CS 419: Computer Security

Week 12: Network Security
Firewalls and VPNs

Paul Krzyzanowski

Network Address Translation (NAT)

NAT: Network Address Translation

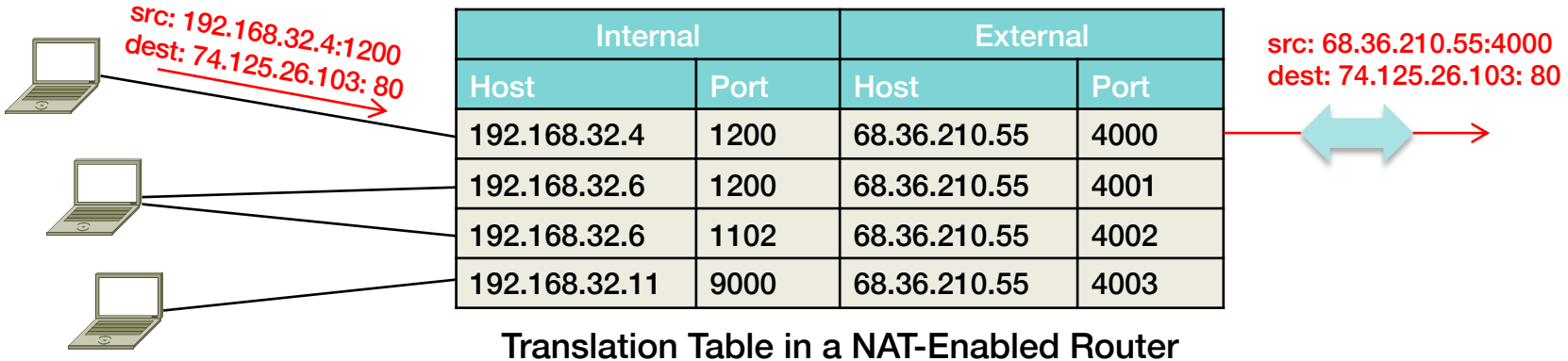
NAT converts between
private (internal) IP addresses and
one or more public-facing (external) addresses

Running out of IP addresses

- **Every device on the Internet needs an IP address**
  - Every address must be unique
    … otherwise, how do you address a host?
- **IP addresses are not plentiful**
  - Does an organization with 10,000 IP hosts really need 10,000 addresses?
  - Prior to 1993, the answer was “YES!”

IP Address Translation

- **Private IP address space in the organization**
  - One external IP address, multiple internal addresses
- **NAT – Translation Table**
  - Map source address:port in outgoing IP requests to a unique external address:port
  - Inverse mapping for incoming requests
- **A NAT-enabled router looks like a single device with one IP address**

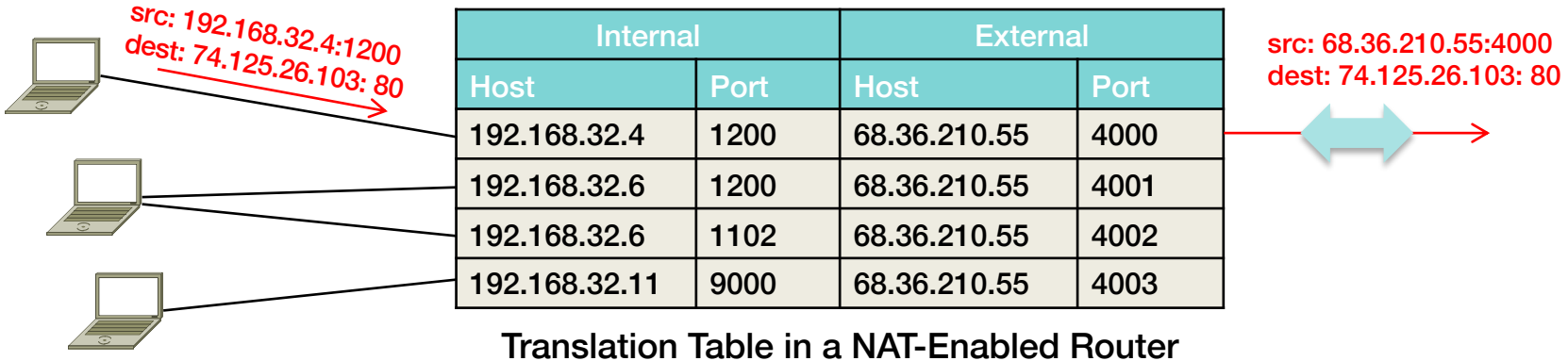


Internal		External	
Host	Port	Host	Port
192.168.32.4	1200	68.36.210.55	4000
192.168.32.6	1200	68.36.210.55	4001
192.168.32.6	1102	68.36.210.55	4002
192.168.32.11	9000	68.36.210.55	4003

Translation Table in a NAT-Enabled Router

IP Address Translation

- **NAT requires a router to look at the transport layer**
  - Source port (outgoing) & destination port (incoming) changes
  - TCP/UDP checksum recomputed



Internal		External	
Host	Port	Host	Port
192.168.32.4	1200	68.36.210.55	4000
192.168.32.6	1200	68.36.210.55	4001
192.168.32.6	1102	68.36.210.55	4002
192.168.32.11	9000	68.36.210.55	4003

Translation Table in a NAT-Enabled Router

Private Addresses

- **We cannot use IP addresses of valid external hosts locally**

  … how will we distinguish local vs. external hosts?

- **RFC 1918: Address Allocation for Private Internets**
  - Defines unregistered, non-routable addresses for internal networks

Address Range	# addresses	IP address block
10.0.0.0 – 10.255.255.255	16,777,216	10.0.0.0/8
172.16.0.0 – 172.31.255.255	1,048,576	172.16.0.0/12
192.168.0.0 – 192.168.255.255	65,536	192.168.0.0/16

Advantages of NAT

- **Internal address space can be much larger than the addresses allocated by the ISP**
- **No need to change internal addresses if ISP changes your address**
- **Enhanced security**
  - A computer on an external network cannot contact an internal computer
    … unless the internal computer initiated the communication
    Even then – it can only contact the computer on that specific port

Network Layer Conversation Isolation:
Virtual Private Networks (VPNs)

Fundamental Layer 2 & 3 Problems

- **IP relies on store-and-forward networking**
  - Network data passes through untrusted hosts
  - Packets can be sniffed (and new forged packets injected)
- **Ethernet, IP, TCP & UDP**
  - All designed with no authentication or integrity mechanisms
  - No source authentication on IP packets – they might be forged
  - TCP session state can be examined or guessed … and then TCP sessions can be hijacked
- **ARP, DHCP, DNS protocols**
  - Can be spoofed to redirect traffic to malicious hosts
  - Man-in-the-middle attacks are possible
- **BGP Internet route advertisement protocols are not secure**
  - Routes may be altered to pass data through malicious hosts

Solution: Use private networks

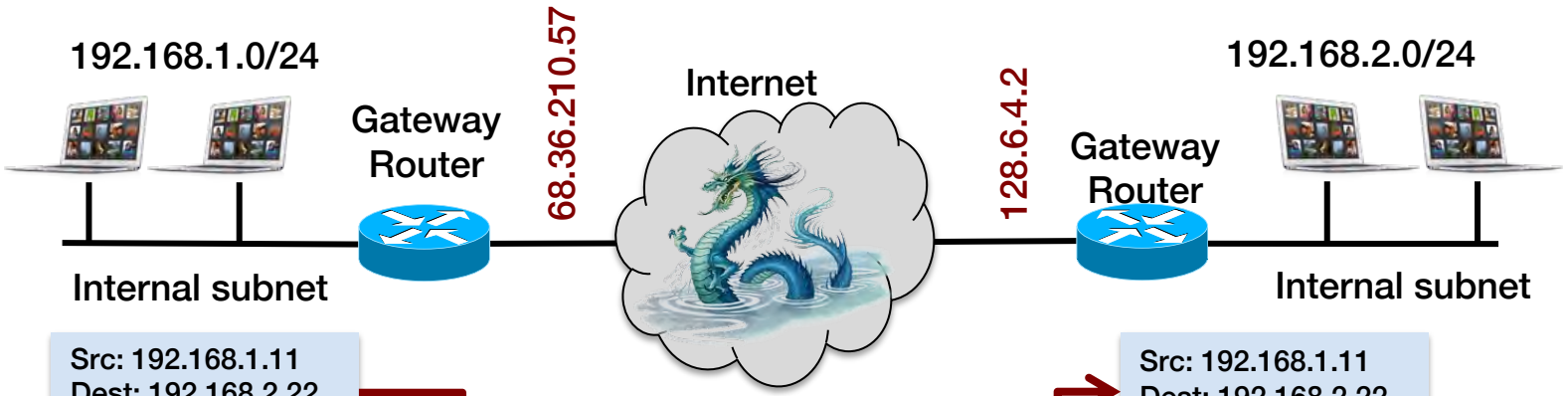
Connect multiple geographically-separated private subnetworks together

But this is expensive … and not feasible in most cases
(e.g., cost, bandwidth, use of cloud servers)

What's a tunnel?

Tunnel = Packet encapsulation

Treat an entire IP datagram as payload on the public network



Tunnel: IP packets to 192.168.2 are encapsulated as data in messages to the router on the public network at 128.6.4.2

Virtual Private Networks

Take the concept of tunneling

… and safeguard the encapsulated data

- **Add a MAC (message authentication code)**
  - Ensure that outsiders don't modify the data
- **Encrypt the contents**
  - Ensure that outsiders can't read the data

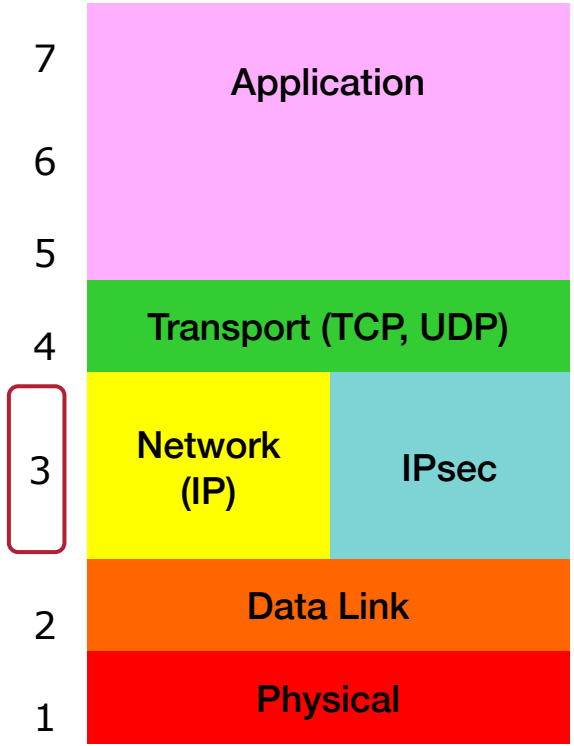
IPsec

Internet Protocol Security

End-to-end security at the IP layer

Two protocols:

- **IP Authentication Header Protocol (AH)**
  - Authentication & integrity of payload and header
  - *Provides integrity*
- **Encapsulating Security Payload (ESP)**
  - AH + encryption of payload
  - *Adds confidentiality*



IPsec is a separate protocol from UDP or TCP – protocols 50 (ESP) & 51 (AH) in the IP header. Layer 3 protocol – gateway routers are responsible for encapsulating/decapsulating

Tunnel mode vs. transport mode

IPsec Tunnel mode

- Communication between gateways: *network-to-network* or *host-to-network*
- The entire IP datagram is encapsulated
  - The system sends IP packets to various addresses on the subnet
  - A router (tunnel endpoint) on the remote side extracts the datagram and routes it on the internal network

IPsec Transport mode

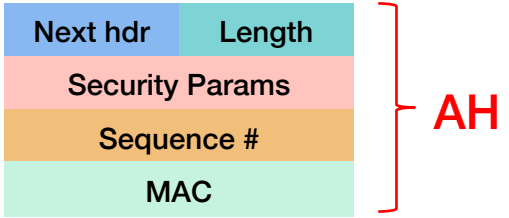
- Communication between hosts
- IP header is not modified
  - The system communicates directly with only one other system

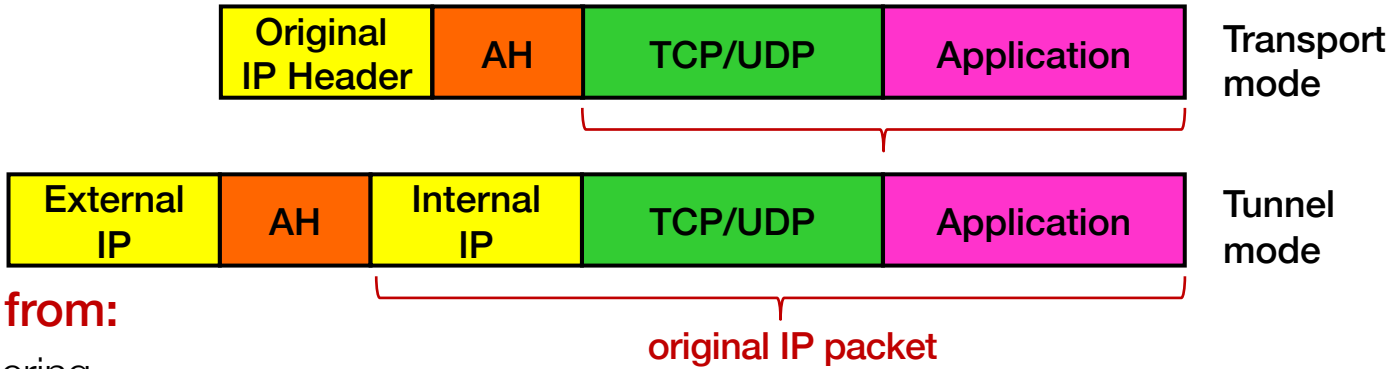
Note: this does not operate at the transport layer – it applies to all IP datagrams between systems or networks, not just a single application

IPsec Authentication Header (AH)

Guarantees integrity & authenticity of IP packets

- MAC for the contents of the entire IP packet
- Computed over unchangeable IP datagram fields (e.g., not TTL or fragmentation fields)





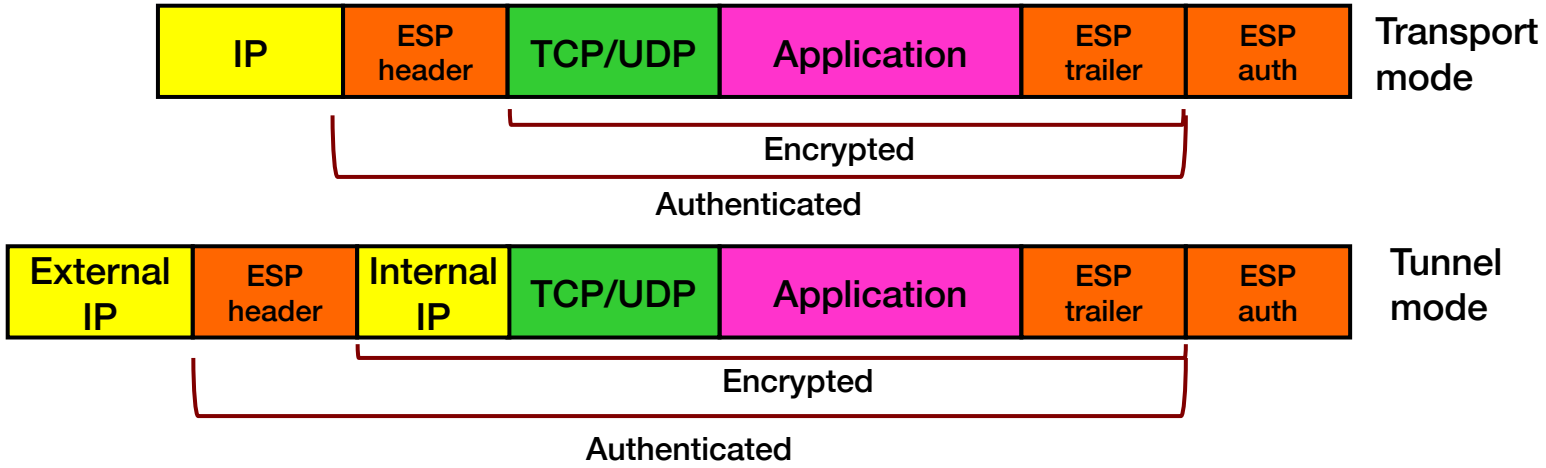
Protects from:

- Tampering
- Forging addresses
- Replay attacks (sequence number in MAC-protected AH)

IPsec Encapsulating Security Payload (ESP)

Encrypts entire payload

– Plus authentication of payload and IP header (everything AH does)
(may be optionally disabled – but you don't want to)



IPsec algorithms

- **Authentication: Certificates or pre-shared key authentication**
  - Public keys in certificates (RSA or ECC) used for authenticating users
    (authenticate by using your private key to decrypt data that was encrypted with the public key in your certificate)
  - Pre-shared keys = authenticate via a shared key that was set up ahead of time
- **Key exchange – Diffie-Hellman**
  - Diffie-Hellman to create a common key for key generation
  - Key lifetimes determine when new keys are regenerated
  - Random key generation ensures Forward Secrecy
- **Confidentiality – symmetric algorithm**
  - 3DES-CBC, AES-CBC, AES-CTR, …
- **Integrity protection & authenticity – MACs**
  - HMAC-SHA1, HMAC-SHA2

Transport Layer Conversation Isolation:
Transport Layer Security (TLS)

Network vs. Transport Layer

VPNs were designed to operate at the network layer

- Connect networks together
- They establish a secure communication channel that can then be shared by multiple applications
- Applications are not aware that the VPN is there

What if applications want to talk securely, such as a web browser & server?

- VPNs aren't an easy answer
- We want to do this at the **transport layer** – for an application talking to a service on a socket

Transport Layer Security

Goal: provide a *transport layer* security protocol

After setup, applications feel like they are using TCP sockets

SSL: Secure Socket Layer

Created with HTTP in mind

- Web sessions should be secure
  - Encrypted, tamperproof, resilient to man-in-the-middle attacks
- Mutual authentication is usually not needed
  - Client needs to identify the server, but the server isn't expected to know all clients
  - Rely on passwords or MFA to authenticate the client after the secure channel is set up

TLS vs. SSL – versions

SSL evolved to TLS (Transport Layer Security)

SSL 3.0 was the last version of SSL
… and is considered insecure

We now use TLS (but is often still called SSL)

- TLS 1.0 = SSL 3.1, TLS 1.1 = SSL 3.2, TLS 1.2 = SSL 3.3
- Latest version = TLS 1.3 = SSL 3.4

Retired versions

- As of the end of 2020, TLS 1.1 & 1.2 (and all older versions) were no longer supported

TLS Goals

Provide authentication (usually one-way), privacy, & data integrity between two applications

Principles

- **Authentication** — *Client should be convinced it is talking with the correct server*
  - Use public key cryptography & **X.509 certificates** for authentication
  - Server side is always authenticated; client optional
- **Data confidentiality** — *Prevent eavesdropping*
  - Use **symmetric cryptography** to encrypt data
  - **Key exchange**: initial keys generated uniquely at the start of each session
- **Data integrity** — *Prevent tampering and man-in-the-middle attacks*
  - Include a **MAC** with transmitted data to ensure message integrity

TLS 1.3 Goals

- **Remove support for older ciphers & hashes**
  - Reduce # of acceptable algorithms & parameters
  - Avoid security risk of **downgrade attacks**

Removed support for SHA-1 & MD5 hashes, DEC, 3DES, RC4, AES-CBC encryption, "export-grade" encryption (shorter keys).

- **Require Diffie-Hellman for key exchange**
  - No longer support RSA public keys; we want Perfect Forward Secrecy
- **Reduce handshake complexity**
  - Assume best-case common protocol options
  - Authenticate all data starting from the first response from the server
- **0-RTT: zero round-trip time for connection restart**
  - Optionally, support near-instantaneous connection resumption
  - After “hello” phase, both sides generate a Resumption Master Key
  - If connection restarts, send a session ticket & data encrypted with Resumption Master Key

TLS Protocol & Ciphers

Two sub-protocols

1. Handshake: authenticate & establish keys

- Authentication
  - X.509 certificates with RSA or Elliptic Curve Digital Signature Algorithm (or pre-shared key)
- Key exchange
  - Ephemeral Diffie-Hellman keys (keys generated for each session)

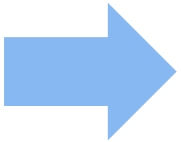
2. Record protocol: communication

- Data encryption options – *symmetric cryptography*
  - AES-128-GCM, AES-256-GCM, ChaCha20-Poly1305
- Data integrity – *message authentication codes*
  - AEAD – Authenticated Encryption with Additional Data – MAC based on selected encryption
  - HMAC-SHA256, HMAC-SHA384

TLS 1.3 Basic Handshake

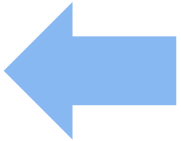
Client

- "Hello"
- Diffie-Hellman public key
- Algorithms/modes



Server

Client



- "Hello"
- Diffie-Hellman public key
- Certificate
- (optional certificate request)
- Proof of private key possession

Server

Both sides now know what algorithms to use & have a D-H common key

TLS 1.3 Key Derivation

- **Both sides have a common key after the handshake**
  - Use that to create all the keys we need – client and server can derive the same sets
- **HKDF - HMAC-based Extract-and-Expand Key Derivation Function (RFC5869)**
  - Specification to create any # of keys starting from one secret key
- **Key Derivation Function**
  - Extracts a fixed-length pseudorandom key, K, from the initial secret:

    hash(non-secret-salt, key)
  - Expands K into any number of additional keys (similar to S/key approach)

    $Key_n = HMAC(K, Key_{n-1}, n)$

TLS 1.3 Communication: AEAD

AEAD: Authenticated Encryption with Additional Data

- **Start with**
  - Message (data to be sent)
  - Secret key
  - Initialization value (IV)
  - Optional additional non-secret data (AD)
- **Use a new derived key for each message**

  Ciphertext = E_{Kn}(message)
- **Create a secondary key for message authentication:**

  HMAC(cipher_text, AD, $length_{cipher\_text}$, $length_{AD}$)

Benefits & Downsides of TLS

Benefits

- Validates the authenticity of the server (if you trust the CA)
- Protects integrity of communications
- Protects the privacy of communications

Downsides

- Longer latency for session setup (only slightly with TLS 1.3)
- Older protocols had weaknesses
  (which is why TLS 1.3 doesn't allow downgrading to weak algorithms)
- Just because a session is over TLS doesn't mean its trustworthy
  - Do you trust the remote side's certificate & that the server hasn't been hacked?

Client authentication Problem

- **TLS supports mutual authentication**
  - Clients can authenticate servers & servers can authenticate clients
- **Client authentication is almost never used**
  - Generating keys & obtaining certificates is not an easy process for users
  - Any site can request the user's certificate – *User will be unaware their anonymity is lost*
  - Moving private keys around can be difficult
    - What about users on shared or public computers?
- **We usually rely on other authentication mechanisms**
  - Usually username and password
  - But there no danger of eavesdropping since the session is encrypted
  - Often use one-time passwords for two-factor authentication if worried about eavesdroppers at physical premises or credential theft (e.g., from the server or phishing attacks)

Some past attacks on TLS

- **Man-in-the-middle: BEAST attack in TLS 1.0**
  - Attacker was able to see Initialization Vector (IV) for CBC and deduce plaintext (because of known HTML headers & cookies)
    - An IV doesn't have to be secret – but it turned out this wasn't a good idea here
  - **Attacker was able to send chosen plaintext & get it encrypted with a known IV**
  - Fixed by using fresh IVs for each new 16K block
- **FREAK**
  - Tricks server into renegotiating a connection with weak RSA encryption keys
- **Man-in-the-middle: crypto renegotiation**
  - Attacker can renegotiate the handshake protocol during the session to disable encryption
  - Proposed fix: have client & server verify info about previous handshakes

Some past attacks on TLS

- **THC-SSL-DoS attack**
  - Attacker initiates a TLS handshake & requests a renegotiation of the encryption key – repeat over & over, using up server resources
- **Heartbleed: vulnerability in popular extension to OpenSSL library**
  - Extension was used to keep the connection alive
    - Client sends payload containing data & the size of the data
    - Server responds with the same message
  - If the client sent false data length, the server would respond with random data
    - That data was memory contents which could include the private key of the server

OpenSSL cert parsing bug causes infinite denial of service loop

BLEEPINGCOMPUTER

Bill Toulas • March 16, 2022

AOpenSSL has released a security update to address a vulnerability in the library that, if exploited, activates an infinite loop function and leads to denial of service conditions.

Denial of service attacks may not be the most disastrous security problem. However, it can still cause significant business interruption, long-term financial repercussions, and brand reputation degradation for those affected.

That is especially the case for software like OpenSSL, a ubiquitous secure communication library used by many large online platforms. Therefore, any vulnerability that affects the library can significantly impact a large number of users.

Certificates causing DoS

In this case, the high-severity OpenSLL problem lies in a bug on the BN_mod_sqrt() function, that if served a maliciously crafted certificate to parse, it will enter an infinite loop.

The certificate has to contain elliptic curve public keys in compressed form or elliptic curve parameters with a base point encoded in compressed form to trigger the flaw.

https://www.bleepingcomputer.com/news/security/openssl-cert-parsing-bug-causes-infinite-denial-of-service-loop/

VPNs and TLS

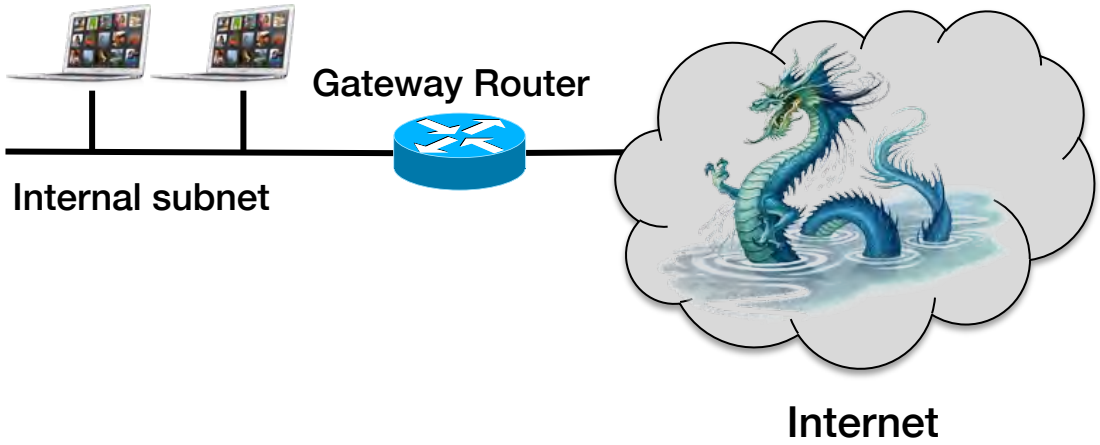
Don't trust the network – manage security at the edges

- **VPNs – Network layer solution**
  - Packet encapsulation
  - Encrypt encapsulated packets & add a MAC
- **TLS – Transport layer solution**
  - Certificate-based public key authentication
  - HMAC for integrity
  - Symmetric encryption for content (AES usually)
    - Diffie-Hellman key exchange for perfect forward secrecy

Firewalls

Network Security Goals

- **Confidentiality**: sensitive data & systems not accessible
- **Integrity**: data not modified during transmission
- **Availability**: systems should remain accessible



Firewall

- **Separate your local network from the Internet**
  - Protect the border between trusted internal networks and the untrusted Internet

- **Approaches**
  - Packet filters
  - Application proxies
  - Intrusion detection / intrusion protection systems

First-Generation Firewalls: Packet Filters

Screening router

Border router (gateway router)

- Router between the internal network(s) and external network(s)
- Any traffic between internal & external networks passes through the border router

Instead of just routing the packet, decide whether to route it

Screening router = Packet filter

Allow or deny packets based on

- Incoming & outgoing interfaces
- Source & destination IP addresses
- Protocol (e.g., TCP, UDP, ICMP, IGMP, RSVP, etc.)
- Source & destination TCP/UDP ports, ICMP command

Filter chaining

An IP packet entering a router is matched against a set of rules: access control list (ACL) or chain

Each rule contains criteria and an action

- **Criteria**: packet screening rule
- **Actions**
  - ***Accept*** – and stop processing additional rules
  - ***Drop*** – discard the packet and stop processing additional rules
  - ***Reject*** – and send an error to the sender (ICMP Destination Unreachable)

  Also
  - ***Route*** – reroute packets
  - ***Nat*** – perform network address translation
  - ***Log*** – record the activity

Filter structure is vendor specific

- **Windows**
  - ***Allow***, ***Block***
  - Options such as
    - Discard all traffic except packets allowed by filters (*default deny*)
    - Pass through all traffic except packets prohibited by filters (*default allow*)
- **OpenBSD**
  - ***Pass*** (allow), ***Block***
- **Linux nftables (netfilter)**
  - Chain types: ***filter***, ***route***, ***nat***
  - Chain control
    - ***Return*** – stop traversing a chain
    - ***Jump*** – jump to another chain (***goto*** = same but no return)

CVE-2021-22555: Turning \x00\x00 into 10000$

Andy Nguyen • July 7, 2021

CVE-2021-22555 is a 15 years old heap out-of-bounds write vulnerability in Linux Netfilter that is powerful enough to bypass all modern security mitigations and achieve kernel code execution. It was used to break the kubernetes pod isolation of the kCTF cluster and won 10000$ for charity (where Google will match and double the donation to 20000$).

Vulnerability

When IPT_SO_SET_REPLACE or IP6T_SO_SET_REPLACE is called in compatibility mode, which requires the CAP_NET_ADMIN capability that can however be obtained in a user+network namespace, structures need to be converted from user to kernel as well as 32bit to 64bit in order to be processed by the native functions. Naturally, this is destined to be error prone. Our vulnerability is in xt_compat_target_from_user() where memset() is called with an offset target->targetsize that is not accounted for during the allocation - leading to a few bytes written out-of-bounds:

https://google.github.io/security-research/pocs/linux/cve-2021-22555/writeup.html

https://nvd.nist.gov/vuln/detail/CVE-2021-22555

Nasty Linux netfilter firewall security hole found

How embarrassing! It turns out there was a security hole lurking in Linux's netfilter firewall program.

Steven Vaughan-Nichols • March 15, 2022

Behind almost all Linux firewalls tools such as iptables; its newer version, nftables; firewalld; and ufw, is netfilter, which controls access to and from Linux's network stack. It's an essential Linux security program, so when a security hole is found in it, it's a big deal.

Nick Gregory, a Sophos threat researcher, found this hole recently while checking netfilter for possible security problems. Gregory explains in great detail his bug hunt, and I recommend it for those who want insight into finding C errors. But, for those of you who just want to cut to the chase, here's the story.

This is a serious bug. Specifically, it's a heap out-of-bounds write problem with the kernel's netfilter. Gregory said it's "exploitable to achieve kernel code execution (via ROP [return-oriented programming]), giving full local privilege escalation, container escape, whatever you want." Yuck!

This problem exists because netfilter doesn't handle its hardware offload feature correctly. A local, unprivileged attacker can use this to cause a denial-of-service (DoS), execute arbitrary code, and cause general mayhem. Adding insult to injury, this works even if the hardware being attacked doesn't have offload functionality!

https://www.zdnet.com/article/nasty-linux-netfilter-firewall-security-hole-found/

https://nickgregory.me/linux/security/2022/03/12/cve-2022-25636/

Network Ingress Filtering: incoming packets

Basic firewalling principle

No direct inbound connections external systems (Internet) to any internal host – all traffic must flow through a firewall and be inspected

- **Determine which services you want to expose to the Internet**
- **Create a list of services and allow only those inbound ports and protocols to the machines hosting the services**
  - E.g., Web server: 10.0.0.10 TCP port 80, TCP port 443
    Mail server: 10.0.0.12 TCP port 587
- **Default Deny model – by default, *deny all***
  - Anything not specifically permitted is dropped
  - May want to log denies to identify who is attempting access

Network Ingress Filtering (inbound)

- **Disallow IP source address spoofing**
  - Restrict forged traffic (RFC 2827)
- **At the ISP**
  - Filter upstream traffic - prohibit an attacker from sending traffic from forged IP addresses
  - Attacker must use a valid, reachable source address
- **Disallow incoming/outgoing traffic from private, non-routable IP addresses**
  - Helps with DDoS attacks such as SYN flooding from lots of invalid addresses

address mask

```
access-list 199 deny ip  192.168.0.0 0.0.255.255  any log
access-list 199 deny ip  224.0.0.0 0.0.0.255  any log
                    ....
access-list 199 permit ip any any
```

Network Egress Filtering (outbound)

- **Usually, we don't worry about outbound traffic**
  - *Communication from a higher security network (internal) to a lower security network (Internet) is usually fine*

- **Why might we want to restrict it?**
  - Consider: if a computer is compromised & all outbound traffic is allowed, it can connect to an external server and download more malicious code
    ... or launch a DoS attack on the internal network

  - Also, log which servers are trying to access external addresses

Second-Generation Firewalls: Stateful Packet Inspection (SPI)

Stateful Inspection – 2nd generation firewalls

Retain state information about a stream of related packets

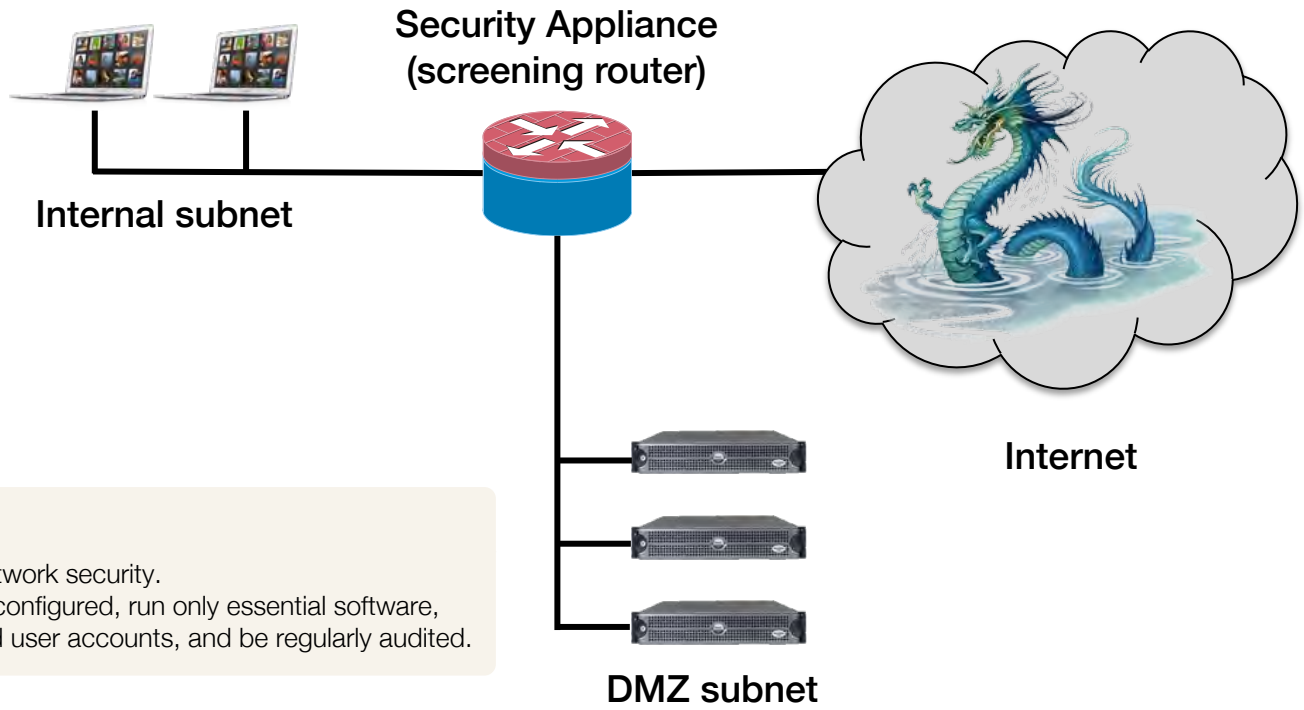
Examples

- **TCP connection tracking**
  - Disallow TCP data packets unless a connection is set up
  - Allow return traffic
- **ICMP echo-reply**
  - Allow ICMP echo-reply only if a corresponding echo request was sent.
- **Related traffic**
  - Identify & allow traffic that is related to a request or connection
  - Example: related ports in FTP
    - Client connects to server on port 21 to send commands
    - Server connects back to client on port 20 to send data

Security Zones

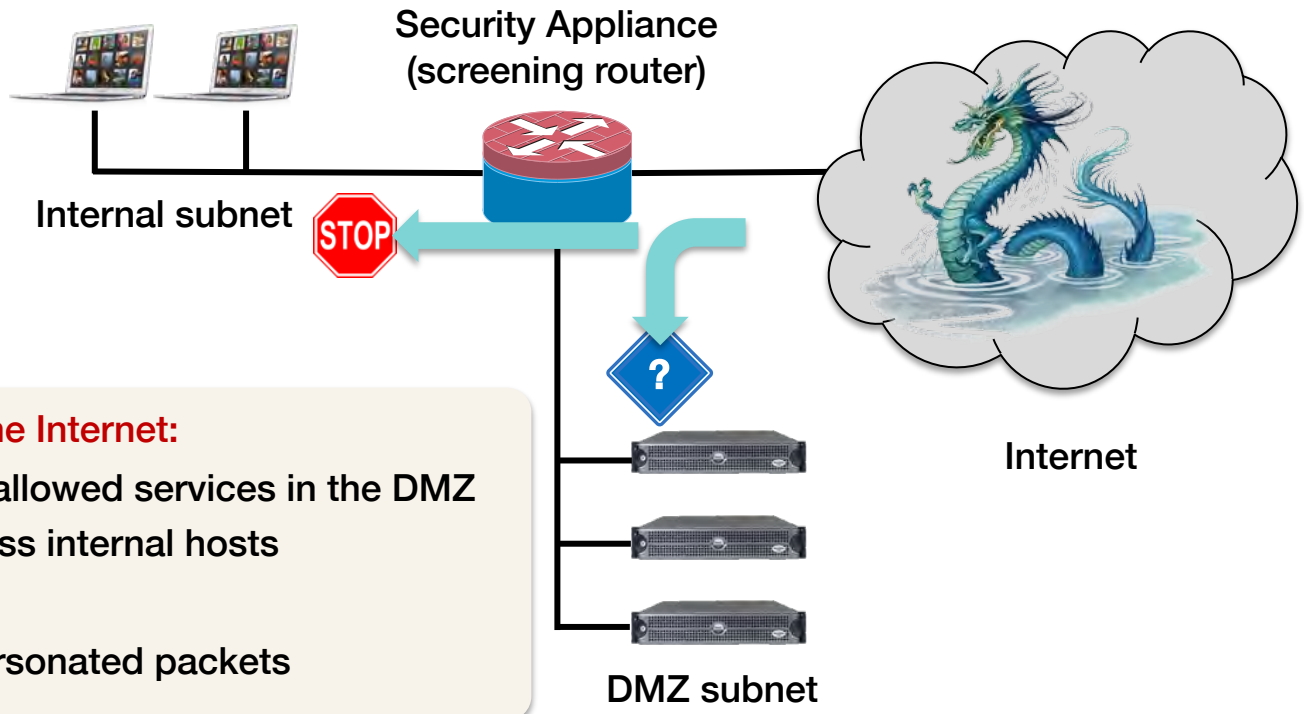
- Packet-filtering firewalls (almost always) live in routers
- Routers connect network zones together
- Firewalls allow you to control traffic between zones

Security Zones: DMZ



Bastion hosts
Systems critical to network security.
They will be carefully configured, run only essential software, have only the required user accounts, and be regularly audited.

Security Zones: DMZ



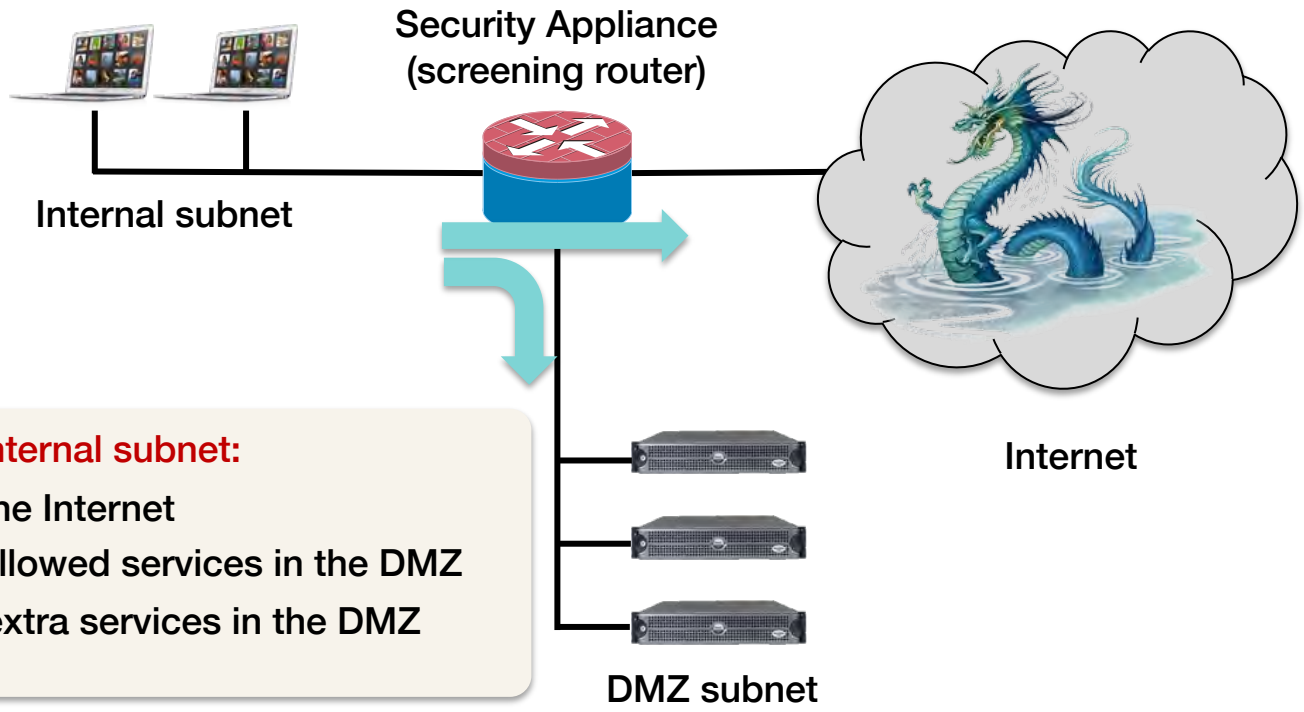
Clients from the Internet:

- Can access allowed services in the DMZ
- Cannot access internal hosts

The firewall:

- Blocks impersonated packets

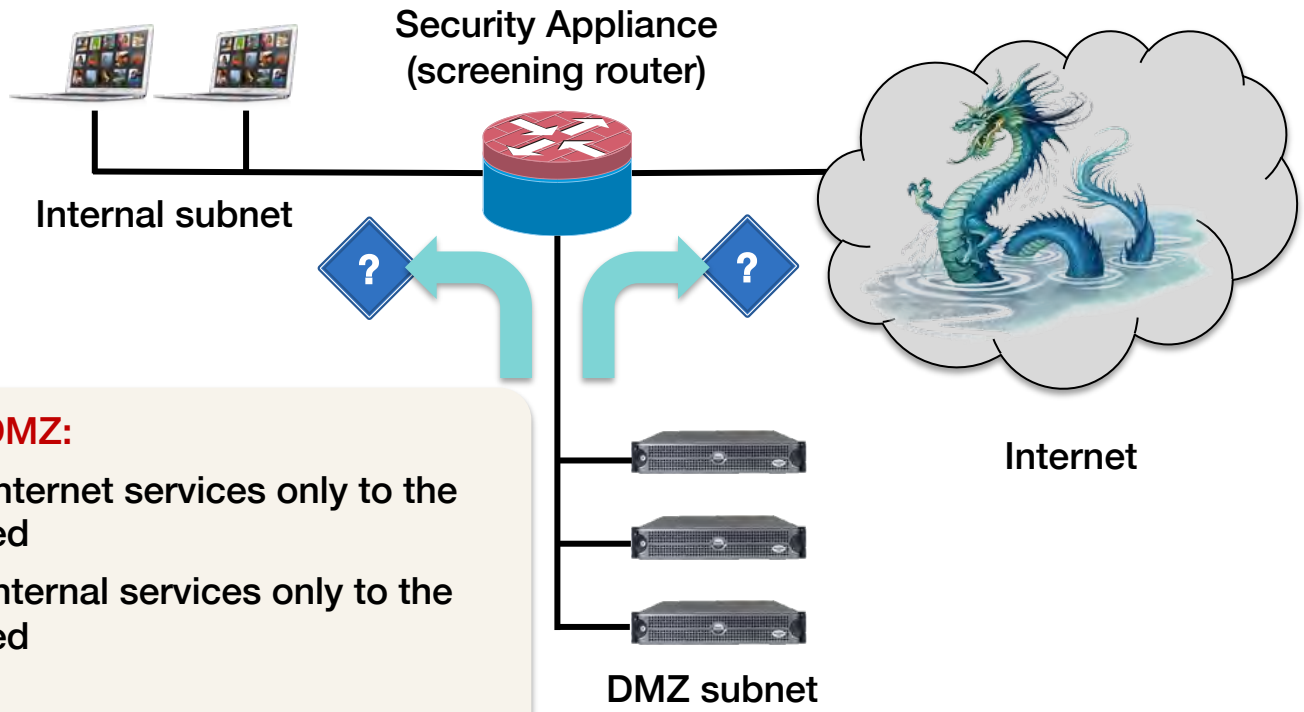
Security Zones: DMZ



Clients in the internal subnet:

- Can access the Internet
- Can access allowed services in the DMZ
- May access extra services in the DMZ

Security Zones: DMZ

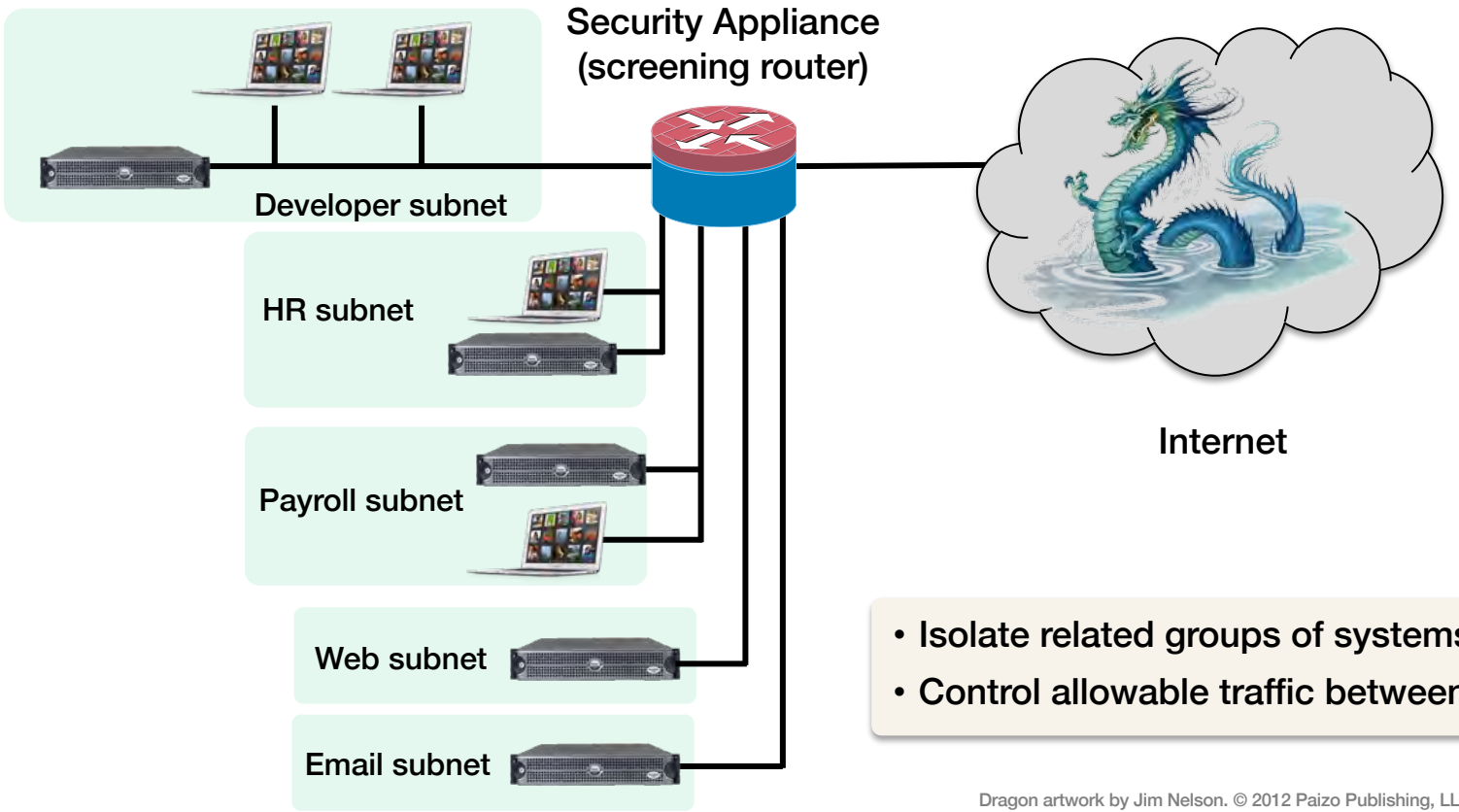


Clients in the DMZ:

- Can access Internet services only to the extent required
- Can access internal services only to the extent required

Goal:
Limit possible damage if DMZ machines are compromised

Security Zones: Segmentation



- Isolate related groups of systems
- Control allowable traffic between the zones

Third-Generation Firewalls: Deep-Packet Inspection (DPI)

Application-Layer Filtering

Firewalls don't work well when everything is a web service

Deep packet inspection (DPI)

- Look beyond layer 3 & 4 headers
- Need to know something about application protocols & formats

Examples

- **URL filtering**
  - Normal source/destination host/port filtering + URL pattern/keywords, rewrite/truncate rules, protocol content filters
  - Detect ActiveX and Java applets, media types; configure specific content as trusted
    - Remove others from the HTML code
- **Keyword detection**
  - Prevent classified material from leaving the organization
  - Prevent banned content from leaving or entering an organization

Design Challenges With DPI

- **DPI matches IP packet data against known bad patterns**
- **This must be done at network speeds**
  - DPI hardware can only hold a limited number of packets for matching
  - DPI hardware can only store a limited amount of malware patterns

Deep Content Inspection (DCI)

Deep Packet Inspection evolves to Deep Content Inspection

- **Deep Packet Inspection systems**
  - Rely on pattern matching and reputation lookup
  - Usually limited to buffering a small set of packets for a stream
- **Deep Content Inspection systems**
  - Unpacks encoded data
    - Example: base64-encoded MIME data in web and email content
  - Signature matching, compliance analysis (including data loss prevention)
  - Behavior analysis via correlation with previous sessions

The difference is largely marketing – focusing on the levels of application-layer inspection that take place

Intrusion Detection/Prevention Systems: IDS/IPS

Intrusion Detection/Prevention Systems

IDS/IPS systems are part of Application-layer firewalls

Identify threats and attacks

IDS: *Intrusion Detection System*

- Monitor traffic at various points of the network and report problems

IPS: *Intrusion Prevention System*

- Sit in between two networks & control traffic between them (like a firewall)
- Enforce admin-specified policy on detection of problems

Types of Systems

– Protocol-based

– Signature-based *We know what is bad; anything else is good*

– Anomaly-based *We know what is good; anything else is bad*

Protocol-Based IDS

Reject packets that do not follow a prescribed protocol

- **Permit return traffic as a function of incoming traffic**
- **Define traffic of interest (filter), filter on traffic-specific protocol/patterns**

Examples

– HTTP inspection: prevent malicious HTTP attacks:
validate headers, cookies, URL string, content types

– DNS inspection: prevent spoofing DNS replies:
make sure they match IDs of sent DNS requests

– SMTP inspection: restrict SMTP command set
… and command count, arguments, addresses

– FTP inspection: restrict FTP command set
… and file sizes and file names

Signature-based IDS

Don't search for protocol violations but for possible data attacks

Match patterns of known "bad" behavior

- Viruses
- Malformed URLs
- Buffer overflows

Need a database of known protocol attacks & malware

- Signature = data segments & order of packets that make up the attack
- Only detects known attacks

Anomaly-based IDS

Search for statistical deviations from normal behavior

Establish baseline behavior first

Examples:

- Port scanning
- Imbalance in protocol distribution
- Imbalance in service access

Challenge

- Distinguish anomalies from legitimate traffic

Next-Generation Firewalls (NGFW)

Term for a firewall that combines

Stateful packet inspection +
Deep packet inspection +
Intrusion prevention

- **Decrypt & re-encrypt TLS & ssh traffic**
  - Breaks end-to-end encryption; firewall is a man-in-the-middle
  - Clients will need to get & validate the firewall's certificate
- **Application awareness**
  - Classify types of apps; assign risk levels & define app-specific policies

Host-based (personal) firewalls

- **Run on the user's systems, not as dedicated firewalls**
- **Application awareness**
  - Manage network-facing effects of malware
    - Allow only approved applications to send or receive data over the network
- ***Important for defense in depth***
- **Problem**
  - If malware gets elevated privileges, it can reconfigure or disable the firewall
- **Personal IDS**
  - E.g., fail2ban on Linux
    - Scan log files to detect & ban suspicious IP addresses
    - High number of failed logins, probes, URLs that try to target exploits

Application proxies

Application proxies

Proxy servers

- Intermediaries between clients and servers
- Stateful inspection and protocol validation

External client

Proxy server

Real server

- Dual-homed host
- Bastion host

Summary

Firewall (screening router)	1st generation packet filter that filters packets between networks. Blocks/accepts traffic based on IP addresses, ports, protocols
Stateful inspection firewall	2nd generation packet filter – like a screening router but also considers TCP connection state and information from previous connections (e.g., related ports for services)
Deep Packet Inspection firewall	3rd generation packet filter – examines application-layer protocols
Application proxy	Gateway between two networks for a specific application. Prevents direct connections to the application from outside the network. Responsible for validating the protocol.
IDS/IPS	Can usually do what a stateful inspection firewall does + examine application-layer data for protocol attacks or malicious content. Usually a part of Deep Packet Inspection firewalls
Host-based firewall	Typically screening router with per-application awareness. Sometimes includes anti-virus software for application-layer signature checking
Host-based IPS	Typically allows real-time blocking of remote hosts performing suspicious operations (port scanning, ssh logins)

Firewall Challenges

Intrusion detection & prevention problems

- **There's a lot of stuff going on**
  - People visit random websites with varying frequencies
  - Software accesses varying services
  - Buggy software may create bad packets
  - How do you detect what is hostile?
- **Attack rates is miniscule … compared to legitimate traffic**
  - Even a small % of false positives can be annoying and hide true threats
- **Environments are dynamic**
  - Content from CDNs or other large server farms has a broad range of IP addresses
  - Malicious actors can coexist with legitimate ones

Intrusion detection & prevention problems

- **Encrypted traffic cannot be easily inspected**
  - Just because you visit a web site using HTTPS doesn't mean the site is secure … or hasn't been compromised
- **Packet inspection is limited**
  - You may need to extract data from multiple packets
  - You may need to reconstruct sessions
  - Both of these are time consuming and can affect performance
- **Threats & services change**
  - Rules must be updated

Deperimeterization

Boundaries & access between internal & external systems are harder to identify

- Mobile systems
- Cloud-based computing
- USB flash memory
- Web-based applications

Zero-Trust Model

Zero Trust Architecture (ZTA), Zero Trust Network Access (ZTNA)

Don't assume everything within your network is secure!

- **Don't allow access to any services until the user is authorized**
  - Enforce the *Principle of Least Privilege*
    - Restrict access and provide permissions only required
  - No device is automatically trusted
- **Rely on multifactor authentication, access control, encryption**
  - Authentication to one resource doesn't mean you have access to others

Challenges with Zero-Trust Access

- **Many apps do not support centrally-managed access control**
  - User authentication credentials, entitlements
- **Networks may need micro-segmentation**
  - Authentication, encryption, and authorization must reach the endpoint
  - Move users or groups of resources into separate network segments so that switches and firewalls can enforce access policies
- **Insider threat is still a problem**
  - So are stolen credentials
  - And compromised devices

The End