CNS UNIT-3

How to launch a DNS remote attack

1) Trigger DNS server (let us call it Apollo) to send out the DNS query.
2) Spoof the reply.
3) Negate the Cache effect.

Challenges for doing a remote dns attack

Challenges for Remote Attackers:

1. **Source Port Number:**
    - DNS queries are transmitted via **a UDP packets, and the source port number is a 16-bit random value.**
    - Remote attackers cannot directly observe the DNS query traffic, making it difficult to determine the correct source port number.
2. **Transaction ID:**

- **The 16-bit transaction ID in the DNS header is another crucial piece of information needed for a successful spoofed reply.**
- Without access to the DNS query traffic, remote attackers can only guess these values, with a chance of one out of 2^32 for each guess.

Computational Challenges:

- If an attacker can send out 1000 spoofed queries per second, it would take approximately 50 days to exhaustively try 2^32 possibilities.
- If a botnet of a thousand hosts is used, the time reduces to 1.2 hours for the exhaustive search.

Overlooking Cache Effect:

- The hypothetical attack scenario overlooks the cache effect of the targeted local DNS server.
- **Even with the ability to guess the correct transaction ID and source port number, the attacker may not succeed on the first try because the real reply from the authoritative DNS server will arrive and be cached by the local DNS server.**

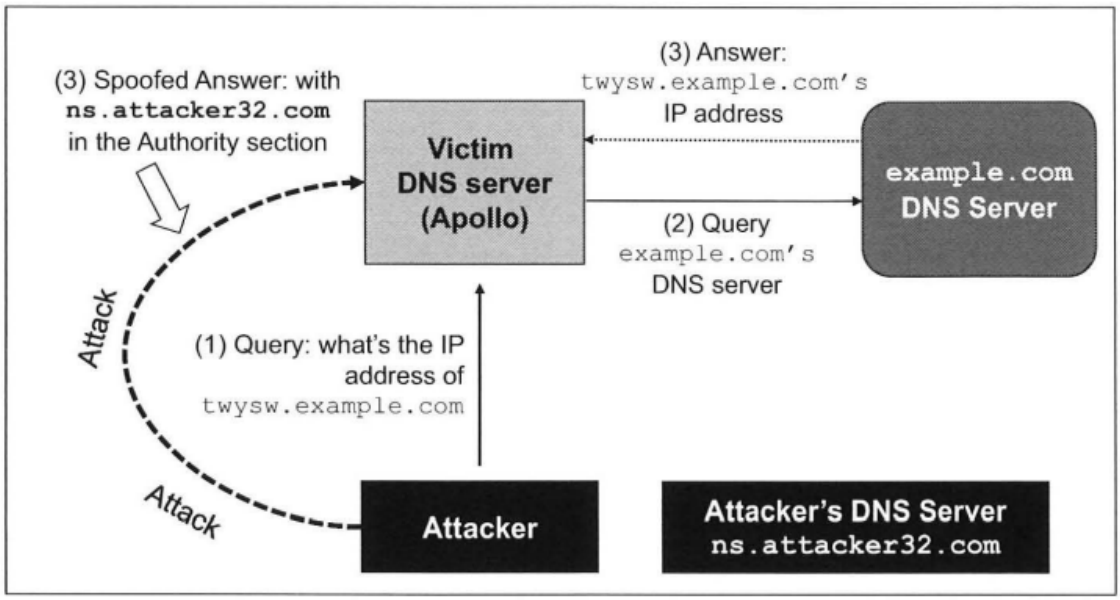
Cache Effect Challenges:

- **The cache effect introduces a waiting period for attackers between attempts.**
- Once the local DNS server has successfully resolved a query and cached the result, it is less likely to send out another query for the same name until the cache entry times out.
- Waiting times for cache expiration can be hours or days, making remote DNS cache poisoning attacks impractical and unrealistic.

Kaminsky Solution

- **Avoid looking at the answer section.**
- **Utilize the authority section to provide nameserver information for the example.com domain.**

- Inform Apollo that the nameserver for the example.com domain is ns.attacker32.com (attacker's machine).
- Once this information is cached, Apollo will query the attacker's machine whenever it needs to resolve a hostname in this domain.
- The attack will then be successful!



The Kaminsky attack is an ingenious approach to conducting DNS cache poisoning attacks remotely without the need to wait for cache expiration. The key challenge is to trigger the targeted DNS server (Apollo) to send out DNS queries for a specific domain, allowing the attacker to exploit the authority section of the DNS reply to inject malicious information into the cache.

Steps of the Kaminsky Attack:

1. **Triggering Queries:**
   - The attacker initiates a DNS query to Apollo for a random hostname within the target domain, such as `twysw.example.com`. This ensures that Apollo sends out a DNS query for the authoritative nameserver of the `example.com` domain.
2. **Legitimate DNS Query:**

- Apollo, not having the answer in its cache, sends a DNS query to the authoritative nameserver of the `example.com` domain. Initially, Apollo may need to query root and .COM servers to obtain the nameserver information, storing it in its cache.

3. **Spoofing Replies:**
   - While waiting for the legitimate reply, the attacker floods Apollo with a stream of spoofed DNS replies. Each spoofed reply attempts different transaction IDs and UDP destination port numbers.
   - In the spoofed replies, the attacker not only provides an IP resolution for the random hostname (`twysw.example.com`) but also includes an NS record specifying `ns.attacker32.com` as the nameserver for the `example.com` domain.
   - If a spoofed reply with a valid transaction ID and UDP port number arrives at Apollo before the legitimate reply, it will be accepted and cached, poisoning Apollo's DNS cache.
4. **Sample Spoofed DNS Response:**

```
QUESTION SECTION:
;twysw.example.com. IN A
ANSWER SECTION:
twysw.example.com. 259200 IN A 1.2.3.4
AUTHORITY SECTION:
example.com. 259200 IN NS ns.attacker32.com
```

5. **Defeating Cache Effect:**
   - If the spoofed DNS response fails, the attacker repeats the process using a different random hostname in the query. Since Apollo's cache does not contain the IP address for the new hostname, it sends out another query, providing the attacker with another chance to conduct the spoofing attack.
   - This cycle effectively defeats the cache effect, allowing the attacker to continuously attempt DNS cache poisoning without waiting for cache expiration.
6. **Successful Attack:**

- If the attack succeeds, Apollo's DNS cache will be poisoned, and the nameserver for the `example.com` domain will be replaced by the attacker's nameserver (`ns.attacker32.com`).

Key Concept - Authority Section:

- **The authority section of the DNS reply is utilized to provide nameserver information for a domain. In the Kaminsky attack, the attacker manipulates this section to influence DNS resolution for the target domain by replacing the legitimate nameserver with the attacker's nameserver.**

By combining these steps, the Kaminsky attack cleverly overcomes the challenges of triggering DNS queries, spoofing replies, and negating the cache effect, allowing for effective remote DNS cache poisoning attacks

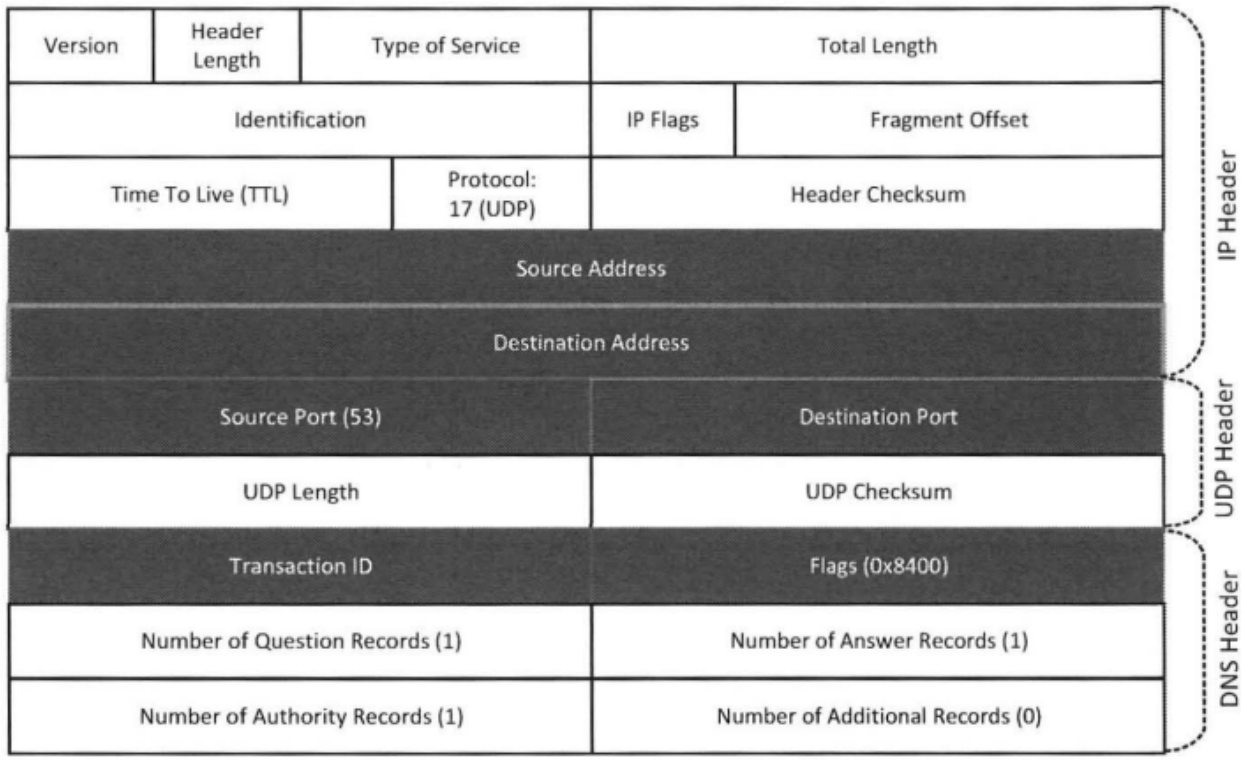


Figure 18.11: The IP, UDP, and DNS headers of the spoofed DNS reply

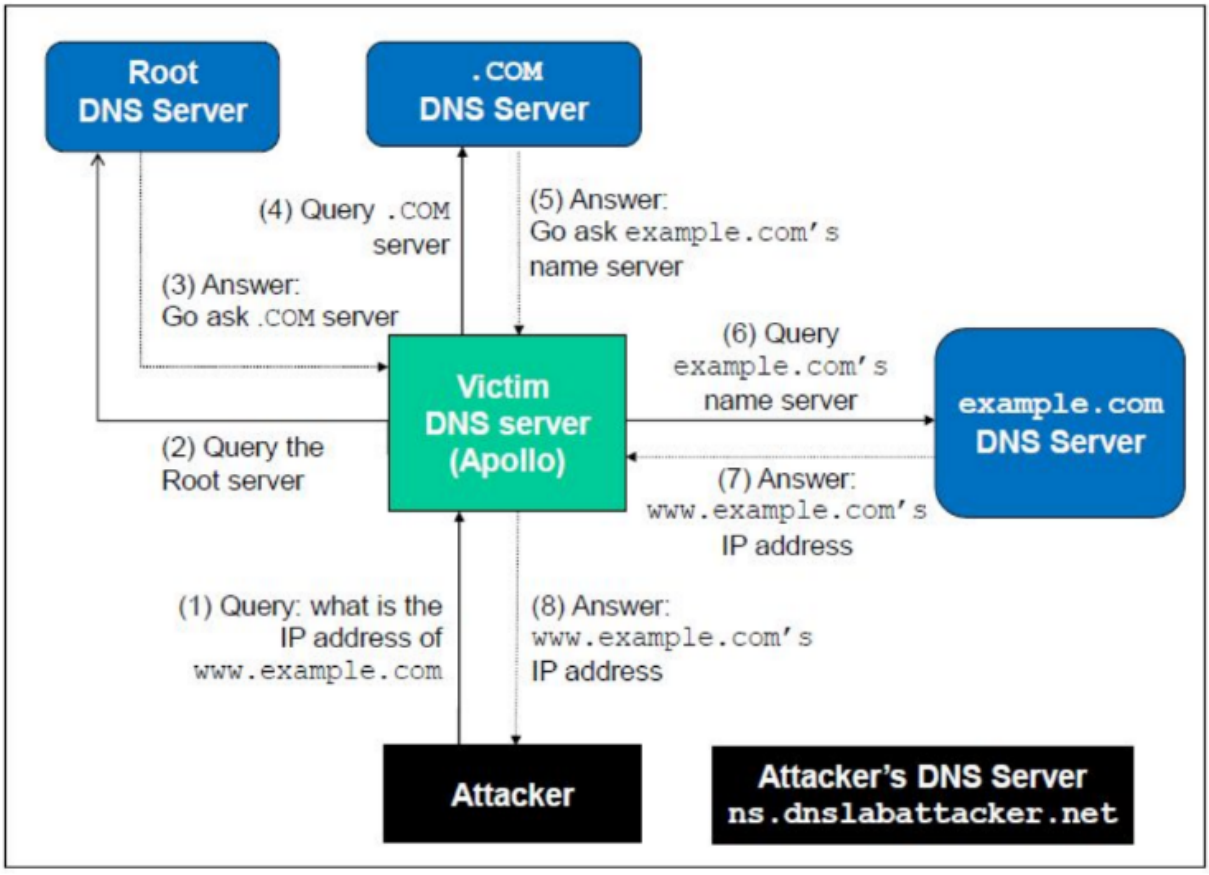
This random name will change for each attack attempt

This answer does not matter

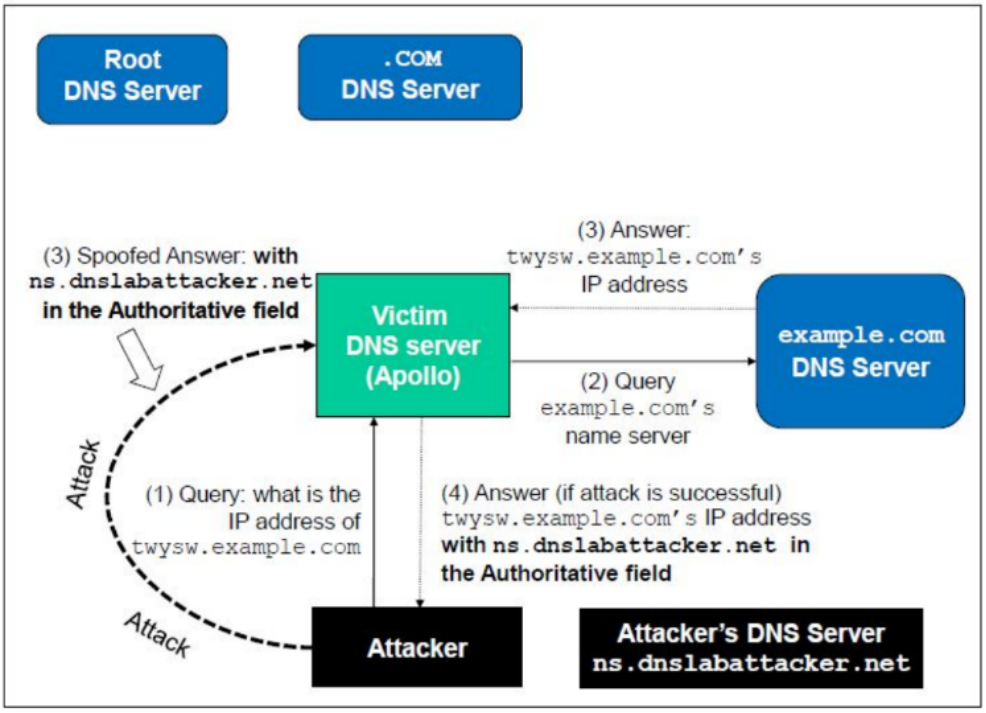
```
;; QUESTION SECTION:
;twysw.example.com.            IN   A

;; ANSWER SECTION:
twysw.example.com.     259200  IN   A   1.2.3.4

;; AUTHORITY SECTION:
example.com.           259200  IN   NS  ns.attacker32.com
```

This is what we want the local DNS server to cache

Tell the DNS server to use this one as the nameserver for the example.com domain



DNS query process when example.com's NS is cached and Attack is successful



Implementation

Python + Scapy and C

- Use Python to construct the packet and write the code in a file.
- Use C to send out the packets by loading the pre-prepared packet.

Steps to Confirm the Success of the Kaminsky Attack:

1. **Search for the Word "Attacker" in DNS Server's Cache:**
    - After running the cache poisoning attack, the attacker should inspect the DNS server's cache for any indication of the injected information. This is done by searching for a specific keyword, in this case, "attacker," within the cache file.
2. **Stop the Attack Program on the Attacker's Machine:**
    - Once the attacker identifies the presence of the keyword "attacker" in the DNS server's cache, it signifies that the cache poisoning attack was

successful. At this point, the attacker stops the attack program on their machine.

3. **Confirmation on User's Machine:**
    - To further confirm the success of the attack, the user's machine is instructed to run specific `dig` commands, a tool for querying DNS servers, with the following outputs:

    a. **$dig NS www.example.com:**

    - This command queries the local DNS server for the authoritative name servers (NS records) of "www.example.com."
    - The expected output: The local DNS server will identify "ns.attacker32.com" as the nameserver for the "example.com" domain. This indicates that the DNS server has been successfully poisoned, and it now recognizes the attacker's nameserver as authoritative.

    b. **$dig www.example.com:**

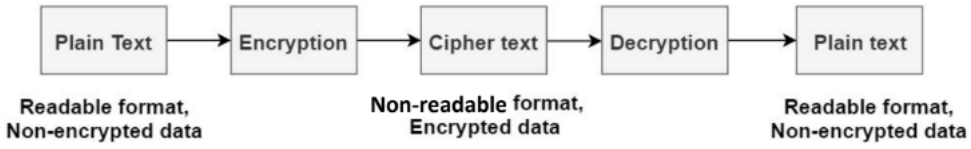
    - This command queries the local DNS server for the IP address (A record) of "www.example.com."
    - The expected output: The user should see the fake IP address for "example.com" in the answer section of the DNS response. This confirms that the DNS cache poisoning attack has successfully redirected queries for "www.example.com" to the malicious IP address controlled by the attacker.

Interpretation of the Outputs:

- **"ns.attacker32.com" Identified as Nameserver:**
    - This signifies that the attacker's crafted authoritative nameserver information (in the authority section of the spoofed response) has been accepted and cached by the DNS server.
- **Fake IP in the Answer Section:**
    - Seeing the fake IP for "example.com" in the answer section confirms that the DNS server is now associating the malicious IP address provided by the attacker with the queried domain.

Cryptograhy

☞ Cryptography, or cryptology is the practice and study of techniques for secure communication in the presence of adversarial behavior



Cryptography basic flow

Symmetric Key Cryptography, also known as Private or Secret key cryptography, involves both the receiver and the sender using a single key to encrypt and decrypt messages. Examples of symmetric key algorithms include AES, DES, Block Cipher, and Stream Cipher.

Asymmetric Key Cryptography, also known as Public key cryptography, uses two separate keys: a private key and a public key. Either key can be used to encrypt data. Examples of asymmetric key algorithms include RSA, DSA, and PKCs.

Hash function: A hash function is a one-way mathematical function that takes input of arbitrary length and produces a fixed-length result called a digest or hash. The hash is almost unique to the original input information. Common hashing algorithms include MD5, SHA-1, SHA-2, NTLM, and LANMAN.

Digital signature: An electronic, encrypted stamp of authentication on digital information, such as email messages, macros, or electronic documents. A digital signature confirms the origin of the information and verifies that it has not been altered.Digital signatures provide assurances of authenticity, integrity, and non-repudiation. **Digital signatures use asymmetric cryptography.**

Example of use of cryptography

- ☞ Arnold writes a document.
  - Document is passed to a hashing algorithm and a digest is created.
  - The digest is encrypted using a Arnold's private key

    --- This is known as Digital Signature
- ☞ Arnold sends the original document + Digital Signature to Jamie.
- ☞ Jamie does the following :
  a. He hashes the document to a digest using the same hashing algorithm.
  b. Decrypts the Digital Signature using Arnold's public key and gets the resultant digest.
  c. If the digests from (a) and (b) match ⇒ document has not been altered and it actually came from Arnold as the document was signed using Arnold's private key .

DNNSEC

DNSSEC is a set of extensions to the DNS (Domain Name System) designed to enhance authentication and ensure data integrity using digital signatures based on public-key cryptography.

Key Components and Concepts:

1. **Digital Signatures:**
   - DNSSEC employs digital signatures to sign DNS data, enhancing authenticity and data integrity.
   - Each DNS zone has a public/private key pair for signing its data.
2. **Chain of Trust:**
   - **DNSSEC establishes a "chain of trust" by signing the public key of each zone with the private key of its parent zone.**
   - **The root zone serves as a trust anchor, and subsequent zones inherit trust through cryptographic key signatures.**
3. **Zone Key Signing:**
   - **Zone owners use the private key to sign DNS data, and the public key is published in the zone.**

- **Recursive resolvers use the zone's public key to validate the authenticity of DNS data.**

4. **Preventing Cache Poisoning:**
    - DNS cache poisoning attempts are defeated by DNSSEC, as any fake data fails the signature checking during validation.
5. **Root Zone's Significance:**
    - **The root zone's public key is crucial for initiating the chain of trust.**
    - Resolvers trusting the root zone's public key can extend trust to subsequent zones.
6. **Trust Anchor:**
    - The public key at the top of the chain (root zone) is known as a trust anchor.
    - Resolvers maintain a list of trust anchors, typically configured with the public key for the root zone.

Key Processes:

1. **Key Pair Usage:**
    - Zone owners use the private key to sign DNS data.
    - Recursive resolvers use the public key to validate the digital signatures on retrieved DNS data.
2. **Chain of Trust Establishment:**
    - The root zone's public key serves as the starting point for trust.
    - Trust extends down the DNS hierarchy through cryptographic key signatures.
3. **Configuration of Trust Anchors:**
    - Resolvers are configured with trust anchors, starting with the root zone's public key.
    - Single trust anchor configurations enable top-down trust establishment.

Benefits:

1. **Authentication and Integrity:**
    - DNSSEC strengthens authentication and ensures the integrity of DNS data.
2. **Preventing Attacks:**
    - Cache poisoning attempts are thwarted, as fake data fails signature validation.
3. **Secure Validation Path:**
    - The chain of trust model ensures a secure and verifiable path for DNS data validation.

DNSSEC Terminology:

1. **Key Pairs for Zone Signing:**
    - **Zone Signing Key (ZSK):**
        - **Public ZSK (PubZSK):** Used to sign and verify zone records.
        - **Private ZSK (Pvt ZSK):** Kept confidential and used for signing/verification.
    - **Key Signing Key (KSK):**
        - **Public KSK (PubKSK):** Used to sign and verify the zone keys.
        - **Private KSK (PvtKSK):** Confidential key for signing/verification.
2. **DNSSEC Resource Record Types:**
    - **DNS Public Key (DNSKEY):**
        - Holds PubZSK and PubKSK for the zone.
    - **Resource Record Signature (RRSIG):**
        - Digitally signs RR sets using PvtZSK or PvtKSK.
    - **Next Secure (NSEC/NSEC3):**
        - Indicates the next authoritative nameserver in the sequence.
    - **Delegation Signer (DS):**
        - Authenticates PubKSK of a child zone.

- Contains a hash/digest of the child zone's PubKSK.

Key Functions:

1. **Zone Signing Key (ZSK):**
   - **Used for signing and verifying zone records.**
   - Secures the authenticity and integrity of zone data.
2. **Key Signing Key (KSK):**
   - **Used to sign and verify the zone keys.**
   - Enhances the security of the DNSSEC infrastructure.

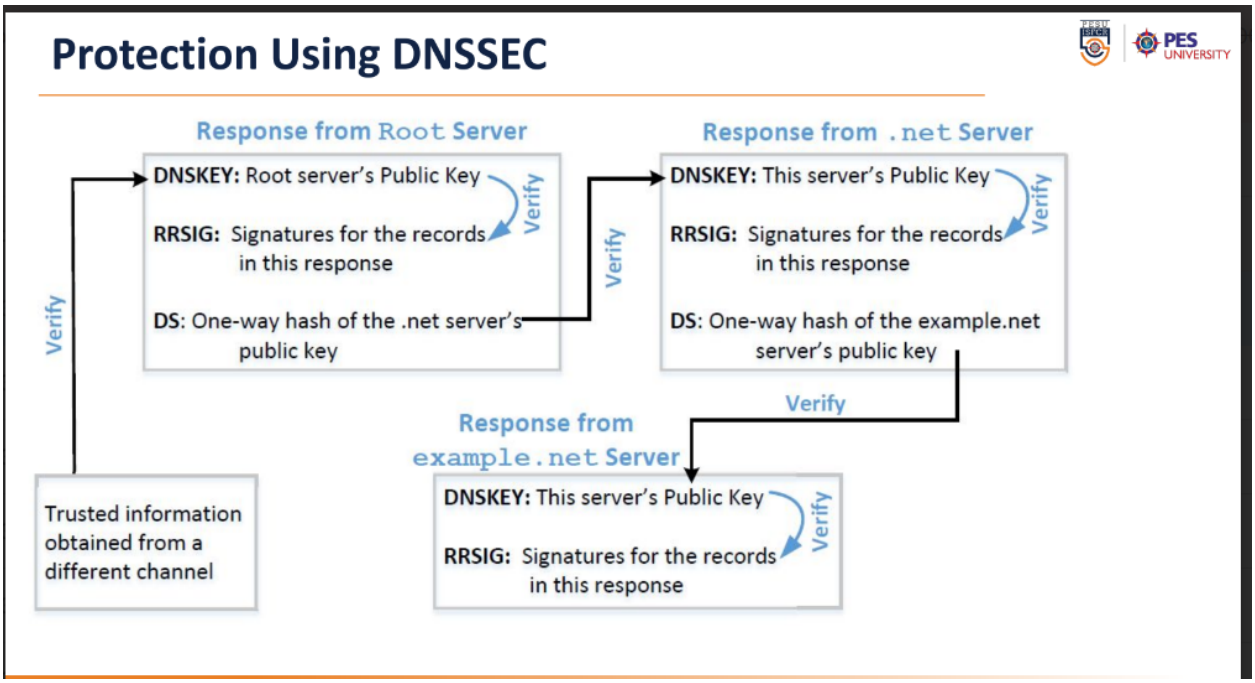
DNSSEC Resource Records:

1. **DNSKEY (DNS Public Key):**
   - **Holds the public components (PubZSK, PubKSK) of the key pairs.**
   - Enables DNS resolvers to verify digital signatures.
2. **RRSIG (Resource Record Signature):**
   - **Digitally signs RR sets using either PvtZSK or PvtKSK.**
   - Ensures the validity of DNS data through signature verification.
3. **NSEC/NSEC3 (Next Secure):**
   - Indicates the next authoritative nameserver in the sequence.
   - Enhances security by preventing certain types of zone walking attacks.
4. **DS (Delegation Signer):**
   - Used to authenticate the PubKSK of a child zone.
   - Contains a hash/digest of the child zone's PubKSK for validation.

Overall Significance:

- **Authentication and Verification:**
  - DNSSEC ensures the authenticity and integrity of DNS data through digital signatures.

- **Preventing Zone Walking:**
  - NSEC/NSEC3 records prevent certain types of zone walking attacks.
- **Secure Key Management:**
  - ZSK and KSK key pairs contribute to secure key management practices.
- **Establishing Trust:**
  - DS records authenticate the PubKSK of a child zone, contributing to the chain of trust in DNSSEC.



HTTP vs HTTPs

HTTP (Hypertext Transfer Protocol):

- **Purpose:** Used for viewing web pages.
- **Data Transmission:** All information is sent over the public internet in plain/clear text.
- **Vulnerability:** Prone to interception, as data is not encrypted during transmission.

- **Security Concerns:** Attackers can easily capture sensitive information.

HTTPS (Secure HTTP):

- **Definition:** HTTPS = HTTP + Security Features.
- **Encryption: Utilizes TLS/SSL (Transport Layer Security/Secure Socket Layer) for data protection.**
- **TLS/SSL Encryption:**
  - **Public Key Encryption: SSL uses public key encryption.**
  - **Secure Key Exchange: Ensures a secure key exchange between the computer and the website.**
- **SSL Certificate Verification:**
  - When a computer connects to an HTTPS-enabled website, the browser requests the website's identity.
  - **The website sends a copy of its SSL certificate to the browser.**
  - If trustworthy, the browser sends an acknowledgment to the web server.
  - The web server responds with an acknowledgment to the browser.
  - The SSL session proceeds, and encrypted data is exchanged.
- **TLS (Transport Layer Security):**
  - Successor to SSL, representing the latest standard cryptographic protocol.

Significance of HTTPS:

- **Data Protection:** Encrypts data during transmission.
- **Secure Communication:** Establishes a secure communication channel between the user's device and the website.
- **Authentication:** Verifies the identity of the website through SSL certificates.
- **TLS Protocol:** Enhances security and privacy on the internet.

SSL Encryption: Hybrid Cryptosystem

SSL Encryption Overview:

- **Asymmetric and Symmetric Encryption:** SSL (Secure Socket Layer) utilizes both asymmetric and symmetric encryption techniques.

Public Key Infrastructure (PKI):

- **Definition:** PKI is a set of hardware, software, people, policies, and procedures designed to create, manage, distribute, use, store, and revoke digital certificates.

Role of PKI in SSL:

- **Hybrid Cryptosystem: PKI uses a hybrid cryptosystem, benefiting from both asymmetric and symmetric encryption.**
- **Certificate Authority (CA):**
  - **A trusted third-party entity that issues digital certificates.**
  - **Manages public keys and credentials for data encryption.**
  - Binds keys with user identities.
- SSL Communications Example:
  - In SSL communications, the server's SSL Certificate consists of an asymmetric public and private key pair.
  - During the SSL Handshake, a symmetric session key is created jointly by the server and the browser.
  - Asymmetric Encryption:
    - The server's public key encrypts data that only its private key can decrypt.
    - Ensures secure transmission of the symmetric session key.
  - Symmetric Encryption:
    - The symmetric session key is used for encrypting and decrypting actual data exchanged during the SSL session.
    - Efficient for bulk data encryption.

Key Components:

- **Asymmetric Keys: Utilized in the SSL Certificate for secure key exchange.**

- **Symmetric Session Key: Created during the SSL Handshake for efficient data encryption.**
- **Certificate Authority (CA): Facilitates trust and binds digital certificates with user identities.**

Protection Using TLS/SSL

Introduction:
Transport Layer Security (TLS/SSL) protocol serves as a robust solution against cache poisoning attacks, enhancing the security of online communications.

Verification Process in TLS/SSL:

1. **DNS Resolution:**
    - After obtaining the IP address for a domain name (e.g., www.example.net) through DNS, the computer seeks to verify the authenticity of the server.
2. **Certificate Presentation:**
    - **The server must present a public-key certificate during the communication initiation.**
    - This certificate is signed by a trusted entity known as a Certificate Authority (CA).
3. **Private Key Demonstration:**
    - **The server demonstrates its ownership of www.example.net by showcasing knowledge of the corresponding private key linked with the presented certificate.**
4. **HTTPS Implementation:**
    - HTTPS (Hypertext Transfer Protocol Secure) operates on top of TLS/SSL.
    - The HTTPS protocol ensures secure data exchange between the client and the server, mitigating the risks of DNS cache poisoning attacks.

Comparison: DNSSEC vs. TLS/SSL:

- **DNSSEC:**
    - **Chain of Trust: Established using DNS zone hierarchy.**

- **Verification: Nameservers in parent zones vouch for those in child zones.**

- **TLS/SSL:**
  - **Chain of Trust: Based on Public Key Infrastructure (PKI).**
  - **Verification: Certificate Authorities (CAs) vouch for the authenticity of computers.**

Firewall

A firewall is a network security device or software that monitors incoming and outgoing network traffic. It determines whether to allow or block specific traffic based on a defined set of security rules, known as the **Ruleset.**

Firewalls have served as the first line of defense in network security for over 25 years. T**hey create a barrier between secured and controlled internal networks, which can be trusted, and untrusted outside networks, such as the Internet.**

Actions taken by Firewall

1. **Allow / Accept:**
   - **Description:** This action permits packets that meet the specified criteria to enter the connected network/host through the firewall.
   - **Use Case:** It is applied to traffic that adheres to established rules and is considered safe for entry.
2. **Deny / Dropped:**
   - **Description:** This action prevents packets that do not meet the specified criteria from entering the other side of the firewall. These packets are dropped without any response.
   - **Use Case:** It is applied to traffic that violates firewall rules, potentially posing a security threat. The firewall simply discards such packets.
3. **Rejected:**
   - **Description:** Similar to the "Deny" action, but with an additional step. When a packet is rejected, the firewall sends **a specifically crafted ICMP (Internet**

Control Message Protocol) packet to the source, indicating the decision to deny the traffic.

- **Use Case:** It is employed when the firewall wants to inform the source of the rejected packet about the decision. This can be useful for diagnostic purposes and to notify the sender of the denial.

Ingress Filtering:

Ingress filtering is a firewall **action that involves inspecting incoming traffic to protect an internal network and prevent potential attacks from external sources.**

- **Objective:**
  - To filter and control traffic entering the internal network.
  - To enforce security policies by allowing only authorized and legitimate traffic.
- **Use Cases:**
  - Filtering out malicious traffic, such as unauthorized access attempts or attacks.
  - Verifying the legitimacy of incoming packets based on predefined rules.

Egress Filtering:

Egress filtering is a **firewall action that involves inspecting outgoing network traffic to control and restrict user access to external networks.**

- **Objective:**
  - To monitor and control traffic leaving the internal network.
  - To prevent users within the internal network from accessing specific external resources.
- **Use Cases:**
  - Blocking access to specific websites or services (e.g., social networking sites in a school or workplace).
  - Preventing unauthorized data exfiltration by restricting certain outbound communications.

Firewall Example

Ingress Filtering:

1. **Allow Inbound Traffic:**
    - **Rule:** Allow incoming traffic on port 80 for web traffic.
    - **Action:** Allow
    - **Description:** Permits incoming web traffic to reach the internal web server.
2. **Deny Inbound Traffic:**
    - **Rule:** Deny incoming traffic on port 22 for SSH.
    - **Action:** Deny
    - **Description:** Blocks incoming SSH traffic, preventing unauthorized access attempts.
3. **Reject Inbound Traffic:**
    - **Rule:** Reject incoming traffic on port 21 for FTP.
    - **Action:** Reject
    - **Description:** Blocks incoming FTP traffic and informs the source with a specifically crafted ICMP packet.

Egress Filtering:

1. **Allow Outbound Traffic:**
    - **Rule:** Allow outgoing traffic to external DNS servers.
    - **Action:** Allow
    - **Description:** Permits internal devices to resolve domain names using external DNS servers.
2. **Deny Outbound Traffic:**
    - **Rule:** Deny outgoing traffic to specific social networking sites.
    - **Action:** Deny

- **Description:** Blocks access to social networking sites from devices within the internal network.

3. **Reject Outbound Traffic:**
   - **Rule:** Reject outgoing traffic to a blacklisted IP address.
   - **Action:** Reject
   - **Description:** Blocks outgoing traffic to the blacklisted IP address and informs the source with a specifically crafted ICMP packet.

Firewall Log Example:

- **Logs for Denied Inbound Traffic:**
  - Time: 2023-11-30 14:30:00
  - Source IP: 203.0.113.1
  - Destination Port: 22 (SSH)
  - Action: Deny
  - Description: Incoming SSH traffic denied.
- **Logs for Rejected Inbound Traffic:**
  - Time: 2023-11-30 15:45:00
  - Source IP: 198.51.100.5
  - Destination Port: 21 (FTP)
  - Action: Reject
  - Description: Incoming FTP traffic rejected; ICMP notification sent to the source.
- **Logs for Denied Outbound Traffic:**
  - Time: 2023-11-30 16:20:00
  - Source IP: 192.168.1.10
  - Destination: facebook.com
  - Action: Deny

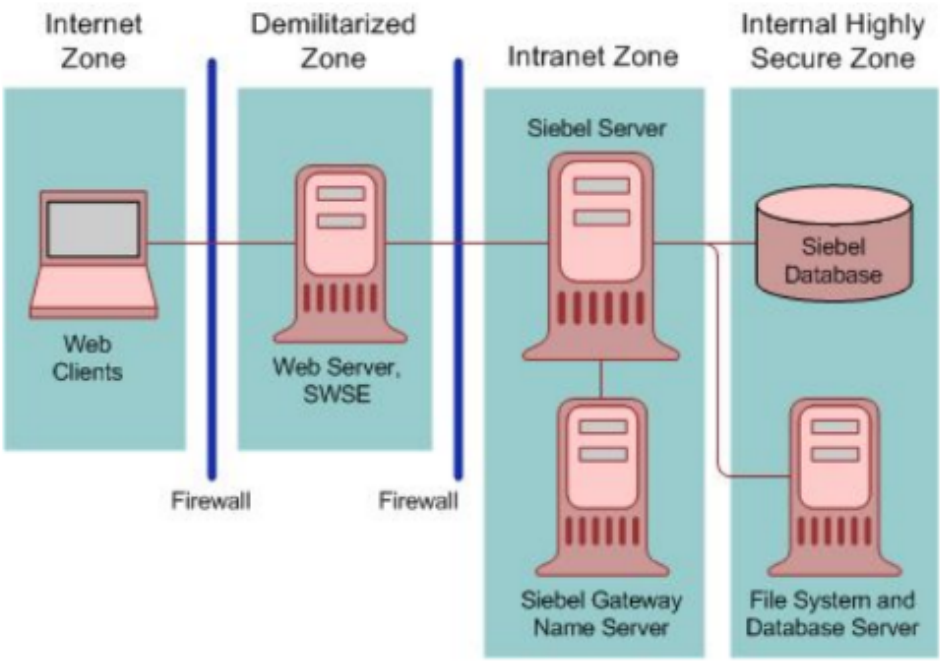
- Description: Outgoing traffic to Facebook denied.

- **Logs for Rejected Outbound Traffic:**
  - Time: 2023-11-30 17:00:00
  - Source IP: 192.168.1.20
  - Destination IP: 198.51.200.15
  - Action: Reject
  - Description: Outgoing traffic to blacklisted IP rejected; ICMP notification sent to the source.

Firewall Location

External/Boundary Firewall

Location: Positioned at the edge of a local or enterprise network



1. **Purpose:**
   a. Safeguards against external threats from the Internet.
   b. Monitors and filters incoming traffic from external sources.

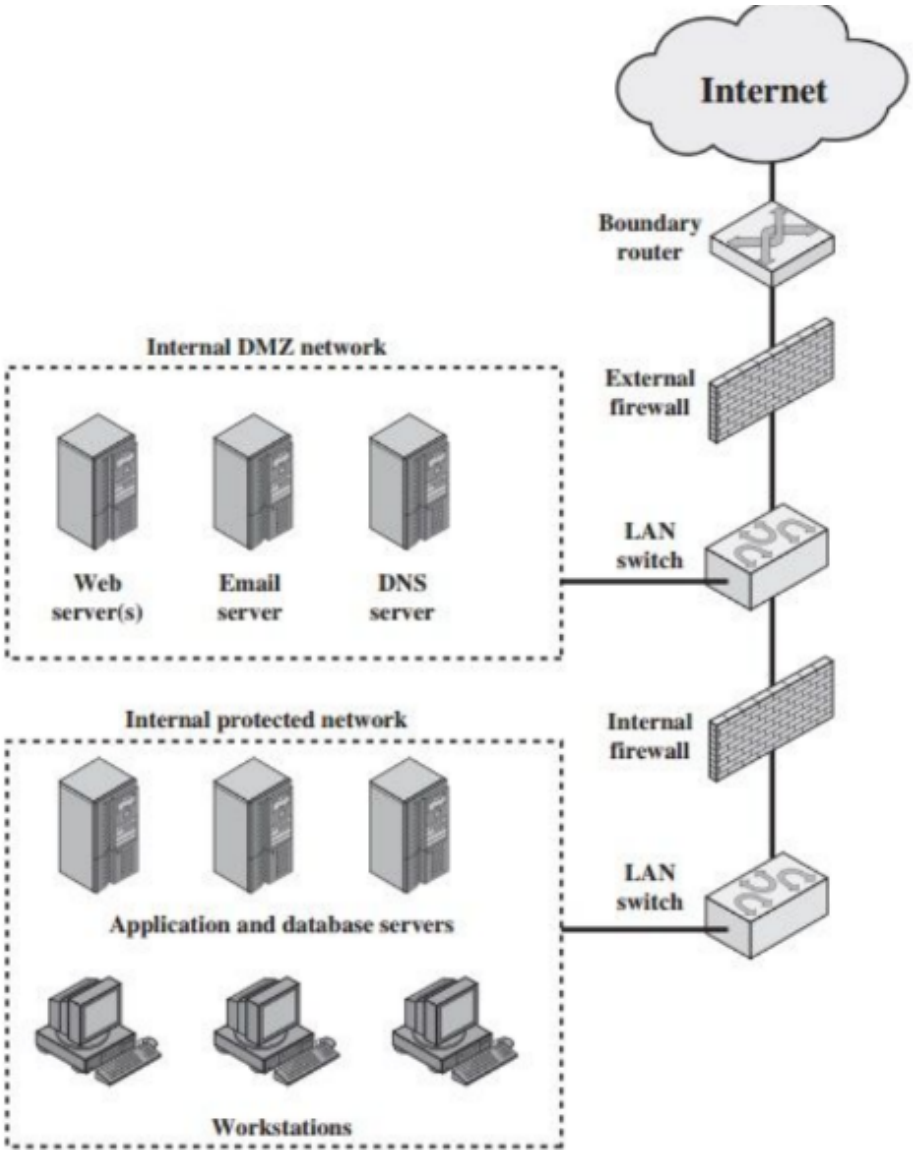
Internal Firewalls

1. **Location:** Deployed within the internal network.
2. **Purpose:**
    a. Protects the bulk of the organization's network.
    b. Adds an additional layer of security, segregating internal segments.

Types of Firewalls

Host-Based Firewalls vs. Network-Based Firewalls

Aspect	Host-Based Firewalls	Network-Based Firewalls
Location	Installed on individual hosts	Placed strategically in the network path
Dependency	Integrated into the host's OS as software applications	Operates independently of the devices it protects
Scope of Protection	Protects a single host	Offers protection for network segments or multiple devices
Implementation	Integrated into the host's OS	Standalone hardware devices or software modules in routers



DMZ

Demilitarized Zone (DMZ) Overview:

1. **Definition:**
    - The DMZ, also known as the Service Network, is a small, isolated network strategically positioned between the Internet and the private/internal network.
2. **Purpose:**

- **Service Provision:** Any service intended for users on the external network can be placed in the DMZ. Examples include web servers, mail servers, FTP servers, etc.
- **Isolation:** It provides a controlled and isolated environment, separating external-facing services from the internal network.

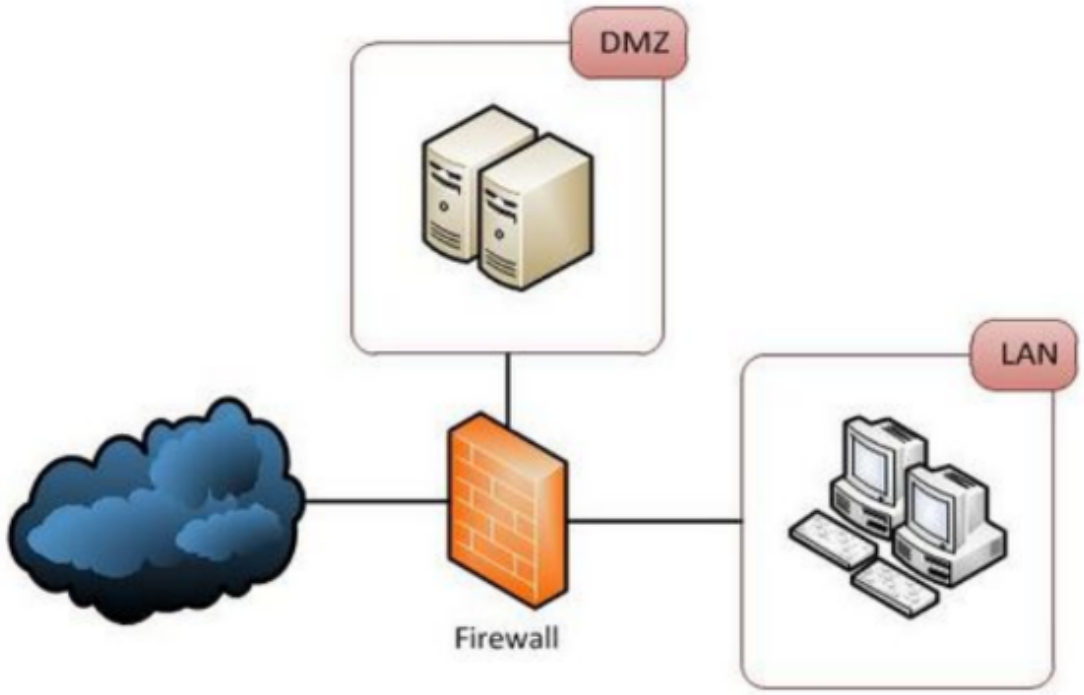
3. **Service Examples in DMZ:**
   - **Web Servers:** Hosted in the DMZ to provide public-facing websites.
   - **Mail Servers:** Positioned for external email communication.
   - **FTP Servers:** Used for file transfer services accessible from the Internet.
4. **Security Positioning:**
   - **Intermediate Security:** The DMZ is designed to be neither as secure as the internal network nor as insecure as the public Internet.
   - **Controlled Connectivity:** Hosts in the DMZ have limited connectivity to specific hosts in the internal network. This controlled connectivity ensures that the DMZ content does not compromise the internal network's security.
5. **Connectivity Principles:**
   - **Limited Internal Access:** DMZ hosts are permitted to have only limited connectivity to specific hosts in the internal network.
   - **Controlled Communication:** Communication between the DMZ and internal network is carefully regulated to minimize security risks.
6. **Design Flexibility:**
   - **Varied Configurations:** There are many different ways to design a network with a DMZ, allowing flexibility based on specific security and operational requirements.
   - **Customizable Security Policies:** Organizations can tailor the DMZ design to meet their unique security and service needs.

Firewall Placement in DMZ - Single Firewall:

1. **Description:**

- **A single firewall with a minimum of three network interfaces is used to establish a network architecture containing a Demilitarized Zone (DMZ).**
- **The firewall acts as a single point of control for traffic going to both the DMZ and the internal network.**

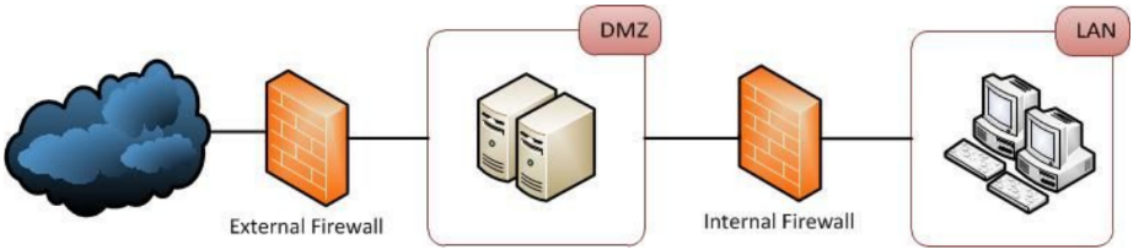
2. **Considerations:**
   - **Single Point of Failure:** The single firewall becomes a potential single point of failure for the entire network.
   - **Traffic Handling:** The firewall must efficiently handle all traffic destined for the DMZ and the internal network.



Firewall Placement in DMZ - Dual Firewall:

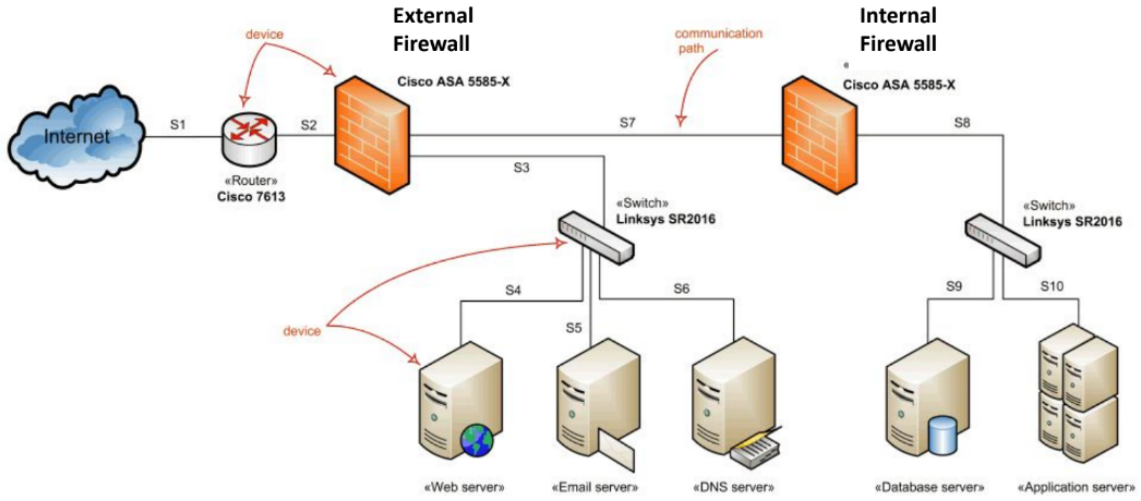
1. **Configuration:**
   - **Two firewalls are employed - the first (front-end/perimeter) firewall allows traffic only to the DMZ, while the second (back-end/internal) firewall permits traffic from the internal network to the DMZ.**
2. **Security Advantages:**

- **Enhanced Security:** Considered more secure as compromising two devices is necessary for an intrusion.
- **Controlled Traffic Flow:** Each firewall is dedicated to specific traffic, providing a controlled flow of information.



Sample Topology:

- **External Firewall:**
  - Provides access control and protection for DMZ systems.
  - Offers basic protection for the overall enterprise network.
- **Internal Firewall(s):**
  - Adds stringent filtering to protect the internal network.
  - Ensures two-way protection for DMZ: protects the internal network from DMZ-based attacks and vice versa.
  - Multiple internal firewalls can segment and protect different parts of the internal network.



The Demilitarized Zone (DMZ):

1. **Advantages of DMZ:**
    - **Three Layers of Protection:** Three separate routers (outside firewall router, bastion host or proxy, inside firewall router) create layers of defense against intruders.
    - **Routing Control:** The DMZ network is selectively advertised to the Internet and the private network, controlling routing and DNS information exchange.
    - **Network Address Translation (NAT):** Allows the use of NAT on the bastion host to eliminate the need for renumbering or re-subnetting the private network.

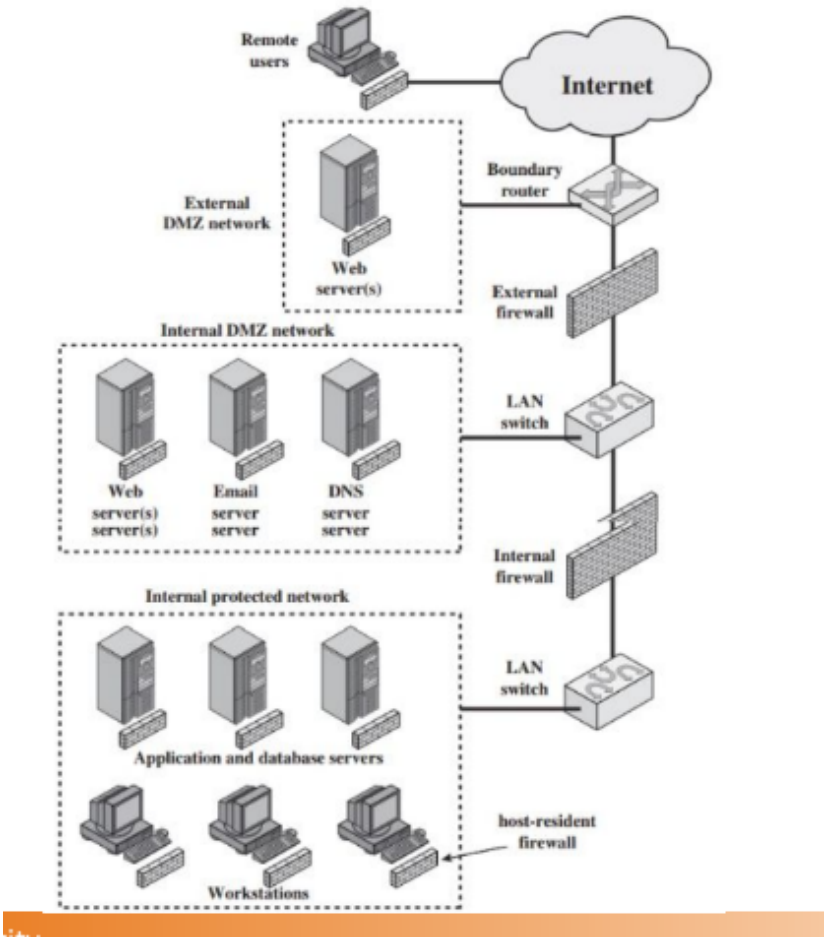
Bastion Host:

1. **Definition:**
    - **A dedicated server facilitating authorized user access from an external network (e.g., the Internet) to a private network.**
    - **Performs authentication and proxy functions, becoming the sole ingress path to internal resources.**
    - Placed outside the firewall or within a DMZ.
2. **Functions:**

- **Authentication:** Verifies user credentials for secure access.
- **Proxy Services:** Acts as an intermediary for communication.
- **Ingress Path:** Becomes the primary entry point to internal resources.
- **Server Types:** Any single-purpose server (e.g., NAT, DNS, web, or mail) can serve as a bastion host.

Location Based Firewalls

Distributed Firewall:



A distributed firewall configuration involves a combination of stand-alone firewall devices and host-based firewalls operating together under centralized administrative control.

Host-Based Firewalls:

- Configured on individual servers and workstations.
- Personal firewalls on local and remote user systems.
- Tailored protection against internal attacks.
- Administrators can configure host-resident firewalls on numerous machines for specific protection.

Stand-Alone Firewalls:

- Provide global protection.
- Include both internal and external firewalls.
- External firewall protects against external threats.
- Internal firewalls provide protection within the network.
- Configuration allows for the establishment of both internal and external DMZs.

DMZ Configuration:

- Internal DMZ for critical servers with sensitive information.
- External DMZ for servers with less critical information.
- Web servers with lower sensitivity can be placed in the external DMZ.
- Host-based firewalls can be employed on these web servers for additional protection.

Firewall Policy

A firewall policy is a set of guidelines that dictate how firewalls should manage network traffic for specific elements such as IP addresses, address ranges, protocols, applications, and content types. These guidelines are based on the organization's information security policies.

- **Policy Development:**
  - Involves risk analysis to identify necessary traffic.
  - Considers types of traffic allowed under specific circumstances.

- Grounded in threat evaluation, vulnerability assessment, and countermeasure effectiveness.
- Reflects potential impact if systems or data are compromised.

- **Documentation and Maintenance:**
  - The firewall policy should be documented in the system security plan.
  - Requires frequent updates based on emerging threats, vulnerabilities, and changing organizational needs.
  - Specific guidance on handling changes to the ruleset should be included.

Requirements/Design Goals of a Firewall:

- **Authorized Traffic:**
  - Only allows authorized traffic defined by the security policy.
- **Zone-to-Zone Traffic:**
  - All traffic between trust zones should pass through the firewall.
- **Immunity to Penetration:**
  - The firewall itself must be immune to penetration, requiring a hardened system with secured operating systems.

Zero Trust Architecture

Zero Trust is a cybersecurity strategy rooted in the principle of "never trust, always verify." It aims to secure organizations by eliminating implicit trust and continuously validating every stage of a digital interaction.

- **Key Elements:**
  - Eliminates implicit trust in traditional security models.
  - Recognizes the need for continuous verification in a dynamic digital environment.
  - Critical for securing hybrid workforces, cloud migrations, and evolving security operations.

Monitoring Outgoing Traffic

Monitoring outgoing traffic is essential for maintaining a secure network environment.

- **Importance:**
  - Equally crucial as monitoring incoming traffic.
  - Mitigates risks arising from internal threats.
  - Addresses intentional or unintentional actions by employees.
  - Parameters like business impact and costs incurred can be considered.

Cybersecurity Risks you can infer

	Risks	C	I	A
1	Inappropriate high bandwidth web usage (e.g. Streaming Media)			X
2	Access to malicious web sites resulting in Host infection or compromise (e.g. Worms, Virus, theft/compromise/destruction of data)	X	X	X
3	Access to BOT command control centre (BOT capabilities include DDoS, Keylogger, SPAM, Phishing, Malware distribution)	X		X
4	Covert channels – Information theft by Insiders	X		
5	Unauthorised remote access (weak authentication leading to host compromise)	X	X	
6	Access to external unofficial/compromised resources (DNS, FTP, NTP)	X	X	X
7	Usage of clear text protocols over internet (sniffing, MITM)	X		
8	Unmanaged Internet Messenger usage (infection/compromise of host)	X		X

Sensible Firewall Policy:

A sensible firewall policy is crucial for effective network security. Here are key principles that make a firewall policy sensible:

1. **Default to Blocking Traffic:**
   - The policy should default to blocking all traffic.
   - Only explicitly allow traffic that aligns with organizational needs and policies.

- The "default deny" approach enhances security by restricting unknown and potentially malicious traffic.

2. **Driven by Institutional Policy:**
   - The firewall policy should align with the broader institutional policies and security objectives.
   - It should reflect the organization's risk tolerance, business requirements, and regulatory compliance.
3. **Linked to Natural-Language Policy:**
   - The firewall policy should have a clear linkage to a natural-language policy.
   - Ensure that technical firewall rules are easily understood and interpreted in the context of the organization's overall policies.
4. **Defined in Simple, Straightforward Terms:**
   - Avoid complexity in rule definitions.
   - Define rules in simple and straightforward terms, making it easier for administrators to manage and audit.
5. **Use of Reusable Objects:**
   - Implement reusable objects, such as address and service objects.
   - This facilitates easier management, as changes to objects reflect across multiple rules, ensuring consistency.
6. **Descriptive Comments:**
   - Include descriptive comments in the firewall rules.
   - Comments provide clarity on the purpose of each rule, aiding administrators in understanding the rationale behind specific configurations.

"Default Deny" Principle:

1. **Preferred Default:**
   - "Default deny" is the preferred default setting for firewalls.
   - This means that all traffic is blocked by default unless explicitly allowed.
2. **Challenges with "Default Allow":**

- Identifying all situations where traffic should be explicitly denied is challenging with a "default allow" policy.
- Enumerating all possible undesired activities in advance is impractical.

3. **Simplicity and Auditing:**
    - "Default deny" simplifies the policy by focusing on identifying supported services and expected traffic.
    - Blocking everything else by default makes auditing more straightforward.

Characterstics used by firewall policy

- IP address
- Application protocol
- User Id
- Network actvity

Firewall Rules for Traffic Controls:

Firewall rules play a critical role in controlling and securing network traffic. The rules are typically defined to provide the following traffic controls on the network:

1. **User Control:**
    - **Objective:** Controls access to data based on the role of the user attempting to access it.
    - **Application:** Applied to users within the firewall perimeter.
    - **Implementation:** User-based rules ensure that individuals or groups have access only to the resources that align with their roles and responsibilities.
2. **Service Control:**
    - **Objective:** Controls access based on the type of service offered by the host.

- **Parameters:** Applied based on network address, connection protocol, and port numbers.
- **Implementation:** Service-specific rules enable or restrict access to hosts based on the nature of services (e.g., web, email) they provide, ensuring that only authorized services are accessible.

3. **Direction Control:**
   - **Objective:** Determines the direction in which requests may be initiated and allowed to flow through the firewall.
   - **Types:**
     - **Inbound:** Traffic from the network to the firewall.
     - **Outbound:** Traffic from the firewall to the network.
   - **Implementation:** Directional rules specify whether traffic is allowed to enter the network from external sources (inbound) or leave the network to external destinations (outbound).

Role of Each Traffic Control:

1. **User Control:**
   - **Benefit:** Ensures that users have access only to the data relevant to their roles, reducing the risk of unauthorized access.
   - **Example:** Different access permissions for employees in finance, marketing, and IT.
2. **Service Control:**
   - **Benefit:** Restricts access to specific services, minimizing the potential impact of unauthorized or malicious services.
   - **Example:** Allowing web traffic on port 80 for HTTP but blocking traffic on other ports.
3. **Direction Control:**
   - **Benefit:** Manages the flow of traffic in and out of the network, preventing unauthorized access and data exfiltration.

- **Example:** Allowing inbound traffic for user requests and outbound traffic for server responses.

Security Policy/Controls - Network Firewall

The table outlines baseline security controls for network-based firewalls, categorized by security levels: High, Medium, and Low.

Control ID	Description	High	Medium	Low
NF.A.01	Segmentation of traffic with a dedicated network firewall	Required	Recommended	Optional
NF.A.02	Enable in default deny mode and permit the minimum necessary services	Required Immediate	Required Immediate	Required Immediate
NF.A.03	Document firewall rules, including purpose, justification, and approvals for allowed services, protocols, and ports. Include additional security features for insecure protocols.	Required Immediate	Required Effective July 2020 (Recommended)	Recommended
NF.A.04	Configure the network to	Required Immediate	Required Immediate	Required Immediate

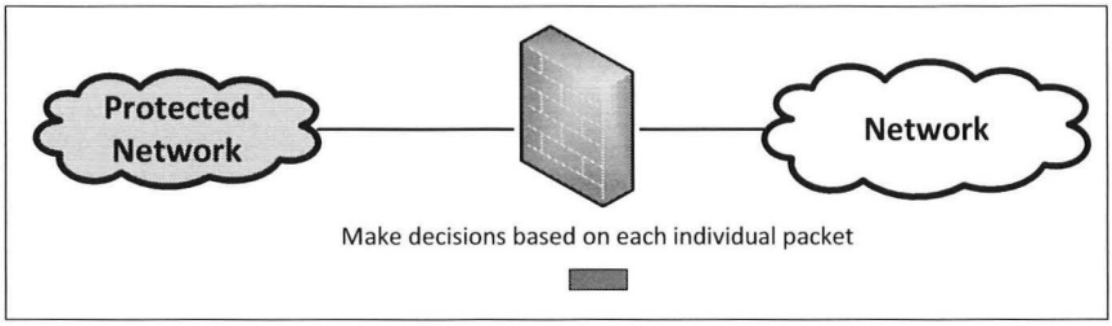
	deny all traffic upon firewall failure			
NF.A.05	Review usage of firewall rules and remove rules that are no longer needed	Required 3-6 months	Recommended 6-12 months	Recommended

Implementation Timeline:

- **Immediate Actions:** NF.A.02, NF.A.03, NF.A.04 (High, Medium, Low).
- **Short-Term Actions (3-6 Months):** NF.A.01, NF.A.05 (High, Low).
- **Medium-Term Actions (6-12 Months):** NF.A.05 (Medium).

Types of Firewalls

1. Packet Filter Firewall (Stateless Firewall)



(A) Packet Filter Firewall

Operation

- Operates at router points in the network.
- Administrator defines rules (ACLs) to manage allowed ports and IP addresses.
- Stateless, performing per-packet inspection.

Access Control List (ACL) Example

```
# Allow incoming traffic to web server (192.168.1.1) on por
t 80
access-list 101 permit tcp any host 192.168.1.1 eq www

# Deny all other incoming traffic
access-list 101 deny ip any any
```

Advantages

- Single device can filter traffic for the entire network.
- Fast and efficient.
- Inexpensive.

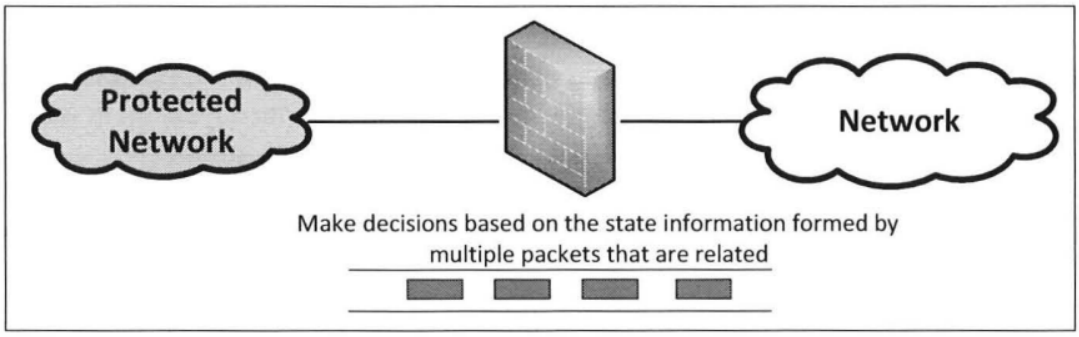
Disadvantages

- Lacks broader context for security.
- Doesn't check payload, susceptible to spoofing.
- Not ideal for every network.

Who Can Use?

- Suitable for small or budget-constrained organizations.
- Larger enterprises can use it as part of a layered defense.

2. Stateful Firewall



(B) Stateful Firewall

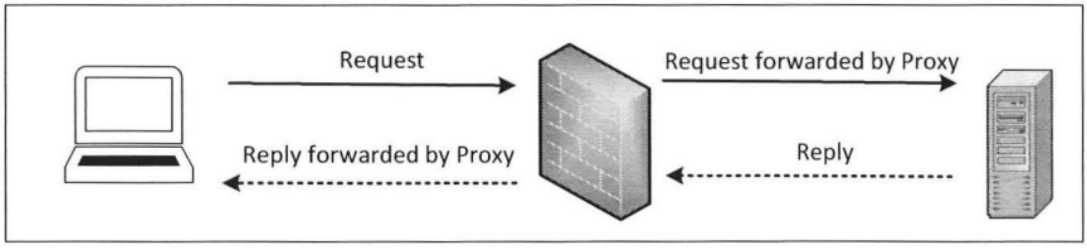
Operation

- Tracks the state of traffic by monitoring connection interactions.
- Maintains a connection state table.
- Allows connections through ports holding open connections.

Stateful Packet Filtering Example

```
# Allow outgoing traffic initiated from inside to outside
permit tcp inside-network any established

# Deny incoming traffic not part of an established connecti
on
deny tcp any outside-network
```

3. Application/Proxy Firewall



(C) Application/Proxy Firewall

Operation

- Controls input, output, and access to/from an application or service.
- Acts as an intermediary, impersonating the intended recipient.
- Analyzes data up to the application layer.

Proxy Configuration Example (Squid Proxy)

```
# Define ACL for allowed networks
acl allowed_networks src 192.168.1.0/24

# Allow access to HTTP for the allowed networks
http_access allow allowed_networks
```

4. Next-Generation Firewall (NGFW)

Operation

- Provides capabilities beyond a traditional, stateful firewall.
- Features include application awareness, deep packet inspection, intrusion prevention, TLS/SSL inspection, antivirus.
- Aware of application-layer data, but doesn't impact data transfer rate.

NGFW Configuration Example (Cisco ASA)

```
# Enable application inspection
policy-map global_policy
  class inspection_default
    inspect ftp
    inspect h323 h225
    inspect h323 ras
    inspect rsh
    ...
```

NG Firewalls Benefits

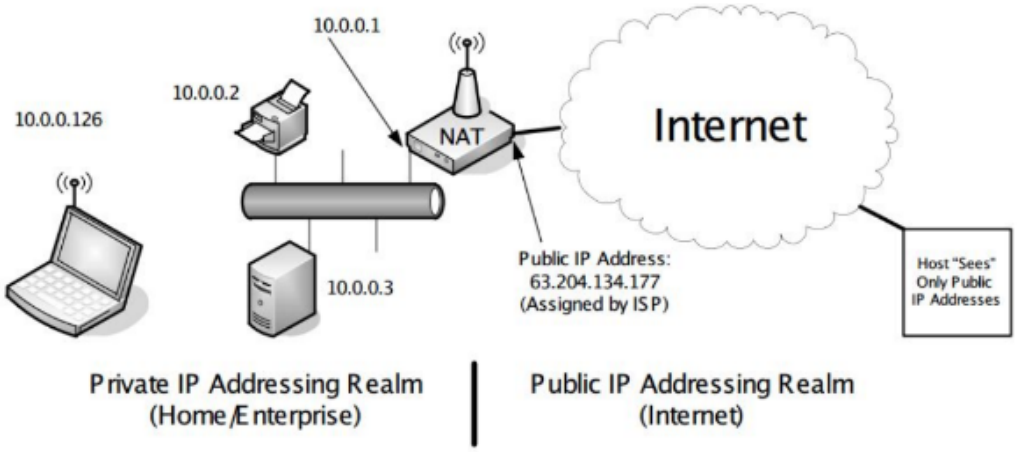
- Single choke point for traffic.

- Location for monitoring security events.
- Protects internal/external systems.
- Filters communications based on content.
- Performs NAT (Network Address Translation).
- Logging for intrusion detection and forensics.

In summary, each firewall type has its specific use case and configuration, and organizations should choose based on their security requirements and network characteristics.

NAT

Most NATs perform both translation and packet filtering, and the packet-filtering criteria depend on the dynamics of the NAT state. The choice of packet-filtering policy may have a different granularity.
For example, the treatment of unsolicited packets (those not associated with packets originating from behind the NAT) received by the NAT may depend on source and destination IP address and/or source and destination port number.



Basic NAT:

Operation:

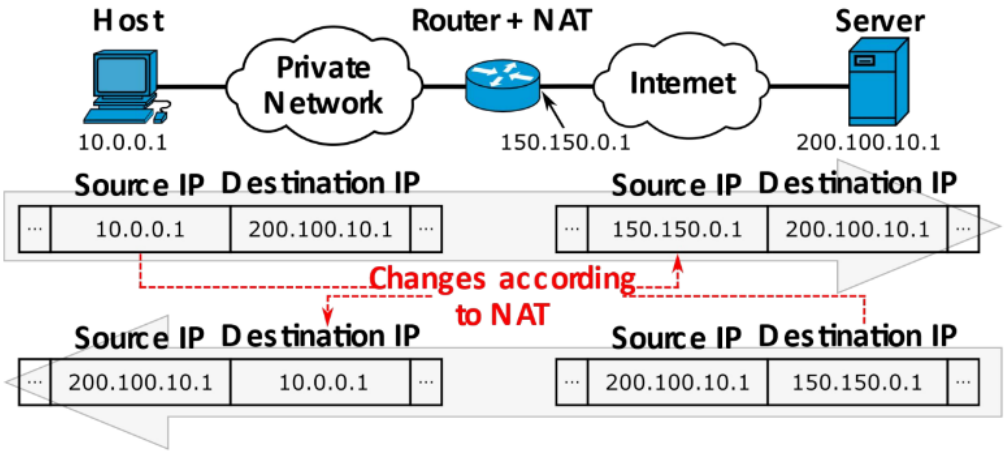
- Performs rewriting of IP addresses only.
- Private addresses are rewritten to be public addresses.
- Often uses a pool or range of public addresses.

Purpose:

- Provides a way to hide internal network structures.
- Facilitates communication between private networks and the public internet.

Limitation:

- Does not significantly reduce the need for globally routable IP addresses.



NAPT

Network Address Port Translation (NAPT) is a type of Network Address Translation (NAT) that adds an extra layer of translation beyond just IP addresses. Here are some details and differences between NAT and NAPT:

NAT Address Pool:

Definition:

- A NAT may have several external addresses available to use.

- This set of addresses is typically referred to as the NAT pool or NAT address pool.

Purpose:

- **Allows the NAT to use multiple external addresses for translating internal addresses.**

Address Pairing in NAPT:

Definition:

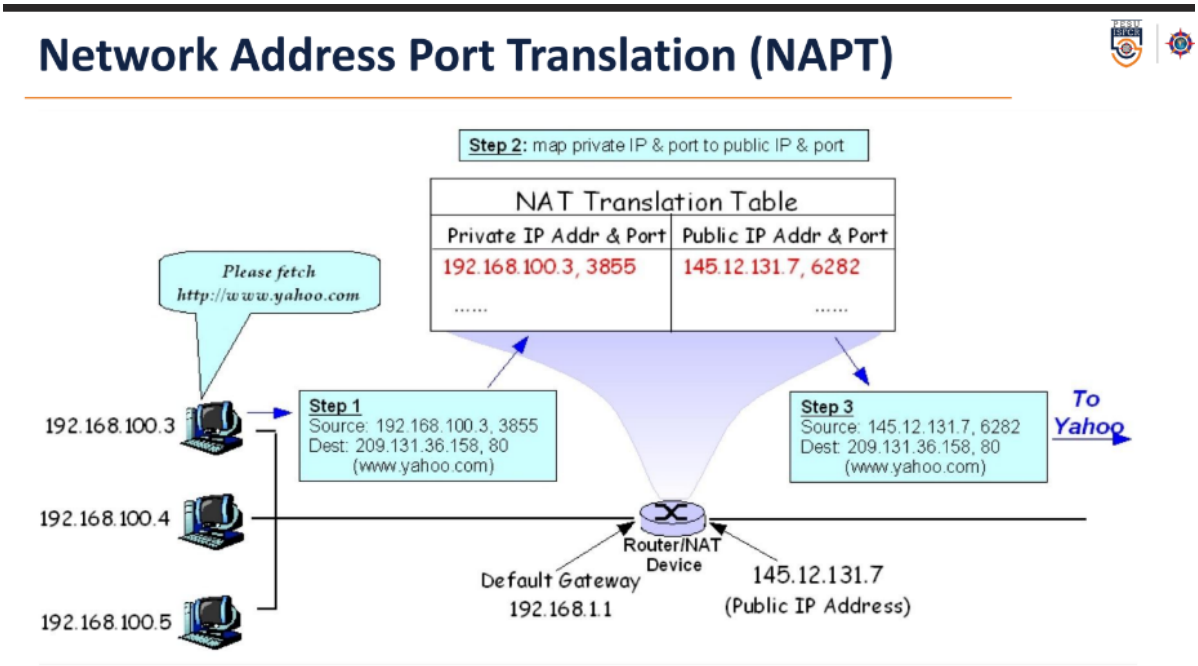
- **Address pairing refers to the behavior of assigning the same external IP address for multiple simultaneous connections initiated by a single host behind the NAT.**

Behavior:

- **Arbitrary:** No restriction on which external address is used for any association.
- **Paired:** Implements address pairing, meaning that the same external IP address is used for all connections initiated by a single internal host.

Recommendation:

- Pairing is the recommended behavior for all transports.
- Without pairing, the communication peer of an internal host may mistakenly interpret it as communicating with different hosts.



Network Address Port Translation (NAPT):

Operation:

- **NAPT extends NAT by also translating transport-layer port numbers.**
- **It uses port information to distinguish between connections from different internal hosts sharing the same external IP address.**

Example Configuration (iptables for Linux):

```
# Enable NAPT for outgoing traffic on eth0 interface
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Difference between NAT and NAPT:

- **NAT: Involves the translation of IP addresses only.**
- **NAPT: Involves the translation of both IP addresses and port numbers**.

IP Masquerading:

- IP Masquerading is another term often used interchangeably with NAPT.

- **It emphasizes the idea of hiding the internal network structure by masquerading internal addresses behind a single external address.**

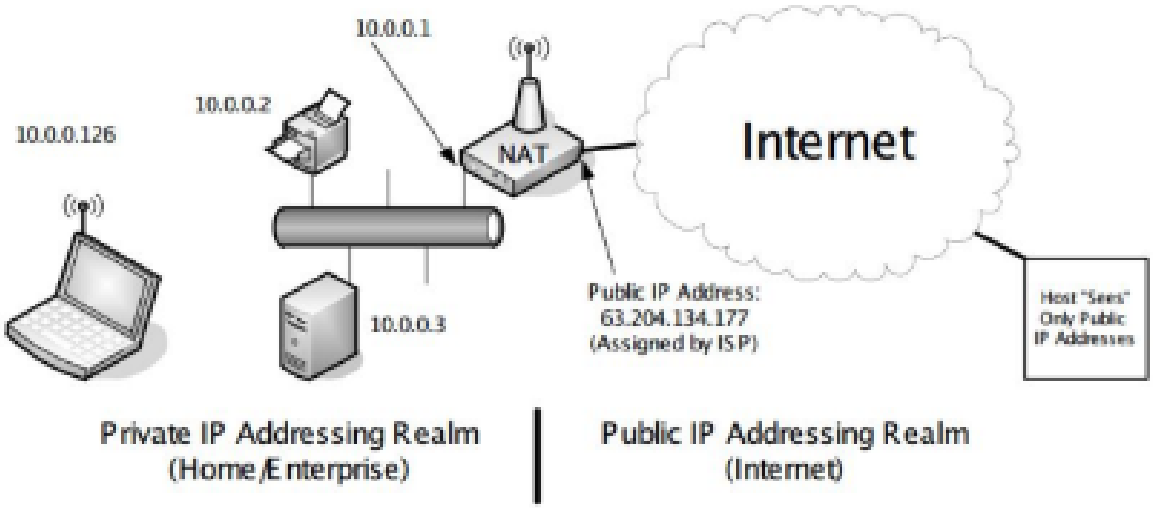
Feature	NAT (Network Address Translation)	NATP (Network Address Translation Protocol)
Definition	Translates private IP addresses to a public IP address.	Extends NAT by allowing translation based on port numbers and protocols.
Address Mapping	One-to-One mapping between private and public addresses.	Many-to-One mapping using ports and protocols.
Port Usage	Doesn't consider port numbers in translation.	Takes into account port numbers for translation.
Granularity	Operates at the IP address level.	Operates at the IP address, port, and protocol levels.
Connection Handling	Basic connection tracking without considering ports.	Advanced connection tracking using ports and protocols.
Use Case	Commonly used for conserving public IP addresses.	Used when a single public IP needs to serve multiple private IPs with different ports.
Security Implications	Limited security due to lack of port-based distinctions.	Enhanced security with better differentiation based on ports and protocols.
Complexity	Simpler configuration and management.	More complex due to considerations of ports and protocols.
Common Protocols	Typically used for basic Internet access.	Useful for protocols that require multiple connections, like FTP.

NAT with Major Transport Protocols

1. NAT and TCP:

1. **Client Sends Request:**
    - A device on a private network (like a home Wi-Fi) wants to access a website.
    - The request is sent with a specific IP address and port.

2. **NAT Steps In:**
    - The home router (NAT) receives the request and notices it's a new connection.
    - NAT changes the device's IP address to the router's own public IP address.
    - It creates a note to remember this change (NAT mapping).
3. **Server Responds:**
    - The website (server) responds to the router's public IP and a specific port.
    - The server uses the same port the device chose for its request (port preservation).
4. **NAT Updates the Response:**
    - The router, using its notes, changes the response back to the device's original IP and port.
    - The updated response is sent back to the device.
5. **Connection Established:**
    - The device receives the response and is now connected to the website.
6. **Keeping Track:**
    - The router keeps track of the connection.
    - It knows when the connection is done (FIN flags).
    - It also removes old connections that are no longer active.



2. NAT and UDP:

- **UDP and Connectionlessness:**
  - UDP being connectionless, NAT uses a mapping timer to clear NAT state.
  - Timers are refreshed when packets travel from inside to outside of the NAT.

3. NAT and ICMP:

- **ICMP Handling:**
  - ICMP provides status information and measurements about IP packets.
  - ICMP error messages from outside passing through NAT have IP addresses rewritten.
  - NAT recognizes outgoing informational requests and sets timers for returning responses.

Security Provided by NAT:

Network Address Translation (NAT) offers a degree of security, akin to a firewall, through the following mechanisms:

1. **Default Inaccessibility from the Internet:**

- Systems on the private side of NAT, which use private addresses, are, by default, unreachable from the Internet. This creates a natural barrier that helps prevent unauthorized access.

2. **Outgoing Traffic Allowance:**
    - A common policy allows outgoing and returning traffic (associated with outgoing traffic) to pass through NAT. However, it blocks almost all incoming new connection requests. This inhibits probing attacks seeking to identify active hosts available for exploitation.
3. **Topology Hiding:**
    - NAT, especially Network Address Port Translation (NAPT), conceals the number and configuration of internal addresses from the outside. This topology hiding provides an additional layer of security by making internal network details less visible.

NAT Filtering Behavior:

- **Binding Establishment:**
    - When NAT establishes a binding for a TCP connection, UDP association, or ICMP traffic, it determines its filtering behavior for returning traffic. This behavior is crucial when NAT acts as a firewall.
- **Filtering Types:**
    - **Endpoint-Independent:** Any incoming traffic is permitted as soon as any mapping is established for an internal host, regardless of the source.
    - **Address-Dependent:** Traffic is permitted only if the source had been previously contacted by the internal host.
    - **Address- and Port-Dependent:** Traffic is permitted only if the source and port had been previously contacted by the internal host.

Port Forwarding:

1. **Service Behind NAT:**

- Systems providing services from behind a NAT are not directly reachable from the outside due to NAT acting as the Internet router and private, non-routable addresses.

2. **Port Forwarding Mechanism:**
   - Port forwarding, or port mapping, is employed to forward incoming traffic to a specific configured destination behind the NAT.
   - It requires static configuration of the NAT with the server's address and associated port number.
   - If the server's IP address changes, the NAT configuration must be updated.
3. **Limitations:**
   - Port forwarding is limited to one set of port numbers for each (IP address, transport protocol) combination.
   - With a single external IP address, it can forward only a single port of the same transport protocol to at most one internal machine.

These security features contribute to the overall protection and controlled access provided by NAT in network environments.

Building Firewall

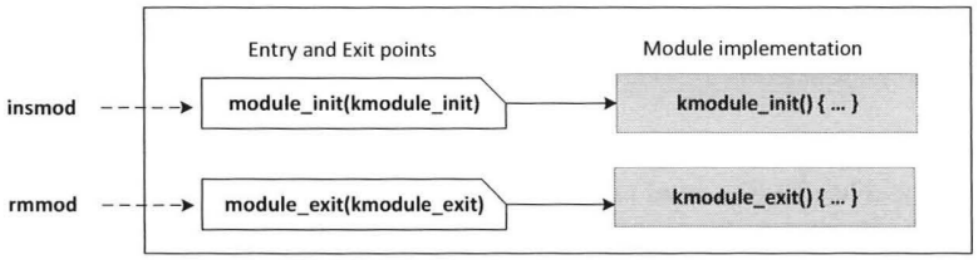
Netfilter provides hooks at the critical places on the packet traversal path inside the
Linux kernel. These hooks allow packets to go through additional program logics that are
installed by system administrators
. Packet filtering is an example for such additional program logics. **These program logics need to be installed inside the kernel, and the loadable kernel module technology makes it convenient to achieve that.** In this section, we first focus on loadable kernel modules, while net filter will be covered in the next section.

Loadable Kernel Module

- **Kernel modules are pieces of code that can be loaded and unloaded on-demand at runtime.**
- **A process needs the root privilege or the CAP_SYS_MODULE** capability to be able to insert or remove kernel modules
- **The Linux kernel is designed to be modular, meaning only a minimal part is loaded into memory initially.**
- **Additional features can be added dynamically through kernel modules, which are pieces of code loaded and unloaded on-demand at runtime.**
- Kernel modules can be used to add support for new hardware, among other functionalities.

Module Entry and Exit Points:

- Each kernel module has two entry points defined by the `module_init()` and `module_exit()` macros.
- The `module_init()` function is executed when the module is inserted into the kernel, while `module_exit()` is executed during removal.



Example Code:

- The example code provided (`kMod.c`) demonstrates a basic kernel module.
- It uses the `printk()` function to print messages to the kernel log buffer.
- The `MODULE_LICENSE("GPL")` macro indicates that the module is released under the GNU General Public License.

Compiling Kernel Modules:

Makefile:

- The text introduces a sample Makefile for compiling loadable kernel modules.
- The `obj-m += kMod.o` line specifies the module to be built.
- The Makefile includes targets for building (`all`) and cleaning (`clean`) the module.

Compilation Process:

- To build a kernel module, a pre-built kernel with configuration and header files is required.
- The `make` command is used with the specified Makefile to build the module.
- The `C` option points to the directory containing kernel source files.

Installing Kernel Modules:

Module Management Commands:

- Once compiled, kernel modules can be managed using commands such as `insmod`, `rmmod`, and `lsmod`.
- Example commands: `sudo insmod kMod.ko`, `sudo rmmod kMod`, `lsmod`.

Checking Kernel Log:

- The `dmesg` command is used to display kernel log messages, including those from kernel modules.
- Example: `dmesg | grep kMod` filters log messages related to the specific module.

The overall process involves compiling, inserting, and managing loadable kernel modules, providing a flexible way to extend kernel functionality in a running Linux system.

Netfilter

- In the Linux kernel, **each protocol stack defines a set of hooks along the path a packet traverses in that stack.**
- Kernel modules can register callback functions to these hooks, allowing developers to **intercept and process packets at different stages of their journey through the network stack.**

Packet Processing with Netfilter:

- When a packet arrives at a netfilter hook, the netfilter framework is invoked, and registered kernel modules are called in sequence.
- **Each registered module has the opportunity to analyze or modify the packet and then return a verdict on how the packet should be handled.**

Return Values:

1. **NF_ACCEPT:**
   - Allows the packet to flow through the network stack without further interference.
2. **NF_DROP:**
   - Discards the packet, preventing it from continuing its journey through the network stack.
3. **NF_QUEUE:**
   - Passes the packet to the user space using the nfqueue facility.
   - Allows asynchronous packet handling in user space.
4. **NF_STOLEN:**
   - **Informs the netfilter framework to forget about the packet.**
   - The responsibility for further processing is transferred from netfilter to the module.
   - The packet remains present and valid in the kernel's internal tables.
   - **Typically used for storing away fragmented packets for later analysis in a single context.**

5. **NF_REPEAT:**
    - Requests the netfilter framework to call the module again.
    - **Useful for scenarios where the module needs to reevaluate the packet or take additional actions.**

Use Cases:

- Netfilter hooks and kernel modules provide a flexible framework for implementing packet filtering, modification, and handling in the Linux kernel.
- Developers can leverage this framework to implement custom packet processing logic based on their specific requirements.

Netfilter Hooks

Netfilter Hooks for IPv4:

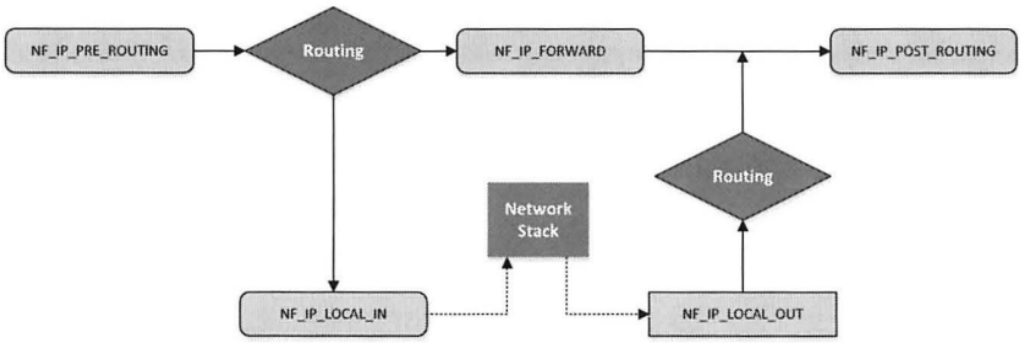


Figure 17.3: `netfilter` hooks in IPv4 stack

1. **NF_IP_PRE_ROUTING:**
    - Incoming packets, excluding those caused by promiscuous mode, pass through this hook.
    - Called before any routing decision is made.
2. **NF_IP_LOCAL_IN:**

- After passing through routing, determines whether the packet is destined for other machines or the host itself.
- If for the host, the packet goes through the NF_IP_LOCAL_IN hook before being processed by the network stack.

3. **NF_IP_FORWARD:**
   - Handles packets that are forwarded to other hosts.
   - Particularly useful for implementing firewall functionality in the context of packet forwarding.
4. **NF_IP_LOCAL_OUT:**
   - Deals with packets generated by the local host.
   - First hook for outgoing packets leaving the host.
5. **NF_IP_POST_ROUTING:**
   - When a packet, whether forwarded or generated, is leaving the host, it passes through this hook.
   - Source Network Address Translation (SNAT) is implemented at this hook.

Packet Movement in the Network Stack:

- **Incoming Packets:**
  - Enter through NF_IP_PRE_ROUTING.
  - Routed to other hosts (NF_IP_FORWARD) or for local processing (NF_IP_LOCAL_IN).
- **Outgoing Packets:**
  - Generated locally and go through NF_IP_LOCAL_OUT.
  - After processing, pass through NF_IP_POST_ROUTING for source NAT.

Use Cases and Relevance to Firewall:

- **NF_IP_FORWARD Hook:**

- Particularly significant for firewall implementations as it deals with packets being forwarded to other hosts.
- Allows for filtering or modification of packets in the context of forwarding decisions.

- **NF_IP_POST_ROUTING Hook:**
  - Important for SNAT (Source Network Address Translation) in the context of outgoing packets.
  - Enables manipulation of packet headers before leaving the host.

Implementing a Simple Packet Filter Firewall

The provided code implements a simple packet filter using the netfilter framework in the Linux kernel to block all packets going out to port 23, effectively preventing Telnet connections.

Here's a breakdown of the key components:

1. **telnetFilter Function:**

```
unsigned int telnetFilter(void *priv, struct sk_buff
*skb, const struct nf_hook_state *state)
{
    struct iphdr *iph;
    struct tcphdr *tcph;

    iph = ip_hdr(skb);
    tcph = (void *)iph + iph->ihl * 4;

    if (iph->protocol == IPPROTO_TCP && tcph->dest ==
htons(23)) {
        printk(KERN_INFO "Dropping telnet packet to %
d.%d.%d.%d\\n",
            ((unsigned char *)&iph->daddr)[0],
            ((unsigned char *)&iph->daddr)[1],
            ((unsigned char *)&iph->daddr)[2],
```

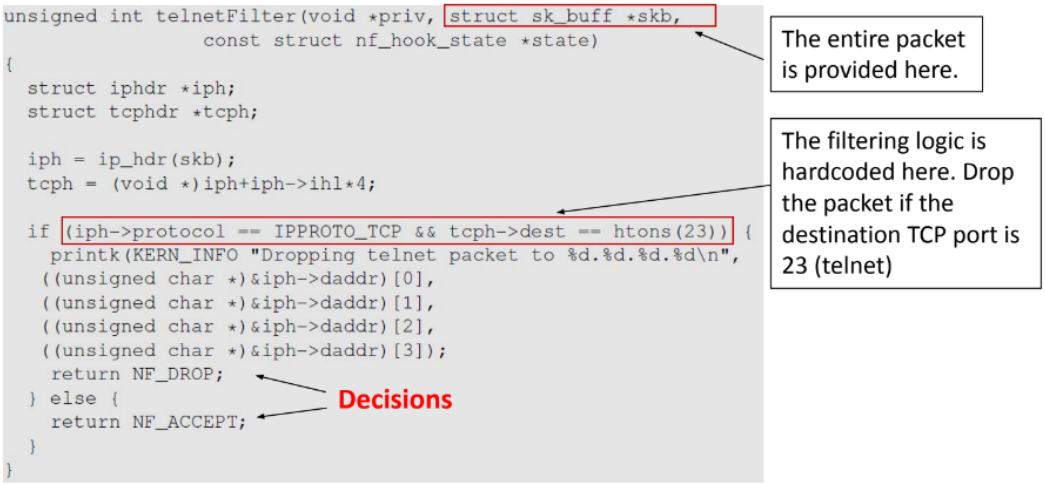
```
            ((unsigned char *)&iph->daddr)[3]);
        return NF_DROP;
    } else {
        return NF_ACCEPT;
    }
}
```

This function examines the destination port of TCP packets. If the destination port is 23 (Telnet), it drops the packet and prints a message to the kernel log. Otherwise, it accepts the packet.

2. **Initialization and Cleanup Functions:**

```
static struct nf_hook_ops telnetFilterHook;

int setUpFilter(void) {
    printk(KERN_INFO "Registering a Telnet filte
r.\\n");
    telnetFilterHook.hook = telnetFilter;
    telnetFilterHook.hooknum = NF_INET_POST_ROUTING;
    telnetFilterHook.pf = PF_INET;
    telnetFilterHook.priority = NF_IP_PRI_FIRST;

    nf_register_hook(&telnetFilterHook);
    return 0;
}

void removeFilter(void) {
    printk(KERN_INFO "Telnet filter is being remove
d.\\n");
    nf_unregister_hook(&telnetFilterHook);
}
```

These functions handle the registration and removal of the netfilter hook. `setUpFilter` is called when the module is loaded (`insmod`), and `removeFilter` is called when the module is removed (`rmmod`).



3. **Module Initialization and Exit:**

```
module_init(setUpFilter);
module_exit(removeFilter);
MODULE_LICENSE("GPL");
```

These lines specify the module's entry and exit points and its licensing information.

4. **Usage:**
   - `sudo insmod telnetFilter.ko`: Load the kernel module.
   - `telnet 10.0.2.5`: Attempt Telnet connection (which will be blocked).
   - `dmesg`: View kernel log messages.
   - `sudo rmmod telnetFilter`: Remove the kernel module.
   - `telnet 10.0.2.5`: Attempt Telnet connection (which should now succeed).

This simple packet filter demonstrates the use of netfilter hooks to intercept and filter outgoing Telnet packets. The module drops Telnet packets and logs messages to the kernel log.

❖ **Ingress Filtering**

```
int setUpFilter(void) {
        printk(KERN_INFO "Registering a Telnet filter.\n");
        telnetFilterHook.hook = telnetFilter;
//      telnetFilterHook.hooknum = NF_INET_POST_ROUTING;
        telnetFilterHook.hooknum = NF_INET_LOCAL_IN;
        telnetFilterHook.pf = PF_INET;
        telnetFilterHook.priority = NF_IP_PRI_FIRST;

        // Register the hook.
        nf_register_hook(&telnetFilterHook);
        return 0;
}
```

Change the hook!! Rest everything is same

iptables Firewall in Linux

iptables is a built-in Linux Firewall based on Netfilter.

The kernel part implementation of the firewall is called Xtables.

Table 17.1: `iptables` Tables and Chains

Table	Chain	Functionality
filter	INPUT FORWARD OUTPUT	Packet filtering
nat	PREROUTING INPUT OUTPUT POSTROUTING	Modifying source or destination network addresses
mangle	PREROUTING INPUT FORWARD OUTPUT POSTROUTING	Packet content modification

1. **Tables:**
   - **Tables in iptables represent the main purpose of the rules.**
   - Different tables serve different functions. The primary tables are:

- **Filter Table (`filter`):** Used for packet filtering rules.

  The filter table is used to make decisions about whether to let a packet continue to its intended destination or to deny its request (Packet Filtering).

- **NAT Table (`nat`):** Used for Network Address Translation (NAT) rules.

  The nat table is used to implement network address translation rules. As packets enter the network stack, rules in this table will determine whether and how to modify the packet's source or destination addresses in order to impact the way that the packet and any response traffic are routed. This is often used to route packets to networks when direct access is not possible.

- **Mangle Table (`mangle`):** Used for specialized packet alteration rules.

  The mangle table is used to alter the IP headers of the packet in various ways. **For instance, you can adjust the TTL (Time to Live) value of a packet, either lengthening or shortening the number of valid network hops the packet can sustain.** Other IP header fields can be altered in similar ways.

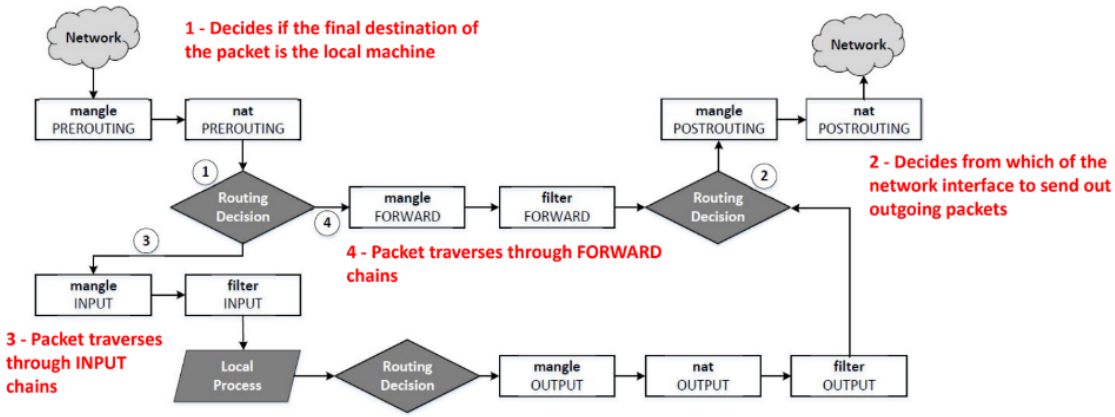
- **Raw Table (`raw`):** Used for configuring exemptions from connection tracking.
- **Security Table (`security`):** Used for Mandatory Access Control (MAC) networking rules.

2. **Chains:**

- **Chains are part of a table and define where the rules are enforced.**
- Each chain corresponds to a specific netfilter hook, indicating when the rules in that chain are applied in the packet processing path.
- Common chains include:
  - **INPUT Chain:** Enforces rules for packets destined for the local system (`NF_IP_LOCAL_IN` hook).

- **FORWARD Chain:** Enforces rules for packets being forwarded through the system (`NF_IP_FORWARD` hook).
- **OUTPUT Chain:** Enforces rules for locally generated packets before routing (`NF_IP_LOCAL_OUT` hook).

- Additionally, users can create custom chains for specific purposes and connect them to existing chains.



3. **Rules:**

- Rules are part of a chain and define the actions to be taken on matching packets.
- Each rule specifies conditions (matching criteria) and an associated action.
- **Possible actions include:**
  - **ACCEPT: Allow the packet to proceed.**
  - **DROP: Discard the packet.**
  - **REJECT: Discard the packet and send an error message to the sender.**
  - **SNAT/DNAT: Source/Destination Network Address Translation for altering packet headers.**
  - Other actions based on specific requirements.

Adding Firewall rule

Assume we would like to increase the time-to-live (TTL) field of all packets by 5. Since we need to make changes to packets, we choose to **add a rule to the mangle table.** This table **provides us with chains in all the five positions provided by the net filter hooks.** We choose the **PREROUTING chain, so the changes can be applied to all packets**, regardless of whether they are for the current host or for other

```
// -t mangle: Add this rule to the "mangle" table
// -A PREROUTING: Append this rule to the PREROUTING chain

$ sudo iptables -t mangle -A PREROUTING -j TTL --ttl-inc 5
```

iptables - General Format

General Format: iptables -t filter - A INPUT <rule specification> -j target

-t specifies the table name

-A specifies the chain name (INPUT in this case) and action (append) to be performed on this chain.

Commonly uses actions are :

-A : append rules to the end of the selected chain
-D : delete rules from the selected chain
-I : insert rule in the selected chain as the given rule number
-R : replace a rule in the selected chain
-L : List all rules in the selected chain or all the chains
-F : Delete all rules in the selected chain or all the chains (Flush all rules)

Iptables Extension

Match Extensions:

1. **conntrack module:**
    - **Functionality:** Used for specifying rules based on connections to build stateful firewalls.

- **Use Case:** Allows tracking and filtering based on the connection state (established, related, new, invalid).

2. **limit module:**
   - **Functionality:** Controls the rate of traffic.
   - **Use Case:** Useful for rate limiting, ensuring that certain types of traffic do not exceed a specified rate.
3. **statistic module:**
   - **Functionality:** Provides another way to control the traffic rate.
   - **Use Cases:**
     - **Random Mode:** Allows dropping packets with a specified probability.
     - **Nth Mode:** Permits dropping one packet for every N packets.
4. **owner module:**
   - **Functionality:** Helps specify filtering rules based on User ID/group ID.
   - **Use Cases:**
     - Works well in the OUTPUT chain.
     - Useful for applying rules to packets generated by specific users or groups.

Target Extensions:

- **Built-in Targets:** ACCEPT, DROP, RETURN.
- **Extended Target Modules:**
  1. **TTL (Time To Live):**
     - **Functionality:** Modifies IPV4 TTL.
     - **Use Case:** Valid only in the mangle table, useful for adjusting TTL values in packets.
  2. **MASQUERADE:**

- **Functionality:** Valid only in the nat table, in the POSTROUTING chain.
- **Use Case:** Replaces the source IP address of packets going out from a specific interface, typically used for NAT.

These extensions add functionalities to iptables, allowing users to define rules based on various criteria and perform actions on packets, such as rate limiting, connection tracking, and modifying packet headers.

Building Simple Firewall using iptables

Step 1: Set up default policies

```
$ sudo iptables -P INPUT ACCEPT
$ sudo iptables -P OUTPUT ACCEPT
$ sudo iptables -P FORWARD ACCEPT
```

- **Explanation:** This sets the default policies for the INPUT, OUTPUT, and FORWARD chains to ACCEPT. This is done to avoid blocking remote access while configuring the firewall.

Step 2: Flush existing configurations

```
$ sudo iptables -F
```

- **Explanation:** This command flushes (deletes) all existing firewall rules. It ensures a clean slate for setting up the new firewall rules.

Step 3: Open TCP ports 22 (ssh) and 80 (http)

```
$ sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT
$ sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
```

- **Explanation:** These rules allow incoming TCP traffic on ports 22 and 80, enabling SSH and HTTP access to the system.

Step 4: Allow all outgoing TCP traffic

```
$ sudo iptables -A OUTPUT -p tcp -j ACCEPT
```

- **Explanation:** This rule allows all outgoing TCP traffic, ensuring that the system can communicate with external servers.

Step 5: Allow loopback interface

```
$ sudo iptables -I INPUT 1 -i lo -j ACCEPT
```

- **Explanation:** This rule allows traffic on the loopback interface, enabling communication between processes on the same system.

Step 6: Allow DNS queries and replies

```
$ sudo iptables -A OUTPUT -p udp --dport 53 -j ACCEPT
$ sudo iptables -A INPUT -p udp --sport 53 -j ACCEPT
$ sudo iptables -A OUTPUT -p udp --sport 53 -j ACCEPT
$ sudo iptables -A INPUT -p udp --dport 53 -j ACCEPT
```

- **Explanation:** These rules allow DNS queries and replies to pass through, enabling DNS functionality.

Step 7: Set default filter policy to DROP

```
$ sudo iptables -P INPUT DROP
$ sudo iptables -P OUTPUT DROP
$ sudo iptables -P FORWARD DROP
```

- **Explanation:** This sets the default policies for the INPUT, OUTPUT, and FORWARD chains to DROP, ensuring that only packets matching the defined rules will be allowed..

Stateful Firewall

Why Connection Tracking?

Overview:

- **Examining packets individually might lead to missing the context, as many packets are part of connections.**
- In an organization, servers may have open ports (e.g., 22, 80, 443) providing services over the public network.
- **Firewall rules are set up to allow only specific ports to be exposed externally.**
- **Outgoing traffic needs to be allowed to receive replies from external connections.**
- The challenge is how to selectively allow outgoing traffic without opening all ports.

Solutions:

1. **Allow All Outgoing TCP Traffic:**
   - Simple but too broad, allowing traffic from all ports.
2. **Specify Rules for Each Port:**
   - Cumbersome, especially with numerous ports or protocols like FTP.
3. **Connection Tracking:**
   - Allow traffic to go out if it's part of an existing TCP connection.
   - Restrict which ports can accept connections and specify that only packets belonging to existing connections can go out.

Mechanism for Connection Tracking:

- **Kernel Module:** Connection tracking is implemented as a module in the Linux kernel.
- **Built on Netfilter:** Connection tracking is built on top of Netfilter, the base for many functionalities, including iptables.
- **Inspects Packets:** The connection tracking module inspects every packet, identifies whether it belongs to an existing connection, and

records that connection using port numbers.

Connection Tracking Framework in Linux:

- **Module:** `nf_conntrack` is the connection tracking framework built on top of Netfilter.
- **Connection States:**
  - **NEW: The connection is starting.**
  - **ESTABLISHED: The connection has been established.**
  - **RELATED: Special state for establishing relationships among different connections (e.g., FTP control and data traffic).**
  - **INVALID: State for packets not following expected behavior.**

Stateful Firewall using Connection Tracking:

- **Stateful Firewall:**
  - **Monitors incoming and outgoing packets over time.**
  - **Records attributes like IP address, port numbers, and sequence numbers.**
  - **A connection state signifies whether a packet is part of an existing flow.**
- **Applied to Protocols:**
  - Applies to both connection-oriented (TCP) and connectionless protocols (UDP and ICMP).
  - Suitable for complex protocols like HTTP, FTP, and IRC.

Connection Tracking - TCP, UDP, ICMP:

- **TCP:** Naturally has a connection.
- **UDP:** Uses heuristics based on port numbers to track connections.
- **ICMP:** Connection tracking considers ICMP request and reply as one "connection."

Dropping Connections:

- **Resource Limitations:** The kernel has a limit on the number of connections it can track.
- **Connection Expiration:** Connections have a lifespan; when expired, the kernel considers the connection finished.
- **Connection Removal:** If the connection pool is full, the kernel removes the oldest connection.

Check Connection Tracking - `conntrack` Command:

- **Tool:** `conntrack` is a command-line tool to view connections established by the connection tracking module.
- **Installation:** Can be installed on Linux machines using `sudo apt install conntrack`.

Example: Set Up a Stateful Firewall:

- Set up a firewall rule to allow outgoing TCP packets only if they belong to an established TCP connection.
- **Allow only SSH and HTTP connections, blocking outgoing TCP traffic if not part of an ongoing SSH or HTTP connection.**
- This approach provides a more refined and secure way of allowing outgoing traffic.

Application Firewall(Proxy)

Overview:

- An application firewall is designed to control input, output, and access to and from a specific application.
- **It inspects network traffic up to the application layer, providing a more granular level of control.**
- **A common implementation of an application firewall is a proxy, specifically an application proxy.**

- A web proxy is a specific type of application proxy used to control what browsers can access.

Application/Proxy Firewall and Web Proxy:

- **Application Firewall:**
  - Controls traffic at the application layer.
  - Inspects and filters communication between applications.
- **Proxy (Application Proxy):**
  - **A proxy acts as an intermediary between clients and servers.**
  - In the context of an application firewall, a proxy can control and filter traffic at the application layer.
- **Web Proxy:**
  - A specialized proxy for controlling web traffic.
  - Common implementations include Squid.
- **Setting Up a Web Proxy:**
  - **Configure each host computer to redirect all web traffic to the proxy.**
  - **Use browser network settings or iptables for redirection.**
  - **Place web proxies on a network bridge connecting internal and external networks.**

Proxy for Evasion:

- **Evading Egress Filtering:**
  - **If a firewall filters packets based on destination address, using a web proxy can help evade the filtering.**
  - Browsing the internet through a web proxy modifies the destination address to the proxy server, bypassing packet filtering rules.

Anonymizing Proxy:

- **Hide Origin of Network Request:**

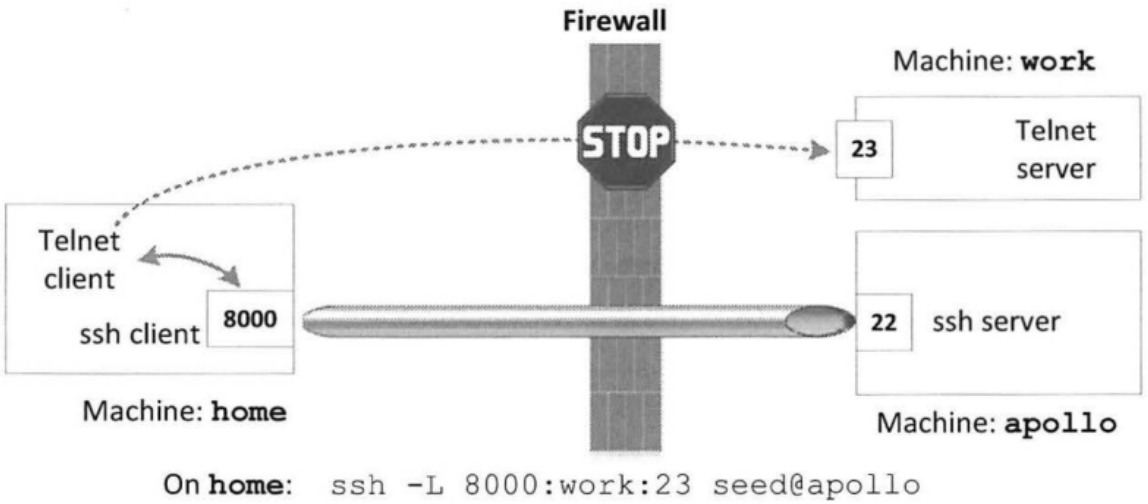
- Anonymizing proxies are used to hide the origin of a network request from servers.
- Servers only see the traffic after it passes through proxies.
- **The source IP seen by servers is that of the proxy, keeping the actual origin hidden.**

Tunnelling

Evading Firewalls with SSH Tunneling

Tunneling Technique:

- In certain situations, firewalls can be overly restrictive, blocking access to specific websites or internet services.
- Egress filtering in many companies and schools can block users from reaching certain sites.
- Tunneling is a common approach to evade firewalls, and it involves hiding the real purposes of network traffic.



VPN and SSH Tunneling:

- Virtual Private Network (VPN) builds tunnels at the IP layer, commonly used to evade both ingress and egress filtering.

- In the absence of VPN, SSH tunneling is a viable option.

SSH Tunneling to Evade Firewalls:

1. **Ingress Filtering Example:**
    - Scenario: Telnet from home to work is blocked by the company's firewall.
    - Solution: Establish an SSH tunnel between home and an internal machine (e.g., apollo) that allows SSH traffic.
    - The SSH tunnel forwards TCP data from the telnet client to apollo, which then sends it to the work machine.
    - The firewall only sees encrypted SSH traffic.

```
# Establish the tunnel from home to apollo
$ ssh -L 8000:work:23 seed@apollo
# Telnet to work from home through the tunnel
$ telnet localhost 8000
```

2. **Egress Filtering Example:**
    - Scenario: Company blocks access to Facebook from work.
    - Solution: Establish an SSH tunnel from work to an external machine (e.g., home) that has access to Facebook.
    - The SSH tunnel forwards HTTP requests from work to Facebook via home.
    - The firewall sees only the encrypted SSH traffic.

```
# Establish the tunnel from work to home
$ ssh -L 8000:www.facebook.com:80 seed@home
# Browse Facebook by typing localhost:8000 in the browser's
URL address bar
```

Dynamic Port Forwarding with SSH for Evading Firewalls

In scenarios where static port forwarding is impractical for multiple sites, dynamic port forwarding using SSH provides a more flexible solution. This technique

establishes a SOCKS proxy through an SSH tunnel, allowing the redirection of traffic to various destinations.

1. Initiate Dynamic Port Forwarding:

```
$ ssh -D 9000 -C seed@home
```

- `D 9000`: Specifies the local port for dynamic forwarding.
- `C`: Enables compression to optimize data transfer.

2. Configure Browser for SOCKS Proxy:

- Open browser settings and configure SOCKS proxy:
  - Proxy Type: SOCKS v5
  - Host: localhost
  - Port: 9000

3. Browsing through the Dynamic SSH Tunnel:

- Type any blocked website URL in the browser.
- Traffic is routed through the SSH tunnel to the specified destination.
- Firewall only sees encrypted SSH traffic, unaware of the actual web traffic.

4. Advantages of Dynamic Port Forwarding:

- **Flexibility**: No need to create individual tunnels for each site.
- **Generic Proxy**: Works independently of application logic, supporting various protocols.
- **Encrypted Traffic**: Provides security through SSH encryption.

5. Limitation:

- **Client Support**: Requires client software with native SOCKS support.
- **Not for Telnet**: Inapplicable to protocols lacking SOCKS support, like Telnet.

Reverse SSH Tunneling:

- Scenario: Internal website inaccessible from outside due to a firewall, and incoming SSH traffic is blocked.
- Solution: Use reverse SSH tunneling to create a tunnel from an internal machine (e.g., apollo) to an external machine (e.g., home).
- Users from outside can access the internal website by sending HTTP requests to the port 8000 on the external machine (home).

```
# Run this command on machine apollo
$ ssh -R 8000:web-server:80 seed@home
```

- Users access the internal website by sending HTTP requests to localhost:8000 on the external machine. The SSH tunnel forwards requests to the internal web server.

Intrusion Detection System

An intrusion is a deliberate unauthorized attempt, successful or not, to break into, access, manipulate, or misuse some valuable property and where the misuse may result into or
render the property unreliable or unusable.

> A security service that monitors and analyzes system events for the purpose of finding, and providing real-time or near real-time warning of, attempts to access system resources in an unauthorized manner.

System Intrusions:

1. Active access to unused logins: Unauthorized entry into dormant user accounts.
2. Login during non-working hours: Suspicious logins occurring outside regular business hours.
3. New user account created automatically: Unauthorized automated generation of user accounts.

4. Modification in system software or configuration files: Unauthorized alterations to system software or configuration files.
5. System logs are deleted: Deliberate removal of logs to cover up unauthorized activities..

Network Intrusions:

1. Identifications of repeated attempts to log in from remote locations: Multiple unauthorized login attempts from different remote locations.
2. Sudden increase in bandwidth consumption: Unexplained spikes in network traffic, indicative of a security incident.
3. Repeated investigations of existing services: Continuous probing or scanning of network services for vulnerabilities.
4. Arbitrary log data in log files: Presence of irregular or unexpected entries in log files.
5. Running a packet sniffer: Unauthorized monitoring of network traffic to capture sensitive information.

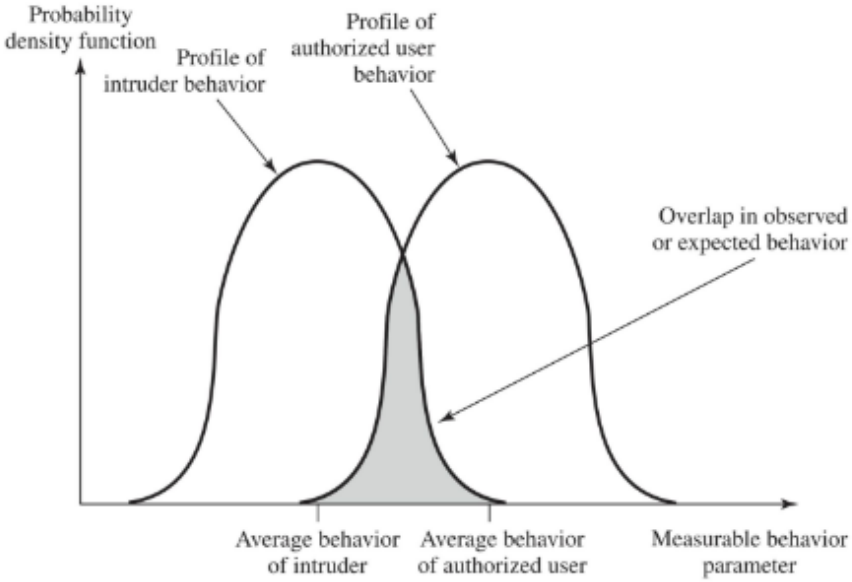
File Intrusions:

1. Identifications of unknown files and programs on your system: Discovery of unrecognized files or programs.
2. File permission modifications: Unauthorized changes to file permissions, potentially allowing unauthorized access.
3. Unexplained modifications in file size: Changes in file sizes without a clear reason.
4. Identifications of strange file presence in system directories: Presence of unfamiliar files in critical system directories.
5. Missing files: Absence of files that were previously present, indicating potential deletion or manipulation.

Drawbacks of Firewall:

1. **Only as effective as the rules they are configured to enforce:**
    - Firewalls rely on predefined rules, and their effectiveness is limited to the accuracy and comprehensiveness of these rules.
2. **Can’t stop attacks if the traffic does not pass through them:**
    - Firewalls are unable to prevent attacks on traffic that doesn't traverse through them, such as attacks within the internal network.
3. **Can’t prevent revealing sensitive information through social engineering:**
    - Firewalls are not designed to counteract social engineering tactics that may lead to the unauthorized disclosure of sensitive information.
4. **Can’t protect against flaws in authorized applications:**
    - Firewalls may not provide protection against vulnerabilities or flaws present in authorized applications, leaving potential security gaps.
5. **Can’t secure against tunneling attempts:**
    - Firewalls may struggle to detect and prevent covert methods like tunneling, where malicious activities are concealed within seemingly legitimate traffic.

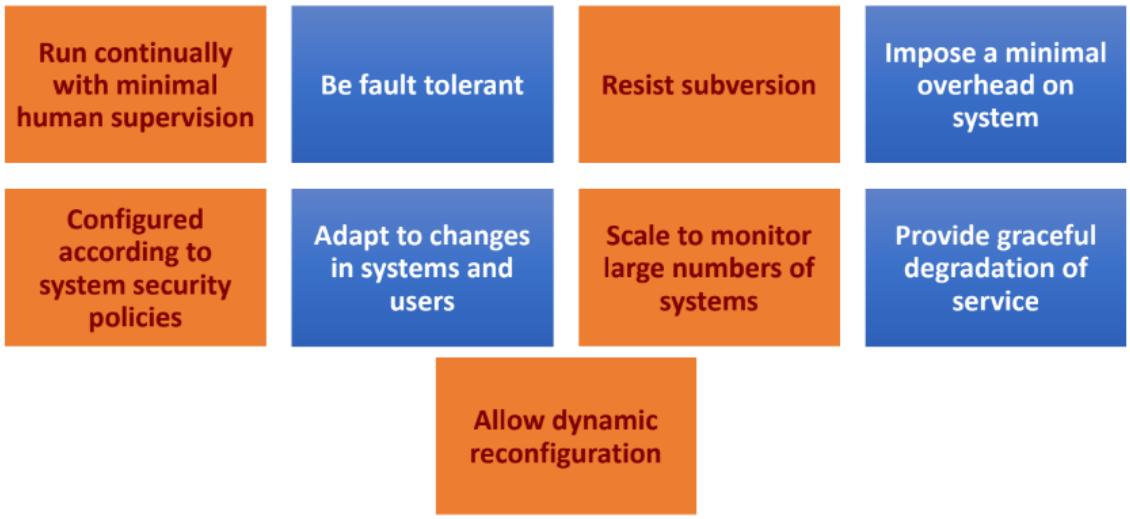
IDS Principles:



Profiles of Behavior of Intruders and Authorized Users

1. **Overlap in behaviors causes problems:**
    - Intruder behavior often overlaps with legitimate user behavior, leading to challenges in accurately distinguishing between the two.
2. **False positives and false negatives:**
    - Overlapping behaviors result in false positives (due to a loose interpretation of intruder behavior) and false negatives (due to a tight interpretation of intruder behavior).
3. **Practical use of IDS:**
    - An IDS should detect a significant percentage of intrusions while maintaining an acceptable level of false alarms to be practically useful.

IDS Requirements:



IDS Defense Mechanisms against Intrusions:

- **Report attacks against monitored systems/networks.**
- **Rules specify which events should generate alerts.**
  - Alerts are generated from Events of Interest (EOI).
- **Alerts can be: a line on screen, beep, email, SMS, or phone call.**
- **Frequently update the antivirus signature database.**
- **Save a trace file of raw packets for future analysis.**
- **Save the attack information (Intruder IP, victim IP, timestamp).**
- **Force a TCP FIN or RST packet to force a connection termination.**

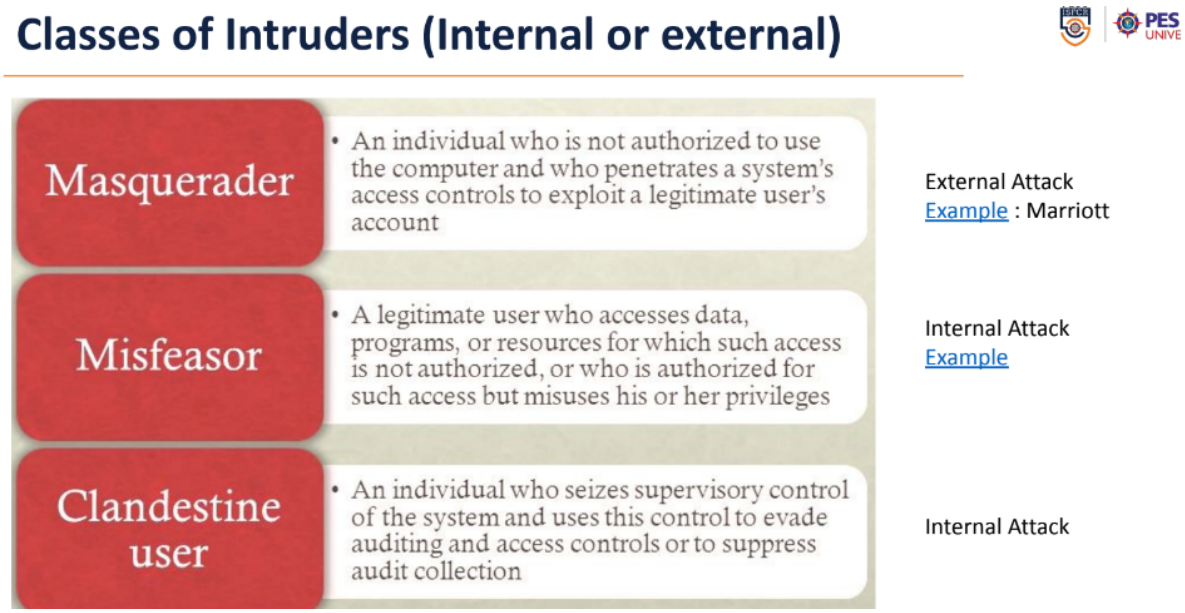
What IDS is not:

- **Not a replacement for:**
  - Firewalls
  - Strong Policies
  - System Hardening
  - Timely Patching
  - Other Defense in Depth Techniques
- **Not a low-maintenance tool**

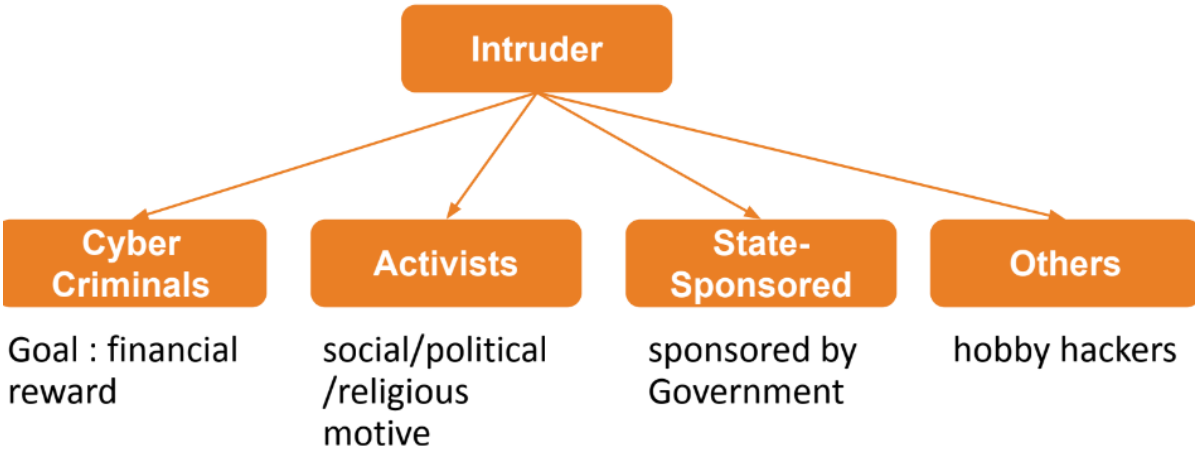
- **Not an inexpensive tool**
- **Not a silver bullet**

Classes of Intruders

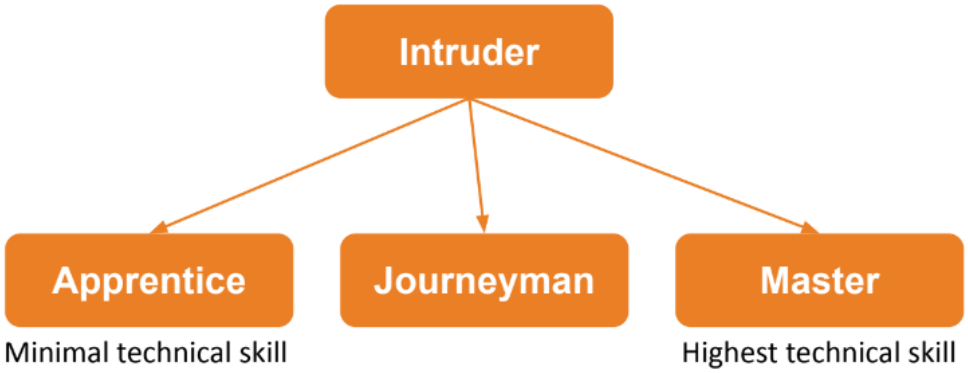
Based on Origin (Internal or External) - Masquerader, Misfeasor, Clandestine User



Based on Motive - Cybercriminal, Activists, State-sponsored, Others



Based on Skill Level - Apprentice“script-kiddies”, Journeyman, Master



Intruder Behaviour

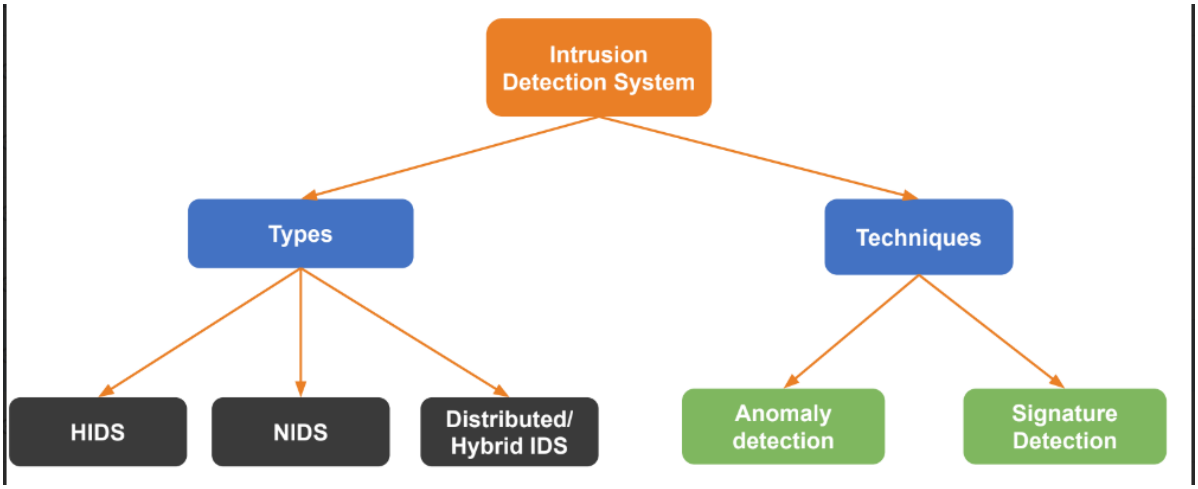
Steps in Common Attack Methodology:

1. **Target Acquisition and Information Gathering:**
   - Identifies the target using publicly available information.
   - Uses network exploration tools to map target resources.
   - Examples include exploring the target website, using DNS lookup tools, mapping network services with tools like Nmap, and identifying potentially vulnerable services or third-party apps used by the target.
2. **Initial Access:**
   - Exploits remote network vulnerabilities or guesses weak authentication credentials.
   - Uses social engineering techniques, such as spear-phishing emails, to install malware on systems.
3. **Privilege Escalation:**
   - Exploits vulnerable applications to gain elevated privileges.

- Installs sniffers to capture admin passwords and uses them to access privileged information.

4. **Information Gathering or System Exploit:**
   - Scans files for desired information.
   - Modifies information/resources on the system.
   - Targets other servers on the network and performs data exfiltration.
5. **Maintaining Access:**
   - Enables continued access after the initial attack by installing backdoors or other malicious software.
   - Modifies or disables antivirus/IDS programs running on the system.
6. **Covering Tracks:**
   - Removes evidence of attack activity.
   - Edits or disables audit logs.
   - Uses rootkits (a type of malware) to hide installed files/code.

IDS Classification



Anomaly-based (Behavior-based) IDS:

1. **Requires understanding of what "normal" is:**

- Establishes a performance baseline of normal traffic evaluation.
- Unexpected conditions are identified as suspicious.

2. **Learning Systems:**
    - Operate by continuously creating "norms" of activities.
    - Anomaly detection compares observed activity against expected normal usage profiles "learned" to detect potential intrusions.

How to Come Up with What's Normal?

1. **Threshold Detection:**
    - Checks excessive event occurrences over time.
    - Determines both thresholds and time intervals.
    - May generate false positives or negatives due to user variability.
2. **Profile-Based Detection:**
    - Profiles developed for users, groups, applications, or system resource usage.
    - Analysis of audit records forms the foundation for this approach.

Example of Metrics for Profile-Based Intrusion Detection:

1. **Counters:**
    - Number of logins by a user, executions of a command, password failures.
2. **Gauge:**
    - Measures the current value of some entity, e.g., logical connections or queued outgoing messages.
3. **Interval Time:**
    - Length of time between two related events, such as successive logins.
4. **Resource Utilization:**
    - Quantity of resources consumed during a specified period, e.g., pages printed or program execution time.

Statistical Tests Based on Metrics:

- **Mean and Standard Deviation:**
  - Reflects average behavior and its variability.
- **Multivariate:**
  - Considers correlations between different metrics.
- **Markov Process:**
  - Establishes transition probabilities among various states.
- **Time Series:**
  - Focuses on time intervals and abnormal timing sequences.
- **Operational:**
  - Based on fixed limits defining abnormal behavior.

Anomaly-Based IDS Classification Approaches:

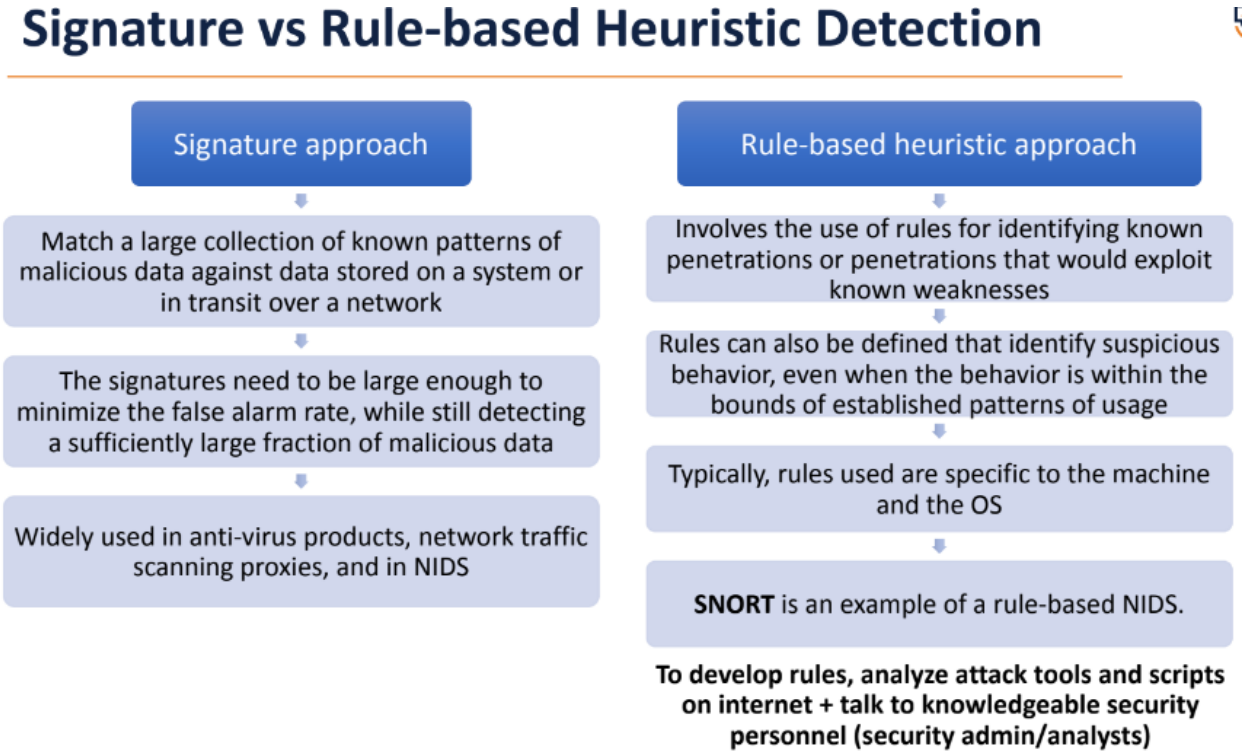
1. **Statistical Approaches:**
   - Simple, low computation cost, no assumptions, but difficulty in selecting suitable metrics.
2. **Knowledge-Based Approaches:**
   - Robust and flexible, but difficult and time-consuming to develop high-quality knowledge.
3. **Machine Learning Approaches:**
   - Require significant time and computational resources, make assumptions, but efficient after model generation.
   - Examples include Bayesian networks, Markov models, Neural networks, Fuzzy logic, Genetic algorithms, Clustering, and outlier detection.

Signature/Heuristic-Based Detection:

1. **Detects intrusion by observing events in the system.**

2. **Classifies observed data as normal or anomalous using two approaches:**
    - **I. Signature Approach:**
        - Applies a set of signature patterns to observed data.
    - **II. Rule-Based Heuristic Approach:**
        - Uses a set of rules to characterize data and make decisions.

Signature vs Rule-Based Heuristic Detection:



Example of Rules:

- Users should not be logged in more than one session.
- Users do not make copies of system password files.
- Users should not read other users' directories.
- Users must not write to other users' files.
- Users who log in after hours often access the same files they used earlier.
- Users do not generally open disk devices but rely on high-level OS utilities.

Host-based Intrusion Detection System (HIDS):

- **Specialized software to monitor system activity:**
  - Detects suspicious behavior.
  - **Primary purpose is to detect intrusions, log suspicious events, and send alerts.**
  - Can detect both external and internal intrusions.
- **Examines User and System Activity:**
  - On a single host (computer).
- **Two Approaches:**
  - **Anomaly HIDS:** Detects deviations from established baselines of normal behavior.
  - **Signature HIDS:** Matches observed patterns against a database of known attack signatures.
- **Distributed HIDS:**
  - **Defends a distributed collection of hosts supported by a LAN or internetwork.**
  - Requires coordination and cooperation among IDSs across the network.

HIDS - Data Sources and Sensors:

- **Common Data Sources (Collected by Sensors) Include:**
  - **System call traces.**
  - **Audit (log file) records.**
  - **File integrity checksums.**
  - **Registry access (Windows specific).**

Audit Records:

- **Native Audit Records:**
  - **Virtually all multiuser operating systems include accounting software that collects information on user activity.**

- Advantage: No additional collection software needed.
- Disadvantage: Native audit records may lack needed information or present it in an inconvenient form.

- **Detection-Specific Audit Records:**
  - **A collection facility generates audit records containing only information required by the intrusion detection system.**
  - Advantage: Vendor-independent and portable to various systems.
  - Disadvantage: Extra overhead with two accounting packages running on a machine.

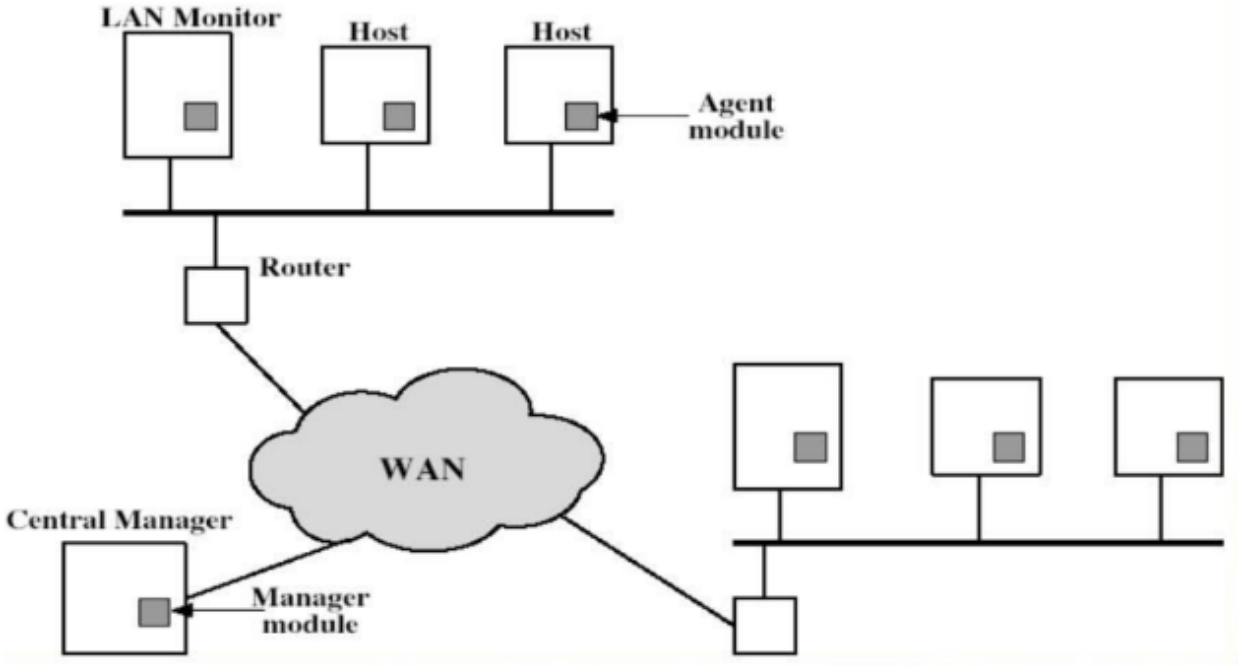
Example: Detection-Specific Audit Record:

- **Components:**
  - **Subject:** Initiators of actions (users or processes).
  - **Action:** Operation performed by the subject on or with an object (e.g., login, read, execute).
  - **Object:** Receptors of actions (files, programs, messages, etc.).
  - **Exception-Condition:** Denotes raised exception conditions.
  - **Resource-Usage:** Quantitative elements representing resource consumption.
  - **Time-Stamp:** Unique time-and-date stamp identifying when the action took place.

Example: Audit Record Creation:

- **Multiple Elementary Actions:**
  - Most user operations involve multiple elementary actions.
  - **Example: A file copy command creates multiple audit records for access validation, reading, and writing.**
  - Enables an audit of all behavior affecting an object.

Distributed Host-Based IDS:

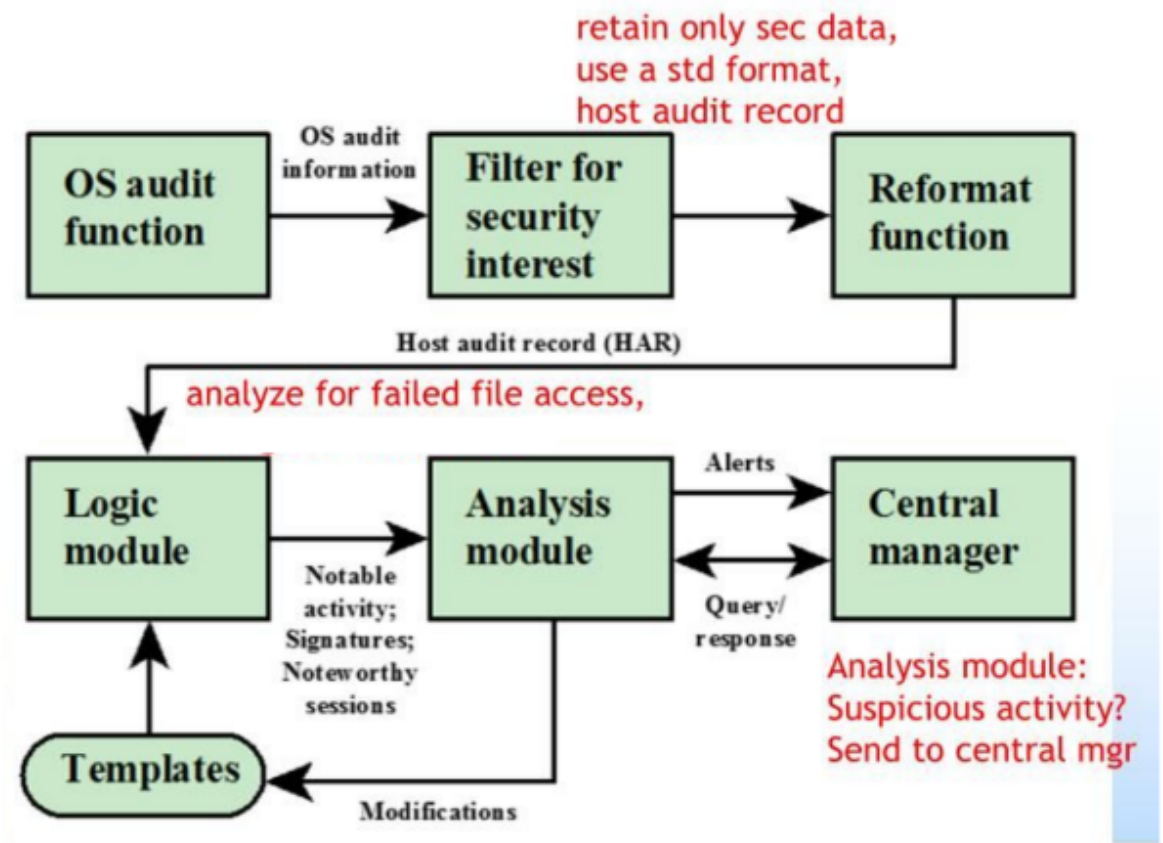


Major Issues in the Design:

1. **Handling Different Audit Record Formats:**
    - In a heterogeneous environment, a distributed IDS may need to deal with various audit record formats.
2. **Node Roles:**
    - One or more nodes serve as collection and analysis points for data from systems on the network.
3. **Data Transmission:**
    - Raw audit data or summary data must be transmitted across the network, requiring assurance of integrity and confidentiality.
4. **Architectural Options:**
    - Centralized Architecture: Eases correlating incoming reports but creates potential bottlenecks and single points of failure.
    - Decentralized Architecture: Involves more than one analysis center, requiring coordination and information exchange.

Architecture for Distributed Intrusion Detection:

1. **Overall Architecture:**
    - Three main components:
        - **Host Agent Module:** Collects data on security-related events on the host and transmits it to the central manager.
        - **LAN Monitor Agent Module:** Analyzes LAN traffic, audits host-host connections, services, and traffic volume, searching for events like sudden changes in network load or suspicious activity.
        - **Central Manager Module:** Receives reports from LAN monitor and host agents, processes, and correlates these reports to detect intrusion.
2. **Independence:**
    - The scheme is designed to be independent of any operating system or system auditing implementation.



Network-based Intrusion Detection System (NIDS):

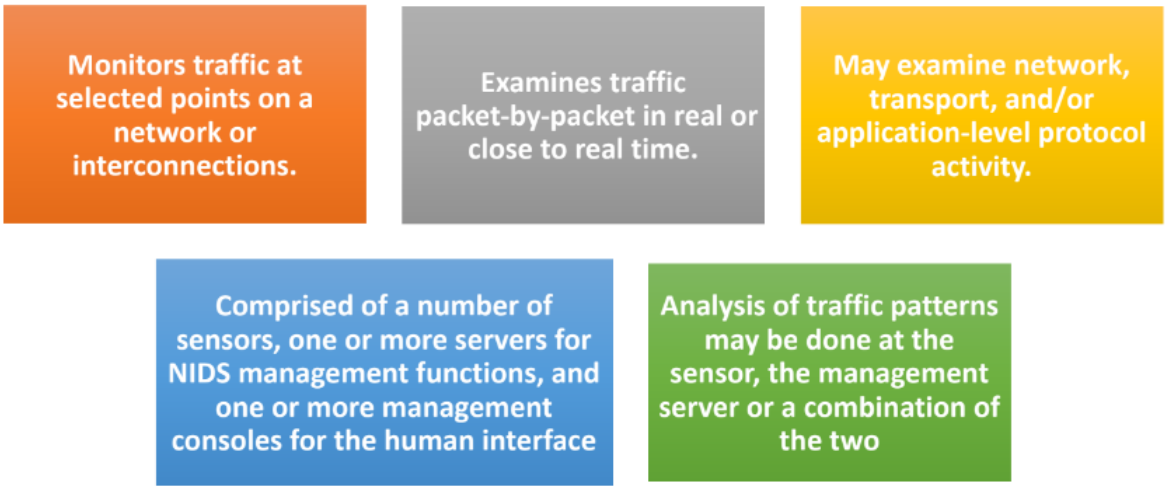
Monitoring Scope:

- **Entire Network:**
  - NIDSs monitor the entire network as their scope.
  - Deployed as passive sensors at network aggregation points.
  - Responsible for detecting anomalous, inappropriate, or unauthorized data on the network.

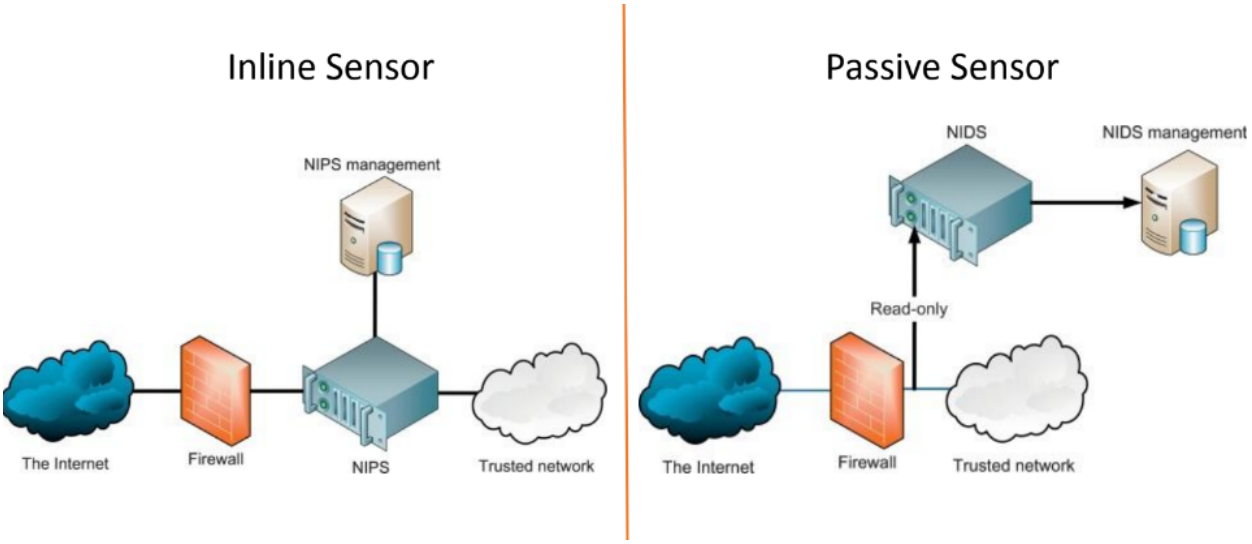
Characteristics:

- **Differences from Firewalls:**
  - Captures traffic like a sniffer.
  - Detects Events of Interest (EOI) on the network.
  - Uses analysis of signatures, anomalies, and transport/application protocol activity.

Components:

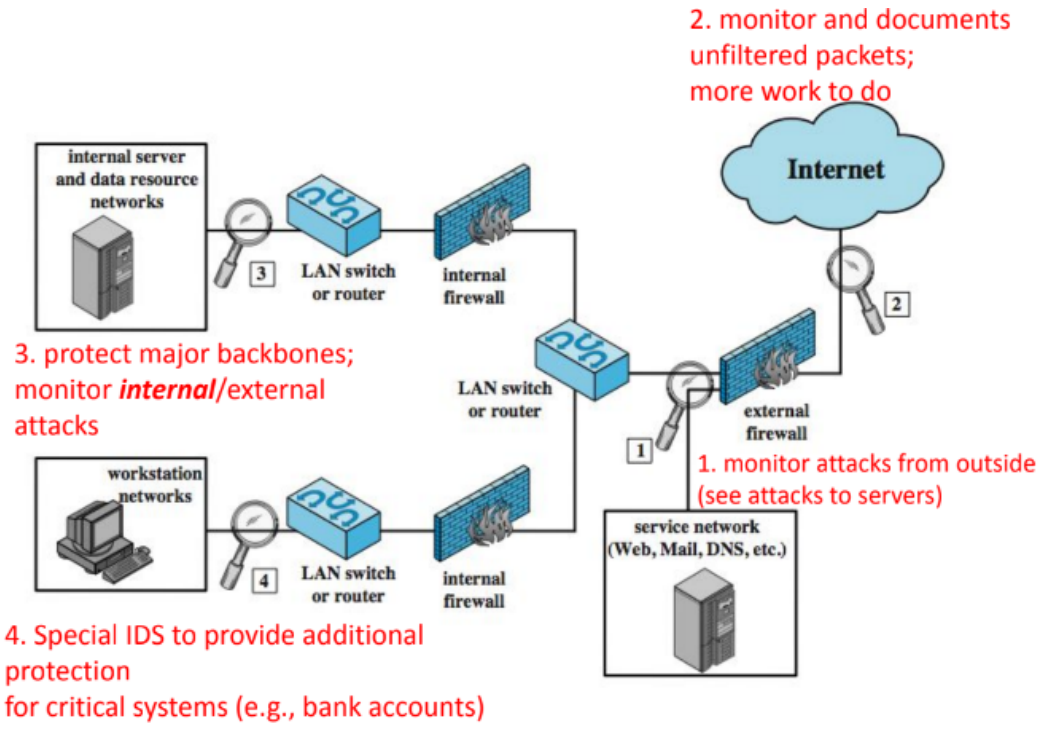


Types of Network Sensors:



1. **Inline Sensors:**
    - Monitored traffic passes through the sensor.
    - Can be combined with other network devices like firewalls or LAN switches.
    - Performs both intrusion detection and prevention functions.
    - May experience packet delay.
2. **Passive Sensors:**
    - Actual traffic does not pass through the sensor; a copy of the network traffic is provided.
    - No extra handling step, hence more efficient with no packet delay.
    - Uses a physical tap to provide a copy of network traffic.
    - Connects to the network with an IP address for communication with a NIDS management server.

NIDS Sensor Deployment:



1. **Location 1: Just Behind External Firewall:**
    - Highlights problems with network firewall policy/performance.
    - Detects attacks originating from outside, targeting web/ftp servers that penetrate external firewalls.
2. **Location 2: Between External Firewall and Internet or WAN:**
    - Monitors all traffic, unfiltered.
    - Documents the number and types of attacks originating from the internet.
    - Has a higher processing burden.
3. **Location 3: To Protect Major Backbone Networks:**
    - Detects unauthorized activity by authorized users.
    - Monitors both internal and external attacks.
4. **Location 4: To Protect LANs Supporting User Workstations:**
    - Detects attacks targeting critical systems and resources.
    - Monitors traffic only to a subset of devices, reducing processing burden.

NIDS - Signature Detection Techniques:

- **Protocol Analysis:**
  - NIDS analyze various application protocols (DHCP, DNS, FTP, HTTP, SMTP, SNMP, Telnet).
  - Looks for attack patterns targeting these protocols.
- **TCP/UDP Traffic Analysis:**
  - Analyzes TCP and UDP traffic.
  - Example: Detects unusual packet fragmentation, scans for vulnerable ports, SYN Flood attacks, etc.
- **IPv4/IPv6, ICMP, IGMP Analysis:**
  - Analyzes network layer protocols (IPv4, IPv6, ICMP, IGMP).
  - Example: Detects spoofed IP addresses, illegal IP header values.
- **Identification of Unauthorized Services:**
  - Example: Detects a host running unauthorized application services.
- **Monitoring for Forbidden Activities:**
  - Example: Detects the use of inappropriate websites or forbidden application protocols.

NIDS - Anomaly Detection Techniques:

Denial-of-Service Attacks:

- **Detection Criteria:**
  - Requires significantly increased packet traffic or connection attempts to overwhelm the target system.

Scanning Attacks:

- **Flow Pattern Analysis:**
  - Attacker probes the target network with various packets to learn system characteristics and vulnerabilities.

- Detected by analyzing flow patterns at the application, transport (port scanning), and network layer (ICMP scanning).

Worms:

- **Propagation and Bandwidth Usage Analysis:**
  - Worms replicate and send copies across network connections.
  - Some propagate quickly and use large amounts of bandwidth.
  - May cause host-to-host communication, use of unused ports, and perform scanning.

Logging of Alerts:

- **Purpose:**
  - When a potential violation is detected, the sensor sends an alert and logs information for:
    - Refining intrusion detection parameters and algorithms in the NIDS analysis module.
    - Assisting security administrators in improving protection and designing prevention techniques.
- **Typical Information Logged:**
  - Timestamp, connection/session ID, event/alert type, rating (priority, severity, impact, confidence), network/transport/application layer protocols, source/destination IP addresses, ports, bytes transmitted, decoded payload data, state-related information (authenticated username).

NIDS Challenges:

- **Deployment Challenges:**
  - Including deployment and access limitations.
- **Analyzing Encrypted Traffic:**
  - Difficulty in analyzing traffic encrypted with secure protocols.
- **Quantity vs. Quality of Signatures:**

    - Balancing the number and effectiveness of signature patterns.
- **Performance Limitations:**
    - Speed of processing and storage size.
- **Cost Challenges:**
    - NIDS management can be costly for proper deployment and maintenance.

Hybrid Intrusion Detection System:

- **Overview:**
    - Combines both Network-based Intrusion Detection Systems (NIDS) and Host-based Intrusion Detection Systems (HIDS).
    - Each patrolling its own area of the network for unwanted and illegal network traffic.
- **Complementary Nature:**
    - **Strengths and Weaknesses:**
        - NIDS and HIDS bring their own strengths and weaknesses.
        - Complement and augment the overall security of the network.
- **Challenges:**
    - **New Technology:**
        - Hybrid systems are relatively new and require significant support for widespread adoption.
    - **Interoperability:**
        - Major drawback is the challenge of getting different technologies to interoperate successfully and efficiently.
    - **Coexistence:**
        - Getting multiple IDS approaches to coexist in a single system can be a challenging task.

Intrusion Detection Exchange Format (IDMEF):

- **Purpose:**
  - Facilitates the development of a distributed IDS that can function across a wide range of platforms and environments.
- **IETF Intrusion Detection Working Group:**
  - Defined standards to support interoperability in intrusion detection and response systems.
- **Key Elements of the Model:**
  1. **Data Source:**
     - Raw data from an IDS.
  2. **Sensor:**
     - Collects and forwards events.
  3. **Analyzer:**
     - Processes data.
  4. **Administrator:**
     - Defines security policy.
  5. **Manager:**
     - A process for the operator to manage the IDS system.
  6. **Operator:**
     - The user of the Manager.
- **Data Formats and Exchange Procedures:**
  - Defined by the Intrusion Detection Working Group for sharing information among intrusion detection and response systems.

Challenges and Conclusion:

- **Challenges:**
  - Newness of hybrid IDS technology.

- Interoperability difficulties.
- Coexistence of multiple IDS approaches.

Incident Response Team (IRT):

- **Purpose:**
  - A primary and centralized group of dedicated individuals responsible for the initial response to incidents.
- **Responsibilities:**
  1. **Stay Informed:**
     - Keep up-to-date with the latest threats and incidents.
  2. **Point of Contact:**
     - Serve as the main point of contact for incident reporting.
  3. **Notification:**
     - Notify relevant parties when an incident occurs.
  4. **Assessment:**
     - Assess the damage and impact of each incident.
  5. **Vulnerability Mitigation:**
     - Determine ways to prevent the exploitation of the same vulnerability.
  6. **Recovery:**
     - Lead efforts in recovering from the incident.

IDS Logs as Evidence:

- **Legal Protection:**
  - IDS logs can serve as evidence to protect the organization in legal proceedings.
- **Consent to Monitoring:**
  - It's essential to have a published policy explicitly stating that the use of the network implies consent to monitoring.

Implementing an Intrusion Detection System (IDS):

1. **Effective Operating Systems:**
   - Utilize operating systems with robust logging and auditing features.
2. **Auditing Features:**
   - Ensure applications, such as web servers and email servers, have logging/auditing features.
3. **Firewalls:**
   - Deploy firewalls with network intrusion detection capabilities.
4. **Network Management Platform:**
   - Implement network management services with tools for setting up alerts on suspicious activity.

Challenges in IDS Implementation:

1. **Ensuring Effective Deployment:**
   - Address performance and scalability concerns.
2. **Managing High Volume of Alerts:**
   - Develop strategies to handle a large number of alerts efficiently.
3. **Understanding and Investigating Alerts:**
   - Provide training and resources for understanding and investigating alerts.
4. **Knowing How to Respond:**
   - Establish clear response protocols for different types of threats.
5. **High Rate of False Alarms:**
   - Work on reducing false positives to enhance the reliability of alerts.

Intrusion Prevention System

Intrusion Prevention System (IPS):

- **Introduction:**
  - **A promising model in the field of intrusion detection and prevention, aimed at preventing attacks.**
- **Continuous Monitoring:**
  - **Monitors the network 24/7, actively searching for signs of intruders or potential attacks.**
- **Preventing Attacks:**
  - Focuses on stopping attacks on systems and networks before they can be successful.
- **Proactive Blocking:**
  - **Capable of attempting to block or prevent detected malicious activity without waiting for user intervention.**
- **Recent Technology:**
  - IPS is a more recent technology compared to traditional Intrusion Detection Systems (IDS).
- **Maturation:**
  - Rapidly maturing as a security technology, adapting to emerging threats and evolving to enhance prevention capabilities.

Key Characteristics:

1. **Real-time Prevention:**
   - Acts in real-time to prevent malicious activities as they are detected.
2. **Automated Response:**
   - Implements automated responses to thwart potential threats.
3. **Signature and Behavioral Analysis:**
   - Utilizes both signature-based and behavioral analysis techniques for threat detection.
4. **Network and Host-based Protection:**
   - Provides protection at both the network and host levels.

5. **Policy Enforcement:**
    - Enforces security policies to ensure compliance and protect against policy violations.
6. **Integration with Other Security Measures:**
    - Often integrated with firewalls, antivirus software, and other security measures for comprehensive defense.
7. **Continuous Evolution:**
    - Evolves continuously to address new and emerging threats through regular updates and improvements.

Advantages:

1. **Preventative Approach:**
    - Focuses on preventing intrusions rather than just detecting them.
2. **Real-time Action:**
    - Takes immediate action to block or neutralize threats as they occur.
3. **Reduced Manual Intervention:**
    - Automates responses, reducing the need for manual intervention.
4. **Enhanced Security Posture:**
    - Contributes to an overall enhanced security posture by actively thwarting attacks.
5. **Adaptability:**
    - Adapts to evolving threats, making it suitable for dynamic cybersecurity landscapes.

Challenges:

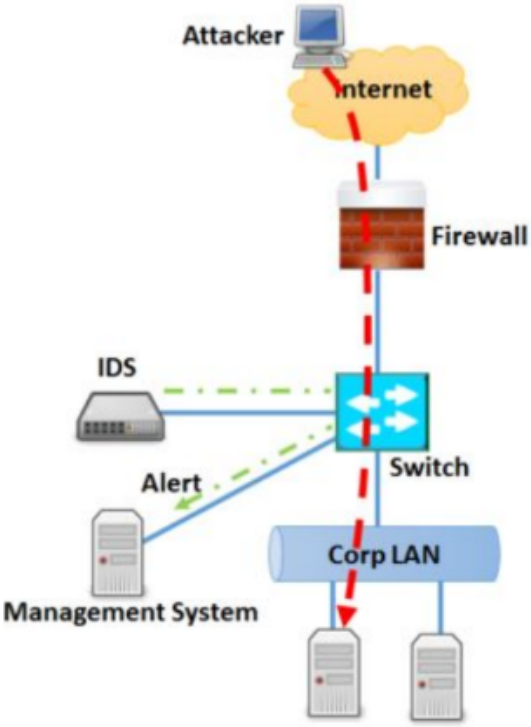
1. **False Positives:**
    - Like IDS, IPS may face challenges with false positives, where legitimate activities are mistakenly flagged as threats.
2. **Performance Impact:**

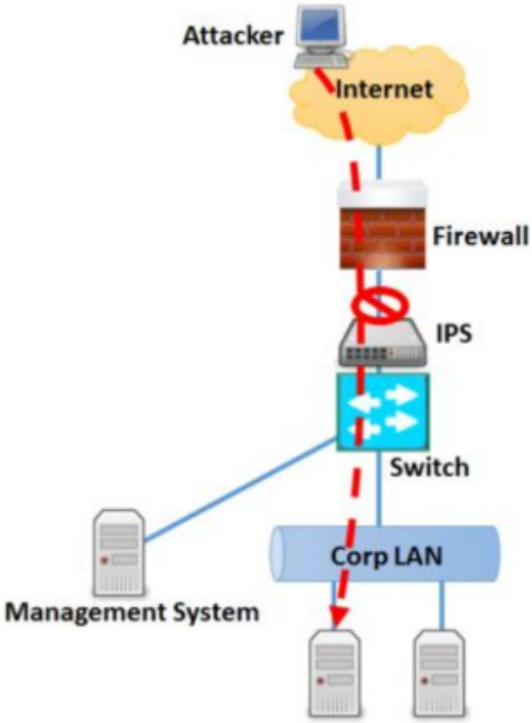
- Introducing real-time prevention measures may impact system performance.

3. **Complexity:**
   - Managing and configuring IPS systems can be complex, requiring expertise.

IDS vs IPS

	IDS	IPS
DESCRIPTION	A system that monitors network traffic for suspicious activity and alerts users when such activity is discovered.	A system that monitors network traffic and alerts for suspicious activity, like an IDS, but also takes preventive action against it.
LOCATION	Resides out-of-band of the communication.	Resides inline within the packet stream.
USE	Warns of suspicious activity taking place, but it doesn't prevent it.	Warns of suspicious activity taking place and prevents it.
FALSE POSITIVE	FPs are usually just a minor convenience. Although the IDS incorrectly labels legitimate traffic as malicious, it doesn't prevent the traffic from entering the network.	FPs can be more serious. When an IPS mistakes legitimate traffic for a threat, it stops it from entering the network, which could impact any part of the organization, not just the IT team.
THREAT DETECTION	Needs help from human being or automated system to interpret the results and decide whether to take action.	Needs to be updated to recognize the latest threats.
PERFORMANCE	No impact in performance because of its placement.	Slows down network performance due to delay caused by inline processing.





Host Intrusion Prevention System (HIPS):

- **Detection Techniques:**
  - Utilizes signature/heuristic or anomaly detection techniques to identify and prevent attacks.
- **Signature Detection:**
  - Focuses on specific content of application network traffic or sequences of system calls, identifying patterns recognized as malicious.
- **Anomaly Detection:**
  - Identifies behavior patterns indicating malware by looking for deviations from normal system behavior.
- **Types of Malicious Behavior:**
  - Examples of malicious behavior addressed by HIPS include modification of system resources, privilege-escalation exploits, buffer-overflow exploits, access to e-mail contact lists, and directory traversal.

Areas of Desktop Protection:

1. **System Calls:**
    - Monitors and controls the system calls made by applications, preventing unauthorized or malicious activities.
2. **File System Access:**
    - Controls and monitors access to the file system, preventing unauthorized modifications or access to sensitive files.
3. **System Registry Settings:**
    - Safeguards the system registry settings, preventing unauthorized changes that could impact system stability or security.
4. **Host Input/Output:**
    - Monitors and controls input/output operations on the host, preventing unauthorized interactions with devices or external systems.

Key Functions of HIPS:

1. **Malicious Activity Prevention:**
    - Actively prevents a variety of malicious activities, protecting the host system from potential threats.
2. **Privilege Escalation Prevention:**
    - Guards against privilege-escalation exploits, ensuring that users and processes operate within their designated permissions.
3. **Buffer Overflow Protection:**
    - Detects and prevents buffer-overflow exploits, a common technique used by attackers to compromise system integrity.
4. **Email Security:**
    - Monitors and controls access to e-mail contact lists, preventing unauthorized or malicious activities related to email communication.
5. **Directory Traversal Prevention:**
    - Protects against directory traversal attacks, where an attacker attempts to access files or directories outside of the intended path.

Benefits of HIPS:

1. **Granular Protection:**
    - Provides granular control and protection at the host level, addressing specific system calls, file access, and other critical functions.
2. **Behavior-Based Defense:**
    - Utilizes behavior-based defense mechanisms to detect and prevent a wide range of attacks.
3. **Customizable Policies:**
    - Allows for the customization of security policies based on the specific needs and risks of the host system.
4. **Desktop Security Enhancement:**
    - Enhances the overall security posture of desktop systems, protecting against various intrusion attempts.

Challenges:

1. **False Positives:**
    - Like many security systems, HIPS may face challenges with false positives, requiring fine-tuning to minimize disruptions to legitimate activities.
2. **Resource Utilization:**
    - Introducing additional security measures may impact system resources, necessitating a balance between security and performance.

Network Intrusion Prevention System (NIPS):

1. Definition:

- Network Intrusion Prevention System (NIPS) is an inline security solution designed to actively prevent and mitigate malicious activities within a network. It possesses the authority to modify or discard packets and can even terminate TCP connections.

2. Key Characteristics:

- **Inline NIDS with Modification Authority:**

- NIPS operates as an Inline Network Intrusion Detection System (NIDS) with the capability to modify or discard packets as a preventive measure.

- **Detection Techniques:**
  - Utilizes both signature/heuristic and anomaly detection techniques to identify and prevent malicious activities within the network.
- **Flow Data Protection:**
  - NIPS may provide flow data protection, requiring the reassembly of application payload in a sequence of packets. This enables a comprehensive analysis of content.

3. Methods to Identify Malicious Packets:

- NIPS employs various methods to identify and take action against packets exhibiting malicious behavior. These methods include:
  - **Signature-Based Detection:**
    - Utilizes a database of known attack signatures to identify and block packets matching known patterns of malicious activity.
  - **Heuristic-Based Detection:**
    - Analyzes the behavior of network traffic, identifying deviations from expected patterns that may indicate potential threats.
  - **Anomaly-Based Detection:**
    - Focuses on identifying abnormal patterns in network behavior, allowing the system to detect novel and previously unknown threats.
  - **Flow Data Analysis:**
    - Involves the reassembly and analysis of application payload across a sequence of packets to ensure a comprehensive examination of content.

4. Preventive Actions:

- NIPS takes proactive measures to prevent and mitigate security threats, including the modification or discarding of packets and termination of TCP connections. This active response helps thwart potential attacks in real-time.

5. Importance in Network Security:

- NIPS plays a crucial role in enhancing network security by actively preventing malicious activities, ensuring the integrity and availability of network resources.

6. Challenges:

- NIPS may face challenges such as false positives, resource utilization, and the need for continuous updates to its signature databases to effectively identify and prevent emerging threats.

Distributed or Hybrid Intrusion Prevention System:

1. Definition:

- A Distributed or Hybrid Intrusion Prevention System is an integrated security solution that gathers data from a multitude of host and network-based sensors. It employs a combination of centralized and distributed components to analyze, correlate, and respond to security threats.

2. Key Features:

- **Data Gathering:**
  - Collects data from a large number of sensors distributed across hosts and networks. These sensors provide real-time information about potential security threats.
- **Central Analysis System:**
  - Relays collected intelligence to a central analysis system. This centralized component correlates and analyzes the data to identify patterns, trends, and potential security incidents.
- **Updated Signatures and Behavior Patterns:**
  - Distributes updated signatures and behavior patterns to all coordinated systems. This ensures that the entire network is equipped with the latest threat intelligence, allowing for a coordinated response against malicious behavior.

3. Example: Digital Immune System:

- **Motivation:**

- Developed initially by IBM and refined by Symantec, the Digital Immune System was designed to address the rising threat of internet-based malwares and their rapid propagation.

- **Objective:**
  - Aims to provide a rapid response time, allowing for the immediate detection and mitigation of malware. The system focuses on stamping out threats almost immediately upon detection.

4. Motivation for Distributed or Hybrid Approach:

- **Rising Threats:**
  - Addressing the increasing threat of internet-based malwares and their swift propagation across networks.
- **Speed of Propagation:**
  - Recognizing the need for a rapid response mechanism to counteract malware before it spreads extensively.

5. Rapid Response Time:

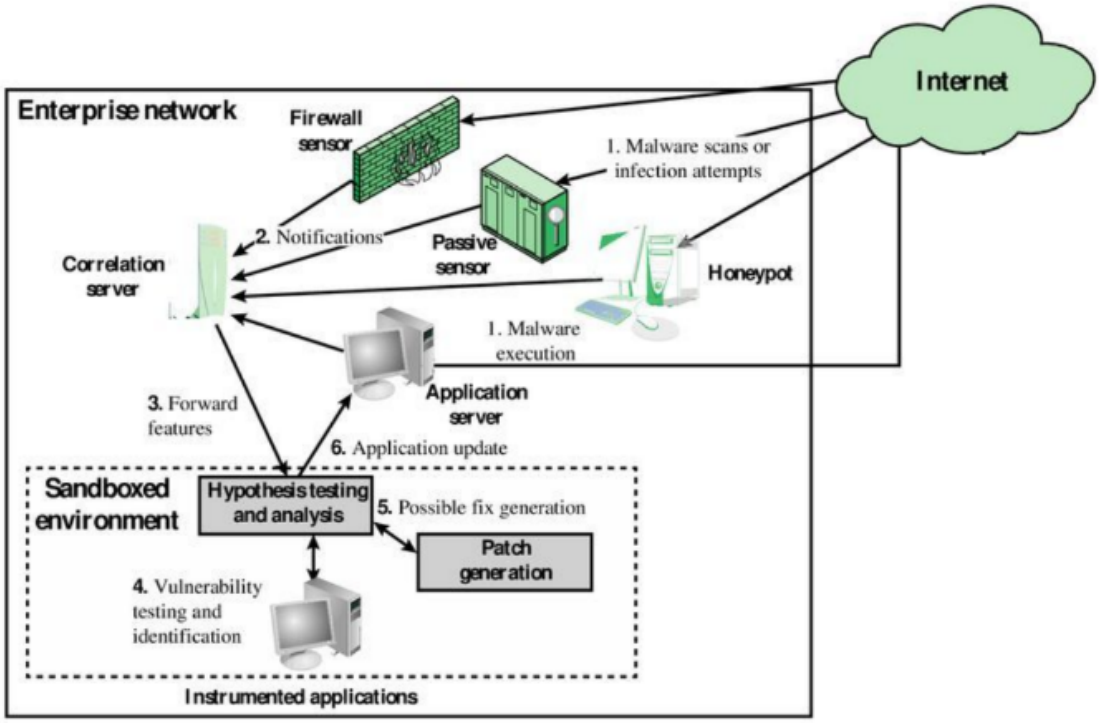
- The primary objective of a Distributed or Hybrid Intrusion Prevention System is to achieve a rapid response time, enabling quick detection and mitigation of security threats. This proactive approach is essential in countering the dynamic landscape of cyber threats.

6. Coordinated Defense:

- By distributing updated signatures and behavior patterns, the system ensures a coordinated defense across all connected systems. This collaborative approach enhances the overall security posture of the network.

7. Significance in Cybersecurity:

- The distributed or hybrid approach to intrusion prevention is significant in addressing the evolving nature of cyber threats. It leverages both centralized analysis and distributed intelligence to create a robust defense mechanism against a wide range of security incidents.



Honeypots

1. Definition:

- A honeypot is a deceptive system designed to mimic vulnerable targets, inviting potential attackers. It serves the purpose of collecting information about the identity, methods, and motivations of intruders.

2. Purpose:

- **Deception:**
  - **Honeypots are deception systems meant to divert attackers from critical systems, allowing administrators to observe and analyze their activities.**

3. Placement:

- **Strategic Location:**
  - Honeypots are strategically placed **either in the DMZ (Demilitarized Zone) or behind the network firewall, depending on the network architecture.**

4. Characteristics:

- **Deception Mechanism:**
  - Unlike sniffer-based Intrusion Detection Systems (IDS) like HIDS and NIDS, honeypots are not strictly focused on sniffing. However, they serve as effective deception systems similar to HIDS and NIDS.

5. Decoy Systems:

- **Lure and Collect:**
  - Honeypots act as decoy systems filled with fabricated information that appears valuable to attackers. The goal is to lure attackers, collect information about their activities, and encourage them to stay on the system for analysis.

6. Importance:

- **Capture New Threats:**
  - Honeypots are considered an advanced technique and are typically employed after other security measures. They are particularly valuable for capturing new worms and analyzing their behavior.

7. Risk Consideration:

- **Potential Risk:**
  - While honeypots offer benefits, there is a risk that attackers might recognize and use them. Therefore, deploying honeypots should be done carefully to avoid alerting potential intruders.

8. Types of Honeypots:

- **Host Trap:**
  - Runs real or simulated services on a sacrificial computer to attract attackers.
- **Network Trap:**
  - Gives the impression of a vulnerable organization, enticing intruders.
- **Email/Spam Trap:**
  - Aims to identify and block spammers by inviting spam emails.
- **Malware Honeypot:**

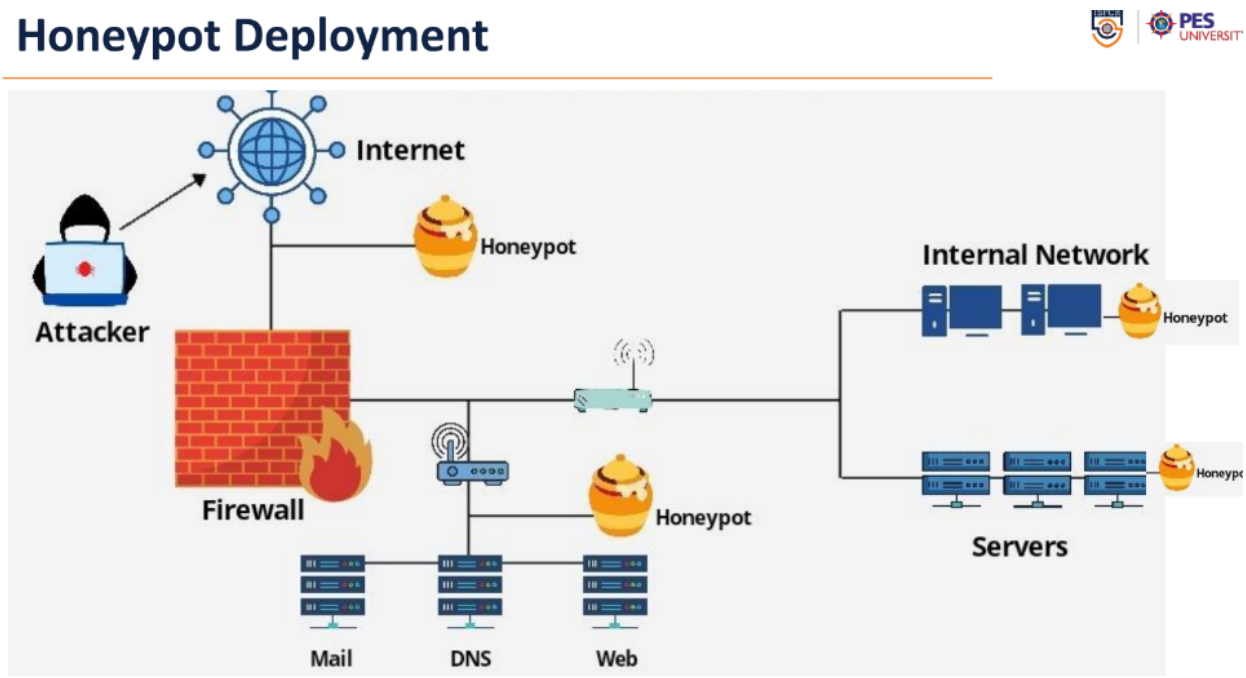
- Mimics software apps and APIs to attract malware attacks.

9. Honeypot Classifications:

- **Low Interaction Honeypot:**
  - Emulates specific IT services but does not execute a full version. Provides a less realistic target, suitable for warning of imminent attacks.
- **High Interaction Honeypot:**
  - A real system with a full operating system, services, and applications. Represents a more realistic target but requires significant resources.

10. Benefits of Traffic Analysis:

- **Insights Gained:**
  - Analyzing traffic to honeypots helps determine the origin of cybercriminals, assess the level of threat, understand attack methods, identify targeted data or applications, and evaluate the effectiveness of existing security measures.



Snort

1. Introduction:

- **Open Source IDS/IPS:**
  - **Snort is an open-source IDS/IPS (Intrusion Detection System/Intrusion Prevention System) available at snort.org.**
- **Primary Uses:**
  - **Snort serves as a packet sniffer, packet logger, and IDS/IPS.**
- **Performance:**
  - **Lightweight and fast, capable of running at Gbits/s with proper hardware and tuning.**
- **Suitability:**
  - Suitable for monitoring multiple sites and sensors.
- **Community Involvement:**
  - **Actively developed by the community, ensuring up-to-date rules.**

2. Network Traffic Monitoring:

- **Packet Sniffing:**
  - Snort can be used for packet sniffing, logging, and intrusion detection.
- **Choke Point:**
  - **It is often deployed close to the "choke point," where all traffic flows through, such as after a firewall.**

3. Snort Design:

- **Packet Analysis Pipeline:**
  - Snort's design includes a packet analysis pipeline that processes network traffic.

4. Snort Rules:

- **Plain Text Files:**
  - Snort rules are plain text files added to the /etc/snort/rules/ directory.
- **Rule Sets:**

- Rules are distributed in two sets: "Community Ruleset" and "Snort Subscriber Ruleset."

5. Snort Detection Rules:

- **Structure:**
  - Rules consist of a rule header and rule options.
- **Options:**
  - Options include meta-data, payload analysis, non-payload analysis, and post-detection triggers.

6. Snort Rule Actions:

- **Inline Deployment:**
  - **Actions like drop, reject, and sdrop are available when Snort is deployed inline, transforming it into an Intrusion Prevention System (IDPS).**

7. Snort Operations:

- **Modes:**
  - Snort can run as a packet sniffer, packet logger, or IDS/IPS.
- **Running Modes:**
  - Snort can be started with various options for sniffing, logging, IDS, or IPS.

8. Tuning Strategies:

- **Optimization:**
  - Tune Snort by enabling only necessary rules, configuring preprocessors, and adjusting variables in snort.conf.
- **Custom Rules:**
  - Caution is advised when writing custom rules to avoid performance impact and false positives.

9. Additional Examples:

- **Rule Examples:**
  - Examples of Snort rules for detecting SubSeven trojan and mountd access were provided.

Snort Rule Examples

Example 1: SubSeven Trojan Detection Rule

```
alert tcp $EXTERNAL_NET 27374 -> $HOME_NET any (msg:"BACKDOOR
subseven 22"; flags: A+; content:"|0d0a5b52504c5d3030320d0a|";
reference:arachnids,485; reference:url,www.hackfix.org/subseve
n/; sid:103; classtype:misc-activity; rev:4;)
```

- **Explanation:**
  - This rule detects traffic related to the SubSeven trojan.
  - It triggers an alert if a TCP packet is detected from an external network to any port on the local network, containing the specified content.
  - The rule includes additional information such as references, classification type, and revision.

Example 2: Mountd Access Detection Rule

```
alert tcp any any -> 192.168.1.0/24 111 (content:"|00 01 86 a5
|"; msg: "mountd access";)
```

- **Explanation:**
  - This rule detects mountd access on TCP port 111.
  - It alerts if a TCP packet from any source port and any source IP address is destined for any port in the 192.168.1.0/24 subnet on port 111.
  - The specified content is used for pattern matching.