Lecture 1: Intro   Fundamentals of Information Security – Notes – Ahmed Nouralla

- **Computer Security**
  - Measures and controls that ensure **confidentiality**, **integrity**, and **availability** of information system assets, including **hardware**, **software**, and **information** being processed, stored, and communicated.
- **The CIA triad**
  - **Confidentiality:** information should not be disclosed to unauthorized parties.
  - **Integrity:** information should not be modified in an unauthorized manner.
  - **Availability:** system or resource shall be available for its intended use.
- **Levels of security breach impact**
  - **Low:** loss will have a limited impact.
  - **Moderate:** loss has a serious effect, but no loss of lives or serious injuries.
  - **High:** loss has catastrophic impact on operations, assets, or individuals.

*"**Threats** exploit **vulnerabilities** (leaky, corrupted, unavailable) through **attacks** (passive/active, insider/outsider)"*

- **Cryptographic systems categorization**
  - **Type of encryption operation:** substitution, transposition.
    - Caesar cipher, rotor machines (substitution), rail-fence (transposition).
  - **Number of used keys:** one (symmetric), two (asymmetric).
  - **How plaintext is processed:** in blocks, as a stream.
    - **Block cipher:** processes input one block at a time, more common and can reuse keys.
    - **Stream/state cipher:** processes input digits continuously given a keystream and is typically faster than block cipher, combining key bit and data bit typically uses XOR operation.
- **Symmetric/Conventional/Single-key/Private-key Encryption**
  - The universal technique for providing confidentiality for transmitted data.
  - Requires sender and receiver to securely share an encryption key (that uses a strong encryption algorithm).
  - Attacks can be based on brute-force, or cryptanalysis.
    - **Brute force attacks** try all possible keys on ciphertext until a meaningful plaintext is obtained.
    - **Cryptanalysis attacks** use smarter approaches that exploit the nature of the algorithm, knowledge of plaintext characteristics, or example plaintext/ciphertext pairs.
      - **Ciphertext only:** only the algorithm is known, hardest to crack.
      - **Known plaintext:** one or more plaintext/ciphertext pair is also given.
      - **Chosen plaintext:** given access to an encryptor (but not the key).
      - **Chosen plaintext:** given access to a decryptor (but not the key).
      - **Chosen text:** given access to an encryptor and a decryptor (relatively the easiest).
  - **Data Encryption Standard**
    - Symmetric block cipher **(block size = 64 bits)** that uses a (**key size = 56 bits)**.
    - Was a standard until broken due to small key size.
    - **Triple DES** applies DES three times and uses **two keys of 112, or 168 bits**, but also has a problem of small **(block size = 64 bits)**.
    - **Advanced Encryption Standard (AES)** uses 128-bit block size and 129-, 192-, or 256-bit key size and is the current standard.

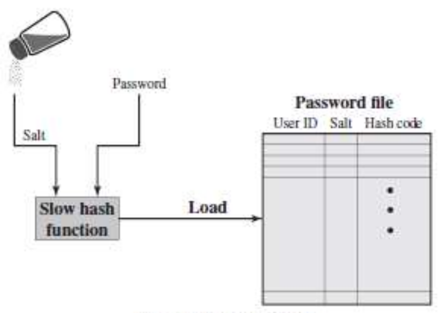
- **Asymmetric/Public-key Encryption (approaches)**
  - **Rivest-Shamir-Adleman (RSA):** public key cryptosystem most widely used, can also be used for digital signatures (discussed below)
  - **Diffie-Hellman (DH):** a scheme (algorithm) used to securely establish a common secret (symmetric key) between two parties.
  - **Elliptic Curve Cryptography (ECC):** like RSA but uses smaller key sizes, used for symmetric key exchange (ECDH) and digital signatures (ECDSA).
- **Message Authentication**
  - Protects against active attacks by ensuring the authenticity (identity) of the sender and that the order and content was not altered in transit.
  - **Message Hash (digest)**
    - One way function, produces fixed length data, can be applied to any size, and is fast to compute.
      - **Weak collision resistance:** given x, it's hard to find y such that $H(x) = H(y)$.
      - **Strong collision resistance:** it's hard to find any pair $x \neq y$ such that $H(x) = H(y)$.
      - **Commonly used functions:** SHA-1 (160-bit hash), SHA-256, SHA-384, SHA-512.
  - **Message Authentication Code (MAC):**
    - Code tagging the message or its encrypted version, it can also be combined with the message and then encrypted together.
    - Used to verify message authentication and integrity.
  - **Digital Signature:**
    - The result of a cryptographic transformation of data that provides a mechanism for verifying the origin authenticity (X authored M), data integrity (M was not tampered with), and signatory non-repudiation (X cannot deny that they authored M).
    - **Digital Signature Standard (DSS)**
      - Standard specifying suite of algorithms to generate digital signatures.
      - Defines the Digital Signature Algorithm (DSA), the RSA DSA, and ECDSA

	Hash	**MAC**	**Digital Signature**
Integrity	Yes	Yes	Yes
Authentication	No	Yes	Yes
Non-repudiation	No	No	Yes
Key type	-	Symmetric	Asymmetric

- **Random Numbers**
  - Used for generating symmetric/asymmetric keys, and the keystream for stream ciphers, for session keys and handshaking to prevent replay attacks.
  - The have to be **random** (choice follows a uniform distribution) and **unpredictable** (next value cannot be inferred from the previous ones).
  - **Pseudorandom** numbers use algorithms and are therefore not truly random.
  - **Truly Random Number Generators (TRNG):** use a nondeterministic source such as measuring a value from natural processes (e.g., radiation level)
- **Steganography**
  - The art of hiding information (covert communication) from 3rd parties.
  - Data is hidden inside images (e.g., LSB method), videos (inside frames), audio and satellite signals (frequency channels), or network protocols (covert channel).
  - Steganalysis is about detecting such hidden information.

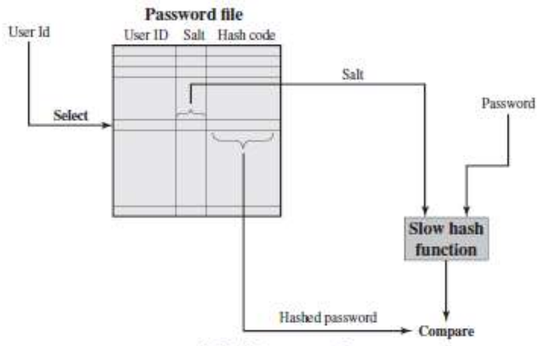
Lecture 2-3: Access Control

- **Four measures of authenticating users**
  - **Something the user knows:** password, PIN, secret question, …
  - **Something the user possesses:** physical key, smartcard, electronic token, …
  - **Something the user is:** fingerprint, face, eye (retina), …
  - **Something the user does:** voice pattern, handwriting, …
- **Unix Password scheme**
  - Originally was limited to 8 characters, and used 12-bit salt.
  - Much stronger algorithms were used later,
  - Hashing function is intended to be slow to make brute forcing harder.
  - Hashing algorithm used or in use for Linux: MD5 (deprecated), SHA variants, and bcrypt (derived from blowfish).



(a) Loading a new password

- **Password cracking**
  - **Dictionary attacks:** uses a large wordlist of possible passwords and brute force.
    - Defeated by using long, non-common passwords.
  - **Rainbow table:** try to construct a reverse lookup table for hashes.
    - Defeated by using longer salts.
  - Modern applications enforce users to choose strong passwords, they may also run their own password cracker to find guessable passwords.



(b) Verifying a password

- **Access Control**
  - Defines who should have access to what and how to enforce such access.
  - Users (**subjects**) are given access (**right**) to **objects** based on their **identity** and/or the **group** the belong to. **Policies** define such access rules and **mechanisms** are used to enforce them.
  - **User identity** determines whether the user is **authorized** to access the system and their privileges.
    - Authentication is then used to verify the claimed identity.
    - A user can be authenticated to enter the system but not authorized to access a certain resource (e.g., admin panel).
  - **Access Control Policies**

Method	Description
Discretionary (DAC)	A user with a certain permission to an object can grant this permission to others.
Mandatory (MAC)	Permissions cannot be delegated; access is granted if an explicit grant rule found.
Role-Based (RBAC)	Policies concern roles, not individuals. A user may have multiple roles, when a user leaves the organization, system doesn't need to be updated.
Attribute-Based (ABAC)	Access is granted based on current user/resource/environment attributes.

  - **Implementing Access Control**
    - In UNIX, each resource is a file, files have owners (identified by UID).
    - Three types of subjects (owner, group, and everyone).

- Three types of permissions (read, write, and execute).
- To work with files, we use system calls: open(), read(), write(), close().
- File metadata is stored in the system as i-nodes.
- Time-of-check to Time-of-use (TOCTOU) problem arises when authentication checks are performed separately of authorization, as access rules may change between the two times.

Method	**Description**
Matrix (ACM)	- List all subjects in one dimension, and all objects in the other. - $ACM[U, O]$ tells what access rights user U has for object O. - Used to define DAC.
List (ACL)	- For each object there is a list: $O_i = [(u_1, rights_1), (u_2, rights_2), \dots]$ - May contain a default entry, that's overridden by explicit user entries. - Can be stored with other object metadata.
Capability List (C-List)	- For each user there is a list: $U_i = [(o_1, rights_1), (o_2, rights_2), \dots]$ - Can be stored in kernel memory (protected).

Lecture 4: Database and Datacenter Security

```
GRANT { privilege | role }
ON object
TO { user | PUBLIC | role }
[ IDENTIFIED BY password ]
[ WITH GRANT OPTION ];
```

```
REVOKE { privilege | role }
ON object
FROM { user | PUBLIC | role };
```

- **Refer to DB Notes for**
  - Relational model terminology.
  - SQL, DBMS, views, and transactions.
- **Database Access Control**
  - SQL **GRANT** and **REVOKE**
- **SQL Injection**
  - Vulnerability in applications that allows malicious SQL commands to be sent to the database.
  - Can impact confidentiality and integrity.
  - Defended against by proper sanitization of user input and rejecting invalid inputs.
- **Inference attacks**
  - Occurs when certain aggregate/statistical queries are allowed to be done by everyone.
  - Inference attacks use knowledge about data to get information about individuals.
  - **Defense**: don't allow aggregate queries when selected tuples are too small or too large.
- **Datacenter**
  - Enterprise facility storing a large number of servers, storage, and network devices with redundant ones for reliability.
  - Used by cloud service providers, search engines, large companies, etc.
  - Concerns about Data/Network/Physical/Site security.
- **Denial of Service (DoS) attack**
  - An action that prevents or impairs the authorized use of networks, systems, or applications by exhausting resources such as CPU, memory, bandwidth, or disk space.
  - Attacks the availability of system/network/application resources.
  - **Flooding attacks:**
    - Aim at filling bandwidth and/or exhausting target resources by sending multiple packets to a target (directly or indirectly).
    - Packets can be ICMP requests (ping, traceroute, etc.), TCP SYN/SYN+ACK/CONNECT, UDP datagrams, or many others.

Attack	Description
Ping flood	- Classic DoS attack in which the system is spammed with ping requests that overwhelm the network capacity. - Source can be identified and blocked, unless using a **spoofed address**. - **On the same subnet:** easier but doesn't completely hide identity. - **On another subnet:** requires misconfigured routers that do not drop packets coming from subnet X and labeled as they came from subnet Y.
SYN flood	- Attacker sends many TCP SYN packets. - Server sends SYN+ACK and allocates resources for the upcoming connection. - Attackers does not send the final ACK and leaves the server waiting. - Server resources are exhausted.
SYN spoofing	- Attacker sends only one SYN with a spoofed address (misconfigured target, or an already overloaded server). - The goal is to have the target not reply to SYN+ACK with RST, so the server will keep resending SYN+ACK on timeout and overload the target.

HTTP flood	- Sending multiple HTTP GET or POST requests to server to allocate resources (e.g., requests to download a large file or looking up items in the DB).
Low and slow HTTP attack	- Sending legitimate HTTP requests but at a very slow rate, keeping the connection open for as long as possible, is very hard to detect.

- **Distributed Denial of Service (DDoS) attack**
    - Takes advantage of many machines (typically compromised, called botnets) to amplify the DoS attack and hide the origin.
    - Compromised machines are typically infected with a worm and may communicate with the attacker's Command and Control (C&C) servers from which they give orders to begin attack.
    - **Distributed Reflective DoS (DRDoS)**
        - Uses publicly available servers as **intermediaries** to amplify attacks.
        - Ideally, intermediaries shouldn't be alerted.
        - **DNS Amplification attack** sends multiple DNS lookups with a spoofed source address of target.
- **DoS defenses**
    - **Prevention:** before the attack.
        - Block spoofed source addresses as close to the source as possible.
        - ISPs should filter packets and ensure origin before passing.
        - Drop incomplete TCP connections on overflow.
        - Block IP-directed broadcasts (packets destined to all, but coming from outside the subnet).
        - Implement human verification (e.g., CAPTCHA) on application level.
        - Use redundant and/or cloud servers that have high performance and can't be easily affected with DoS.
    - **Detection and filtering:** during the attack.
        - Detect patterns and filter suspicious services and activities.
    - **Source tracing and identification:** during and after the attack.
    - **Reaction and mitigation:** after the attack.
        - Have a response plan, identify attacks and add them to filters.
        - Have a backup deployment plan (new servers, new addresses) to keep the service running.

Lecture 5: Intrusion Detection

- **Defense-in-depth**
    - The concept in which multiple security layers (physical, technical, and administrative) are deployed to provide stronger system.
    - Examples include fences (physical), encryption (technical), and security regulations (administrative).
- **Intrusion:** any unauthorized activity in the network or system.
    - **Intruder behavior**
        - Information gathering ⟹ Initial access ⟹ Privilege escalation ⟹ System exploit ⟹ Maintaining access ⟹ Covering tracks.
    - **Intrusion Detection System (IDS):**
        - **How it works**
            - IDS is designed to monitor the system passively and detect common threats based on recorded and analyzed data (system and network activity).
            - Upon detecting malicious behavior, it generates an alert, depending on the security policy, an action might be taken.
            - Gets less effective as threats get more targeted/sophisticated and will not help at all with zero-day exploits.
            - Requires that the system is observable and that normal behavior is distinguishable from malicious behavior.
            - IDS can use statistical, knowledge based, or machine learning techniques to detect common attack vectors and/or they can be explicitly programmed by experts.
            - IDS can be deployed on **hosts**, or in the **network**/subnet perimeter.

Host-based IDS (HIDS)	Network-based IDS (NIDS)
- Uses system logs, call traces, and file hashes as data sources. - *ptrace* (system call used for tracing call origin).	- Uses a packet capturing tool to sniff network traffic. - *libpcap* (interface for user-level packet capturing software). - May use a **passive** or **inline** sensor. - Is fail-open (passes unknown traffic), unlike firewall.

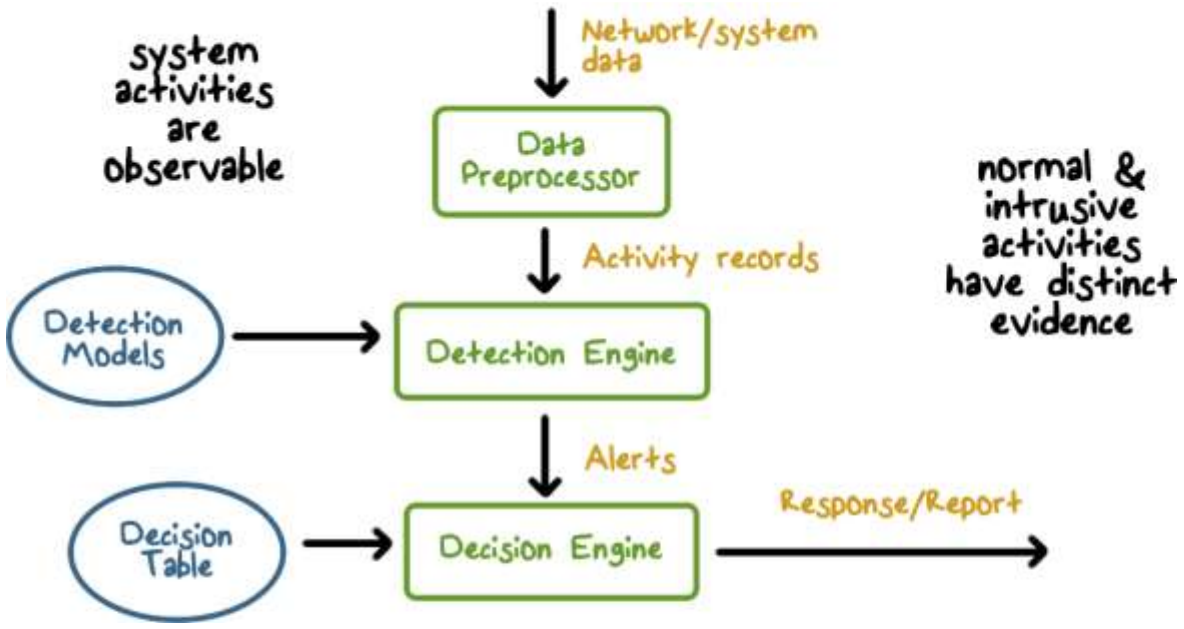
Inline network sensor	Passive network sensor
- Is placed in the network segment such that all traffic pass through it for analysis. - May block traffic, thus implementing IDS and IPS at the same time.	- Analyze a copy of the traffic, can only detect threats but cannot block them. - Is typically faster than the inline sensor.

        - **Algorithmic components:**
            - **Features:** capture intrusion evidence
            - **Models:** pieces evidence together.
        - **System components**
            - **Audit-data Processor:** observes and collects activities.
            - **Knowledge Base:** stores evidence.
            - **Decision Engine:** processes stored data and decide if it's an intrusion.
            - **Alarm Generator:** alerts administrators.

- **Detection methods**

Misuse (signature-based) detection	Anomaly detection
- Compare current behavior with previously known attack patterns. - Alert on known pattern match. - Can only detect previously known attacks. - Commonly used in Antiviruses.	- Compare current behavior to previously collected data about legitimate behavior. - Alert on unknown behavior. - May generate many false alarms.

- **IDS Components**



Lecture 6: Firewalls

- **Network-based Firewall**
  - Network security system that monitors and controls incoming and outgoing network traffic based on predetermined security rules.
  - Typically stands between a trusted network and an untrusted network, such as the Internet.
  - Should **enforce all rules** and **be dependable** (can't be disabled by an attack).
  - **DeMilitarized Zone (DMZ)**
    - Physical/logical subnet facing the public internet, provides an extra layer of security by having users communicate to the internal LAN only through them
    - Usually contains organization's publicly available servers (e.g., DNS, FTP, HTTP, …).
    - Typically placed along with the internal network behind the firewall.
  - **Firewall Access Policy**
    - Lists the traffic patterns that can pass (e.g., address ranges, protocols, applications, …).
    - Developed by the organization's risk assessment team given the nature of traffic the company supports.
    - Implemented within the appropriate firewall topology.
  - **Additional services that can be provided by firewall**
    - Traffic logging for analysis.
    - (Stealth mode) hiding system from outside by blocking discovery packets (e.g., pings).
    - User authentication and authorization.
    - Network Address Translation (NAT)
  - **Filtering types**

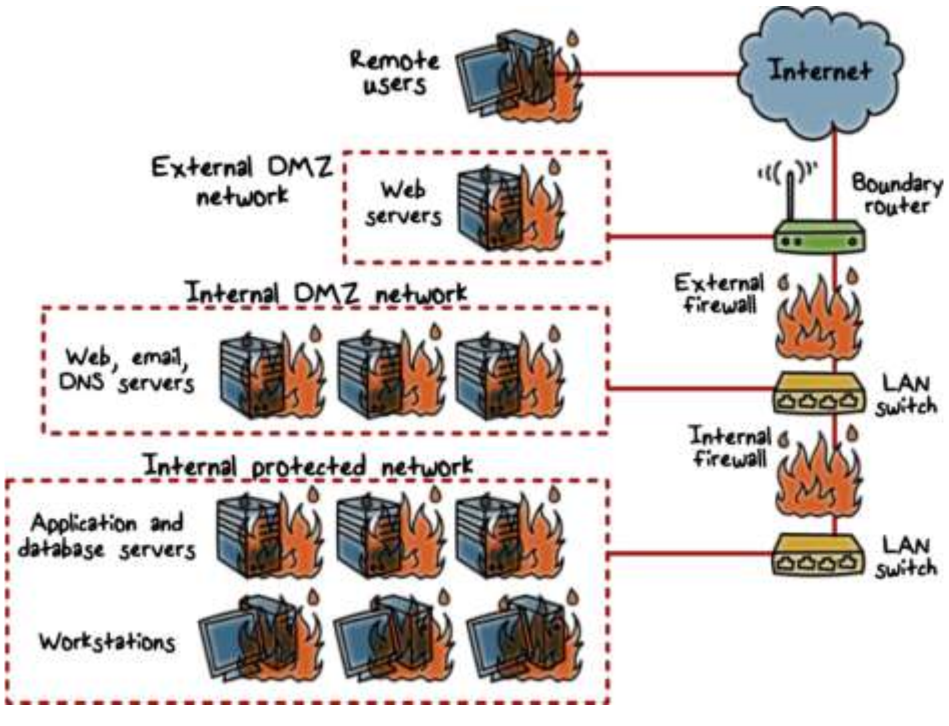
Packet Filtering	**Session Filtering**
- Filtering decisions are made per-packet. - Used by stateless firewalls. - Filtering based on packet contents (source and destination IP and port, protocol, and interface). - Simple and fast, but cannot prevent application-specific exploits.	- Filtering decisions are made per-session. - Used by stateful firewalls. - Stores a table of active connections, only sessions that look benign are kept. - Can inspect protocol-specific data.

  - **Filtering action**
    - If a rule was found take action (forward or drop).
    - If a rule was not found, use the default policy which can be to forward (easier to manage but less secure) or to drop (more restrictive).
  - **Typical attacks on Packet filtering firewall**

Attack	**Description**	**Countermeasure**
IP address spoofing	Source address is spoofed to be an internal address, to bypass firewall source address checks.	Drop packet with an internal source address if it came from outside.
Source routing attack	Attacker packet comes with routing information to bypass firewall.	Drop packets containing routing information in their destination.
Tiny fragment attack	Attacker uses IP fragmentation to send the header in multiple tiny packets to bypass firewall header checks that are enforced on the first packet.	Enforce rules on the minimum amount of header fields of the first packet.

- **Application Level/Layer Gateway/Proxy (ALG)**
  - Relays traffic from users to the application server after implementing other functionalities such as Firewall or NAT.
  - Provides an additional layer of security but requires much overhead in application development and may hinder some features.
  - **Bastion host**
    - Hosting one or more ALGs that are designed to withstand attacks (acts as a layer of defense).
    - It stands between users and the application to verify their legitimacy.
    - Blocks malicious users even before they communicate with the application.
    - Users cannot directly communicate with the application; proxy relays data for legit connections.
    - **Characteristics:**
      - Minimal, simple, runs secure OS, only essential services.
      - May authenticate users before relaying data to the application.
      - It can restrict some features or block access to certain application servers.
      - Proxies are isolated and each one runs a non-privileged user in a separate directory.

- **Host-based Firewall**
  - Used to secure an individual host (typically a server) by filtering incoming and outgoing traffic.
  - Provided by the OS or can be installed separately.
  - Can be customized for this certain host, independently from the network, thus providing an additional layer of security.
- **Personal Firewall**
  - Controls traffic between PC or workstation and the Internet or enterprise network
  - Can be installed on router that connects home computers to DSL.
  - Simpler than enterprise firewall and is only there to deny unauthorized remote access.
  - May also monitor outgoing traffic to block worms and malware activities.
- **Internal firewall**
  - Used inside the organization to add yet another layer of security.
  - Used to protect and isolate internal subnets from each other in case of malware spread.
- **Distributed firewall deployment**



Lecture 7: Malicious Software

- **Malware:**
  - Software intentionally designed to do a malicious behavior (e.g., disrupt systems and networks, gain unauthorized access to systems or information, etc.).
  - Past malware was just made for fame/fun (e.g., defacing web pages and spreading a message).
  - Modern malware is often for profit and political gain, is sophisticated and advanced.
- **Malwares that need host program**

Malware	Description
Backdoor (trapdoor)	- Secret entry point to the system - Typically works by recognizing some special inputs.
Logic bomb	- Embedded in a legitimate program. - Performs malicious activities only when certain conditions are met.
Trojan horse	- Hidden in an apparently legitimate host program. - Performs malicious functions when the host program is executed.
Virus	- When executed, replicates itself by infecting other programs. - Modifies programs code and can insert its own code.
Browser plug-ins, extensions, scripts	- Hijack existing software to do malicious behavior.

- **Virus**
  - **Stages:**
    - **Dormant:** the virus is idle.
    - **Propagation:** the virus is being spread.
    - **Triggering**: the host program is executed, virus is triggered.
    - **Execution:** virus code runs, performing malicious activities and looking for other programs to spread.
  - **Structure:**
    - Virus code is appended into program code.
    - When program runs, virus **executes** first, then program code.
    - After program terminates, virus may also clean up to avoid detection.
    - It may also compress or decompress files to keep program size unchanged.
  - **Types:**

Type	Description
Parasitic	- Attached to executables, runs when they are executed.
Memory-resident	- Attached to the OS programs. When executed, gets loaded into memory. - **Rootkit:** a type of memory resident virus used to hide malware from the user by modifying OS functions (e.g., modify *ls* and *ps* commands to not list the virus)
Macro virus	- Written in a macro language, the same one the host program uses. - Typically embedded in documents, executes as programs try to expand the macro.
Boot sector	- Resides in boot sector. Runs when the system is booted before the bootloader.
Polymorphic	- Changes form without changing functionality to avoid detection - May encrypt itself with randomly generated keys to change forms.

- **Independent malware**
    - **Worms:**
        - Use network connection to self-propagate, doesn't need a host program to execute.
        - Exploits system vulnerabilities while remaining undiscoverable.
        - Can hide itself by loading code in memory and removing downloaded files, encrypting files, and periodically changing name and process ID.
        - Worms later evolved into botnets.
        - The internet worm
            - Exploited security flaws in $sendmail$ (backdoor), $fingerd$ (buffer overflow), and weak guessable passwords.
            - When downloaded, uses a loader with password authentication to fetch the rest of the code and spread to other systems on the network.
            - Some systems were infected multiple times, leading to resource exhaustion.
            - Drew attention to the importance of security scanning and patching.
    - **Botnet:**
        - Network of bots controlled by an attacker to carry a strong and coordinated malicious behavior.
        - Used for spamming, DDoS attacks, click fraud, phishing and pharming, keylogging and data/ID theft, password cracking, anonymized communications, cheating in online games and polls.
        - **Check botnet DDoS (Lecture 4)**
        - **Bot (zombie):**
            - A compromised machine under the control of an attacker.
            - Communicates with the attacker server (C&C) and gets instructions.
                - Communication has to be efficient, stealthy and hard to block.
            - A naïve approach would be to hard-code the author address in the botcode
                - Source will be traceable and can get banned, single point of failure.
            - A smarter approach will be to use dynamic DNS with the bots communicating to C&C domain names that can change IP to avoid getting blocked easily.
                - The DNS provider may be able to detect that this host is used for C&C (e.g., if it's referenced too often by too many machines while not being indexed by search engines).
                - The DNS provider can then point the hostname to a **DNS sinkhole** (a server where all bot requests will go and get logged) instead of the botmaster server.
    - **Advanced Persistent Threat (APT)**
        - **Name**
            - **Advanced:** specially crafted malware to do a certain operation (zero-day exploit).
            - **Persistent:** stays for a long-term, does its function slowly in multiple steps without raising attention (no traffic graph spikes).
            - **Threat:** targeted at high-valued information and organizations.
        - **Lifecycle**
            - Define and research (scan) the target for vulnerabilities and infrastructure.
            - Deployment (through email or by social engineering means)
                - Compromise core network elements and search for valuable targets.
                - Craft special social-engineering attack that is convincing.
            - Establish outbound connections to pass information or get commands from master.
            - Avoid detection and stay hidden for as long as possible.

- **Defenses against malware**
    - Limit contact to the outside world, do it only through firewalls, proxies, and gateways.
    - Deploy systems for malware detection and identification.
    - Remove or isolate detected malware.

- **Antivirus software**

Type	Description
Simple scanners	- Scans file signatures against signatures of well-known viruses. - Not effective against polymorphic viruses. - But we still use them as first line defense for detecting known malware quickly.
Heuristic scanners	- Check files regularly and detect when their signature change as it may indicate virus infection. - Generate many false positives.
Activity traps	- Look for specific known malicious activities (e.g., modification of passwd file). - Not effective against new malware.
Full-featured analyzers	- Sophisticated software that can be deployed on hosts and networks. - Allows executing suspicious files inside isolated environment (sandbox) for analysis and reporting.

Lecture 8: Software Security

- **Buffer overflow**
  - An anomaly where a program, while writing data to a buffer, overruns the buffer's boundary and overwrites adjacent memory locations.

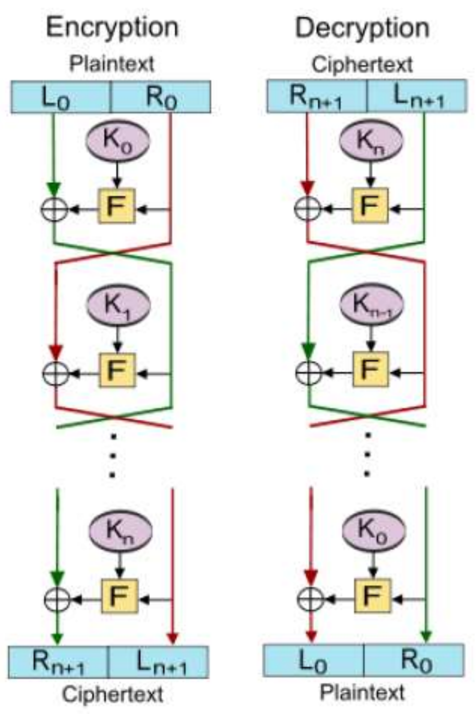
- **Buffer overread**
  - A similar anomaly where the program reads adjacent memory outside of its reserved data structure, resulting in sensitive data exposure.
  - Heartbleed: a vulnerability in OpenSSL that allowed buffer overread due to a missing bound-check.

- **Stack memory:**
  - Region of program address space used mainly for
    - Storing program static variables and parameters.
    - Keeping track of routine calls and their return addresses.
  - It grown downwards (towards lower address) as data is pushed into it and shrinks upwards as data is popped.
  - This is opposed to the heap memory (grows upwards) used for dynamically allocated structures.

- **Stack buffer overflow**
  - Common problem that happens when programs write to memory locations outside of their allocated data structures (which is usually a fixed-length buffer).
  - This causes memory corruption which can lead the program to crash, leak data, or do other unintended behavior.
  - Attackers can exploit such problem (i.e., **stack smashing**) to inject malicious code and take control of the program/system.
    - This is especially possible when program stores un-sanitized user input in memory.
    - The injected code can be crafted such that it replaces the function return address with another address of malicious code or an address of already accessible code (i.e., **return-to-libc attack**)
  - A similar concept (**heap overflow**) allows overwriting heap memory to change the value of a function pointer or linked list node pointer, resulting in an unintended behavior.

- **Defenses against buffer overflow**
  - **Address Space Layout Randomization (ASLR):**
    - A technique used by operating systems to lower the chances of buffer-overflow attacks, by loading system executables into random locations in memory, making it harder for the attacker to guess the location of sensitive data and feed a certain input to the program that exposes such data.
  - **Applying secure programming practices**
    - Sanitizing user input.
    - Type and bounds checks.
    - Memory management.

Lecture 9: Symmetric Encryption



- **Encryption is computationally secure if one of the following is true:**
    - Cost of breaking it exceeds the value of information
    - Time required to break it exceeds the useful lifetime of information.

- **Feistel cipher:** general structure for symmetric encryption algorithms
    - Plaintext is divided into two blocks (L_0, R_0)
    - The following formulas are applied for $1 \leq i \leq n$
      $$\begin{cases} L_{i+1} = R_i \\ R_{i+1} = L_i \oplus F(R_i, K_i) \end{cases}$$
    - Where:
        - F is the Feistel function (implements encryption logic)
        - K_i is the round subkey, extracted from the master K
    - Ciphertext = (R_{n+1}, L_{n+1})
    - Decryption applies keys in reverse order.
    - Different symmetric encryption algorithms (e.g., DES and Blowfish) are based on Feistel structure with different block size, key size, round function, number of rounds, subkey generation algorithms which leads to different algorithm characteristics in terms of efficiency and ease of analysis.

- **Data Encryption Standard (DES)**
    - **Check Lecture 1 for recap.**
    - **Uses a Feistel structure with:**
        - 64-bit block size, 64-bit output, 56-bit key, 16 rounds,
        - An initial round of permutation of the plaintext to achieve diffusion (to be inverted at the end).
        - The Mangler (Feistel) function used involves 4 stages:
            - **Expansion:** function that expands the 32-bit half-block into 48-bits.
            - **Key mixing:** XORing the expanded result with the subkey.
            - **Substitution:** substituting and shrinking the output using S-Boxes (fixed lookup table)
            - **Permutation:** rearranging output according to a fixed permutation.
    - **Triple DES**
        - A proposal to fix the problem of small key size in DES.
        - Using $K_1 = K_3$ results in an effective key size of 112-bits
        - Using distinct keys result in an effective key size of 168-bits.

- **Advanced Encryption Standard (AES) - Visualizer**
    - **Involves four steps**
        - **SubBytes:** S-Box is used to achieve confusion.
        - **ShiftRows:** rows are shifted to achieve diffusion.
        - **MixColumns:** multiplication by a fixed matrix over a finite (Galois) field to achieve better diffusion.
        - **AddKey:** XOR round subkey with the output.

- **Block cipher modes of operation**
    - Decides how divide (and possibly pad) long messages to have a fixed block size.
    - **Electronic Code Book (ECB)**
        - Simplest mode of operation in which each block is encrypted separately with the same key.
        - Same plaintext block results in the same ciphertext block.
        - Not secure for long messages as repeated plaintext is seen in repeated ciphertext, allowing cryptanalysis inferences about the ciphertext.
        - Provides opportunity to substitute or rearrange blocks, doesn't do integrity checks.
    - **Cipher Block Chaining (CBC)**
        - Plaintext block is XORed with the previous ciphertext block before being passed to the encryption algorithm.
        - First plaintext block is XORed with an Initialization Vector (IV) that's generated randomly and needed for decryption.
            - IV may be sent using ECB mode.
        - This mode solves both problems with ECB as
            - The same plaintext block is encrypted into different ciphertext blocks.
            - Blocks cannot be reordered or substituted as it will produce garbage when decrypting.

Lecture 10: Asymmetric Encryption

- **Prerequisite Math**
  - **Assume** $a, b, m, r, p, n \geq 0$ unless explicitly stated otherwise.
  - **Divisibility**
    - b divides $a \Leftrightarrow a = mb$
    - We say that b is a divisor of a and write $b \mid a$
    - $r = a \bmod b$ is the remainder (modulo) when dividing a by b, it follows that $a = mb + r$
  - **Greatest Common Divisor: gcd$(\boldsymbol{a}, \boldsymbol{b})$**
    - Largest integer that is a divisor of both a and b
    - a and b are coprime (relatively prime) if $\gcd(a, b) = 1$
    - Euclidean algorithm to find **gcd$(\boldsymbol{a}, \boldsymbol{b})$** where $a \geq b$

$$\gcd(a, b) = \begin{cases} a & b = 0 \\ \gcd(a, a \bmod b) & \text{otherwise} \end{cases}$$

  - **Prime numbers**
    - Prime numbers have exactly two divisors.
    - Any integer $a > 0$ can be represented as $a = \prod_{p \in P} p^n$ where P is the set of all prime numbers.
  - **Modular arithmetic**
    - Normal arithmetic operations but results are taken modulo n where n is called the **modulus**
    - This limits the results of computations to the finite set $\{0,\ n-1\}$
    - a and b are **congruent modulo** n if $a \bmod n = b \bmod n$ and we write $a \equiv b \pmod n$
    - **Useful properties:**
      - $(a \pm b) \bmod n = a \bmod n \pm b \bmod n$ (Same for multiplication)
      - $x^y \bmod n = x^{y \bmod \phi(n)} \bmod n$ (Check ϕ function below)
    - a is the **additive inverse** of b modulo n if $a + b \equiv 0 \pmod n$.
      - **This can be useful for encryption:**
        - **Let** K_{enc}, K_{dec} be the additive inverse of each other.
        - **Then** $C = P + K_{enc}, P = C + K_{dec}$
      - All integers have an additive inverse.
    - a is the **multiplicative inverse** of b modulo n if $a * b \equiv 1 \pmod n$.
      - a has a multiplicative inverse modulo n if a and n are coprimes.
      - Not all integers have a multiplicative inverse.
    - **Fermat Theorem**
      - If a and b are coprimes then $a^{p-1} \bmod p = 1$
      - If p is prime and a is a positive integer then $a^p \equiv a \pmod p$
  - **Euler totient function $\boldsymbol{\phi}$**
    - The number of positive integers less than n and relatively prime to it.

$$\phi(n) = \begin{cases} 1 & n = 1 \\ n - 1 & n \in P \\ (p-1) * (q-1) & n = pq, \quad p, q \in P \end{cases}$$

    - For $n = ab$ where a, b are coprime: $\phi(n) = \phi(a)\,\phi(b)$
    - **Euler theorem**
      - $a^{\phi(n)+1} \equiv a \pmod n$ for $a, n > 0$
      - $a^{\phi(n)} \equiv 1 \pmod n$ if a and n are coprimes.

- **Asymmetric Encryption (Public Key Cryptography)**
  - Uses a pair of keys, public (known to everyone) and private (kept secret).
  - **Requirements:**
    - Easy to generate such a pair of keys.
    - Data encrypted with one key can only be decrypted with the other easily.
    - Hard to deduce one key from the other or decrypt without the other key.
  - **Encrypting with public key:** for secrecy (only owner can decrypt).
  - **Encrypting with private key:** for authentication (only owner could have encrypted).

- **RSA Algorithm**
  - Choose large primes p, q and calculate $n = pq$
    - Small values allow brute-forcing p, q from n which makes calculation of ϕ easy.
    - Thus, breaking RSA requires factorizing p, q back from n which is not an easy task.
  - Choose e such that $\gcd\big(e, \phi(n)\big) = 1,\ \ 1 < e < \phi(n)$
    - e and $\phi(n)$ are coprimes.
  - Find d such that $de \bmod \phi(n) = 1$
    - e and d are multiplicative inverses modulo $\phi(n)$.
  - We have

$$\begin{cases} \text{PublicKey} = \{e, n\} \\ \text{PrivateKey} = \{d, n\} \end{cases}$$

  - The following relations hold and are used for encryption and decryption:

$$\begin{cases} C = P^e \bmod n \\ P = C^d \bmod n \end{cases}$$

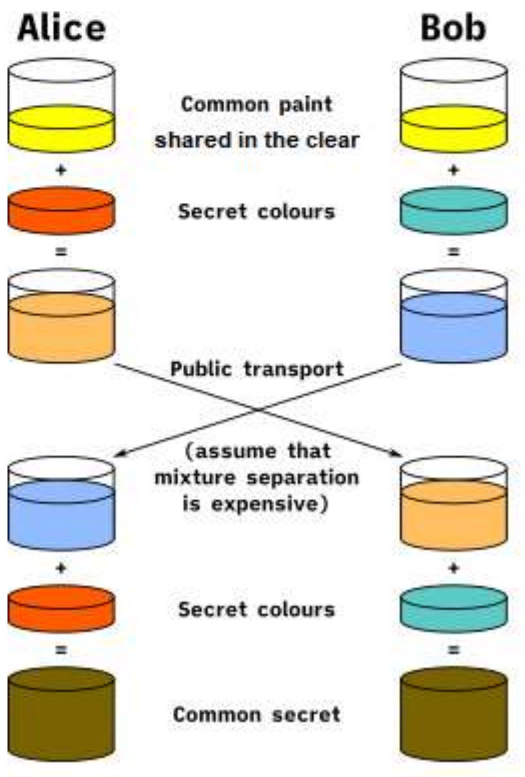
    - Where C is the ciphertext and P is the plaintext.
    - Mathematical properties discussed above can make this exponent calculation feasible.

- **Timing attack**
  - Exploits the time taken by the operation (RSA exponentiation) to infer operand size which helps with brute-forcing.
  - **Countermeasures**
    - Use constant exponentiation time.
    - Add random delays.
    - Blind values used in computation.
      - **Blinding:** instead of computing $f(x)$, compute $D\Big(f\big(E(x)\big)\Big) = f(x)$ where E and D provide blinding of the actual function argument.

Lecture 11: Key Management and Distribution

- **Key distribution/exchange**
  - The problem of having two parties (A and B) agree on the same key securely over an insecure network.
  - **Solutions:**
    - Physical delivery of key (directly or through a third-party that chooses the key for both).
    - Using an existing encryption channel to deliver the new key (directly or through a third-party).
    - Make A, B agree on the same key without sharing it explicitly (e.g., Diffie Hellman).

- **Diffie-Hellman Key Exchange**
  - Practical method for establishing a shared secret between two parties (cannot be used to share messages like RSA).
  - The value of that shared secret depends on the choice of private keys.
  - Based on exponentiation in a finite (Galois) field.
  - Security relies in the complexity of computing discrete logarithms.
  - **Algorithm**



- A and B agree on
  - q: large prime integer.
  - a such that $\{a^i \bmod q = i+1\}_{i=0}^{q-2}$
    - a is called a primitive root modulo q
- A and B each generate their own keys such that:
  - $\text{PR} < q$
  - $\text{PU} = a^{\text{PR}} \bmod q$
- Shared symmetric key K can then be calculated as:

$$K = a^{\text{PR}_A * \text{PR}_B} \bmod q = \text{PU}_A^{\text{PR}_B} \bmod q = \text{PU}_B^{\text{PR}_A} \bmod q$$

  - Attacker wants to calculate K and have q, a, PU_A, PU_B, but doesn't have PR_A, PR_B
    - That won't be possible without calculating private keys for both A, B then calculating K
    - Calculating private keys will require calculating the log function base a
    - For large a this won't be feasible.

- **Link Encryption vs End-to-End Encryption**

Link Encryption	End-to-End Encryption
- Encrypt data over individual network links.	- Encrypt data only at the end points (hosts or applications).
- Each end-points of a link share a secret key.	- Each pair of hosts share a secret key.
- Requires all links to support encryption.	- Doesn't rely on intermediate devices.
- Encryption is not done by application programmers, less opportunity for human error.	- Used today.
- If traffic levels are constant, defeats traffic analysis.	

- **Sharing session keys**
    - The longer the key is used, the weaker its security.
    - **Solution:** share a master key once (using a secure method), use it to share session keys that change frequently.
    - **Approaches:** KDC and decentralized key distribution.

- **Key Distributed Center (KDC)**
    - A trusted authority to which A and B submit their master keys.
    - A and B both have an encrypted communication channel with KDC.
    - When A and B want to communicate, initiator contacts the KDC.
    - The KDC automatically generates and distributes a session key for them and renews it periodically.
        - TCP uses a new session key for each connection.
        - UDP renews session key periodically or after transmitting a certain number of packets.
    - **Problem:** it introduces a security and performance bottleneck.

- **Decentralized keys distribution**
    - All users share their master keys with all other users.
    - **Problem:** too many key exchanges, changing key introduces large overhead.

- **Sharing public keys**
    - How to ensure that a received public key indeed belongs to the claimed source, attackers may send their own public keys pretending to be someone else.
    - **Public directory:** storing public keys, users provide their identity upon submitting a key
        - **Problem:** Has to be secured and trusted.
        - **Public authority:** a public directory that has its own keys and encrypted channels with users.
            - **Problem:** introduces a bottleneck.
    - **Digital certificates**
        - Proves that a certain party owns a certain public key.
        - Approved by the Certification Authority, which everyone can query to verify the certificate.
        - Certificates has expiration dates and should be renewed.
        - Commonly used format is X.509.
        - Certificates may be revoked before expiry in some conditions (CA private key was compromised, or the organization didn't adhere to policy requirements).
            - This is done through Certificate Revocation List (CRL) that contain the list of revoked certificates and is checked by applications before trusting a certificate.
            - Or through Online Certificate Status Protocol (OCSP).