Intrusion detection system (IDS) overview

- Detects intrusions in real time and alerts
- Can filter the traffic and alert the security personnel
  - Also known as ***intrusion detection and prevention systems (IDPS)*** or ***intrusion prevention systems (IPS)***
- Inspects both incoming (inbound) and outgoing (outbound) traffic
- Can be a software or hardware
- Usually placed near the firewall
  - Inside or outside depending on which traffic is being monitoring
  - 💡 Good to deploy on both places (before and after DMZ) for layered defense

Intrusion types

- **Filesystem intrusion**
  - Unexpected creation/deletion/modification of files or file attributes (e.g. permissions)
- **Network intrusion**
  - Increase in bandwidth consumption
  - Unexpected incoming connections e.g. attempted logins
  - Sudden increase of logs can be caused by DoS/DDoS
- **System intrusion**
  - Missing/modified for log, system or configuration files
  - Degradation in system performance
  - Unfamiliar processes, system reboots, crashes

IDS types

Network-based vs Host-based IDS

- Comparison

	NIDS	HIDS
Strength	Sensing attacks from outside	Sensing attacks from inside that NIDS cannot examine
Packet headers	Examines	Does not understand
Host	Independent	Dependent
Bandwidth	In need of	Does not require
Performance	Slows down networks where it's installed	Slow down hosts where it's installed
Attack types	Senses network attacks as payload is analyzed	Senses local attacks before they hit the network
False positive rate	High	Low

- See also WIDS (Wireless Intrusion Detection system)

Network-based intrusion detection systems (NIDSs)

- Also known as ***network-based IDS***
- Inspects each incoming packet for anomalies and suspicious patterns.
- Can detect DoS attacks, port scans, or break-in attempts.

Network tap

- Typically a hardware device, which provides a way to access the data flowing across a computer network.
- Provide IDS visibility into the traffic flowing over the network
- E.g. a hub connected on the segment or a network appliance created specifically for the task

Snort

- Open-source NIDS that's most widely deployed
- Rule-based IPS to detect and stop packages
- Can block expressions such as
  - `▯ /(\%27)|(\')|(\-\-)|(\%23)|(#)/ix`
  - `/((\%27)|(\'))union/ix`

Snort configurations

- Alerts are defined in Snort configuration file
  - Configuration file is at `/etc/snort`, or `C:\Snort\etc`
- Can be configured to use as:
  - packet sniffer
    - E.g. `snort -vde`
  - packet logger

        - E.g. `./snort -dev -l ./log`
    - Network intrusion detection system by
        - Does not drop packets
        - Evaluates packets to check all alert rules, logging the matches.
        - E.g. `./snort -dev -l ./log -h 192.168.1.0/24 -c snort.conf`
    - Network intrusion protection System

Snort rules

- All rules are checked for each packet
- If multiple matches are found:
    - Alerts the most unique (specific) rule ignoring the more generic one.
- 📝 **Syntax**
    - Action protocol address port -> address port (option:value;option:value)
    - E.g. `alert tcp 10.0.0.1 25 -> 10.0.0.2 25 (msg:"Sample Alert"; sid:1000;)`

Host-Based intrusion detection systems (HIDSs)

- Also known as ***host-based IDS***
- 📝 Analyzes behavior and events on a particular host e.g. a desktop PC or a server.
- Can detect both anomalies and unauthorized changes in the filesystem.
- Log file monitoring (LFM): Monitoring logs files for malicious events.
- **File integrity checking**
    - Checking for modified files e.g. ossec-hids
    - Compares the current hash value of the file against its known-good hash value.
- E.g. Windows Defender, Norton Internet Security..

Active vs passive IDS

Active IDS

- Also known as **Intrusion Detection and Prevention System (IDPS)** or **Intrusion Protection Systems (IPS)**
- Configured to automatically block suspected attacks without any intervention required by an operator

Passive IDS

- Configured to only monitor and analyze network traffic activity and alert
- Does not perform any protective or corrective functions on its own

Signature-based vs Anomaly-based IDS

- 💡 Recent systems uses both (hybrid approach) to cover each others flaws

Signature recognition

- Also known as ***misuse detection***, ***signature based IDS*** or ***signature-based IDS***
- 📝 Compares incoming and outgoing traffic to the signatures of already known attacks
- Based on a database of previous attack signatures and known system vulnerabilities.
- A signature is a recorded evidence of an intrusion or attack
- 📝 **Pros**
    - Little false positives
    - No need for a training phase, starts working out of the box
- 📝 **Cons**
    - Vulnerable to unique attacks, easy to fool
    - High dependency of latest updates, constant maintenance
    - Signature data consumes traffic

Anomaly detection

- Also known as ***not-use detection***, ***behavior based IDS*** or ***behavior-based IDS***.
- 📝 Analyzes characteristics of the system's users and components and looks for deviations.
- Learns pattern of normal system activity to identify active intrusion attempts.
- Deviations from this baseline or pattern cause an alarm to be triggered.
- Can use artificial intelligence or can be based on heuristics or rules
- 📝 **Pros**
    - More suitable for blocking future unknown attacks
    - Low dependency of latest updates, constant maintenance
- 📝 **Cons**
    - Higher false positive alarm rates
    - Challenging to construct a model thoroughly on a regular network.

Protocol anomaly detection

- Identifies anomalies specific to a protocol
- Uses a model of the different ways vendors deploy the TCP/IP protocol.

IDS alerts

- 📝 **IDS alert types**
    - **True positive**: Attack + Alert
    - **False positive**: No attack + Alert
    - **True negative**: No attack + No alert
    - **False negative**: Attack + No alert
    - 💡 False negatives are considered far worse than false positives
- 📝 **IDS alert thresholding**
    - Also known as ***alert throttling*** or ***event filtering***.
    - Reducing the volume of repeated alerts
    - E.g. ignore alerts after nth times during X minutes

Firewall vs IPS vs IDS

	Firewall	IPS	IDS
Abbreviation for	-	Intrusion Prevention System	Intrusion Detection System
Firewall	Filters incoming and outgoing network traffic based on predetermined rules	Inspects traffic, detects it, classifies and then proactively stops malicious traffic from attack.	Monitors a traffic for malicious activity or policy violations and sends alert on detection.
Working principle	Filters traffic based on IP address and port numbers (layer 3), state of the connection (layer 4), or contents of packet (layer 7)	Inspects real time traffic and looks for traffic patterns or signatures of attack and then prevents the attacks on detection	Detects real time traffic and looks for traffic patterns or signatures of attack and them generates alerts
Configuration mode	Layer 2 to 7	Layer 3 and 4	Layer 2 to 7
Usual placement	First line of defense	After firewall	After firewall
Action on unauthorized traffic detection	Block the traffic	Block the traffic	Alerts/alarms

Evading IDS

- See also SQL evasion
- See also bypassing IDS and firewall when scanning, evading firewalls

Obfuscation

Path obfuscation

Type	Clear-text	Obfuscated-text
Self-referencing directories	`/etc/passwd`	`/etc/./passwd`
Double slashes	`/etc//passwd`	`/etc/passwd`
Path traversal	`/etc/passwd`	`/etc/dummy/../passwd`
Windows folder separator	`../../cmd.exe`	`..\..\cmd.exe`
IFS (Unix shells)	`/etc/passwd`	`CMD=X/bin/catX/etc/passwd;eval$CMD`

URL encoding

- E.g. `http://cloudarchitecture.io/paynow.php?p=attack` becomes `http://cloudarchitecture.io/paynow.php?p=%61%74%74%61%63%6B`
- **Null-byte attacks**
  - Evasion technique and attack at the same time (to get unauthorized access to server files)
  - Effective against applications
    - developed using C-based languages
    - using native file manipulation
  - Can be done by appending `%00`

Unicode encoding

Unicode

- Provides unique identifier for every character in every language
- Facilitates uniform computer representation of the world's languages
- Each character can be represented by U+xxxx where x is a hexadecimal digit.

Unicode encoding attack

- Also known as **UTF-8 encoding**
- Presenting information in an unusual way to confuse the signature-based IDS
- 📝 A very common way to evade IDS
- E.g. instead of `http://vulneapplication/../../appusers.txt` using `http://vulneapplication/%C0AE%C0AE%C0AF%C0AE%C0AE%C0AFappusers.txt`

Encryption

- 📝 Most effective evasion attack
- IDS becomes unable to analyze packets going through these encrypted communications
- E.g. SSH, SSL/TLS, or OpenVPN tunnel

Polymorphism

- Using polymorphic shellcode to create unique network patterns to evade signature detection
- E.g. by encoding payload by XORing and putting the decoder in the start of the payload where the target runs the decoder when it executes the code
- Tools include ADMMutate: A shellcode mutation engine, can evade NIDS

Denial of service

- If IDS fails, it allows the traffic to go through
- Passive IDSes are vulnerable as they are fail-open.
- E.g.
  - by exploiting a bug in the IDS, consuming all of the computational resources on the IDS
  - deliberately triggering a large number of alerts to disguise the actual attack.

False positive generation

- Also known as ***flooding*** or ***false-positive generation***
- 📝 Designed to create a great deal of log noise in an attempt to blend real attacks with the false
- Attackers craft packets known to trigger alerts within the IDS, forcing it to generate a large number of false reports
- Similar to the DoS method is to generate a large amount of alert data that must be logged
- Make is difficult legitimate attacks and false positives by looking at logs
- Can even generate false positives specific to an IDS if attacker has knowledge of IDS used.
- Tools include inundator: intrusion detection false positives generator.

Insertion attack

- Exploited by sending packets to an end-system that it will reject but IDS will think are valid.
- By doing this the attacker is ***inserting*** data into the IDS
- Allows attacker to defeat signature analysis and to slip attacks past an IDS.
- An IDS can accept a packet that an end-system rejects.
  - also misbelieving that the end-system has accepted and processed the packet
- As signature analysis use pattern-matching in a stream of data to detect strings.
  - E.g. IDS can easily detect `phf` in HTTP request.
    - But the attacker insert data and make it look like e.g. `pleasdontdetectthisforme` where only `phf` part is sent to the original stream.
- A countermeasure would be making IDS as strict as an end-system to minimize this attacks
  - however it then allows for evasion attacks.

Session splicing

- Splits the attack traffic in to many packets such that no single packet triggers the IDS.
- Network level attack
- ❗ Not the same as IP fragmentation
  - Session splicing concerns just HTTP payload in chunks to prevent string matches by IDS.
- Send parts of the request in different packets
  - E.g. `"GET / HTTP/1.0"` may be split across multiple packets to be
    - `"GE"`, `"T "`, `"/"`, `" H"`, `"T"`, `"TP"`, `"/1"`, `".0"`
- Tools include Nessus or Whisker

Tools

- `fragroute` for packet fragmentation
- Different scanners such as `nmap` has also options to evade IDS.
- Also many web vulnerability scanners can be used such as Nikto], Whisker and Nessus

Whisker

- Also known as `libwhisker`
- Open-source perl module for HTTP-related functions, including vulnerability scanning and exploitation.
- 📝 Helps also to evade IDS with session splicing and tactics including:

Name	Explanation/Example
Method matching	GET -> HEAD
URL encoding	HEX %xx notation
Double slashes	`/` -> `//`
Reverse traversal	`/dir/blahblah/../`
Self-reference directories	`/dir/./././././ == /dir/`
Premature request ending	stop at the first `HTTP/1.?\r\n`
Parameter hiding	`%3f` -> `?`
HTTP mis-formatting	`%20` -> `%09 (TAB)`
Long Urls	`GET /<random>/../dir/a.cgi`
DOS/Win directory syntax	`'/'` -> `\`
NULL method processing	`GET\0`
Case sensitivity	`'abc' -> 'ABC`

Firewall overview

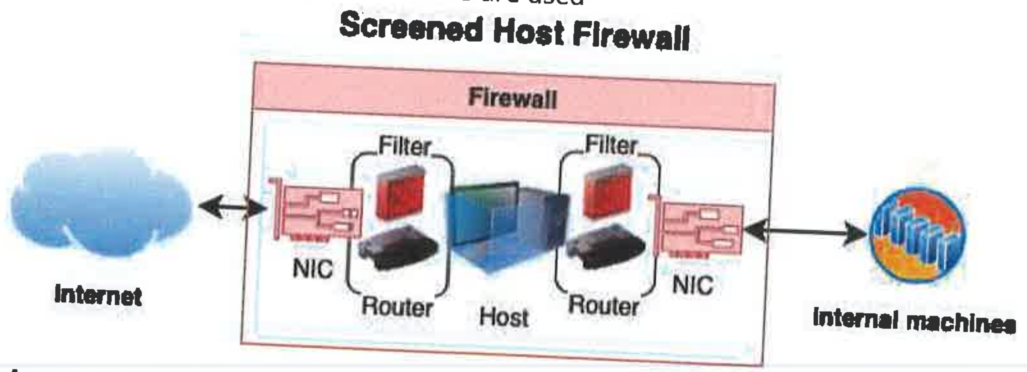
- Monitors network traffic and allows or blocks traffic based on a defined set of rules
- Predefined rules ensure only allowed incoming and outgoing traffic can pass through.
- Gateway/filter between two networks usually between private and public (internet)

Firewall architecture

Multi-homed firewalls

- Also known as ***multi-homed hosts***, ***multihomed hosts***, ***multi homed hosts***, ***multi homed firewalls*** or ***multihomed firewalls***.
- 📝 A host / firewall that has more than single network interface (NIC)
- Each interface are connected to separate network segments

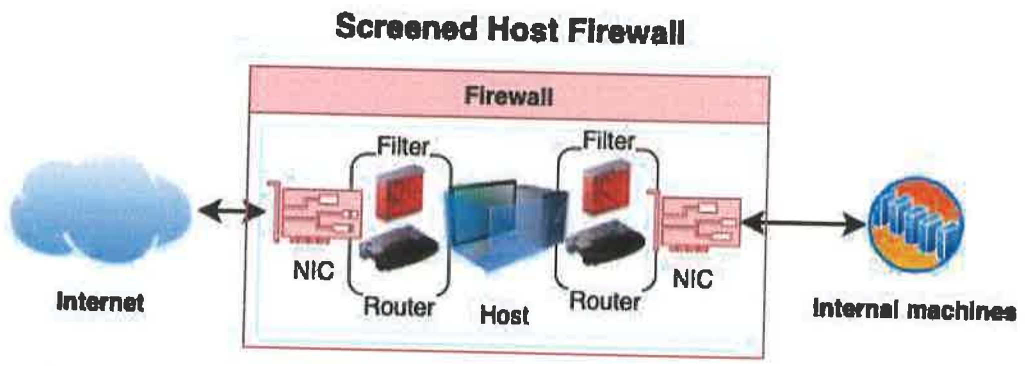
Dual-homed firewalls

- Dual-home can be a proxy, gateway, firewall etc.
- A special case of bastion hosts.
- Allows them to transfer data between the two networks
- Has two interfaces
    - External or public, usually to untrusted network such as Internet
    - Internal or private, usually to trusted network such as Intranet
- **Screened Host**
    - Firewall architecture where two filters are used
    - 

Bastion hosts

- Mediates traffic between inside and outside networks
- Designed and configured (hardened) to withstand attacks
- Usually hosts a single service e.g. a proxy server

Screened subnet firewalls

- Screen subnet is also known as **DMZ** or **perimeter network.**
- Used to secure servers that are accessible from the internet.
- Consists of three zones:
    1. External (e.g. to internet)
    2. 📝 Demilitarized Zone (DMZ)
        - Placed in-between internal (trusted) and external (untrusted) network
        - Usually where bastions are placed
    3. Private (to internal network e.g. intranet)
- 

- Two routers
    1. External router separating traffic from a perimeter network (DMZ)
    2. Internal router separating perimeter from internal network
- Can be achieved through either
    - Single firewall with three interfaces
        - See also zone-based firewall
    - Three different firewalls
        - 💡 Better as compromising one won't compromise all

Firewall categories

- **Hardware Firewall**
    - Device placed on the network's perimeter
    - Uses packet filtering technique to filter the traffic
    - Can be a standalone device or part of a router
- **Software Firewall**
    - Filters traffic on the installed machine
    - Protects its host from unauthorized access, trojans, viruses, and worms.

Software vs Hardware firewalls

Attribute	Hardware firewall	Software firewall
Price	More expensive	Cheaper
Maintainability	Hard	Easy
Speed	Faster response time	Slower response times
Interference	Minimal, can easily remove/replace etc.	Difficult to uninstall

Firewall types per OSI Layer

- 📝 Technologies used per OSI layer

OSI Layer	Firewall technologies
7. Application	• Virtual Private Network (VPN) • Application Proxies • Web Application Firewall (WAF) • Request filtering based on headers and payload
6. Presentation	• Virtual Private Network (VPN)
5. Session	• Virtual Private Network (VPN) • Circuit-level gateway
4. Transport	• Virtual Private Network (VPN) • Packet filtering based on port numbers
3. Network	• Virtual Private Network (VPN) • Network Address Translation (NAT) • Packet filtering, Stateful multilayer inspection
2. Data Link	• Virtual Private Network (VPN) • Packet filtering

- 📝 All vulnerabilities in one layer is independent of the other layer.
  - E.g. cross-site scripting (application layer) vulnerable application would be vulnerable to it regardless of any protection on other layers.
  - Also vulnerabilities exist at network layers would not be visible to a stateful
- 💡 In most cases, you'd use both a L3 and an L7 firewall and the two complement each other.
- See also OSI model

Packet filtering firewalls

- Implemented on the Network Layer, usually part of routers
- Designed to analyze each packet individually to apply a set of filters
- 📝 Examines the packets headers for source, destination, protocol, destination port, flags..
- Packet is dropped (not forwarded to its destination) if it does not comply with the predefined rules
- 📝 Can be stateful (mostly, newer) or stateless (older).
- Ineffective in preventing Web Application attacks as port 80 and 443 would not be blocked.

Access Control Lists (ACLs)

- Usually packet filtering rules are defined using ACLs (access control lists)
- Known also **Wireless Access Control List (WACL)** as in wireless routers.
- 📝 Type of rule-based access control
- 📝 E.g.
  - In Linux using iptables, disable all incoming SSH using `iptables -A INPUT -p tcp --dport 22 -j DROP`
  - In Windows it's controlled with `netsh advfirewall` (older: `netsh firewall`)
  - E.g. on Cisco routers using `access-list 101 deny tcp any host 100.100.100.1 eq 22`
    - where `101` is sequence number that helps with ordering of the rules
      - the lower the number is the higher priority it gets in the ordering
    - ACL are processed in top down meaning if a condition is met all processing is stopped.

Packet inspection

Port-based classification

- E.g. TCP 80 = HTTP
- Old way, today it's useless as assumptions can be wrong

QoS markers (DSCP)

- Similar to port-based but based on QoS tags for prioritization
- Ignored as it's easy to cheat and forge

Statistical traffic classification

- Based on manual rules or ML (machine learning) dataset
- Hard to create a good dataset and poor accuracy for cases outside of the set

Deep Packet Inspection (DPI)

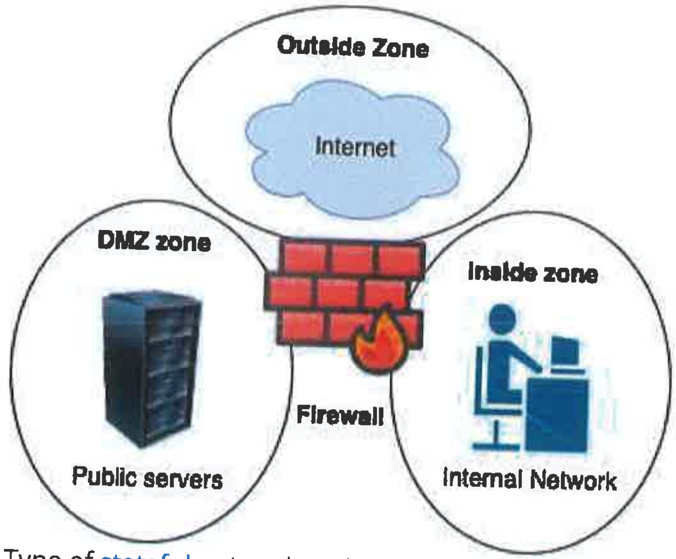
- Inspecting packet payload
- Encrypted payload (e.g. HTTPS) ensures privacy and confidentiality
- Can see DNS query name, HTTP Host/Server fields and SSL/QUIC SNI (Server Name Indication)
- Important for cloud applications sharing same IPs
- Used in e.g. state censorship, and cloud-generation firewalls
- Tools: ntop that's based on nDPI

Stateful firewalls

- Also known as ***stateful inspection firewalls***
  - Stateful inspection is also known as **dynamic packet filtering**
- Keeps state of open connections.
- When a packet arrives, its header is examined to determine if it belongs to an already established connection.
- If it belongs to a connection, then it's allowed to go through
- Analyzed against defined set of rules to determine if connection is allowed to flow through

- E.g. will filter `ACK` without TCP connection establishment (with `SYN`)

Zone-based firewall

- 

- Type of stateful network and multi-homed firewalls.
- Allows setting up rules such to filter / inspect traffic between zones.

Firewall zone

- Logical area in which the devices having same trust levels resides.
- An interface is assigned to a zone.
- By default, traffic is not allowed from one zone to another.
- Common zones
  - **Private**: inside | Most trusted network
  - **Public**: outside | Untrusted network
  - **DMZ (demilitarized)**: neutral | Sits in-between private and outside

firewalld

- Zone-based network level firewall on Linux
- Rules can be
  - **Runtime**: For duration of the session
  - **Permanent**: Persists through reboot or reload of a firewall
    - ▯ Requires `firewall-cmd --reload` to apply the rules.
- Commands:
  - Install: `yum install firewalld`
  - Start: `systemctl start firewalld`
  - Add rule (allow access): `firewall-cmd --permanent --zone=public --add-port=80/tcp`
  - Remove rule (deny access): `firewall-cmd --permanent --zone=public --remove-port=80/tcp`
  - Reload to apply changed rules: `firewall-cmd --reload`
  - Block access from different countries through geo IP block

1. Download ip blocks at e.g. ipdeny.com
    1. Download: `wget https://www.ipdeny.com/ipblocks/data/countries/all-zones.tar.gz`
    2. Extract: `tar -vxzf all-zones.tar.gz`
2. Create a list called e.g. `blacklist`: `firewall-cmd --permanent --new-ipset=blacklist --type=hash:net --option-family=inet`
    - `--type`: `hash` is the storage type, `:net` is to block whole subnet
    - `--inet`: ipv4
3. Add entries to list: `firewall-cmd --permanent --ipset=blacklist --add-entries-from-file=us.zone`
4. Apply entries in list
    - Deny access: `firewall-cmd --permanent --zone=drop --add-source=ipset:blacklist`
    - `firewall-cmd --reload`

- In `/etc/firewall/firewall.d` file you can e.g. enable logging for denied traffic.
    - 💡 Careful for what you log as they can grow rapidly as servers, web applications etc. are also logging.

Circuit level gateway firewalls

- Also known as **circuit-level gateway**
- 📝 Monitors TCP handshakes to determine if the requested connection is legitimate.
- 📝 Implemented on
    - OSI model: Session Layer (5)
    - TCP/IP stack: Between application (4) and transport layer (3)
- It acts as a proxy server using address translation
    - Maps all of internal IP addresses to one "safe" IP address for incoming packet
    - Address is associated with the firewall from which all outgoing packets originate
    - Provides security as untrusted network is only aware of single IP address

Application level firewalls

- Also known as **proxy firewall**, **application firewall** or **gateway firewall**.
- Installed on a proxy server to act as a barrier between internal and external networks.
- Implemented on the application layer.
- 📝 Designed to filter traffic only for the protocols for which they are configured.
- Exposes single address instead of exposing internal network.
    - Clients first establish a connection with a proxy firewall, and then a new network connection is initiated on the client's behalf.
    - Utilizes NAT (Network address translation) to make the translations
- Can function in two modes:
    - **Active application-level firewalls**: Actively reject or deny requests
    - **Passive application-level firewalls**: More like IDS, does not filter

Web Application Firewall (WAF)

- Type of an application firewall that filters, monitors, and blocks HTTP traffic to and from a web service.
- 📝 It can prevent attacks exploiting a web application's known vulnerabilities
  - E.g. injection, cross-site scripting (XSS), file inclusion, and improper system configuration.

Application level vs Network level Firewalls

Capability	Application level	Network level
Configuration	Advanced	Narrow (IPS and ports)
Coverage	Small (protocol-specific)	Wider (any IP-based)
Speed	Slower	Faster
Application threats	Greater security	Blind

- 💡 Use multiple layers of defense in depth
  - Level 3 firewall at the edge that only allows inbound traffic on the specific ports used by apps
  - Route those ports to an L7 firewall for deeper inspection.

Stateful multilayer inspection firewall

- Combination of different firewalls:
  - packet filtering *(network layer)*: to filter packets
  - circuit level *(session layer)*: to verify legitimate sessions
  - application level *(application layer)*: to evaluate packets
- A type of **hybrid firewall** as it's a mix of some of the firewalls already

Network Address Translation (NAT)

- Implemented by many firewalls just like routers
- Enables LAN to use different sets of IP addresses for external and internal traffic.
- NAT modifies the packet's IP header and translates one address space into another
- NAT allows to hide the layout of the internal network.
- **Basic NAT**
  - One-to-one mapping where each internal IP is mapped to a unique public IP.
  - Too expensive to implement
- **Port address translation (PAT)**
  - Also known as ***network address and port translation (NAPT)***, ***IP masquerading***, ***NAT overload*** and ***many-to-one NAT***.
  - Allows multiple internal IP addresses to be mapped to single public IP
  - Uses different port (and other items) for each web conversation.
  - Typically used as is the cheaper option.
- Dynamic vs Static NAT

    - **Static NAT**: one-to-one internal to public static IP address mapping
    - **Dynamic NAT**: uses a group of available public IP addresses.

Virtual Private Network (VPN)

- A network which enables a secure connection to a private network through the Internet.
- Information is protected by encryption and integrity checks.
- Can use e.g. IPSec or OpenVPN tunnelling protocol.

Evading firewalls

- See also • Bypassing IDS and firewall | Scanning Networks • Evading IDS

Firewall evasion techniques

- Source routing to avoid the route of the firewall

Using fragmented packets

- The idea is to split up the TCP header over several packets to make it harder
- E.g. `-f` command in `nmap`: `nmap -f 192.168.1.12`
  - utilizes 16 bytes per fragment which diminishes the number of fragments
  - to specify own offset size: `nmap --mtu 16 192.168.1.12`
- ❗ Most modern firewall and IDS detect fragmented packets.

Firewalking

- 📝 Discovers firewall rules using traceroute-like technique with IP TTL expiration
- Works by sending out TCP or UDP packets with a TTL one greater than the targeted gateway
  - Tests if gateway allows the traffic to find firewalls
- Requires knowledge of:
  1. Known gateway (can be firewall) before the host (serves as waypoint)
  2. IP address of a host located behind the firewall.
- ❗ If a host on the other side of the firewall cannot be targeted then firewalking will not be successful
- Also known as ***port knocking***
  - Externally opening ports on a firewall by generating a connection attempt on a set of prespecified closed ports
- **Tools**
  - `firewall` script in `nmap`: e.g. `nmap --traceroute --script=firewalk --script-args=firewalk.max-probed-ports=-1 192.168.3.11`
  - Firewall tool e.g. `firewalk 192.168.1.2 192.168.3.11`
    - Responses can be interpreted as:
      - `ICMP_TIME_EXCEEDED`: Gateway forwards packets to next hop where they're expired.
      - **No response**: Port is probably blocked
- **Countermeasures**
  - Use Network Address Translation to hide the addresses on your internal networks
  - Block all outgoing TTL Exceeded in Transit packets in the firewall

HTTP and ICMP tunneling

- Can be used to bypass firewalls rules through obfuscation of the actual traffic
- Works by injecting arbitrary data into packets sent to a remote computer
- Hard to detect without proper deep packet inspection or log review
- HTTP tunneling (port 80) is almost never filtered by a firewall.

DNS tunneling

- Also known as **TCP over DNS**
- Provides a TCP tunnel through the standard DNS protocol
- Used to evade firewalls as most firewalls allow DNS traffic to freely pass into and out of the network.
- 🤫💡 May browsing internet in coffee shops for free
- Tools include • iodine • ThunderDNS

Banner grabbing

- Used to identify firewalls.
- Tools
  - Using Nmap banner script `nmap -sV --script=banner <target-ip>`
  - Using `netcat`: `nc -v -n 192.168.51.129 21`

Honeypots

- Designed as a trap for attackers who try to access the network.
- Any interaction with a honeypot points to a malicious activity.
- E.g. free proxy servers, VPNs, WiFis...
- Can be used by law enforcements to e.g. get IP of attackers, or malicious people to blackmail you.
- IDP (intrusion detection prevention) systems or admins can redirect intruders to a virtual machine as honeypot.

Honeypot types

- **Low-interaction honeypots**
  - Mimic small number of applications and services that run on a system or network.
  - Capture information about network probes and worms.
- **Medium-interaction honeypots**
  - Mimic real operating system, applications and services
  - Capture more data compared to low-interaction honeypots
- **High-interaction honeypots**
  - Run real operating systems and applications
  - Gather information about the techniques and tools used in the attack.
- **Production honeypots**
  - Mimic the organizations real production network allowing more attacks
  - Helps network admins to take preventive measures to reduce the probability of an attack
  - Differs from high-interaction honeypots as they do not run real operating systems or applications.
- **Research honeypots**
  - High-interaction honeypots
  - Mainly used by security analysis and researchers
  - Goal is to understand how the attack was performed

Evading honeypots

- Goal is to avoid being trapped in a honeypot
- Tools are used to detect honeypots that are installed on the network.
- Well configured honeypot is nearly impossible to detect.
- Best to target specific IPs known ahead of time to be valid machines.
- Some giveaways (see discussions, paper):
  - They can be to good too obviously insecure e.g. sitting near DMZ.
  - No network traffic
  - Unrealistic configurations e.g. IIS server on Linux, file names, drivers (e.g. VMWare defaults) etc.
  - Attacker can detect if it's running in a VM, disrupt the VM.

    - Performance degradation or fails under a sustained attack because of e.g. insufficient bandwidth.
    - Logging instructions affects total execution time of hacker commands.
- There are some attempts to automate such as Honeypot Hunter (commercial scanner) or using machine learning.

Setting up a proxy server as honeypot

- 🤗 This walkthrough is out of scope to to get better understanding, unrelated to exam.
- Setup the honeypot
    - Install `squid` the proxy server: `yum install squid`
    - Start `squid`: `systemctl start quid`
    - Start automatically on reboot (good for cloud machines): `systemctl enable quid`
    - Configure in `vim /etc/squid/squid.conf`:
        - Has ACL (access list) rules to e.g. allow source ip ranges and ports for access
    - People can now use the proxy server with its public ip and port `3128` as default.
    - It will be detected by automated crawlers on internet that's looking for e.g. vulnerabilities.
- Monitor the traffic using sniffing tools such as `tcpdump` or Wireshark
    - Create a named pipe (aka FIFO) file: `mkfifo myPipe.fifo`
    - Redirect proxy server logs to a local file `ssh root@<proxy-server-ip> "tcpdump -s 0 -U -n -w - -i eth0 not port 22" > myPipe.fifo`
        - `-s 0`: sets snapshots length to default `262144` bytes to take
        - `-U`: unbuffered, dump anything
        - `-n`: don't convert addresses to names
        - `-w`: write file instead of parsing and printing them out.
            - `-`: means standard output, so it writes to standard output.
        - `-i eth0`: capture traffic on `eth0` interface
        - `not port 22`: filter out own connection to the server
    - Run `wireshark-gtk -k -i myPipe.fifo` to start wireshark
- You can now use proxy using e.g. Firefox and see the traffic.