard to defend against if the attacker is running code inside the trust boundaries
Modifying data they've supplied	Supplies data to a pass by reference API, then changes it	Works because of TOCTOU issues
Time-of-check to time-of-use (TOC) TOU, TOCTTOU or TOC/TOU	Supplies data into a shared memory segment, then changes it	

Tampering with a Network

Threat Example	What the Attacker Does	Notes/Examples
Redirects the flow of data to their machine	Uses an attack at some network layer to redirect traffic	Pakistan/YouTube
Modifies data flowing over the network		Easier (and more fun) with wireless networks
	Uses network tampering to improve spoofing attacks	

Understanding Integrity - Tampering

Integrity means to interfere with something in order to cause damage or make unauthorized alterations.

Integrity can apply to data wherever it is, including:

- Disk
- Network
- Memory

Tactics for Integrity

System Defenses:

- Use permissions (operating system/program).

Cryptography Defenses:

- Use digital signatures.
- Use hashes/MACs.

Logging and Audit:

- These do not prevent but may deter.
- Generally used as a fallback or defense in depth.

Developer Ways to Address Integrity

- Use permissions as provided.
- Cryptography is required over a network.
- Implementing a permission system is hard. Lots of mistakes have been made & documented.

Operational Ways to Address Integrity

- Add additional protections:
    - Tripwire-like systems on the local machine.
    - Tunneling over the network.
- Good alert design is a prerequisite.
    - Too many alerts, people will be overwhelmed.

- Too few, they'll miss stuff.

Protect files with:

- Digital signatures.
- ACLs/permissions.
- Hashes.
- Windows Mandatory Integrity Control features.
- Unix immutability.

Protect network traffic with:

- SSL.
- SSH.
- IPSec.
- Digital signatures.

Repudiation

Threat Example	What the Attacker Does	Notes/examples
Repudiating an action	Claims to have not clicked	Maybe they did, maybe they didn't, maybe they're honestly confused
	Claims to not have received	1. Electronic or physical 2. Receipt is strange; does a client downloading email mean you've seen it? Did a network proxy pre-fetch images? Was a package left on a porch?
	Claims to be a fraud victim	
	Uses someone else's account	

Threat Example	What the Attacker Does	Notes/Examples
	Discovers there are no logs	
Modifies data flowing over the network	Puts data in the logs to confuse you	</tr></html>

Repudiation threats occur when users deny performing actions without any means for others to prove otherwise. For instance, in a system lacking the capability to trace prohibited operations, a user might perform an illegal action and later deny it. Non-repudiation, on the other hand, refers to a system's ability to counter such threats. For instance, when a user purchases an item, they might be required to sign for it upon receipt. This signed receipt serves as evidence that the user did indeed receive the package, thus preventing them from illegitimately claiming that the transaction was not completed. Non-repudiation involves tools and technologies that establish what happened, ideally to the satisfaction of all parties involved or impacted, bridging both business and technical levels.

Tactics for ensuring non-repudiation include:

1. **Fraud prevention:**
    - Preventing internal fraud such as embezzlement.
    - Preventing customer fraud.
2. **Logs:**
    - Keep logs for as long as possible.
    - Store logs securely.
3. **Cryptography:**
    - Use cryptographic techniques such as digital signatures to ensure the integrity and authenticity of transactions.
4. **Customer fraud prevention:**
    - Preventing account takeovers and abuse.
    - Managing stable and predictable customers.
    - Utilizing technologies and services such as validation services, customer history sharing, multi-merchant data, and purchase device tracking.

Ways developers can address non-repudiation include:

- **Logging business logic:** Log relevant business logic to track user actions.
- **Cryptography:** Implement cryptographic digital signatures, especially useful between business partners.

Operational strategies to address non-repudiation include:

- **Table-top exercises:** Use exercises to identify issues that may not be apparent from logs.
- **Scaling:** Implementing mechanisms to handle the scaling of logs, such as utilizing dedicated personnel and specialized tooling.

Technologies for addressing repudiation threats include:

- **Logs:**
  - Logging mechanisms.
  - Log analysis tools.
  - Secured log storage.
- **Digital signatures.**
- **Secure timestamps.**
- **Trusted third parties.**

Information Disclosure

Threat Example	What the Attacker Does	Notes/Examples
Extracts user data	Exploits bugs like SQL injection to read db tables	Can find this by looking to data stores, but here the issue is the process returning data it shouldn't
	Reads error messages	
Extracts machine secrets	Reads error messages	Cannot connect to database 'foo' as user 'sql' with password '&IO*(^&'
	Exploits bugs	"Heartbleed"

Information disclosure threats involve the exposure of information to unauthorized individuals. This could include scenarios such as users accessing files they weren't granted permission to, or intruders intercepting and reading data transmitted between two computers. For instance, an unencrypted message being intercepted off the network.

Sub-category	What the Attacker Does
Permissions	Take advantage of missing or inappropriate ACLs
	Take advantage of bad database permissions
	File files protected by obscurity
Security	Find crypto keys on disk or in memory
	Get data from logs/temp files
	Get data from swap files
	See interesting information in filenames/directory names
Network	See data traversing a network
Misc	Obtain device, boot in new OS

Understanding Confidentiality

Confidentiality ensures that information is only disclosed to authorized parties. It involves keeping secrets within data, which could be financial results, new product plans, private data entrusted to an entity, or even metadata.

Sub-category	What the Attacker Does
Network	Read data on a network
	Redirects traffics to enable reading data on the network
Metadata	Learns secrets by analyzing traffic
	Learns who talks to whom by watching the DNS
	Learns who talks to whom by analyzing social network information

Tactics for Confidentiality

To ensure confidentiality, various tactics can be employed:

- **On a system:**
  - Access Control Lists (ACLs)/permissions
  - Cryptography
- **Between systems:**
  - Cryptography
- **To hide the existence of information:**
  - Steganography

Developer Ways to Address Confidentiality

Developers can address confidentiality through:

- **Permissions/ACLs**
- **Cryptography:**
  - Data encryption (file on disk, email message)
  - Container encryption (volume encryption, email connections)
  - Proper key management is crucial, and it's important to remember that encryption alone doesn't provide authentication or integrity.

Operational Ways to Address Confidentiality

Operational strategies to address confidentiality include:

- Adding permissions/ACLs
- Volume encryption:
  - Protects data if the machine is stolen and powered down, although it doesn't protect against an attacker who breaks in.
- Network encryption (SSH, SSL, IPSec)

Technologies for Confidentiality

Technologies employed to ensure confidentiality include:

- **Protecting files:**
  - ACLs/Permissions
  - Encryption
  - Appropriate key management

- **Protecting network data:**
  - Encryption
  - Appropriate key management
- **Communication headers/act of communication:**
  - Mix networks
  - Onion routing
  - Steganography

Denial of Service

Denial of Service (DoS) attacks aim to deny service to valid users, such as making a web server temporarily unavailable or unusable. Protecting against certain types of DoS threats is essential to improve system availability and reliability. For instance, a DoS attack could flood a system with requests until the web server fails.

Threat Example	What the Attacker Does	Notes/Examples
Against a process	Absorb memory (ram or disk)	
	Absorb CPU	
	Uses a process as an amplifier	
	Against business logic	"Too many login attempts"
Against a data store	Fills the data store	
	Makes enough requests to slow the system	
Against a data flow	Consumes network resources	

Understanding Availability

Availability is the ability to meet a defined or implied Service Level Agreement (SLA). Attacks can target any system resource, including disk space, network bandwidth, or CPU usage. These attacks can be transient or might require manual intervention to resolve.

Tactics for Availability

To ensure availability, consider the following tactics:

- Ensure there are enough resources to serve requests.
- Implement proof of work mechanisms, such as high-cost proofs used in Bitcoin.
- Implement proof of communication mechanisms.

Developer Ways to Address Availability

Developers can address availability issues by:

- Avoiding fixed-size buffers that can lead to resource exhaustion.
- Considering the resources consumed per request and how many requests the system can handle.
- Being aware of clever attacks that can inflate resource usage.
- Planning for recovery in case of an attack.

Operational Ways to Address Availability

Operational strategies to address availability include:

- Implementing quotas to limit resource usage.
- Using elastic cloud systems to dynamically add more resources as needed.

Technologies for Addressing DoS

Technologies used to address DoS attacks include:

- Access Control Lists (ACLs) and filters to block malicious traffic.
- Implementing quotas such as rate limits, thresholding, and throttling.
- Designing systems for high availability.
- Having extra bandwidth to absorb sudden increases in traffic.
- Utilizing cloud services for scalability and redundancy.

Elevation of Privilege

Threat Example	What the Attacker Does	Notes/Examples
EoP Against process via corruption	Sends inputs the code doesn't handle properly	Very common, usually high impact
	Gains read/write access to memory	Writing memory more obviously bad
EoP via misused authorization checks		
EoP via buggy authorization checks		Centralizing checking makes consistency, correctness easier
EoP via data tampering	Modify bits on disk	

Elevation of privilege threats occur when an unprivileged user gains privileged access, allowing them to compromise or destroy the entire system. In such situations, attackers penetrate all system defenses and become part of the trusted system itself.

Understanding Authorization

Elevation of privilege is a class of authorization bypass threat. For instance, when an attacker effectively becomes part of the trusted system by gaining unauthorized access, it poses a significant threat. An example of this is when an attacker changes group membership.

Tactics for Authorization

To address elevation of privilege threats, consider the following tactics:

- **Limit the attack surface:**
  - Minimize the number of setuid programs.
  - Use sandboxes for network-exposed code.
  - Avoid running processes as root/admin.
  - Be aware of elevation paths for semi-privileged accounts.
- **Implement comprehensible, manageable permission systems.**

Developer Ways to Address Authorization

Developers can address authorization threats by:

- **Limiting the attack surface.**
- **Carefully defining the purpose and validation rules for inbound data.**

- **Defining what will be accepted, rather than what will be rejected.**
- **Rejecting bad input instead of attempting to sanitize it.**
- **Implementing looped canonicalization routines.**
- **Transforming data from one form to another (e.g., markdown to HTML).**

Operational Ways to Address Authorization

Operational strategies to address authorization threats include:

- **Defense in depth.**
- **Running each target as its own unique limited user.**
    - For example, in Unix, using the "nobody" account, which initially has limited privileges. However, it's crucial to ensure that over time, this account does not accumulate unnecessary privileges.

Use of STRIDE

Using the STRIDE model, we can identify and address potential threats to the system. Each letter in STRIDE represents a different type of threat:

- **S**poofing identity
- **T**ampering with data
- **R**epudiation of actions
- **I**nformation disclosure
- **D**enial of service
- **E**levation of privilege

Applying STRIDE to the Model

- **Spoofing Identity:**
    - How could an attacker spoof user identity to gain unauthorized access to the system?
    - How could an attacker pretend to be a legitimate user or system component?
- **Tampering with Data:**
    - How could an attacker tamper with data in transit or at rest?

    - How could an attacker modify data to cause a system malfunction or gain unauthorized access?
- **Repudiation of Actions:**
    - How could an attacker perform actions without leaving a trace?
    - How could an attacker deny performing certain actions?
- **Information Disclosure:**
    - How could an attacker gain access to sensitive information?
    - How could an attacker intercept data in transit between system components?
- **Denial of Service:**
    - How could an attacker overload system resources to deny service to legitimate users?
    - How could an attacker disrupt system functionality?
- **Elevation of Privilege:**
    - How could an attacker gain elevated privileges within the system?
    - How could an attacker exploit vulnerabilities to escalate their privileges?

What Can Go Wrong?

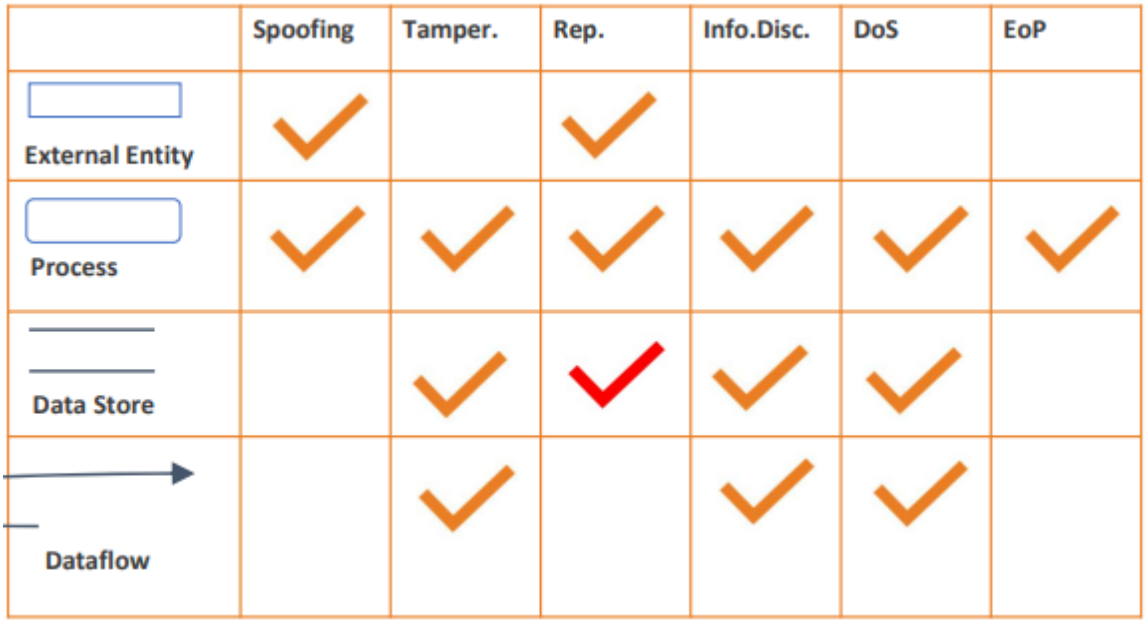
- Track issues as they are discovered:
    - Identify potential attack vectors (e.g., "attacker could pretend to be a client & connect").
- Track assumptions:
    - Verify assumptions made about system security (e.g., "I think that connection is always over SSL").
- Use these lists as inputs to determine how to mitigate potential threats.

STRIDE Variants

- **STRIDE per Element:**
    - Analyze each system element for potential STRIDE threats.
- **STRIDE per Interaction:**

- Analyze interactions between system elements for potential STRIDE threats.

- **Elevation of Privilege Game:**
  - Use training, structured analysis, and execution to address elevation of privilege threats effectively.



This is Microsoft's chart; it may not be the issues you need to worry about (privacy)

#	ELEMENT	INTERACTION	S	T	R	I	D	E
1	Process (Contoso)	Process has outbound data flow to data store.	x			x		
2		Process sends output to another process.	x		x	x	x	x
3		Process sends output to external interactor (code).	x		x	x	x	
4		Process sends output to external interactor (human).			x			
5		Process has inbound data flow from data store.	x	x			x	x
6		Process has inbound data flow from a process.	x		x		x	x
7		Process has inbound data flow from external interactor.	x				x	x
8	Data Flow (com-mands/ responses)	Crosses machine boundary		x		x	x	
9	Data Store (database)	Process has outbound data flow to data store.		x	x	x	x	
10		Process has inbound data flow from data store.			x	x	x	
11	External Interactor (browser)	External interactor passes input to process.	x		x	x		
12		External interactor gets input from process.	x					

Attackers Respond to Your Defenses:

- Start looking for threats at the beginning of your project:
  - Create a model of what you're building.
  - Conduct a first pass to identify potential threats.
- Dig deeper as you work through features:
  - Consider how threats apply to your mitigations.
- Check that your design and model match as you get close to shipping.

Playing Chess:

- The ideal attacker will follow the road you defend:
  - Ideal attackers are like spherical cows—they're a useful model for some things.
- Real attackers will go around your defenses.

- Your defenses need to be broad and deep.

Orders of Mitigation:

- Window smashing is a first-order threat, cutting alarm wire is a third-order threat.
- Don't get stuck arguing about orders:
  - Focus on the interplay between mitigations and further attacks.

Order Threat Mitigation

1st Order: Window smashing → Reinforced glass

2nd Order: Window smashing → Alarm

3rd Order: Cut alarm wire → Heartbeat signal
Cryptographic signal integrity

4th Order: Fake heartbeat

How to Approach Software:

- Depth First:
  - It's the most fun and "instinctual."
  - Keep following threats to see where they go.
  - Can be useful for skill development, promoting "flow."
- Breadth First:
  - It's the most conservative use of time.
  - Most likely to result in good coverage.

Order	Threat	Mitigation
1st	Window smashing	Reinforced glass
2nd	Window smashing	Alarm
3rd	Cut alarm wire	Heartbeat signal
4th	Fake heartbeat	Cryptographic signal integrity

Diagram Element	Threat Type	Threat	Bug ID
Data flow #4, web server to business logic	Tampering	Add orders without payment checks	4553 “Need integrity controls on channel”
	Info disclosure	Payment instruments sent in clear	4554 “need crypto” #PCI

Threat Type	Diagram Element(s)	Threat	Bug ID
Tampering	Web browser	Attacker modifies our JavaScript order checking	4556 “Add order-checking logic to server”
	Data flow #2 from browser to server	Failure to authenticate	4557 “Add enforce HTTPS everywhere”

Both are fine, help you iterate over diagrams in different ways

Example Assumption Tracking:

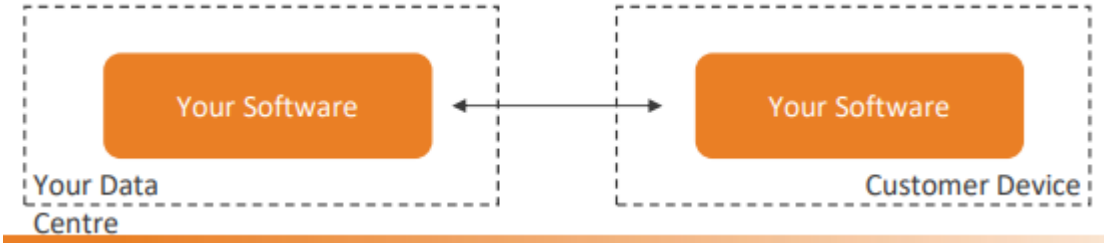
- **Bug #:** 4555
- **Follow-up by date:** April 15
- **Who's following up:** Alice
- **Who to talk to:** Bob
- **Impact if it's wrong:** Availability will be below spec
- **Assumption:** It's okay to ignore denial of service within the data center.

Assumption	Impact if it’s wrong	Who to talk to	Who’s following up	Follow-up by date	Bug #
It’s ok to ignore denial of service within the data centre	Availability will be below spec	Alice	Bob	April 15	4555

The Customer/Vendor Boundary:

- There is always a trust boundary when:
  - Your code goes to someone else’s (device/premises).

    - Their data comes to your code.
    - Lawyers: pretending does not eliminate human trust issues.
- You need to think about it while deciding what happens over the data flow shown.



Generic API Threat Model:

- Perform security checks inside the boundary.
- Copy before validation for a purpose:
    - Is `http://evil.org/pwnme.html` "valid"?
- Define the purpose for data, validate near that definition.
- Manage error reporting.
- Document what checks happen where.
- Do cryptography in constant time.
- Address the security requirements for your API.

Threat Modeling Agenda

For each STRIDE Threat:

- **Defensive tactics and technologies**
- **Operations and development**

Technologies for Addressing:

- Access Control Lists (ACLs)
- Groups or role membership
- Role-based access controls
- Windows privileges (runas)/Unix sudo

- Chroot, apparmor, other Unix sandboxes
- MOICE Windows sandbox
- Input validation for defined purposes

What Are You Going to Do About It?

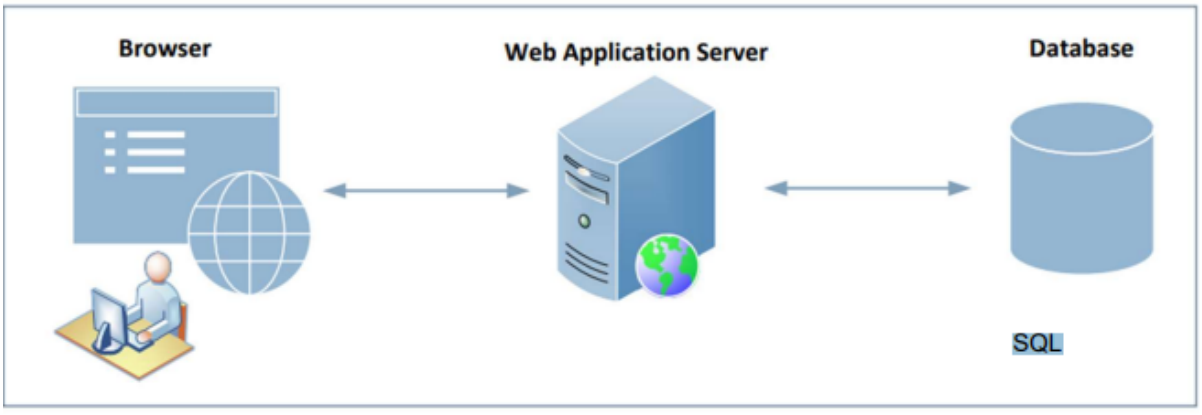
For each Threat:

- Different ways to address threats:
  - Avoid / Fix the problem
  - Mitigate with standard or custom approaches (Standard approaches were covered in the previous session)
  - Accept it
  - Transfer the risk
  - Economic impact (cost of damage)

For each assumption:

- Check it
- Wrong assumptions lead to reconsideration of potential threats.

The Web Architecture



HTML (Hypertext Markup Language)

- HTML is used for creating web pages. It provides the structure and content of the web page.

CSS (Cascading Style Sheets)

- CSS is used to specify the presentation style of HTML elements. It separates the content of the web page from its presentation style, allowing for easier maintenance and more flexible design.

Dynamic Content

- Web pages should be able to show animations, play audio and videos, and change their appearance based on various user inputs.
  - Adobe Flash
  - Microsoft Silverlight
  - ActiveX
  - Java applets
  - JavaScript

JavaScript (ECMAScript)

- JavaScript, also known as ECMAScript, is a scripting language for web pages. It is used to make web pages interactive and dynamic.

Different ways to include JavaScript code:

- Inline:

```
<script>
    // JavaScript code here
</script>
```

- External file:

```
<script src="script.js"></script>
```

These components form the basic architecture of the web.

Web Server: HTTP Server and Web Applications

- The primary function of a web server is to deliver web content to clients.
- There are two general types of content:
  - Static content

- Dynamic content

How HTTP Server Interacts with Web Applications

- **CGI (Common Gateway Interface):**
  - Starts the CGI program in a new process.
- **FastCGI:**
  - A variation of CGI that is faster.
- **Modules:**
  - Directly execute script-based programs.

Inline Approach

```
<!doctype html>
<html>
<body>
<h1>PHP Experiment</h1>
<h2>Current time is
<?php echo date("Y-m-d h:i:sa") ?>
</h2>
</body>
</html>
```

Template Approach

```
<?php
 $title = "PHP Experiment";
 $time = date("Y-m-d h:i:sa")
?>

<!doctype html>
<html>
<body>
<h1><?=$title?></h1>
<h2>Current time is <?=$time?></h2>
</body>
</html>
```

Browser-Server Communication: The HTTP Protocol

- Browsers communicate with servers using the HTTP protocol.
- HTTP is an application layer protocol.
- A server processes the HTTP request and sends back a HTTP response.

Example HTTP Request:

```
GET /index.html HTTP/1.1                ①
Host: www.example.com                   ②
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) ...
Accept: text/html,application/xhtml+xml,application/xml; ...
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

```
HTTP/1.1 200 OK
Content-Encoding: gzip
Age: 434007
Cache-Control: max-age=604800
Content-Type: text/html; charset=UTF-8
Expires: Mon, 22 Mar 2021 12:13:26 GMT
Last-Modified: Thu, 17 Oct 2019 07:18:26 GMT
Server: ECS (ord/4CDD)
Content-Length: 648
```

```
GET /index.html HTTP/1.1
Host: www.example.com
```

Example HTTP Response:

```
HTTP/1.1 200 OK
Content-Type: text/html

<!DOCTYPE html>
<html>
<head>
    <title>Example</title>
</head>
<body>

<h1>Hello, World!</h1>

</body>
</html>
```

Types of HTTP Requests: GET and POST

- Main difference is how they send data to the server.

GET request

```
GET /post_form.php?foo=hello&bar=world HTTP/1.1
Host: www.example.com
Cookie: SID=xsdfgergbghedvrbeadv
```

POST request

```
POST /post_form.php HTTP/1.1
Host: www.example.com
Cookie: SID=xsdfgergbghedvrbeadv
Content-Length: 19
foo=hello&bar=world
```

GET request:

- Data is sent in the URL.

POST request:

- Data is sent in the request body.

Privacy Concerns:

- GET requests put data in the URL, making it visible to others if the URL is shared.
- Browsers maintain browsing history by recording visited URLs. For POST requests, data is not recorded in history, but for GET requests, data, as a part of the URL, is recorded.

HTTPS

- The HTTP protocol runs directly on top of TCP, and the data is sent in plaintext.
- Plain HTTP connections are subject to eavesdropping and MITM (Man-in-the-Middle) attacks.
- To secure the HTTP protocol, HTTPS was developed.
- HTTPS is HTTP over TLS (Transport Layer Security) or formerly known as SSL (Secure Sockets Layer).
- Communication is encrypted using TLS, ensuring the confidentiality and integrity of data during transmission.

Cookies

- The web server is stateless, meaning it does not maintain a long-term connection with the client.
- HTTP Cookies are used to save information on the client side.
- Browsers save cookies and attach them to every request sent to the server.
- Websites can send new cookies to browsers or replace old cookies using the `Set-Cookie` header field in the HTTP response.

```
GET: HTTP/1.1 200 OK
Date: Wed, 25 Aug 2021 20:40:15 GMT
Server: Apache/2.4.41 (Ubuntu)
Set-Cookie: cookieA=aaaaaa
cookieB=bbbbbb; expires=Wed, 25-Aug-2021 21:40:15 GMT; Max-Age=3600
Content-Length: 28
Keep-Alive: timeout=5, max=99
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

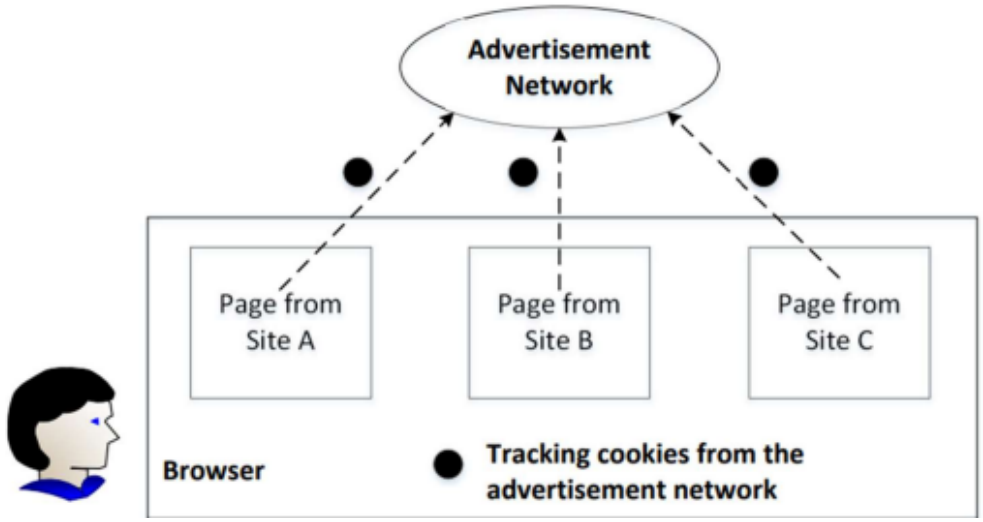
Attaching Cookies

- When a browser sends an HTTP request to a website, it looks at its cookie storage, finds all cookies belonging to the website, and attaches them in the request using the `Cookie` header field.

```
http://www.bank32.com/index.html
Host: www.bank32.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; ...
Accept: text/html,application/xhtml+xml,..
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Cookie: cookieA=aaaaaa; cookieB=bbbbbb
Upgrade-Insecure-Requests: 1
```

Tracking Using Cookies

- Cookies have many applications, and tracking is one of them.
- To help users understand how this is done, we use a shopping analogy.



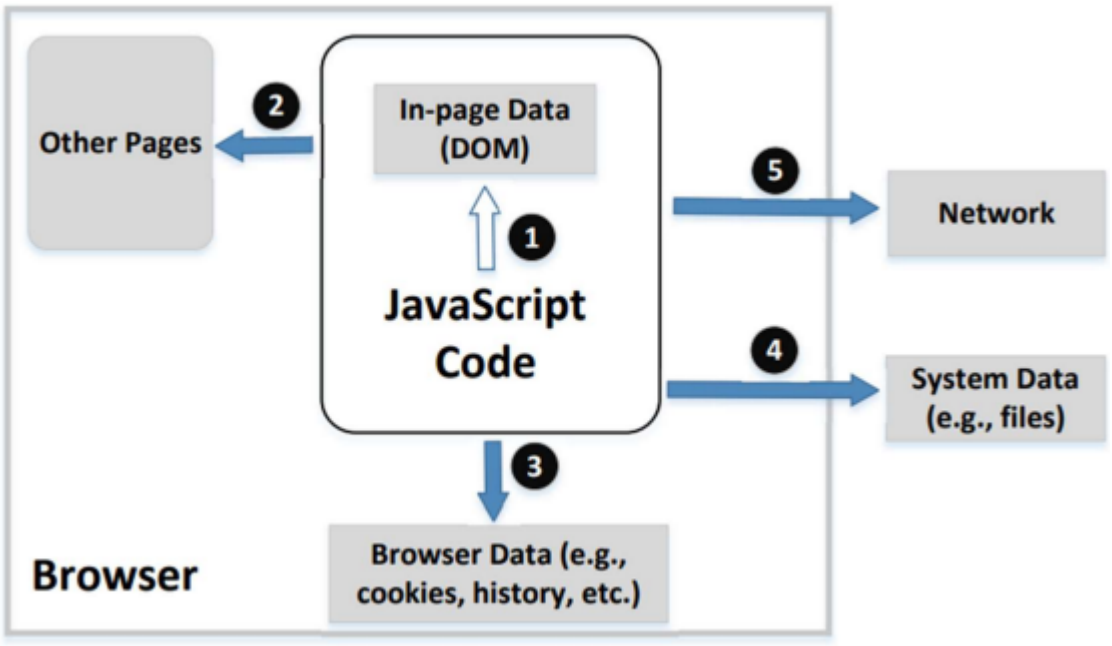
Prevent Tracking Using Cookies

- Use anonymous mode in browsing.
- Block third-party cookies:
  - First-party cookies are essential for browsing.
  - Third-party cookies are mainly used for advertisement, information collection, etc.

Sessions and Session Cookies

- **Session:**
  - A session is a way to persist data across multiple requests from the same client.
  - It allows the server to recognize the client between requests.
- **Session Cookies:**
  - Session cookies are cookies that are stored temporarily and are destroyed when the browser is closed.
  - They are used to store session information, such as user authentication tokens, shopping cart contents, etc.

Sandboxing JavaScript

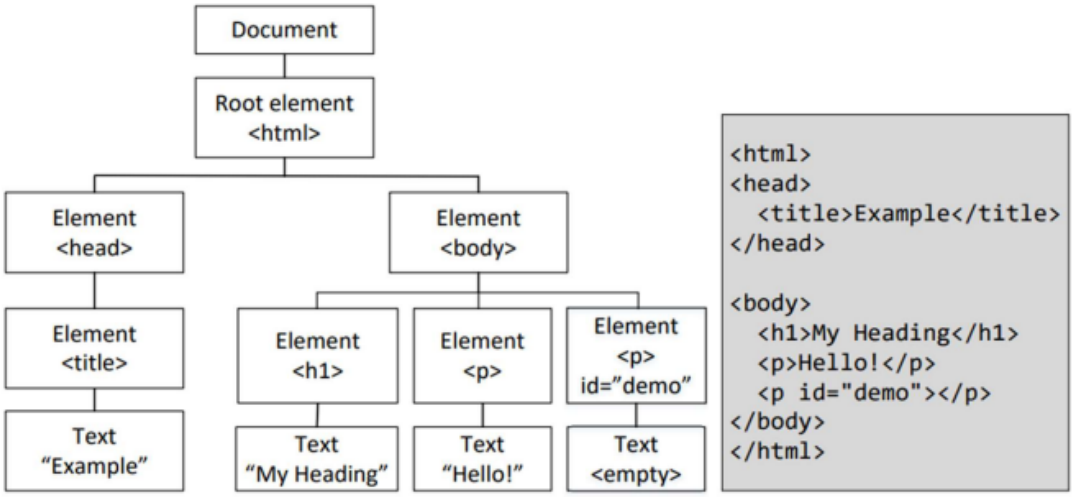


To ensure security when running JavaScript, it is important to sandbox it, meaning restricting its access to certain types of data.

Types of Data that JavaScript can Access:

1. **In-page data:**
   - JavaScript should be allowed to access data within the current page.
2. **Other pages' data:**
   - JavaScript should not be allowed to access data from other pages.
3. **Browser data:**
   - JavaScript can access browser data, but with restrictions to prevent unauthorized access.
4. **File system:**
   - JavaScript can access the file system, but with restrictions to prevent unauthorized access.

Access Page Data and Document Object Model (DOM)




```
document.getElementById('demo').innerHTML = 'Hello World'
```

Access Browser data

- Browsers expose the `window` global variable to JavaScript, which allows JavaScript code to access browser-related information.
- Two examples of accessing browser data:
  - `window.history`:
    - Returns a reference to the browser's navigation history.
  - `window.navigator`:
    - Returns GPS location of the browser.
    - Generates a pop-up notification requesting permission from the user.

Access File Systems

- JavaScript cannot directly access the local file system due to security restrictions.
- Users need to grant permission via file selection.
- File selection involves granting permissions by selecting files and getting the file handlers.

Ajax Request and Security

- Three communication mechanisms:
  - Normal HTTP

    - Ajax
    - WebSocket
- Security policies are different for each mechanism.
- With Ajax, JavaScript code in a web page sends HTTP requests asynchronously.
- When the response arrives, the browser gives the data to a callback function, which updates the existing page using the data.

```
function send_ajax()
{
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {     ②
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "http://www.bank32.com/getdata.php", true); ③
  xhttp.send();
}
```

Same Origin Policy on Ajax

- The Same Origin Policy restricts JavaScript from making requests to a different origin than the one from which the script was loaded.
- If an Ajax request is made from web server A, only responses from the same web server can be given to the callback function.
- If the Ajax request's target URL is B, the browser will not give the response to the callback function, even if the response comes back.
- This policy helps prevent cross-origin access, which could compromise privacy.

```
sudo tcpdump -i br-3f00b5edf2b0 -n -v
0.9.0.1.42580 > 10.9.0.5.80: ... HTTP, length: 316
  GET /getdata.php HTTP/1.1
  Host: www.bank32.com
  Origin: http://www.bank99.com
  Referer: http://www.bank99.com/ajax.html

0.9.0.5.80 > 10.9.0.1.42580: ... HTTP, length: 224
  HTTP/1.1 200 OK
  Server: Apache/2.4.41 (Ubuntu)
  Content-Type: text/html; charset=UTF-8

  Data from Bank32!
```

Cross Domain Ajax Request

- In many applications, it's necessary to get data from different origins, making the same origin policy too restrictive.
- To solve this, a new standard called CORS (Cross-Origin Resource Sharing) was created.
- CORS provides a mechanism for the server to tell the browser whether its response should be given to the client.
- This is done using HTTP headers created for CORS like `Access-Control-Allow-Origin`.

WebSocket

- Ajax uses HTTP, which is half-duplex:
  - Browser sends a request, server responds.
  - No "push" mechanism from the server.
- WebSocket is full-duplex:
  - Both browser and server can send data without a request.

WebSocket: Security Policy

- Browser does not restrict data from WebSocket, unlike Ajax.
- Access control is conducted on the client side.
- Access control is also conducted on the server side by checking the "Origin" of the request.

WebSocket: Cable Haunt Attack

- Discovered in January 2020.
- Affected many Broadcom-based cable modems.
- These modems run a WebSocket-based server program.
- JavaScript code can interact with the server, providing a door for attackers.
- Attacker exploits a buffer overflow vulnerability on the affected modems.