Network Security fiche

Pierre Colson

January 2023

Contents

Markdown version on *github*
Compiled using *pandoc* and *`gpdf` script*

TLS

General

- **TLS**: **T**ransport **L**ayer **S**ecurity
- It's goal is to provide a **secure channel between two peers**
- **Entity authentication**
  - **Server** side of the channel is *always authenticated*
  - **Client** side is *optionally authenticated*
  - Via **Assymetric crypto** or a symmetric *pre-shared key*
- **Confidentiality**
  - *Data* send over the channel is *only visible to the endpoints*
  - TLS does *not hide the length* of the data it transmits (but allows padding)
- **Integrity**

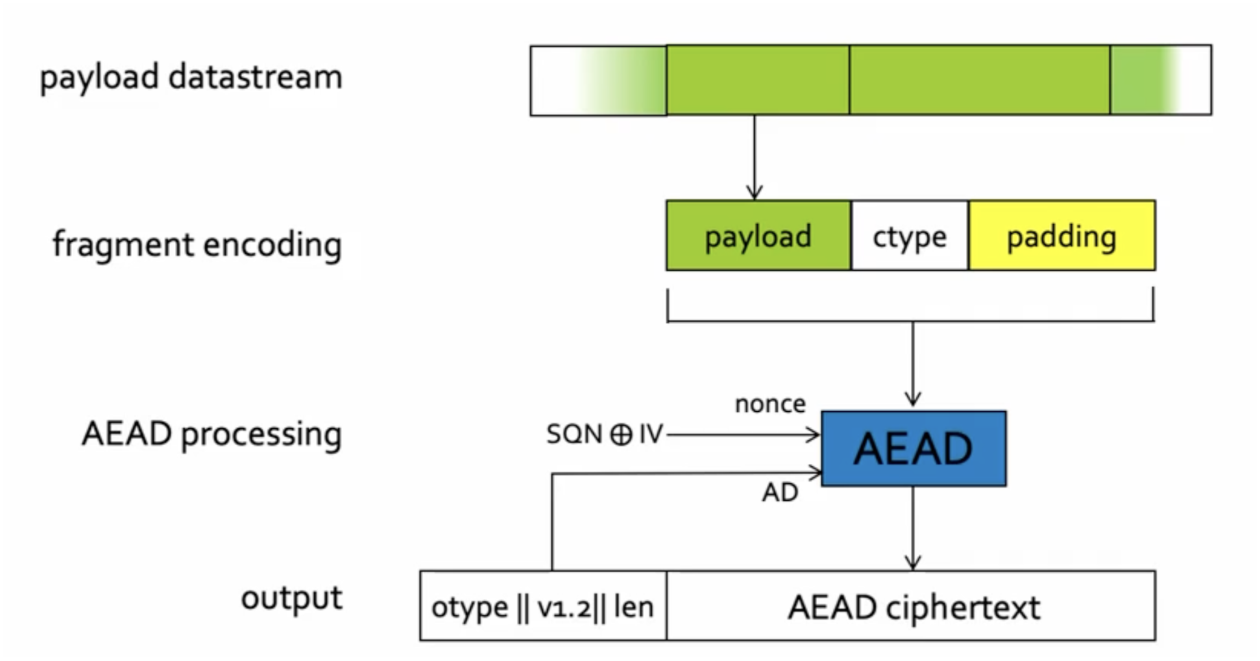
  - *Data* sent over the channel *cannot be modified* without detection
  - Integrity guarantees also cover reordering, insertion, deletion of data
- **Efficiency**
  - Attempt to minimise crypto overhead
  - Minimal use of public key techniques; maximal use of symmetric key techniques
  - Minimise number of communication round trips before secure channel can be used
- **Flexibility**
  - Protocol supports flexible choices of algorithms and authentication
- **Self negociation**
  - The choice is done in hand, i.e. as part of the protocol itself

  - The is done through the version negociation and cipher suite negociation process: the client offers, server selects
- **Protection of negocation**
  - Aim to prevent MITM attacker from performing version and cipher suite downgrade attacks
  - So the cryptography used in the protocol should also protect the hsoice of cryptography made
- **TLS** aims for security in the face of *attacker who has complete control of the network*
- Only requirement from underlying transport: reliable, in order data-stream
- **Handshake protocol**: Authentication, negociation and key agreement
- **Record protocol**: Use those keys to provide confidentiality and integrity
- **TLS 1.3** design process goals
  - *Clean up*: get rid ot flawed and unused crypto & features
  - *Improve latency* : for main handshake and repeated connections (while maintaining security)
  - *Improve privacy*: hide as much of the handshake as possible
  - *Continuity*: maintain interoperability with previous versions and support exisiting important use cases
  - *Security Assurance (added later)*: have supporting analyses for changes
- TLS uses mostrly ‘boring’ cryptography yet is a very complex protocol suite
- Some protocol design errors were made, but not too many
- Legacy support for EXPORT cipher suites and liong tial of old versions opened up seious vulnerabilities
- Lack of formal state-machine description, lack of API specification, and sheer complexity of specifications have let to many serious implementations errors
- Poor algorithm choices in the Record Protocol should have been retired more aggressively
- Most of this had been fixed in TLS 1.3
- TLS 1.3 was developed hand-in-hand with formal security analysis
- The design changed many times, often changes driven by security concerns identified through the analysis
- Cryptography has evolved significantly in TLS
- The largest shift was from RSA key transport to elliptic curve Diffie-Hellman, and from CBC/RC4 to AES-GCM
- A second shift now underway is to move to using newer elliptic curves, allowing greater and better implementation security
- A third shift is the move away from SHA1 in certs
- A future shift is being considered to incorporate post-quantum algorithm
- But Implementation vulnerabilities are bound to continue to be discovered

Record Protocol

- The TLS Record Protocol provides a **stream oriented** API for applications making use of it
  - Hence TLS may fragment into smaller units or coalesce into larger units any data supplied by the calling application
  - Protocol data units in TLS are called **records**
  - So each record is a fragment from a **data stream**
- Cryptographic protectionin the TLS Record Protocol
  - Data origin authentication & integrity for records using a MAC

  - Confidentiality for records using a symmetric encryption algorithm
  - Prevention of replay, reordering, deletion of records using per record sequence number protected by the MAC
  - Encryption and MAC provided simultaneously by use of AEAD in TLS 1.3
  - Prevention of reflection attack by key separation
- *Datastream* is divided in different **payload**
- Each *payload* in concanated with a bit (**ctype**) and an optional **padding**; this give a **fragment**
- This is then given to **AEAD** encryption
  - Needs in input a *nonce*, some *associated data* (AD) (otype, v1.2, and len field) and a plaintext
- **ctype field**
  - Single byte representing content type - indicates wheter contetn is handshake message, alert message or application data
  - AEAD-encryption inside record; header contains dummy value otype to limit traffic analysis
- **padding**
  - Optional features that can be used ot hide true length of fragments
  - Not needed for encryption
  - Sequence of 0x00 bytes afer non-0x00 content type field
  - Removed after integrity check, so no padding oracle issues arise (Time side channel attack to recover lenght on plaintext)
- **AEAD nonce**
  - $Nonce = SQN \bigoplus IV$
  - Constructed from 64 bits sequence number (SQN)
  - SQN is incremented for each record sent on a connection
  - SQN is masked by XOR with IV field
  - IV is a fixed (per TLS connection) pseudorandom value deirved from secrets in TLS handshake protocol
  - IV masking ensures nonce sequence is 'unique' per connection, good for security in multi-connection setting
- **Record header**
  - Contains dummy field, legacy version field, length of AEAD ciphertext
  - Version field is always securely negociated during handshake
  - SQN is not included in header, but is maintained as a counter at each end of the connection (send and receive)



Handshake Protocol

- TLS 1.3: **full handshake in 1 RTT**
  - Achieved via feature reduction: we always do (EC)DHE in one of a shortlist of groups
  - Client includes DH shares in its first message, along with `Clienthello`, anticipating groups that server will accept
  - Server respons with single DH share in its `ServerKeyShare` response
  - If this works, a forward-secure key is established after 1 round trip
  - If server dos not like DH groups offered by client, it sends a `HelloRetryRequest` and a group description back to client
    * In this case, the handshake will be 2 round trips
- **0-RTT handshake** when resuming a previously established connection
  - Client + server keep shared state enabling them to derive a PSK (pre-shared key)
  - Client derives an 'early data' emcryption key from the PSK and can use it to include encrypted application data along with its first handshake message
  - *sacrifices* certain securitty properties
- Because of reliance oc Ephemeral DS key exchange, TLS 1.3 handshake is **forward secure**
- This means: compromise of all session keys, DH values and signing keys has no impact on the security of earlier sessions
- Use of ephemeral DH also means: if a server's long term (signing) key is compromised, then an attacker cannot passively decrypt future sessions
- Compare to RSA key transport option in TLS 1.2 and earlier: past and future passive interception using compromised server RSA private key

Public Key Infrastructure (PKI)

- In symmetric cryptography, main challenge is key distribution as keys need to be distributed via **confidential and authentic** channels
- In public-key system, main challenge is key authentication (i.e., which key belongs to who) as keys need to be distributed via **authentic channel**
- **Public-key infrastructure (PKIs)** provide a way to validate public keys
- **CA**: certificate Authority
- A **public-key certificate** (or simply **certificate**) is signed and dinds a name to a public key
- **Trust anchor, trust root**: self-signed certificates of public keys that are allowed to sign other certificate
- **X.509** strandard format of digital certificate
- **Root of trust** is used to establish trust in other entities
- *Cryptography operations enable transfer of trust from one entity to another*
- Trust roots do not scale to the world
  - *Monopoly model*: single root of trust
    * Problem: world cannot agree on who controls root of trust
  - Obligarchy model: numerous roots of trust
    * Problems: Weakest link security: single compromised enables man-in-the-niddle attack; not trusting some trust roots results in unverifiable entities
- **Let's Encrypt**
  - Goal: provide free certificate based on automated domain validation, issurance, and renewal
  - Based on ACME; Automated Certificate Management Environment
- **Certificate Revocation**
  - Certificate revocation is a mechanism to invalidate certificates
    * After a private key is disclosed
    * Trusted employee / administrator leaves corporation
    * Certificate expiration time is usually chosen too long
  - CA periodically publishes Certificate Revocation List (CRL)
    * Delta CRLs only contains changes
    * What to do if we miss CRL update?

  - What is general problem with revocation
    - CAP theorem (Consistency, Availability, tolerance to partition): impossible to achieve all 3, must select one to sacrifice
- **DANE**
  - DNS-Based Authentication of Named Entities
  - Goal: Authenticate TLS servers without a certificate authority
  - Idea: use DNSSEC to bind certificate to names
- **Certificate Transparency**
  - Will make all public end-entity TLS certificate public knowledge, and will hold CAs publicaly accountable for all certificates they issue
  - And it will do so withou introducing another trusted third party
  - A CT log is an append-only list of certificate
  - The log server verifies the certificate chain
  - Periodically append all new certificates to the append-only log and sign that list
  - Publish all updates of the signed list of certificates to the world
  - A CT log is not a “Super CA”
    - The log does not testify the goodness of certificates; it merely notes their presence
    - The log is public: everyone can inspect all the certificates
    - The log is untrusted: since the log is signed, the face that everyone sees the same list of certificate is cryptographically verifiable
  - How CT improves security
    - Browser would require SCT for opening connection
    - Browser contacts log server to ensure that certificate is listed in the log
  - Consequence
    - Attack certificate would have to be listed in public log
    - Attacks become publicly known
  - Advantages
    - CT is fully operational today
    - No change to domain’s web server required
  - Disadvantages
    - MitM attacks can still proceed
    - Browser still needs to contact Log eventually to verify that certificate is listed in log
    - Current CT does not support revocation
    - Malicious Log server can add bogous certificate
    - Management of list of trusted log server can introduce a kill switch
- **Summary**
  - Cannot tolerate additional latency of contacting additional server during SSL/TLS handshake
  - A key has to be immediately usable and verifiable after initial registration
  - Users shouldn’t be bothered in the decision process if certificate is legitimate
  - Need to cover entire certificate life cycle, including revocation, handing stolen and lost certificate
  - Secure crypto and secure protocols are insufficient
    - Numerous failure possibilities
    - User interface security and certificate management are critically important
  - The entity who controls the root keys, controls all authentication and verification operations
  - PKI and revocation can result in a powerful ‘kill switch’, which can enable shouting down part of internet
    - Sovereign PKI continues to be an important research challenge

Virtual Private Networks (VPNs)

- VPN creates a **Secure channel** between two networks over an **untrusted network**
  - **Set-up phase**: the gateways (tunnel endpoints) *authenticate* each other and *set up keys*
  - **Tuneling phase**:
    - Packets are encapsulated at the first gateway

    - ∗ … and decapsulated at the second
- Simalar security properties as the TLS record protocol
  - Authentication of the source (handshake) data integrity (MACs)
  - Secrecy (symmetric encryption)
  - Replay suppression (sequence numbers)
- VPN setup 1: secure connection between two physically separared networks (site to site)
  - Replace private physical networks and leased lines
    - ∗ Even for leased lines, encryption may be desirable
- VPN setup 2: secure connection of a remote host to company/university network (host to site)
  - Remote host can access resources in private network
    - ∗ Private IP addresses can be accessed without port forwarding
    - ∗ Services do not need to be exposed to the Internet
  - First gateway located at the host
    - ∗ All traffic between host and private network is secure
- VPN setup 3: VPN as a 'secure' proxy (to get a different IP address)
  - Circumvent censorship
  - Avoid trackigng by your ISP or in a public Wi-Fi network
  - Hide your IP address from websites
  - Spoof your location
  - Access restricted content
  - Downloads torrents (only legal ones of course)
- Inportant: VPN provider has access to metadata of all traffic
- $PVN/neqanonimity$
- VPNs provide some limited anonimity properties
  - Local network and ISP only see that you send traffic through some VPN
    - ∗ They do not see which websites you access
  - Web servers do not see you real IP address
    - ∗ Of course, if you use cookies or log in, anonimity is lost
- VPN server can monitor and record all traffic
- Why do we need VPNs when we have TLS?
  - VPNs protect *all* traffic: *blanket* security
    - ∗ DNS requests
    - ∗ Access to services that do not support TLs
  - VPNs can give some access to services in private networks or behind firewalls
- Why do we need TLS when we have VPNs?
  - Data is only secure in the tunnel: no security outside of it
  - VPN server can see all uncrypted traffic → TLS still necessary
  - With a VPN it is not possible to authenticate the webserver, only the tunnel endpoint
- VPNs can *negatively impact performance*
  - Additional cryptographic operations
  - Potential detours
  - Limited bandwidth at VPN server
- Generally, VPNSs do not provide higher availability
  - No build in defense against DoD or routing attack
- VPNs *can defend against targeted packet filtering*
  - Routers can recognize VPN packets but not content
  - Would need to drop all VPN packets
- VPNs themselves can become targets for DoS attacks
- VPN vs **VLAN** (virtual local area network)
  - VPN (securely) *connect/combine* two different networks
    - ∗ One virtual network over multiple physical networks
  - VLAN: set up multiple *isolated virtual networks* on a single physical infrastructure
    - ∗ Virtual networks are identified by tags, which are added to Ethernet frames
    - ∗ Often used in cloud-computing environments for isolating communication betweens VMs

- **Authentication mechanism**
  - Pre-shared key (PSK)
  - Public keys and certificates
  - Client: username/password
- **Tunneling mechanism** (tunnel protocol)
  - Custom protocols (IPsec)
  - Tunnel over TLS (SSTP)
- **Layer of connected networks** (inner protocol)
  - Layer 3 (Network Layer)
  - Layer 2 (Link Layer)
- **Implementation**
  - User space
  - Kernel module
  - Hardware
- VPN creates **virtual network adapter**
- Can be used like any other network adapter
- VPN interface can be used to all traffic or only selectively
- **IPsec** is a very large and complicated protocol
  - A typical IPsec session
    - Set up a security associaction (SA) via IKE
    - Encapsulate packets and tunnel them between SA endpoints
- **Wireguard**
  - No cryptographic agility
    - Only use state-of-the-art primitives
    - Simplify negociation and remove insecure promitives
  - Very simple configuration - similar to `autorized_keys` file in ssh
  - Very small codebase, minimal attack surface, formally verifiable
  - handshake follows the Noise Protocol Framework
    - Built exclusively on (elliptic curve) Diffie-Hellman exchanges
  - Each peer has a *static key* pair
  - Each peer creates *ephemeral key* pair
  - Derive symmetric keys from four Diffie helman combinations
  - 1-RTT handshake
  - Wireguard does not store state before authentication and does not send responses to unauthenticated packets
    - Invisible to attackers
    - Prevent state-exhaustion attacks
  - Initial message contains a timestamp to prevent replay attacks
- VPNs create **secure channels** on the network or link layer
- VPNs and end-to-end security (TLS) **complement each other**
- Many different VPN protocols and applications
  - **IPsec has** a long history and *numerous configuration options*
    - Very versatile but difficult to set up
  - **WireGuard** is a new VPN protocol with a focus on simplicity
    - Very few configuration parameters, no cryptographic agility
    - Simple to set up
    - Small codebase $\rightarrow$ small attack surface

Anonymous-Communication Systems

- IP address leak metadata information
  - Who talks to whom, at what time, for how long, how frequently
  - NSA can log connection metadata, and later incriminate Snowden
- **Anonimity** and related concepts is tricky

    - Anonimity is not a property of individual messages or flows; *You cannot be anonymous on your own*
- **Sender anonimity**
    - Adversary knows/is receiver
    - Adversary may learn message
    - Sender is unknown
    - **Sender anonimity set**
        * Set of all senders/individuals indistinguishable from real sender
        * Can be used as a rough metric
        * Small set $\implies$ little anonimity
    - **Return address** Tolen provided by the sender
- **Receiver anonimity**
    - Adversary knows/is sender
    - Adversary may choose message
    - Receiver is unknow
    - How does destination receive traffic
        * **Onion service** (pseudonym known)
- **Unlinkability**
    - Adversary knows senders
    - Adversary knows receivers
    - Link between senders and receivers is unknown
    - Multiple users need to communicate at the same time
- **Unobservability**
    - Adersary cannot tell whether any communication is taking place
    - Always send traffic
- Plausible deniability
    - Adersary cannot prove that any particular individual was responsible for a message
- **Threat models**
    - There are various types of adversaries that can be considered
    - Degree of control: *local* or *global*
    - Type of contorl: *network* or *compromised infrastructure*
    - Tyoe of behavior: *passive* or *active*
- User multiple proxies to avoit single point of failure (*cascade*)
    - Each proxy only sees addresses of two neighbors
    - Should work if the message addresse traverses at least one honest proxy
- Message and forwarding information is encrypted multiple times (onion)
    - All keys are necessary to decrypt
- **Mix-nets**
    - Intented for sending anonymous emails
        * Latency is not a big concern
        * No connection setup, only individual messages
    - Built on asymmetric cryptography
    - Each mix has a public/private key pair
    - Public keys and addresses are known to the sender
    - Problem: network attacker can observe in and outgoing messages
        * Each proxy should perform **batching**: Collect several messages before forwarding
        * Additionally, the proxies should change the order of (**mixing**) the messages, this is called **threshold mix**
        * Important: messages need to be padded to a *fixed length* to make them indistinguishable
    - To achive full Unobservability, user **cover traffic**
    - How to send reply?
        * Idea: Inlcudes an *untraceable* path return address in its message
    - Problems of mix-nets: high latency dut to batching and mixing; overhead due to asymmetric cryptography

- **Forward Security**: if long term keys are compromised, anonimity of previously establisged circuits is preserved
- **Circuit-based anonimity networks** (onion routing)
  - *Layered encryption*, no batching and mixing, no cover traffic
  - Flow-based: establish a *virtual circuit* (keys) once per flow, reuse it for all packets in the flow using only *symmetric key crypto*
  - The *nodes* are called **relays**
  - The virtual circuit is also called **tunnel**
  - **Circuit setup**
    - Initially, sendre knows long-term public keys or relays
    - The sender negociates shared keys with all relays on the path; this require (expensive) *asymmetric cryptography*
    - The relays store the necessary state
  - **Direct circuit setup**: Establish state on relays by using a normal packet as for mixes
    - Message for each node contains address of next node and ephemeral Diffie-Helman share
    - Each node replies with its own ephemeral Diffie-Helman share
    - Ecnryption of setup packet uses long-term Diffie-Helman share
    - Relatively fast
    - Does not provide (immediate) forward security for long between communication patners
  - **Telescopic circuit setup**
    - Keys are negociated one relay at a time
    - The circuit is 'extended' by one hop at the tine
    - The setup is slower but it offers immediate forward security
  - **Data forwarding**
    - Packets for one or more flows are forwarded along the circuit
    - Only *symmetric cryptography* is used (AES)
  - **Circuit tear-down**
    - The circuit is destroyed to free state on relays or to prevent attacks
    - Can be both by sender and by intermediate ralays
    - Circuits have a limited lifetime, so they will eventually be destroyed

.	Mix-net	Onion routing
Forwarding system	Messag-based	Circuit based
Layered encryption	yes (asymmetric)	yes (symmetric)
Mixing and batching	yes	no
Cover traffic	yes (optional)	no
Forward Security	no	yes (Telescopic setup)
Latency	high	low/medium

- **Tor**
  - Most widely used anonymous-communicatioin system
  - Circuits established over *3 relays*
  - *Telescopic setup*
  - *Per-hop TCP*, established on the fly
    - Avoid TCP stack fingerprints
  - *Per-hop TLS* (except on the last hop)
    - Multiple circuits over the same TLS connection
    - End to end HTPPS is possible
  - Exit policies (exit can restrict the destinations they connect to)
  - **Onion services**
    - Provide receiver anonimity
    - Use `.onion` URL (not in DNS)
    - How can we authenticate the onion service if that wants to be anonymous? The hash of Bob's public key is the identifier of his hiddent service

        - Bod has conenctions to a set of special ralays called *introduction points* (IP)
        - To communicate, Alice connects to an IP and suggest a *rendez vous*
        - Bob can connect to the *rendezvous* and start the communication
    - **Tor cells**
        - Basic unit is the cell (512 bytes)
        - It contains a circuit ID and ac ommand field (cleartext)
        - Same for cells in both directions
    - A *relay* cell's payload is decrypted and its digest is checked
        - If correct (this means the current relay is the intended recipient) check command
        - Otherwise (it is an intermediate node just forwarding the cell): replace circuit ID and forward cell along
        - Only exit relays sees unencrypted payload
    - **Directry authorities**
        - How do the clients know what relays there are?
        - *10 directory authorities* running a consensus algorithm
        - The authorities track the state of relays, store their public keys
        - Client software comes with a list of the authorities's key
        - The centralized authorities are an *important weakness* or Tor
        - Every relay periodically reports a signed statement
        - DAs also act as bandwidth authorities: verify bndwidth of nodes
    - Censorship resistance in Tor
        - Relay nodes are publicaly listed and can be blocked
        - The Tor network contains several *bridge relays* (or *bridges*); not listed in main Tor directory, downloaded on demand; use to circumvent censors which block IP address of Tor delays

Border Gateway Protocol (BGP) Security

- **Rerouting attacks** issues
    - Not all traffic is encrypted/authenticated: DNS, HTTP
    - Even encryptted traffic leaks timing information
    - Rerouting can cause dropped packages and widespread outages
    - Hard to notice and impossible to solve without ISP cooperation
    - Undermine and invalidate other security protocols (can get a fake certificate using acme → TLS becomes useless)
- **IP prefeix origination** into BGP
    - Prefix advertised/announced by the AS who owns the prefix
- **IP prefix hijacking**
    - A malicious (or misconfigured) AS announces a prefix it does not own
    - Today, no proper verification in place
- **BGP does not validate the origin of advertisements**
- **BGP Interception**
    - Selectively announcement of hijack prefix only to some neighbors
        - Problem: neighbors may still learn hijacked routes from their peers
    - Use **BGP poisonning**
        - Only some of the neighbors use hijacked route
    - Use BGP communities to ensure the announcement only reaches certain ASes
        - Can tell ans AS not to forward announcement to specific other ASes using the 'NoExportSelected' action
    1. Set up an AS and border router or compromise someone else's router
    2. Configure router to originate the target (sub-)prefix
    3. Get other ASes to accept the wrong route
- **BGP does not validate the content of advertisements**
- ASes can modify the BGP path
    - *Remove ASes from the AS path*; Motivation:

        - Attrack traffic by making path look shorter
        - Attrack sources that try to avoid a specific AS
    - *Add ASes to the AS path*; Motivation
        - Trigger loop detection in specific AS (DoS, BGP poisonning)
        - Make your AS look like it has richer connectivity
- *Security Goal*
    - Only an AS that owns an IP prefix is allowed to announce it
        - Can be proven cryptographically
    - Routing message are authenticated by all ASes on the path
        - Cryptographic protection
        - ASes cannot add or remove other ASes in BGP announcements
- Applying **Best Current Practices** (BCPs)
    - Securing the BGP peering session between routers (authentication, priority over other traffic)
    - Filtering routes by prefix and AS path
    - Filters to block unexpected control traffic
- Enter prefices into Internet Routing Registries and filter based on these entries
- **Resource Public Key Infrastructure (RPKI)**
    - *Required*: ability to prove ownership of resources
    - RPKI cryptographically asserts the cryptographic keys of ASes and the AS numbers and IP prefixes they own
    - Root of trusts are ICANN and the five regional Internet registries
    - Enables the issuance of *Route Origination Authorizations* (ROAs)
    - ROA can states which AS is authorized to annouce certain IP prefixes
        - Can specify the maximum length of the prefix that the AS is allowed to advertise → avoid sub-prefix hijacking
        - Certificates follow same delegation as IP addresses from RIRs
    - ROAs are signed, distributed, and checked out-of-band
    - *Distribution of ROAs*
        - ASes and/or RIRs create ROAs and upload them to repositories
        - Each AS periodically fetches repositories
        - All BGPs routers of an AS periodically fetch a list of ROAs from the local cache
        - When a BGP update message arrives, the router can check wheter a ROA exisits and it is consistent with the first AS entry of the BGP message
- **BGPsec**
    - Secure version of BGP
    - Secures the AS-PATH attribute on BGP announcements
    - Idea: Origin authentication + cryptographic signatures
    - Include Next AS in the signature so that both ASes confirm the link between them
    - Path prepending is no longer possible
    - *Problems*
        - Routing policies can interact in ways that can cause BGP wedgies
        - Still vulnerable to protocol downgrade attacks
        - Performance degradation
    - Unless security is the first priority or BGPsec deployment is very large, security benefits from partially deployed BGPsec are meager
    - Deployement is challenging
- **BGP** was not designed with security in mind
- **SCION** Scalability, Control, and Isolation on Next Generation Networks (Replacement of BGP)
- *"BGP is one of the largest threats on the internet"*
- Proposals to improve BGP or competely replace it are emerging, but large-scale deployment is difficult

Firewall, Intrusion Detection and Evasion

- A **Firewall** is a system used to protect or separate the *trusted network* from an *untrusted network*, while *allowing authorized communications* to pass from on side to the other
- **Firewall** enforce an access control policy between two networks
- **Network firewall** are a software appliance running on specitfic hardware or as virtual instance taht filter traffic between two or more network
  - *Protect differect network segement*
- **Host firewall** provide a layer of software on one host that controls network traffic in and out of that single machine
  - *Protect single host*
- **Ingress**:
  - Filter incoming traffic
  - From low security to high security net
- **Egress**:
  - Filter outgoing traffic
  - From high security to low security net / outside
  - Gets often forgotten
- **Default policy**:
  - Define what to do when no rule matches
  - *Default accept* versus *default reject* policy
- **Deny Access**:
  - Techniques to deny access
    - *DROP*: Silently drop the packet
    - *REJECT*: Drop packet and inform sender
- **Stateless Firewall**
  - Examine a packet at the network layer
  - Decision is based on packet header information
  - *Pros*:
    - Application independant
    - Good performance and scalability
  - *Cons*:
    - No state or application context
- **Statefull Firewall**
  - Keep also track of the state of the network connections
  - Decision also based on session state
  - *Pros*:
    - More powerful rules
  - *Cons*:
    - State for UDP?
    - Inconsistent state: Host vs Firewall
    - State explosion
- Enforcing access control is not enough
- Firewall can't block all malicious traffic
  - Many ports must be kept open for legitimate applications to run
  - Users unwittingly download dangerous applications or other forms of malicious code
  - Peer-to-peer and instant messaging have introduced new injection vectors
- The lagacy firewall technology is effectively blinded by this evolution
- **Next Generation of Firewall** (NGFW)
  - Deep packet (content) inspection
  - Take aplication and protocol state into account
  - *Pros*:
    - Even more powerful rules
    - Application & protocol awarness

  - *Cons*:
    * Need to support many application protocols
    * Performance, scalability
    * Inconsistent state: host vs firewall
- **Web Application Firewall** (WAF)
  - Protect web-based applications from malicious requests
  - *Request filtering*
    * Request patterns (signature)
    * SQL injection, cross-site scripting, buffer overflow atttempts, checking number of form parameters
    * Static or dynamic blacklisting/whitelisting
  - *Authentication*
    * User authentication
    * Session management
  - *TLS Endpoint*
    * WAFs are often implemented as a reverse proxy to protect public facing web applications
    * Reverse proxy: client is outside the internal network
- **Deployemnt Challenges**
  - **Scaling**: protecting large number of hosts, endpoints, network segments is not trivial
  - **Complexity**:
    * Firewall rulesets are complex and grow over time
    * Thousands of rule on a single firewall are no exception
    * Detection between channels of sync
  - **Management**:
    * Tools are needed to mange hundreds of firewalls and their rules securely
    * What is the process to change rulesets
    * Who has permission, monitoring of changes
  - **Incentives**:
    * Infrastructure Team: paid for providing connectivity blamed for disruptions
    * Security Team: paid to protect and disrupt connectivity
- **Firewall attacks**
  - Firewalls are just complex machine, with vulnerabilities and assumptions
  - **IP Source spoofing**: Spoofing the IP address to bypass filters (Works for stateless protocols)
  - **Artificial Fragmentation**:
    * Fragment packets to bypass rules
    * Without proper reassembly at the firewall the atteck gets through
    * Attack sent in multiple packets
    * Out of sequence packets
    * Fragmentation overlap
  - **Vulnerabilities**:
    * Exploiting vulnerabilities in firewall software/firmware/OS
    * Exploit vulnerabilities in target application
  - **Denial of Service**: Firewall state explosion
    * Data in ICMP ping packets, or use DNS requests as channel
    * Attack through VPN
- **Payload encoding**
  - Different encodings or mapping confuse detection
  - Different encoding and obfuscation schemas can be combined with noise insertion in millions ways
  - Encoding are not necessarily unique
  - Undefined or border cases are very effective for detection evasion
  - Different implementation of decoding on target application vs detection engine decoding
- Being a security product from a security vendor does not imply better code quality
- **False Positive**: I raised an alert but it was nothing
- **Attack Detection**

  - Basic detection approaches
  - **Reactive**: System can only detect already known attacks
  - **Proactive**: System can detect known and new, yes unknown attacks
  - **Deterministic**:
    * System always performs the same given the same input
    * The same stimuli always result in same action
    * Reason for alert in known
  - **Non-deterministic**
    * System detection is fuzzy (heuristic, machine learning, sandboxing) and depends on current state of the world
    * Reason for alert typically not known
  - **Static**
    * Signature based detction
    * Static analysis and identification of known malware
    * Create and distribute signature of known malware
    * *Pros*: Reliable, low rate of false positives
    * *Cons*: Reactive, only known malware can be detected; Problem of large scale distribution of signatures
  - **Behavior**
    * Catatlogue and identify suspicious behavior
    * Run samples in sandbox/device
    * Classify upon observed behavior
    * *Pros*: Proactive, detection on inknown malware
    * *Cons*: Complex and computationally expensive; higher rate of false positives
  - **Protocol analysis**:
    * Analysis and decoding of protocols
    * Reassembly and normalization of traffic
  - **Signatures**: Compare attributes of observables
    * to a deny list or allow list
    * to patterns of known attacks/exploits/malware
  - **Sandboxing**:
    * Suspicious file is executed within a virtual environment
    * Specific actions are categorized and labeled as good or bad
  - **Machine learning**
    * Recognize complex patterns
    * Make decisions based on the data and assumptions formed from previous data
- **Signature based detection**
  - Deterministic and non-proactive
  - Signature-based concepts still lie at the heart of all modern detection systems … and will continue to be integral for the foreseeable future
  - Able to promptly identify and label a threat
  - Different signature systems used together to accurately label a known threat
  - Different signature systems used together to accurately label a known threat
  - For each new threat, a unique signature needs to be created
  - Frequent updates to signature database or online lookups
  - Progression and sophistication of signature-based detection systems depend upon human signature writers
- **Sandboxing Based Detection**
  - Run malware in detonation chamber
  - Proactive and mostly deterministic
  - Sandoxing product typically run a samples in a (instrumented) sandbox environment
  - Examine/monitor runtime behavior of sample
  - Compare the behovior agains a list of rules previously developed by the vendor in their lab and applly machine learning for behavior classification

  - *Pros*:
    - Proactive, can detect unknown threats;
    - No signature updates required
  - *Cons*:
    - Resource intensive
    - High latency
    - Difficult to scale
- **Machine learning**
  - Proactive and partially deterministic
  - Security vendors are now applying increasingly sophisticated machine learning elements into their cloud-based analysis and classification systems, and into theri products
  - *Proven*: These techniques have already proven their value in Internet search, targeted advertsing, and social networking
  - *No humans*: Machine learning largely removes humans and their biases to the developmen of an dimesional signature (or classification model)
  - Machine learning estimates probability distribution it approximates from training data
  - *Golden rule*
    - Data you are going to work on needs to come from approximately the same distribution as the data you are training on
- **Detection Evasion by Design**
- **Malware Developtment Lifecycle**
  - Develop new malware with desired functionality
  - Automatically create numerous permutated samples of the initial malware at massive scale
  - Protect samples from analysis
  - Make samples aware of sandboxing/detection technologies
  - Quality Assurance: Test sample against current anti-malware solutions before deployment
  - Malware used in a target attack will not be detected by anti-malware tools at the time of attack
  - Because it was tested for detection beforehand
- **Polymorphism Techniques**
  - Tools manipulate the structure of the source code of the malware by reordering and replacing common programmatic routing
  - Swapping of equivalent code constructs
  - Changing the order of the code
  - Inserting noise
  - Compiler modulation
- Challenges and limitations of **intrusion detection**
  - Unable to inspect encrypted traffic
  - High number of false positive
  - Packet capturing and analysis at high link speed
  - Latency introduced by inspection engine
  - Application level attack
  - Policy/signature management
- **Accuracy**: how close the measured values are to the target value
- **Precision**: Values of repeated measurements are clustered and have little scatter
- "It is better to be roughly right than precisely wrong" *John Maynard Keynes*
- **True Positive**: alert on true intrusion
- **False Positive**: alert but no intrusion
- **False Negative**: Intrusion but no aler
- **True Negative**: No alert on non-intrusion
- Perfect detection
  - Difficulty: Built a detector with optimal balance between FPs and FN
  - Costs of detection errors
    - False Positive: Mobilize incident response team, stop service, interrupt business
    - False Negative: Getting compromised, forensics, downtime, cleanup

- Detection Performance
  - A high number of false positives is a major challenge for detection system
  - Accurate detection is very challenging when rate of attacks is very low
  - Different detecton techniques achieve different sensitivity and specificity
  - Vendors claims to detect almost 100% of the sample are meaningless without indicating rate of false positives
  - Use a combination of diverse tests to increase precesion
  - Context is important
- **What to remember**
  - 100% protection is an illusion: assume you're already compromised, and get compromised over again
  - A successful breach shoud be planned for, and handled in controlled process (rather than being treated as an execption)
  - Deploy tools and processes to quickly detect and remediate successful breaches
  - We need anti-virus, anti-malware, intrusion detection et al, butl we also need to know their limitations
  - Most devices can achieve a high detection rate
    - At the price of an unacceptably high rate of false positives
  - *An installed security patch/update is better than millions of detection signatures*

Botnets & Internet of Things (IoT)

- The word **botnet** is a combination of the words 'robots' and 'network'. The term is usually used with a negative or malicious connotation
  - **Numerous devices**: Build and manage a large network of compromised machines or devices
  - **Command and Control** (CnC): Controlling a massive number of bots is a key requirement and challenge
- **Bot agents**
  - *Targets*
    - Criminals compromise an array of victim computers
    - A *boot agent* is installed on every victim/target
  - **Challenge**: how to communicate with the bot master
    - To receive commands and new payloads
    - To exfiltrate data
- **Proxy layer**
  - A distributed and redundent layer of *proxy agents*
  - *Proxy agent* are itself compromised machines
  - *Bot agents* connects to proxies to:
    - Receive commands
    - Downloads payloads
    - Exfiltrate data
- **Bot master**
  - Control server
    - The *bot master* connects to **crntrol server**, which communicate with proxy agents
    - *Asynchronous communication* between bot master and control server
  - *Drop Zone*: Host where exfiltrated data is stored until pickup by bot master
- **Botnet Command Models**
  - *No Control*
    - Default malicious behavior
    - Less flexible, detected by signatures
    - Most resistant to global shutdown
  - *Public infrastructure*
    - Use common application API's
    - Generally reliable and 'anonymous'

        - Mostly IRC, some P2P & micro blogging
        - Majority of today's botnets
    - *Resilient hybrids*
        - Default malicious behavior
        - Fallback plans id CnC unavailable
        - Pre-programmed contact points
    - *Private channels*
        - Custom and covert channels
        - Abuse & alteration of common protocols
        - Short-term stealth
        - Signature detection easy once CnC observed
- Locate the Command and Control Infrastructure
    - The ability of a bot agent to locate the CnC infrastructure is a critical requirement for maintaining control of the entire botnet
    - A *bot agent* that cannot connect to the control infrastructure cannot be controlled
    - The bot has to somehow identify the CnC infrastructure
    - The CnC communication can be intercepted by competitors and/or law enforcement to
        - Shutdown ot take over the botnet
        - Identify the bot master
- **Fast Flux**: A CnC resource with a given Fully Qualified Domain Name (FQDN) is mapped to a new set of IP addresses as often as every few minutes
- Key methods for robust communication
    - Typical methods for a bot agent to locate and connect to a command and control instance or the bot net
    - Use *fixed ip*:
        - Easy to identify bot agent
        - Easy to identify controller
        - No flexibility
        - Easy to block botnet
    - Use *fixed domain*:
        - Easy to identify bot agent
        - Harder to identify controller
        - Harder to shut down botnets
        - More flexibility
    - Use *IP or domain flux*
        - Domain names and/or IP addresses change frequently
        - Very dynamic, moving target
        - Hard to shut down botnet
- **Sinkholing**: A botnet defense
    - A technique that is used to redirect the traffic from bots to an analysis server
    - A **Sinkhole** server gathers analytics and controls bots (if the authentication is also reverse engineered)
    - Reverse engineering of infected machines enables security researchers to replicate the DGA
    - This allow the identification and registration of some of the 'rendez vous' domains, and thereby redirect traffic of infected bots to the sinkhole server
    - Sinkhole server gather valuable telemetry and control bots (if the authentication is also reverse-engineered)
- Bot configuration files control key bot agent functionality on the target system
    - **Blacklists**: An array of domains that the botnet permanently blocks on the target host
        - Prevent the victim from automatically, or manually, updating the machien or any anti-malware solutions
        - The victims machine can no longer get security protection
    - **Web Injects**: Specific HTML code injectd into victims browser session to exfiltrate sensitive data
        - When accessing a site of interest, the attackers html code is injected into the session

        - Exfiltration of user names & passwords upon authentication, or hidden inserstion of fraudulent transactions

- **Safety** is the protectionm against random, unwanted incidents - resulting from coincidences or driven by the environment
    - The environment does not adapt to bypass safety measures
- **Security** is the protection against intended incidents - resulting from a deliberate and planned act
    - Driven by targeted attacker
    - Deliberate acts driven by an adaptative attacker
- *When is the last time you updated your connected fridge or your printer ?* → problems
- No perceptiohn of risk
    - People need highly visible incidents before they act
    - Insecure systems cannot be identified without extensive testing → Illusion of control
    - We face considerabe difficulty to get resources (from C-level) to protect against abstract risks → Accumulation of risks
- **Internet of Things** (IoT)
    - System of interrelated and connected computing devices
    - Global network of 'smart' physical objects of various kinds for monitoring, data gathering, reporting, remote control etc. . .
    - Ability to transfer data without requiring interaction
- *Industrial Internet of Things* (IIoT): Subset of IoT specific to industry
- *Critival Infrastructure* (CI): Processes, facilities, technologies, networks and systems that control and manage essential services
- **Operational Technology** (OT) and **Information Technology** (IT) systems have different operational requirements which impact their ability to respond and adapt to these threats
    - **Operational Technology**
        - High availability & integrity are vital with less stringent confidentiality requirements
    - **Information Technology**
        - Confidentiality & Integrity are vital while availability is important
- Security issues in IoT because their are such small devices that it is not always possible to fulfill the security requirement (e.g. not enough computatiom power for cryptography or random number generators)
- **IoT attack surface**
    - IoT connects innumerable everyday devices and Systems
    - *Device*
        - Insecure software & defaults
        - Lacking update mechanism
        - Vulnerabilities
    - *Communication*
        - Insecure communication
        - Weak or no cryptography
        - Lack of authentication
    - *Backend service*
        - Central control
        - Erosion of privacy
        - Data breaches
    - Attacker perspective: IoT are easy target, way more than computer
    - *Personal Computer*
        - Networked and *continuously hardened* in battle
        - Designed to withstand *external threats*
        - *Secure defaults*
        - Exploit mitigation
        - *Frequent secutity updates*
    - IoT, IIoT & ICS devices
        - Ran isolated for decades

        - Designed for *high availability* and *safety*, *not secutity*
        - *Insecure defaults*
        - Old code, *no protection*
        - *No security updates*
- Cyber devices require continued 'security maintenance'
    - IoT devices often does not have a convinient way to be updated
- Traditional products rarely change after delivery whereas digital products constantly require security updates
- Lifespan of decades for embedded or IoT devices
    - Replacement of devices is very difficult or prohibitively expensive
    - Options before purchase
        - **Code Escrow**: Agree to have a copy of source code deposited with trusted third party
        - **Open source**: Mandate to open source code in case of manufacturer exiting market
- Many digital devices have to be certified to be used
    - By applying a security patch the certification is lost, further use of the device is illegal
    - Current (re)certification cannot keep up with dynamic security requirements
    - *Certification timeline is outpaced by cyber security, and industry wide challenge*
    - You're doomed if you patched - you're doomed if you don't

Challenges	**What is needed**
The security of individual components *does not imply the security of the complete system*	• Design systems with *redundancy and resiliency* • Realistic testing for the *complete system (end-to-end)*
The continued innovation of attacker, treats technologies, society, and use-cases creates *a danamic and adaptative threats* landscape	• Prepare *for continued adaptation* to new threats, no matter what the driver or domain behind the attacker or threat • Comprehensive and continued security monitoring
Softrware drives everything, and there is no such thing as a secure software. Prepare for continued *discovery and publication of publication of vulnerabilities in sotware and hardware*	• Active *management of vulnerabilities* • Robust and *scalable process to deploy security updates* timely and effeciently on *any connected device*
We depend on a *complex supply chain of hardware and software* components, which *can not be fully controleld*. Assume that some components are already compromised	• Systematic *security and integrity testing* of all crtitical components • Comprehensive security appendix in supplier contacts

DNS Security and Privacy

- **D**omain **N**ame **S**ystem (DNS)
    - A distributed global lookup mechanism for translating names into other objects
    - Most used and least understood protocol in the internet
    - DNS is a mission-critical component for any online business
    - Yet this component is often overlooked and forgotten, until something breaks
    - DNS is the phone book of the internet: it tells computers where to send adn retrieve information
    - DNS is a *globally distributed*, loosely coupled, scalable, reliable, and dynamic database
    - DNS data is *maintained locally* and *retrievable globally*
    - No single computer has all DNS data

  - **Namespace**: hierarchical namespace defines names and ownership
  - **Resolvers**: Clients to query servers
  - **Protocol**: simple protocol, TCP or UDP port 53, no built in encryption, integrity, authentication
  - **Server**: Make namespace available and store data DNS security and privacy [SF]
- **Security**
  - Client expects `txid` to match (else drops response)
  - Random `txid` introduces 16 bit of entropy
  - If source port is random, mac 16 bit of entropy
  - *No confidentiality*
  - *No integrity verifivation*
  - *No authentication*
- Why attack DSN? Control DNS resolution of all clients server name server/resolver
- Why study DNS security?
  - Use DNS to explain attack classes irrespective of protocol
  - Understand security impact of specific features in protocol design or implementation
- What could possibly go wrong? Cache poisonning, replay & amplification, session state, dependencies, org structure, ... DNS security and privacy [SF] DNS security and privacy [SF]
- **DNS Root Zone**
  - DNS root name server are the key to the internet kingdom
  - **Root server** only resolve the IP addresses for the top level name servers
  - Every name resolution on the Internet either start with a query to a root server of, uses information that was once obtained from a root server
- Key challenge for root Server System: *Denial of Services attacks*
- **Cache poisoning**
  - Attacker inserts incorrect resolution information
  - *Assumption*
    - The attacker is off path
    - If required, the attacker could make client to resolve a FQDN the attacker controls
  - *Prerequisite* to inject a fake response, the attacker needs to
    - Raply faster - before true response arrives
    - Guest relevant parameters of query to get attacker reply processed
  - Response is only processed with matching TXID of the query
  - Attacker has to wait for TTL for next attempt
  - Response must match source IP/port of query
  - Attack method
    - Attacker controls authoritative names server and a domain *D*
    - Attacker tricks user to resolve *D* (hacked site, mail, social media, etc...)
  - Attacker Execution
    - Client resolves *D*
    - Name server replies with an additional section in DNS response, adding unrelated information for `bank.com` for example
    - DNS do not cache information not related to the query anymore
  - Lond TTL = high security, low TTL = low security
  - **Forced lookup**: Attacker can force a server to look something up
  - **Multiples replies**: Who said you can only reply once
  - **Multiples domains**: Who said you can only resolve one domain name
  - Attack patern
    - Manipulate DNS configuration settings on internal nerwork or local host
    - Have target point to attackers name server
  - *WAN Network*
    - Scan ISP networks, identify vulnerable routers or weak/default passwords
    - Attack poorly protected client router of Internet Service Providers
  - *LAN network*
    - Attack client router or DHCP server direclty

        * Attack DHCP exchange in local network
    - *Local host*
        * Manipulate DNS local host settings on compromised machine
- Protocol cannot be understood in isolation
    - If you are not aware of how all the protocols and your systems interact, you will miss vulnerabilities that are obvious to an attacker
    - **DNS**
        * DNS protocol has only 65 536 possible transactions IDs (16 bits)
        * Relying on TTL for security
    - **TCP/IP**
        * IP protocol has only 65 536 possible source port (16 bits)
        * Implementation with no random source port
- One standard to ensure integrity and authenticity: **DNSSEC Extensions**
    - DNSSEc is aset of security extensions for verifying the identity of DNS root and aithoritative nameservers
    - Designed to prevent DNS cache poisoning & other attacks
    - It does not encrypt communications
    - DNS over TLS or HTTPS (DoT & DoH) do encrypt DNS queries

(Distributed) Denial of Service (D)Dos

Introduction and Generic Attacks

- **Denial-of-Service** (DoS) attacks try to *make a service or network resource unavailable* to its intended/legitimate users
- Typically achieved by *exhausting available resources* by sending an excessive amount of traffic/packets/request
- **Distributed DoS** attacks use many different sources simultaneously, often by creating and using so-called *botnets*

	Network links	**Network Devices/ Network stack**	**Applications**
Description	Volumetric attack	Protocol attack	Application-layer attacks
Unit of measurements	Bits per second (bps)	Packets per second (pps)	Request per second (rps)
Used mechanism/ examples	• Reflection and amplification • Shrew attack	• Reflection • State exhaustion • SYN/ACK floods • Fragmentation	• Computational complexity • Hash collisions • Slowloris
Defenses	• Filtering traffic scrubbing • Black-hole filtering	• Cookies • Rate-limiting	• Randomized/keyed hash functions

- Features that faciliate DoS attacks
    - Attacker controls significantly more resource than victim
    - Attacker needs to expend significantly less resources thatn victim
    - Attacker can hide its identity or continually change it
    - Victim needs to expend a significant amount of resources before being able to assess the legitimacy of requests

    - Attacker can instruct/trick other entities to send traffic on its behalf
- **Link flooding** causes congestion & high loss rates for incoming traffic
    - Very low throughtput for >5% loss rates
    - Result: Few legitimate clients served during DDoS attack
- **Volumetric attacks**
    - Consume the bandwidth either *within the target network/service, or between the target network/service* and the rest of the internet
    - These attacks are simply about causing congestion
- **Protocol attacks**
    - Designed to exhaust resrouces available on the target or on a specific device between the tarher and the Internet
    - Devices can include routers, load balancers, or security devices
    - Once the attack consumes a resource such as a device's TCP state table, no new connections can be opened
    - *Protocol DDoS attacks need not consume all of the tagets's available bandwidth to make it inaccessible*
    - They can take down even high-capacity devices capable of maintaining state for millions of connections
- **Applications Layer Attacks**
    - Target various aspects of an application or service at Layer 7
    - Most *sophisticated and stealthy attacks as they can be very effective with as few as one attacking machine* generating traffic at a low rate
    - Attacks very difficult to proactively detect with traditional flow-based monitoring solutions
    - Application-layer attacks are often referred to as stealthy or low-and-slow attacks
- **Address Spoofing**
    - Source address in IP header can be set by sender
    - In a connection protocol (UDP), server cannot confirm actual sender
    - *Defenses*
        - Address filtering by ISPs: ensure that hosts use their own addresses
        - Use connection-based protocols (e.g. TCP)
        - Cryptographic source authentication
- **Reflection and amplification**
    - *Requirements*
        - Ability to spoof source address
        - Publicly accessible servers
        - Ideally: response is (much) larger than request → amplification
    1. Choose open service (e.g. open DNS resolver) as reflector
    2. Craft request that triggers (much) larger response
    3. Send packet where source address is set to victim's address
    4. Reflector sends reply to victim
- **Mitigations**
    - Prevetn address spoofing
    - Perform access control
        - For example, DNS servers deployed within an organization or ISP should only server server clients from this organization
    - Implement response rate limiting
        - Limit the number of responses to a client IP
    - Ensure small emplification factors (ideally < 1)
        - Examples: WireGuard ensures that the responders's first message is smaller than the iniator's

Specific Attacks and Defense Mechanisms

- **Shrew attacks**
    - Exploits TCP congestion control feature
    - Sends periodically short burst to the target link/router

        - $\rightarrow$ Low traffic volume
    - Denies the bandwidth of legitimate TCP flows as it makes TCP believe there is a long-term congestion
    - *The attacker forces TCP flows to repeatedly enter a retransmission timeout state by sending high-rate but short-duration bursts*
- **Temporal Lensing**
    - Idea: ‘multiple rounds simultaneously impact’
    - Concentrate a flow in time
    - Even a low-bandwidth source can perfrom shrew attack
- Volumetric attack: **Coremelt and Crossfire**
- **Coremelt attack**
    - Adversaty controls many bots distributed accross the Internet
    - Bots send traffic between each other, thus all traffic is desired by destination
        - Traffic is not sent to victim as in regular DDoS attacks
    - Adversary can exhaust bandwidth on victim link
    - Result: attack traffic exhausts bandwidth in per-flow fair sharing systems
- **Crossfire attack**
    - Adversary controls distributed bot army
    - Observation: dut to route optimization, few links are actually used to connect a target region to rest of Internet
    - Adversaty can contact selected servers to overload target links
    - Result: disconnect target region from ramainder of Internet