Network Security

1. Introduction

1.1. Security Trends
Network security is an issue since *critical infrastructures* in the open systems with a growing user base ($\rightarrow$*increasing risk*) are threatened by *organized crime*.

1.2. Security Threats
Asymmetric Threat: Defenders must protect against all exploits on all systems but attackers can attack only a few.
Attacker Motivation: Ego, Revenge, Destruction, Criminal intend, Acquisition of resources, Acquisition of sensitive information

1.3. Security Concepts
CIA Triad
- Confidentiality (prevention of unauthorized disclosure)
- Integrity (prevention of unauthorized modification or deletion)
- Availability (prevention of unauthorized withholding)

Also: Authenticity, Accountability(Responsability), Nonrepudiation (Can't deny), Privacy
Passive attacks: Confidentiality (Eavesdropping, Content compromise, Traffic analysis)
Active attacks: Availability (Denial of service), Integrity and Authenticity (Modification, Replay, Fabrication)
Secure Channel: secure = authentic (of the sender) and confidential (no eavesdropping)
Security on OSI-Layers: Physical (Link encryption), Network (IPSEC), Transport (SSL), Application (SSH)

1.4. OSI (Open Systems Interconnection)

OSI Model	Data Seq.	Protocols + Port
Application	Message	FTP 20-21, SSH 22, Telnet 23
Presentation*		SMTP 25, HTTP/s 80/443
Session*		DHCP, NTP, BGP 179, TLS
Transport	Segment	TCP, UDP
Network	Datagram	IP, IPSec, ICMP, IGMP, OpenVPN
Data Link*	Frames	Ethernet, ARP
Physical	Bit Stream	802.11 (WiFi)

* *Don't exist in* **TCP/IP Model:** Internet Layer Link Layer

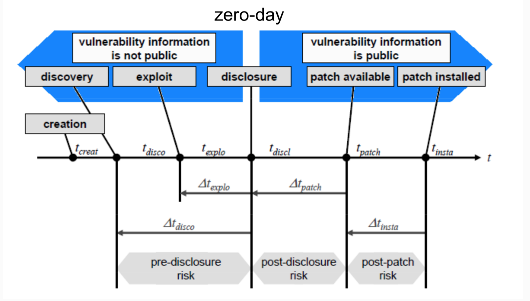
1.5. ARP (Address Resolution Protocol)
Use to find MAC add. corresponding to IP add. in LAN $\rightarrow$ request is broadcast MAC add. (FF:FF:FF:FF:FF:FF) $\rightarrow$ answer store in ARP cache
ARP Spoofing(=ARP cache poisoning/ARP poison routing) $\rightarrow$ possible because ARP stateless + no authentication $\Rightarrow$ MitM + DOS attacks
$\Rightarrow$ Defense: static ARP entries +/ certify/cross-check ARP answers
Cache lifetime: a few 10 seconds (avoiding frequent ARP requests)
Switch: extends LAN (layer 2) $\rightarrow$ forward Frames
Router: forward IP datagram (layer 3)
Note: ARP req. + MAC broadcasts don't go across routers

1.6. HTTP (Hypertext Transfer Protocol)
- Stateless protocol (Server doesn't know which client connects again)
- Status Code : 1xx info, 2xx success, 3xx redirection, 4xx client error, 5xx server error
- HTTP_REFERER : header field, identifies address of webpage that linked to the resource being requested (due to proxy not always set)
- HTTP methods : GET, POST, PUT, DELETE, HEAD
- HTTP cookie : name$\leftrightarrow$value (expiration date, stored by client)
  - Use for session management, secure cookie (HTTPS only)

2. (In)securtity, Risk & Vulnerability Lifecycle

2.1. (In)security Landscape
General: complexity is bad (but increases fast)
Security of a system := Security of its weakest link

2.2. Vulnerablity Lifecycle
Security vulnerability: A weakness in a system allowing an attacker to violate the confidentiality, integrity, availability of the system or the data and applications it hosts disagreement on what is a vulnerability possible (it's a a feature not a vulnerability)
CVE: Standard names for all publicly known vulnerabilities and security exposures. De facto standard. (Form: *CVE-Year-AnyDigits*)



Zero-day Date when the vulnerability becomes known by the public
Zero-day-exploit Attack that exploits a previously unknown vulnerability
Numbers:
50 % of vulnerabilities known to insiders 30 or more days before disclosure (less-than-zero-day)
At disclosure (zero-day): $\approx 50\%$ unpatched + $\approx 80\%$ exploits available
Exploit availability jumps from 15% to 80% after disclosure
A month after disclosure: $\approx 30\%$ unpatched

2.3. Dynamics of (In) Security
- extremely high dynamics around the disclosure
- exploit availability stays higher than the patch availability
- insiders: may know undisclosed vulnerabilities

Gap of insecurity: Difference between the exploit and patch (the bad a consistently faster than the good)
Security is slow: Many vulnerabilities are unpatched even 100 days after disclosure.
Market for new vulnerabilities: ZeroDayInitiative and Black Market, pricing from 1000 to more than 200 000 USD

2.4. Risk Management
There is no such thing as absolute security
Trade-Off between security and financial, social, functional, . . .

2.4.1 Risk analysis
- What assets are we trying to protect?
- what are the risk to those assets?
- How well does the security solution mitigate those risks?
- What other risks does the security solution cause?
- What costs and trade-offs does the security solution impose?

$\rightarrow$Is the trade-off worth if?

2.4.2 Risk management
Security is relative $\Rightarrow$manage risks
Options: avoid, decrease, transfer, accept risk
Business sense: risk < opportunity

3. Identity and Authentication

3.1. Identity and Identity Theft
Authentication: Authentication is the process of verifying an identity claim of an entity. It binds the principal to an identity
Authentication Criteria:
- Something an entity knows (e.g. password, PIN)
- Something an entity has (e.g. key, card)
- Something an entity is (e.g. biometric characteristic)
- rarely used: location, ability

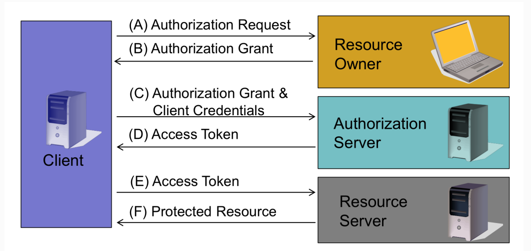
Types of ID Theft
- Financial Identity Theft: using someone else credit card, using another's name and SSN to obtain goods and services (e.g. open a bank account, purchase on invoice)
- Criminal Identity Theft: posing as another when apprehended for a crime
- Identity Cloning: using another's information to assume his or her identity in daily life
- Business/Commercial Identity Theft: using another's business name to obtain credit

Weak authentication: checking only one authentication criteria
Strong authentication: checking two or more authentication criteria
FIDO (Fast IDentity Online) Alliance
Non-profit organization addressing:
- lack of interoperability among strong authentication devices
- problems users face with creating and remembering multiple usernames and passwords

Members : Google, Microsoft, PayPal, Samsung, intel, RSA etc

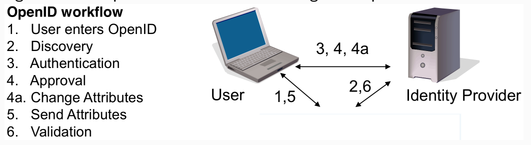
3.2. Authentication Protocols

3.2.1 OAuth
Allows sharing of private resources across web sites.
To access the data, tokens are used instead of credentials.
Limitation: possibility of impersonation of resource owner (via phishing)
Many eggs in one basket if central account becomes hacked, or you simply choose to close the account, then the serious repercussions are felt across several sites instead of just one.



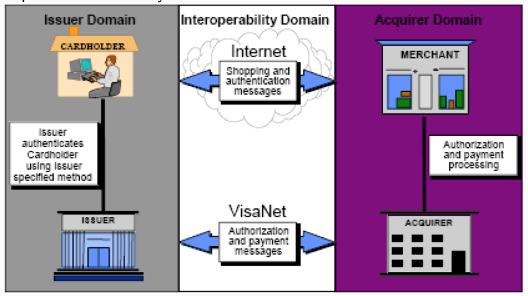
Resource Owner: Entity capable of granting access to a protected resource. When resource owner is a person, it is referred to as an end-user.
Client: An application making protected resource requests on behalf of the resource owner and with its authorization (e.g. a web browser for a printing service webpage).
Authorization server: The server issuing access tokens to the client after successfully authenticating the resource owner and obtaining authorization
Resource server: Server hosting the protected resources, capable of accepting and responding to protected resource requests using access tokens.

3.2.2 OpenID
Standard for decentralized user authentication: use existing account to sign in to multiple websites without creating a new password



Need to trust the identity provider.

3.2.3 3D Secure
A protocol used widely to authenticate online card transactions.

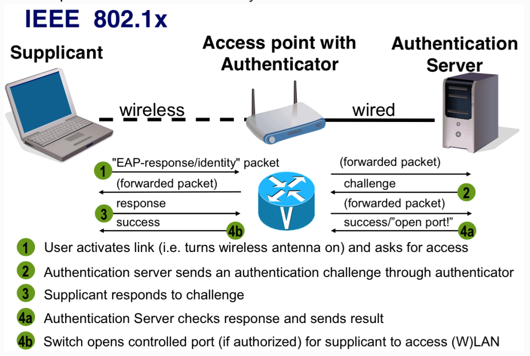


Issuer Domain
Interoperability Domain
Acquirer Domain
CARDHOLDER
Internet
Shopping and authentication messages
MERCHANT
Issuer authenticates Cardholder using Issuer specified method
Authorization and payment processing
VisaNet
Authorization and payment messages
ISSUER
ACQUIRER

After entering payment card details on merchant site:
- 3D Secure pops up a password entry form to a bank customer
- customer enters a password and, if it was correct,
- customer is returned to the merchant website to complete the transaction and
- the merchant gets an authorization code to submit to his bank

Problems: Pop-Up blockers (changed to iframes), Activation during shopping (phishing attacks), SLL verification not visible, liability shift (customer becomes liable for losses), weak bank authentication, no password reset procedure, privacy issues

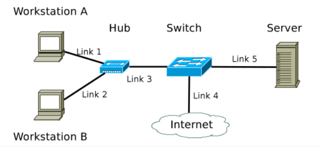
3.2.4 802.1x : Port-Based Network Access Control
Client-server based access control protocol that restricts unauthorized devices from connecting to a (W)LAN through publicly accessible ports.
For each 802.1x switch port, the switch creates **TWO virtual access points** at each port. The controlled port is open only when the device connected to that port has been authorized by 802.1x



IEEE 802.1x
Supplicant
Access point with Authenticator
Authentication Server
wireless
wired
1 "EAP-response/identity" packet
(forwarded packet)
(forwarded packet)
challenge 2
3 response
(forwarded packet)
success 4b
success/"open port!" 4a
1 User activates link (i.e. turns wireless antenna on) and asks for access
2 Authentication server sends an authentication challenge through authenticator
3 Supplicant responds to challenge
4a Authentication Server checks response and sends result
4b Switch opens controlled port (if authorized) for supplicant to access (W)LAN

EAP (Extensible Authentication Protocol): Authentication framework on the data link layer supporting multiple methods (MD5, OTP=one-time passwords, TLS, TTLS) without requiring an IP.
Roles : Identity and Security: Authentication (Who?) , Authorization (What?) , Access Control, Policy enforcement
Benefits: Standard based technology, control at link layer, interoperating wifi and wired, centralized user administration
Drawbacks: Authenticator authentication, Man-in-the-middle, session hijacking

A small company relies on the IEEE 802.1X protocol to grant access to their network. To simplify the task, we assume that the network consists only of two workstations and one server providing a DHCP and an authentication service (RADIUS). The nodes are interconnected with the help of one hub and one switch as illustrated in Figure 4. Please note that the switch acts as IEEE 802.1X authenticator. Assume, that in the beginning no workstation is authenticated.



1. Workstation A sends a DHCP request toward the DHCP server → there is no reply, the request is blocked at the switch. Only EAP queries pass through the uncontrolled port are allowed by the switch.
2. Workstation A sends an EAP-Response packet → links 1,2,3,5 can see this packet.
   Now assume the Workstation A is authenticated using 802.1x.
3. Workstation B tries to access http://www.example.com on the Internet → He receives an answer because A opened the port on the switch for both machines.
4. In this network, IEEE 802.1x does NOT prevents malicious users/hosts to steal HTTP session cookies. 802.1x does not provide any encryption and both users are connected with a cable.
5. The IEEE 802.1x protocol does NOT require fully functional DHCP and DNS services since it operates on OSI layer 2 (link/MAC layer), nothing above this layer is required.

3.3. Anonymization

There is no perfect anonymity in reality.
Levels of Anonymity: Identity (principal known), Pseudonymity (indirectly kown as pseudonym), anonymity (part of anonymity set but no distinguishing within)
Pseudonyms: public (→identification), non-public (→pseudonymity), unlinkable (→anonymity)
Use cases:
physical: feedback, voting, whistleblowing, censorship
digital: digital cash, digital voting, illegal activities
Oninon routing: Data is encapsulated multiple times and only the next hop is known to each node (Default TOR = 3 nodes)
Mixnet: proxy handles messages in batches (transformed and premutated) ⇒unlinkability of incoming and outgoing messagesat each proxy
Attacks: Traceback (break systems on path - defense : more proxies), Collusion (bad nodes - defense : reputation system), Traffic analysis (tagging with bit errors, replay messages - defense: heartbeats, traffic shaping, padding messages to constant size) , Logging (most initiators want to continue end-to-end communication across path changes, compromised proxy records predecessor;Required server relays:TOR first and last, Mixnet all)

4. Firewalls, IDS and NAT Traversal

4.1. Firewall Attack techniques

- IP source spoofing (doesn't work well with TCP based attacks)
- Artificial Fragmentation
- Vulnerabilities
- Denial of Service (state explosion)
- Tunnelling/Covert channel (ICMP, DNS. . .)

4.2. Firewall detection

- **Port scanning**: traceroute, source IP of response → improve obscurity by spoofing src address as target host
- **TTL**: let expire TTL one hop past firewall → reset low TTL, spoof or create response
- Trying to keep existence of firewall secret is OK, but it's **not a security technique**

4.3. Oranizational challenges

- **Large Rulesets**: complex, hard to manage and understand
- **Big Organizations**: tools for hundreds of FW, process to change?
- **Conflict**: Networking vs. security staff

4.4. Firewalls

Hardware or software device which is configured to permit, deny or proxy data through a computer network with different levels of trust. The configuration is called *policy*.
Types: simple packet filter, stateful filter, application layer proxy
Rules: Filtering (outgoing (egress), incoming (ingress)), Default policy (accept, reject), Deny (Drop silently, Reject), Addressing Transparency (firewall and network fingerprinting)

4.4.1 Stateless Firewall

Examine on network layer, decision based on header
Pro: application independent, good performance and scalability
Cons: no state or application context, can't prevent probing

4.4.2 Stateful Firewall

Keeps track of state. Decision based on *session state*
Pro: more powerful rules
Cons: no state for UDP, host vs. firewall state (inconsistent), state explosion

4.4.3 Application Layer Firewall

Take application state into account
Pro: application aware
Cons: many application protocols, performance, scalability

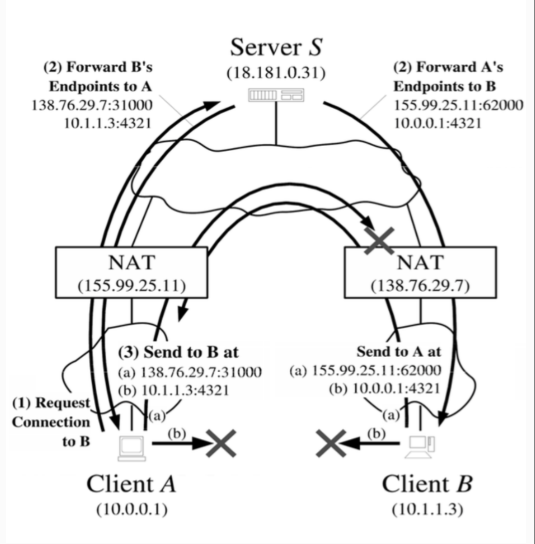
4.4.4 Web Application Firewall

Protects web-based applications from malicious requests (often reverse proxy)
Filtering: signatures, black/whitelisting

4.5. NAT (Network Address Translation)

One-to-one IP Address rewriting, keeps port numbers.
NAPT = Network Address and Port Translation (Most used one) is many-to-one (multiple hosts on private network using the same public IP address)
Benefits: Saves address space + Prevents malicious activity from outside (Don't use it for security as a Firewall)
Drawbacks: no end-to-end connectivity + P2P and IPSec don't work

4.5.1 NAT UDP Hole Punching (using RDV server)



4.5.2 NAT Security

No protection against application vulnerabilities once a connection is established. Prevents outsiders to initiate a connection. Network discovery through NAT is possible. NATs allowing hole punching for P2P may be vulnerable

4.6. Firewall configuration : IP Tables

Rule (source destination port/IP and target): `iptables -A INPUT -p tcp -s 0/0 -d 0/0 --dport 80 --syn -j ACCEPT`
Rule Targets(when packet matches rule)
ACCEPT; DROP; REJECT; QUEUE–(rarely used); RETURN–(rarely used);
MASQUERADE–only in nat table: rewrite source or destination address with address of outgoing or incoming interface

4.7. Intrusion Detection and Prevention Systems

Try to detect intrusions on the network by comparing to attack signatures. Block traffic after detection. **Output:** alerts, action, reporting, analysis

4.7.1 Classification

Object of observation:
Packet: (-)IP fragmentation, (-)not scalable (explosion)
Flow: (+)scalability, (+)encryption not problem, (-)not content-based
Point of observation:
Host: (-)hard deployment, (+)context information, (+)one sensor per machine
Network: (+)easy deployment, (+)multiple machines, (-)unknown machines, (-)performance (no context)
Method of observation:
Signature: (+)precision, (-)not detecting unknown attacks
Behavior: (+)unknown attacks, (-)difficult to find ground truth, (-)false positives

4.7.2 Challenges

- False alarms and false silences
- Cope with high network speeds
- Sensor management and signature distribution
- Policy management
- Manpower for interventions 24x7

4.7.3 Attacks and IDS evasion techniques

DoS attacks can also be used as a component of an attack. It enables other attacks to remain undetected.
- Flooding / resource exhaustion
- algorithmic Complexity Attacks (e.g. on hash table) → use some randomness

5. The Domain Name System Security

5.1. Routing

ASes(Autonomous System): define by AS Number $\simeq 40000$
- Tier 1ISP: can access all internet for free or by reciprocal agreement
- Transit: AS transiting msg for other (for **$**)
- Tier 2ISP: need to purchase transit

5.1.1 BGP (Border Gateway Protocol)

Used to exchange routing and exchangeability information (inter ASes → **iBGP**, between ASes → **eBGP**)
BGP routers have Routing Information Base (RIB) → if update its RIB ⇒ notify others around
Attacks on BGP: eavesdropping, replay, msg insertion, deletion and modification, MitM, DOS + TCP attacks (reset attack)
Solutions: BGP + TCP MD5 signature (not efficient), SBGP, SCION

5.2. Attacks on DNS

5.2.1 Denial of Service on DNS

Targeting the 13 DNS root servers which are a bottleneck resource.
Defense: Over-provisioning (13 clusters) + Anycast (>500) (1 address to multiple potential receivers) prevents most attacks.

5.2.2 Account-Takeover

Attacking the web interfaces of registrars and try to enter malicious name servers. Possible solution: Multi-channel authentication

5.2.3 Manipulate local DNS settings

- Manipulate local host configurations (access to the local machine nescessary)
- Spoof DHCP replies (access to LAN nescessary) →countermeasure: authentication for DHCP messages
- Set up Malicious DHCP server that is faster than the valid one

Example: DNS-Changer-Botnet (4million infected hosts, difficult malicious DNS servers shut-down : still used by at least 250 000 hosts)

5.2.4 DNS Spoofing

Successful insertion of incorrect resolution information by a host that has no authority to provide that information. Attacker sniffs traffic for a victim request and get the TXID. He can reply with the right TXID. Works only if attacker is faster than legitimate DNS Response.

5.2.5 Manipulate DNS lookup process (Cache Poisoning)

Inject faked Domain,IP information into caching name server.
Process: attacker asks caching server for attack.com, attacker server replies with fake records in the additional section (e.g. www.bank.com)
This worked because early DNS implementations accepted records in the additional section without proper checking.
Bailiwick Checking: Is the server authorized to respond to DNS requests for the domain. They check the hierarchy : 83.www.bank.com can return any record for www.bank.com but cannot for bank.com (which cannot for .com)
Weak authentication: Port, TXID, Query string - first good answer wins

5.2.6 The Kaminsky Attack

1. Inject query on the client *$Random*.www.bank.com
2. reply multiple times (TXID 0-200)
3. Send name server redirections in additional section
4. if it fails, return to step 1

⇒ Fix: Source Port Randomization. Chance: $65536 \cdot (65536 - 1024)$ to 1

5.3. DNS

DNS is a global, distributed, robust system for name to IP address resolution

DNS is the glue of the internet: Nearly all activities start with a DNS lookup (Orginally: no security considerations)
Design:
- Client-Server application
- Hierarchical System
- Use UDP on port 53

Zone: Collection of hostnames/IP pairs managed together by same entity
Domain: just a name, part of hierarchy of the DNS
Nameserver (authoritative): Server that answers DNS queries (Responsible DNS Server for each zone)
Resolver: Client part of DNS that resolves domain names
Recursive name server: Answers queries for all zones (1- Root Server, 2- TLD Server, 3- Authoritative DNS Server)
Stub resolver: Forwards request to recursive name server (typically used by end-hosts)
Caching: Reduces overhead. Lifetime controlled by TTL
`dig URL_TO_LOOKUP @DNS_SERVER` = make a request to a specific DNS
`whois IP_OR_URL_TO_LOOKUP` = get information on a specific IP or URL
`nslookup URL_TO_LOOKUP` = get IP address from a URL (don't use OS local DNS as `dig`)
3 possible answers to DNS queries:
- Here is your answer
- go away
- I don't know ask . . .

5.4. DNSSEC

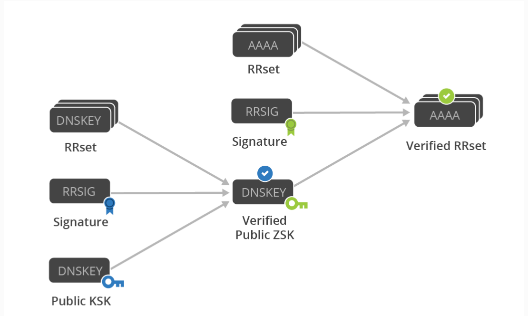
Provides: Authenticity, Integrity, Backward compatibility
Drawbacks: No confidentiality, No protection against DoS, Must be installed from a trusted source, Must be deployed at each step in the domain lookup.
All records are signed: Key pairs for each zone → Integrity
Chain of trust → Authenticity

DNS record types specific to DNSSEC

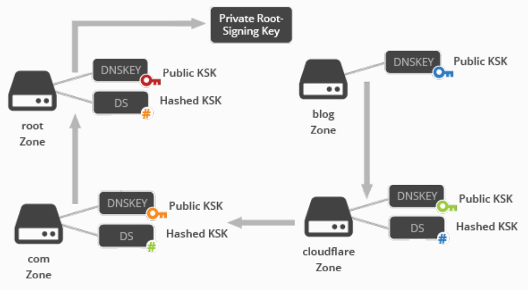
- **ZSK**: Zone-Signing Keys (RRset signed with ZSK_priv → RRSIG)
- **RRSet**: Resource Record Set of the same type records
- **RRSIG**: Contains a cryptographic signature of RRsets
- **DNSKEY**: Contains a public signing key (ZSK or KSK)
- **KSK**: Key-Signing Keys, signs DNSKEY records of public ZSK creating a RRSIG record
- **ZSK**: Zone-Signing Keys, each zone has a/several pair(s) used to sign all other records (priv: signs, pub: verifies)
- **DS**: Contains the hash of a DNSKEY record

To validate .ch:

1. Request the KSK key of .ch by querying the parent zone (here, root) for a DS record (record set signed in RRSIG by root's ZSK)
2. Verify RRSIG of DS record with root's public KSK (must be stored localy)
3. Request desired RRset to contact .ch (containing A and NS records for .ch), with the corresponding RRSIG to sign it (using the .ch's ZSK)
4. Request DNSKEY of .ch with public ZSK(s) and KSK with the RRSIG of DNSKEY RRset (using the domain's KSK)
5. Verify RRSIG of RRset with the .ch's public ZSK
6. Verify the .ch KSK key by hashing it and comparing it to the DS received in 1



There are two signing keys (ZSK and KSK) because the KSK is longer (i.e more secure and less often changed and used) than the ZSK (which can be changed more often and is used to sign a lot more records). The parent zone signs the KSK (not done often) which in turn can sign the ZSK (faster to use shorter keys).
The DS (Delegation Signer) record glues the chain of trust :



6. Availability and Denial of Service

6.1. System Level Agreements (SLA)

Level	%	Downtime (year)	Downtime (month)
Two-nines	99	3.65 days	7.2 hours
Three-nines	99.9	8.76 hours	43.8 minutes
Four-nines	99.99	52.56 minutes	4.38 minutes
Five-nines	99.999	5.26 minutes	25.9 seconds
Six-nines	99.9999	31.5 seconds	2.59 seconds

How to achieve 99.999:

- High Redundancy + Fast Failover (Quick Change)
- Failure Resilience
- Over Provisioning
- Backup, Close Monitoring and Fast Recovery

6.2. Denial of service (DoS)

DoS attack : aims to prevent legitimate users from accessing a specific service. → Resource starvation of CPU, Storage, Network

Attacks:

- SYN-Flood: Spam Server with TCP SYNs → needs to keep state *Solution*: Choose carefully constructed initial sequence number (crypto based) and SYN cookies (client keep state and present it later)
- Compression Bomb *Solution* :restrictions on size, depth and time
- Source Spoofing/Reflector Attack from any vulnerable service, by botnet or broadcast addresses (Smurf attack)
- Mail Bounce: reply-to: @victim.com, To: and multiple invalid Bcc: @target.com → victim will receive many bounce emails from target.com
- DNS Amplification: Send small request to recursive servers with spoofed source

6.3. Dos Countermeasures

Prevention:

- Secure systems: (e.g. prevent machine compromise to build botnets by patching, IDS/IPS, firewall etc.)
- Secure protocols: (e.g. prevent IP spoofing, use TCP SYN cookies, HashCash: add client CPU cycles)
- Resource accounting: (server commits resources only to authenticated/trustworthy clients)
- Resource multiplication: (resource over provisioning for the case of an attack)
- Late resource binding: (bind resources as late as possible)

Reaction:

- Detect (using pattern, anomaly, third party)
- React (rate-limit, filter, reconfigure, identify agents)

6.4. Data Sizes : SI Vs Binary symbols

SI symbol (name)	value	Binary symbol (name)	value
kB (kilobyte)	10^3	KiB (kibibyte)	2^10
MB (megabyte)	10^6	MiB (mebibyte)	2^20
GB (gigabyte)	10^9	GiB (gibibyte)	2^30
TB (terabyte)	10^12	TiB (tebibyte)	2^40
PB (petabyte)	10^15	PiB (pebibyte)	2^50
EB (exabyte)	10^18	EiB (exbibyte)	2^60
ZB (zettabyte)	10^21	ZiB (zebibyte)	2^70
YB (yottabyte)	10^24	YiB (yobibyte)	2^80

Note: Historically, 1 MB = 10^6 or 2^20 depending on the context

7. Secure Channels: Principles, VPN, SSH

7.1. Security by Layer of the TCP/IP Model

Properties of a secure channel: secure = authentic and confidential

7.1.1 Security at Link Layer

- All traffic over a specific Link (WPA2 uses AES)
- Often implemented in hardware (Quantum Cryptography)

Pro: Speed, Seamless
Cons: every link separately, trust in link operator

7.1.2 Security at Network Layer

Secure traffic over multiple links between end points (IPSec, L2TP)
Pro: seamless security to above layers, IPSec part of IPv6 + IPv4
Cons: complex configuration, tunnel mode encrypts only part of a route

7.1.3 Security at Transport Layer

Implemented in end-hosts (e.g. TLS/SSL, SSH)
Pro: can be added to existing apps, provide and easy configuration
Cons (of TLS): protocol specific(TCP), application must be TLS/SSL aware

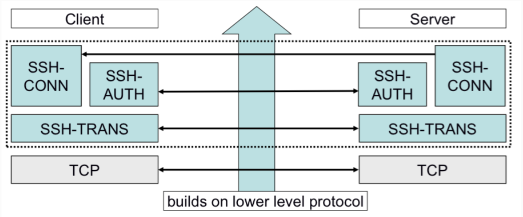
7.1.4 Security at Application Layer

implemented in end hosts (PGP, S/MIME, Skype, ...)
Pro: extend app without involving OS, applications understand data
Cons: separate security for each app

7.2. SSH

Standard for remote login and encrypted file transfer, provide PFS. Security algorithms and parameters used are negotiable.

7.2.1 Protocols



- **SSH-CONN**ection protocol: connection multiplexing
- **SSH-AUTH**entication protocol: client authentication
- **SSH-TRANS**port prot.: server authentication, confidentiality, integrity

SSH-CONN could be used without SSH-AUTH (e.g. anonymous FTP)

7.2.2 SSH-Trans

- algorithm negotiation
- session key exchange (e.g. Diffie-Hellman)
- session id
- server authentication
- encryption, integrity, data compression

7.2.3 SSH-Auth

- authenticates the client with multiple auth mechanisms
- defines format of auth-requests (Username, Method name, Service name)

7.2.4 SSH-Conn

- multiplexing multiple steams
- port forwarding
- compression handling
- interactive and non-interactive sessions

7.2.5 Attacks SSH cannot counter

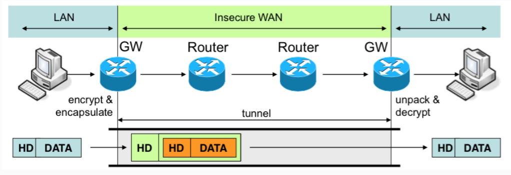
- password cracking (passwords must be strong and not reused)
- traffic analysis
- IP and TCP attacks

7.3. VPN

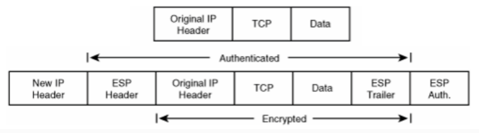
Securely interconnect networks or machines over an existing network. IPSec, L2TP, OpenSSH, SSL/TLS can tunnel entire network's traffic (OpenVPN).

- Encryption → confidentiality
- Randomized Initialisation Vectors (IVs)
- Replay protection → unique ID or timestamp in packets before signature
- Authentication (HMAC) → integrity and authenticity

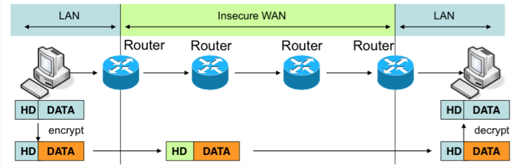
7.3.1 Tunnel Mode (Gateway-to-Gateway)



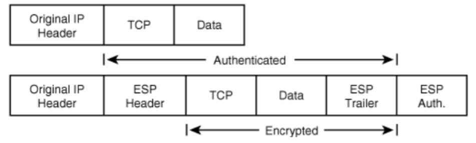
- Encrypt and Authenticate original IP packet
- Transport it as payload in new IP packet (IP encapsulation)
- **IPSec Tunnel Mode** packet architecture :



7.3.2 Transport Mode (End-to-End)



- End-to-End communication between hosts
- Encrypt and Authenticate payload only
- **IPSec Transport Mode** packet architecture :



7.3.3 IPSec Security Association (SA)

A Security Association (SA) is the establishment of shared security attributes between two network entities to support secure communication.
SA is uniquely defined by a triple of :

- Security Parameter Index (SPI): number placed in ESP datagrams.
- IP Destination Address
- Security Protocol Identifier:AH or ESP

7.4. Message Authentication Code (MAC)

Provide data integrity and authentification

- message not modified in transit
- source is authentic
- not delayed
- sequence order

`MAC (K, M) = DES (K, M)` take last 32 bits

HMAC (Mac with Hash functions)

- use hash functions (MD5, SHA)
- faster
- easy available
- less affected by collisions than hash functions → if MD5 or SHA-1 broken, corresponding HMAC can still be secure.

`HMAC (K, M) = H(K' xor opad | H (K' xor ipad | M))`

8. TLS (Transport Layer Security)

- Session or Application layer (OSI model)
- SSL (Secure Socket Layer) = predecessor of TLS
- SSLv3 and TLS almost same except some crypto algo.

8.1. Diffie-Hellman Key Exchange (DH)

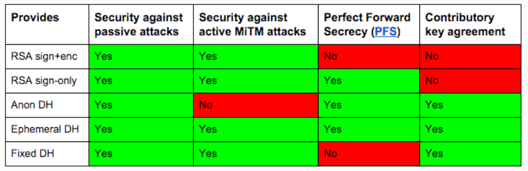
- **Public values**: large prime p, generator g
- **Secret values**: Alice has a, Bob has b
- Alice → Bob: $A = g^a \pmod p$
- Bob → Alice: $B = g^b \pmod p$
- Bob computes $A^b \pmod p = g^{ab} \pmod p$
- Alice computes $(B)^a \pmod p = g^{ab} \pmod p$
- Alice and Bob have now a shared secret $g^{ab} \pmod p$
- Eve cannot compute $g^{ab} \pmod p$

8.2. Generation of Crypto Parameters

Master Secret(MS) created from pre-master secret(PS), client random(CR), server random(SR): MS = $h('A')||h('BB')||h('CCC')$ with $h(X) = MD5(PS||SHA(X||PS||CR||SR))$
Key materials pairs client & server: MAC key, write key and write IV
`key_block` = $h_2('A')||h_2('BB')||h_2('CCC')||\cdots$
with $h_2(X) = MD5(MS||SHA(X||MS||CR||SR))$

- `key_block` is the concatenation of the 6 keys
- Nb of round depends on size of keys → MD5 = 16 bytes per round
- MS ≃ pseudorandom seed value and CR and SR ≃ salt values

8.3. Key Exchange Methods



Provides	Security against passive attacks	Security against active MiTM attacks	Perfect Forward Secrecy (PFS)	Contributory key agreement
RSA sign+enc	Yes	Yes	No	No
RSA sign-only	Yes	Yes	Yes	No
Anon DH	Yes	No	Yes	Yes
Ephemeral DH	Yes	Yes	Yes	Yes
Fixed DH	Yes	Yes	No	Yes

- **Contributory Key Agreement** = Both parties contribute to session key, no party can fully determine session key.
- **Perfect Forward Secrecy** = knowing long-term private key does not reveal session key.

8.3.1 RSA

Pre-master secret(PMS)= 48-bytes (2 protocol version + 46 random)
sign+enc: client generates pre-master and encrypts it with PU_{server}.

- No `server_key_exchange` msg needed in phase 2
- Always same public/private keys used ⇒ no PFS ⇒ Not recommended

sign-only: Server RSA keys used for signature only

- Server creates temporary RSA keys, signs them with PR_{server} and use `server_key_exchange` to send them
- PMS encrypted with temporary server public key

8.3.2 Diffie-Hellman

Anonymous DH: No authentication of client nor server

- `server_key_exchange`: DH public values + server public DH key
- `client_key_exchange`: client public DH key
- Vulnerable to MitM attacks

Ephemeral DH: create ephemeral (one-time) secret keys

- `server_key_exchange`: DH public values + server public DH key + signature of those parameters
- `client_key_exchange`: client public DH key
- Don't provide authentication but signing values with RSA does.
- Most commonly used

Fixed DH : DH public values + server public values are fixed

- All DH parameters in server certificate signed by CA ⇒ no need of `server_key_exchange`
- server can asked client to be authenticated with `certificate_request`
- Rarely used

8.4. Attacks against SSL/TLS

8.4.1 Dumbing Down

- attacker forces to use old breakable algo. for communication (using the `ciphersuite`)
- only DOS except with anon DH where it can become real MitM

8.4.2 Attacks against CAs

- Attacks against CA to get false certificates signed
- any CA can issue certs for any domain (weakest link)
- 2010: Stuxnet signed itself 2 compromised Taiwan CAs' private keys
- 2011: Comodo was duped into creating certificates for Google sites

8.4.3 BEAST

- Vulnerability in the Initialization Vector (IV) of the CBC mode of AES
- MitM can read msg by encrypting it multiple times
- mitigated in TLS1.1 and above

8.4.4 Assumption for Secure TLS

- CA, Crypto, Browser, User
- if a single assumption does not hold ⇒ TLS not secure

8.5. Handshake Protocol

8.5.1 Phase 1: Establish Security Capabilities (hello_messages)

1. C → S: `client_hello`
   - **version**: highest supported version.
   - **ciphersuite**: Supported ciphers listed in ↘ order of pref. := Key exchange algo. + Cipher algo. (RC4, AES, 3DES...) + MAC algo. (MD5 or SHA-1) + ...
   - **random**: 32-bit timestamp + 28 bytes random ⇒ prevent replay
   - **compression**: list of compression methods
   - **SessionID**: update (value=which session) or create (value=0)
2. S → C: `server_hello`: Reply to every param.

8.5.2 Phase 2: Server Authentication and Key Exchange

1. S → C: `certificate`*: except for Anon DH, contains public DH param if DH fixed used
2. S → C: `server_key_exchange`*: required for Anon DH, Ephemeral DH, RSA sign-only
3. S → C: `certificate_request`*: request types of cert. from client
4. S → C: `server_hello_done`: no param. + wait for client response

8.5.3 Phase 3: Client Authentication and Key Exchange

Client should verify server certificate (if required) + server parameters

1. C → S: `certificate`*: client cert. or no_certificate alert
2. C → S: `client_key_exchange`
   - **RSA**: 48 bytes pre-master secret encrypted with server's public key
   - **DH**: client public DH values
3. C → S: `certificate_verify`*: hash of master secret + previous msgs encrypted with client private key (except fixed DH)

8.5.4 Phase 4: Finish

1. C → S: `change_cipher_spec`: Send 1 byte to confirm encryption.
2. C → S: `finished`: hash(`master_secret` || pad2 || hash(handshake_messages || Sender || master_secret || pad1))
   - once with MD5(...) and once with SAH-1(...)
   - `handshake_messages` = all messages until now
3. S → C: `change_cipher_spec`: server confirm encryption too
4. S → C: `finished` 1st encrypted msg = hash of all previous msg

** = optional or situation-dependent messages that are not always sent*

9. Certificates and PKI (Public Key Infrastructure)

9.1. Public-Key Certificates

Certificate = (public key + id of key owner)*signed by a trusted 3rd party

1. All can determine name + public key of the certificate's owner
2. All can verify signature of CA
3. Only CA can create and update certificates
4. All can verify the currency of the certificate

9.1.1 Revocation of Certificates

1. User's private key is assumed to be compromised
2. User is no longer certified by this CA
3. CA's certificate is assumed to be compromised.

CRL(Certificates Revocation List): listing of all the revocated certificates by a CA.

9.2. OCSP (Online Certificate Status Protocol)

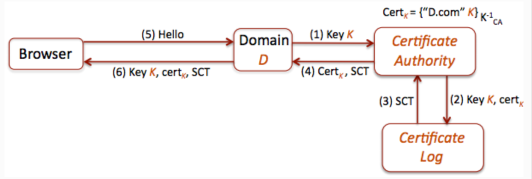
Verify certificate status, ensure it is valid and has not been revoked.

- Small bandwidth needs ⇒ Real-time checks but not always reliable
- Privacy issues ← OCSP responder can track users
- OCSP servers need to answer every client request (no cache)
- if connection fails, client has to make the choice on its own
- OCSP stapling: certificate owner give OCSP response with its certificate ⇒ **Not secure** active attacker dropping requests, default behavior is to validate for no responder

9.3. GL: Certificate Transparency

Protocol for publicly logging the certificates activity.
⇒ augment chain-of-trust for the entire SSL certificate system.

- **Logs**: log cryptographically secured in a Merkle hash tree
- **Monitors**: watch for suspicious or missing certificates in logs
- **Auditors** : verify the overall integrity of logs (Signed Certificate Timestamp (SCT))



Benefits: Gradual rollout + Minimal impact on existing infrastructure
Disadvantages: MitM (but can be detected externally), no support revocation, still need to contact log to verify

9.4. AKI (Accountable Key Infrastructure)

- New public-key validation infra. to reduce lvl of trust in CAs
- Distribute trust → no single point of failure
- support revocation and update of certificates
- Based on integrity tree (Efficient representation of the current state of all domains)

10. Web Application Security

10.1. Top 10 Application Security Flaws

A1	Injection (SQLi, LDAP, XPATH, OS Command)
A2	Broken Authentication and Session Management
A3	Cross-Site Scripting (XSS)
A4	Insecure Direct Object References
A5	Security Misconfiguration
A6	Sensitive Data Exposure
A7	Missing Function Level Access Control
A8	Cross-Site Request Forgery (XSRF)
A9	Using Components with Known Vulnerabilities
A10	Unvalidated Redirects and Forward

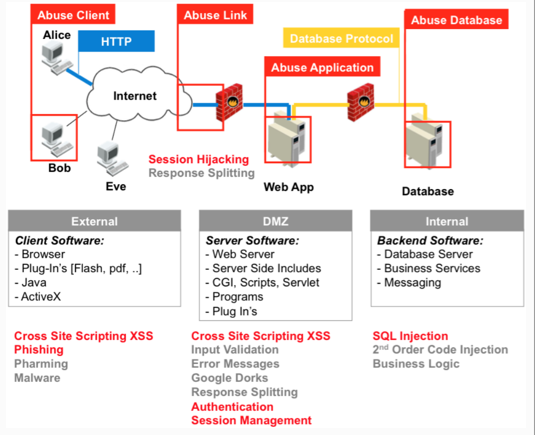
OWASP (Open Web App Security Project), Nov. 2013

Session Management

10.2. Attacks

- Network sniffing → use HTTPS to prevent it
- Don't mix HTTP and HTTPS SessionID
- Random SessionID is important
- Brute Force → Not predictable Username and Password

10.3. Anatomy of a Web App & Threat Vectors



10.4. HTTP and Session Management

HTTP is stateless → state needs to be implemented with session ID
SessionID: generated on server, stored on client, transmitted with every request
After login → Identification by SessionID (Target for attacker)

10.4.1 Generation

- Strong → not predictive + impossible to guess next value
- Random → not based on predictable values (no MD5 or SHA-1)

10.4.2 Transport

- GET: easy (in URL), (+)compatible , (-)logged on intermediate devices, (-)HTTP referer
- POST: (+)compatible(works with cookies disabled), (+)not obvious, (-)more complex development, (-)slightly harder to manipulate
  → Both GET and POST are vulnerable to *session fixation* attacks !
- Cookie: (+)more options, (+)not recorded, (+)https restriction, (-)store on client, (-)may be disabled by users

10.4.3 Revocation

- Session Validity: client and server side revocation + time limited
- Session Timeout: important with shared computers, expiry time should be minimum

10.4.4 Destruction

SQL Injection (SQLi)

10.5. Code Injection

Attack which insert code that is afterwards interpreted by a process.
Process:

1. Data enters app from untrusted source
2. Data is part of a string executed as a command by app
3. App gives attacker privilege or info that he would otherwise not have

10.6. SQL Injection Defense

- constrain and sanitize all client data on the server (input validation)
- use parametrized prepared statements
- use stored procedures (restraint possible actions)
- avoid disclosing error information
- run DB with reduced privileges

10.7. SQL code injection

SQLi = insertion of SQL query via input data from client to app.
What can you do with it ?

- Read / Modify data in the database
- Execute admin operations on the DB
- issue commands on the OS

10.7.1 Tautology

Inject code in conditional statement(s) ⇒ always evaluate to true

```
SELECT uid FROM users WHERE login='' OR ''='' AND pwd='' OR
''='';
```

10.7.2 Union Query

Inject UNION query ⇒ get union of the 2 queries. Need same number(#columns, add same on in case) and type of parameters from both selected queries.

```
SELECT uid FROM users WHERE login='' UNION SELECT cardNo from
CreditCards WHERE uid=123; -- AND pass='';
```

10.7.3 Piggy-Backed Queries

Inject query at the end of another query (⇐ Misconfiguration)

```
SELECT * FROM users WHERE uid='abc' AND password=''; DROP
TABLE users; --';
```

10.7.4 Stored Procedure

Execute stored procedures present in the database

```
SELECT * FROM users WHERE uid='abc' AND password='';
SHUTDOWN; --';
```

10.7.5 Inference / Determine DB Engine

If DB doesn't return error msg ⇒ change behaviour of DB to guess (Blind Injection or Time Injection (e.g. example)

```
http://www.example.com/product.php?product_id=100 AND
IF(version() like '5%', sleep(15), 'false'));--
```

→ check MySQL version 5.x or not ⇒ if yes server delays answer 15s

10.7.6 Alternate Encodings

Fool scanning and detecting techniques with alternate encoding.

```
SELECT * FROM users WHERE uid='abc' AND password='';
exec(char(0x73687574646f776e)); --'
```

- ; → terminates a command
- -- → considers everything afterwards as a comment.
- Error messages can provide hints on which DB is used.
- **IDS Detection Evasion** by varying the command (e.g. through the insertion of whitespaces or comments UNION/**/SELECT)

Cross-Site Scripting (XSS)

10.8. Same-origin policy

A script can only access content and properties of a document loaded from the same origin (⇔ same protocol, same hostname, same port but ignoring URL path)
Interaction of different origins

- Link (href)
- iframe: Shown inside the website but can't exit iframe
- POST: POSTing data to external source (php form ...)
- script included: Evaluated in context of the website (⇒ Dangerous)
- Asynchronous Requests (AJAX): Different origin only possible if allowed by target domain (with Access-Control-Allow-Origin Header)

10.9. Request Forgery (XSRF)

A form from one domain posts a request to a different domain through an authenticated session (= write-only attack) (E.g.: Reset password)
Countermeasures: (CSRF) hidden security Token (unique and cannot be read by attacker), ask for (old) password, check HTTP referer headers (indicates the page making the request, not always set → not really efficient), show CAPTCHA

10.10. Script inclusion (XSSI)

Never include a script you do not trust!
Direct sourcing: honest server includes untrusted code from an external source
Call-back: attacker's website includes code (e.g.: scripts to send money) from an honest server which do not verify that the user really intended to execute it.
Countermeasures: security tokens derived from the cookie and a server challenge, use only HTTP POST for XSSI call-back, check HTTP referer

10.11. XSS Attacks

XSS flaws occur whenever a web app takes user supplied datas and sends it to another web browser without first *validating* or *encoding* the content (⇒ allows injection of malicious code).
Target: The user of the web app
Infection path:

- Reflected(1:1,URL): email, chat, uploaded files, third party content ...
- Stored: script in database from forum messages, comments, username ...
- DOM Based: stays in browser (DOM)

Dangerous content: advertisements, user contributed content, widgets (counters, scripts ...)
Example: Data Leakage = script reads session cookies and sends them to an evil server (→ Solution: HTTPOnly cookies → browser prevents client side script to access it, not supported by all browsers)
XSS Shell = XSS backdoor and zombie manager → can interactively send requests and get responses from victim (≃ backdoor the page)

10.11.1 Countermeasures

- Test your web app
- Input sanitization (check user submitted content) → filter out metacharacters (<, >, ...)
- HTML output encoding (escaping) → e.g. < is replaced by &lt;

11. Malware

11.1. Definitions

Malicious software (Malware) is software that is intentionally included or inserted in a computer system for a harmful purpose.
Trojan: seemingly offers useful functionalities but contains hidden functionality which undermine system security
Worm: Self-contained software that can replicate itself from computer to computer across network connections. Upon arrival, may be activated to perform unwanted operation and continue propagating.
Bot: malicious software agent running on a compromised machine executing commands by the bot master by listening to the command and control center (**botnet**= all bots listening to the same C&C)
Rootkit: activate during boot, hides from OS, can interfere with reflashing process or reinstall itself (self-healing)
Keylogger: records keys pressed to steal information
Ransomware: encrypt data and extort money for decryption key

11.2. Malware on Portable Storage

- Uses Sneakernet(people) as physical propagation
- Variants: boot code virus (replaces boot sector), host program infection (embedding into executables)
- E.g. Stuxnet infected nuclear plants in Iran from USB sticks

11.3. Problems cause by Malwares

- Unauthorized use of system/network resources
- Sabotage
- Espionage
- Lowering of system security

11.4. Malware detection

Goal: Detect malware and disinfect system
Deployment: Endpoint, Network
Evaluation: Recognition rate (high true positive), False alarm rate (low)

11.4.1 Signature Based

- Reactive (based on known malware binaries)
- Signature database (keeps growing, polymorphism)
- Metrics: update delay, coverage, resource consumption

11.4.2 Behaviour Based

- Requires behavior model (ground truth, baseline)
- Problematic: user alerting (when is it a problem ?)

11.4.3 Evaluation Antivirus Softwares

- Detection (reactive, proactive)
- Support for compression
- Self-protection
- Cleanup of already infected system
- Enterprise deployment features

11.5. Resistant Malware

11.5.1 Antivirus Detection Evasion

- Polymorphism (signature won't recognize them)
  - different code with same effect (while instead of do - while)
  - change order of code
  - insert noise (sleep(0), if (1==1) or NOPs)
  - compiler setting modulation (using different compiler options).
- Code obfuscation (Disguise actual purpose)
- Encryption of code and messages
- Code changes to prevent disassembly
- Sandbox detection (act normal if monitored)
  - read flags/API that would indicate the process is being debugged
  - self debugging: failure indicates that the program is already being debugged
  - measuring deltas between timestamps around exception handlers
- Bootstrapping / Multi Stage Worms

11.5.2 Advanced Persistent Threat (APT)

Sophisticated stealth customized attack on selected high value targets

- **Advanced**: sophisticated attack techniques
- **Persistent**: might launch any time and hide well, hard to detect
- **Threat**: specific objective, skilled, well funded

Examples: Aurora(2009) open backdoor, Stuxnet(2010)

11.6. Worm Propagation Mechanism

1. Select a target host address (hitlist, random scanning...)
2. Contact target host
3. Check if target is vulnerable
4. Exploit vulnerability
5. Install copy of worm
6. Start copy and Goto 1

Propagation speed: scan rate, vulnerable population, system compromise delay, worm transfer speed

11.6.1 SIS Model (Susceptible-Infected-Susceptible)

$$d\rho(t)/dt = \underbrace{\overbrace{\beta \cdot k \cdot (1-\rho(t))}^{\text{\# susceptible nodes connected to infected node}} \cdot \rho(t)}_{\text{\# susceptible nodes that get infected}} - \overbrace{\delta\,\rho(t)}^{\text{\# spontaneously cured nodes}}$$

- Only two possible states: Susceptible or Infected
- $\rho(t)$ fraction of infected nodes
- β: infection rate
- δ: cure rate
- k: number of outgoing edges at each node

11.7. Massive Worm examples

- "Morris" Internet worm (1988)
- SQL Slammer, UDP database worm (2003)
- Blaster, RPC(=Remote Procedure Call) internet worm (2003)
- Sobig.F , Email worm (2003)
- Witty worm , "firewall" worm (2004)

11.8. Worm Network Impact

Primary network impact is due to scanning activity

- ARP flooding
- High volume of ICMP destination unreachable msg
- Higher network and traffic flows

Network activity anomaly detection: Entropy measurement of flow header fields. $H = -\sum_{i=1}^{n} p_i \log(p_i)$; if $H \leq T$ then raise alert. Determine threshold T using training data.

11.9. Malware countermeasures

- End system: patching, firewall, IPS, Antivirus and educate users (awareness)
- Anti-Virus: useful but not too effective
- Monocultures are targeted first (e.g. Windows)
- Slow patching is a huge risk
- Application whitelisting
- No trusted intranet
- Thin Client (e.g. ChromeOS)

12. Botnets + Malware Development

Bot Agent: crime-ware tool installed on victims
Botnet: collection of all bot agents
Bot Master: the criminal(s) operating the botnet
Command and Control(CnC): botnets management system
Opt-in Botnet: Voluntary botnets (e.g. Low Orbit Ion Cannon)

12.1. Objectives to best utilize infected machines

- **Buildup** efficient infection and spreading
- **Persistence**: prevent detection and Removal
- **Modularity**: address changing functionality needs
- **Scalability**: handle large number of machines
- **Adaptive**: to business and technology challenges
- **Anonymity**: prevent identification of operator

12.2. Target exploitation

- **Persist and avoid detection**: explore, steal information, load functionality, maximize yield
- **Move to active attacks**: load attack functionality → higher chance of detection
- **Throw away agent**: sell to idiots

12.3. Bot Command Modeles

- **No Control**: default malicious behaviours, self propagation defaults, less flexible, most likely detected by signatures, most resistant to global shutdown
- **Private Channel**: custom and covert channels, abuse and alteration of common protocols, short-term stealth, signature detection easy once CnC observed
- **Public Infrastructure**: use common applications API, generally reliable and anonymous, mostly IRC, some P2P and microblogging (95% of today's botnets)
- **Resilient hybrids**: (all models) default malicious behaviors, fall-back plans if CnC unavailable, pre-programmed contact points (drop boxes)

12.4. Exploitation Phases

1. ID Theft (Keylogger, sniffer...)
2. Peer and Social Attacks (Trust misuse)
3. Local Attacks (Local Network, USB, VPN)
4. Attack Support (Host of Phishing Site, Proxying)
5. External/Noisy Attacks (Scanning, Spam, DDoS)

12.5. Command and Control Topology

The ability of a bot agent to locate the CnC infrastructure is a critical requirement for maintaining control of the entire botnet.

- **Star**: Single centralized CnC (+ Speed of control, - 1 point of failure)
- **Multi-server**: Multiple CnC communicating together (+ No single point of failure, + Geographical optimization, - Advanced planning required)
- **Hierarchical**: Multiple layers below CnC (+ Botnet awareness: single bot agent don't give entire botnet, Ease or resale, - Latency)
- **Random**: No centralized CnC (+ Highly resilient, - Latency, - Botnet enumeration)

12.6. Fluxing Technologies

Botnet takedown attempts typically target the CnC infrastructure

- **IP-Flux**: Constant change of IP address information related to a particular fully-qualified domain name (FQDN) (cnc.net→multiple IPs)
- **Domain Flux**: Constant change and allocation of multiple fully-qualified domain names (FQDN) (cnc.net, cnc.ru, abc.net→ multiple IPs)
- **Single-Flux**: The FQDN of the CnC's host has multiple IPs addresses assigned (DNS A record)
- **Double-Flux**: The name servers of the CnC's FQDN as well as the FQDN of the CnC's host have multiple IP addresses assigned (DNS A and NS records)

Defenders advantage: Sinkholing: register domain generated by the botnet's DGAs. "Confickr" generates 50,000 domains per day, and introduced non-determinism in DGA. Taking over all domains costs between $91.3 million and $182.5 million per year. Domain registrars are the entity best positioned to mitigate malware that relies on DNS.

12.7. Malware Development Process

1. **Create** unique samples of malware at massive scale
2. **Crypter**: encrypt malware to protect against signature detection
3. **Protector**: prevent debugging (against reverse engineering, sandboxing)
4. **Packers**: make it smaller + use polymorphism
5. **Binders**: hide it in another application
6. **Quality Assurance**: Test detectability before deployment

12.8. Takedown

Legal and technical measures to sever the connection between the CnC structure and the bot agents

12.9. Defense

- Firewalls
- Antivirus (up-to-date) but know its limits
- Patch your system
- brain.exe

13. Security Ecosystem and Detection Failures

A vulnerability is a weakness in software (or hardware) that enables an attacker to compromise the software or the data that it processes

13.1. Correlation of Detection Failures

Detection failures are **correlated** – they are not independent events
The combined failure rate of layered security is typically considerably higher than the product of individual failure rates. $P_{AB} \gg P_A \cdot P_B$
Ignoring correlation predicts zero exploits to bypass four or more devices

13.2. Security Information Provider

SIP = Private and government organizations that **collect** vulnerabilities, efficiently **monitor** the primary sources of security information, **validate** the content found, and **publish** their findings as security advisories in a consistent format (NVD, CVE). (e.g. CERT, X-Force(IBM), Secunia...)
NVD = National (USA) Vulnerability Database
CVE = Common Vulnerabilities and Exposures
SIPs inform the vendor in order to release a patch → average exposure time: 153 days
When the vendor is not informed about new vulnerabilities → average 0-day attack persists: 312 days

13.3. Disclosure Options

- Keep it secret (No disclosure)
- Quietly alert vendor
- Exploit it by-yourself
- Sell on Black Market (Cybercriminals, Governments) between 40k-160k
- Sell on White Market (purchase programs: ZeroDayInitiative, VulnerabilityContributorProgram or vendor) → Coordinated/Responsible Disclosure (if fails then Full Disclosure)
- Full Disclosure

Choice depends on **discoverer incentives**: economics, ethics, resources, past experience, publicity, legal constraints

14. E-Mail Spam

14.1. Generalities about Spam

More than 60% of email traffic is spam

14.1.1 Types of Spam

UCE: Unsolicited Commercial Email
UBE: Unsolicited Bulk Email (not necessarily commercial, sent in masses)

14.1.2 Where to get emails addresses

- Directory Harvest Attack (DHA) against mail server
- Email address crawlers
- Malware
- Harvesting email addresses from databases

14.1.3 Spam Sending Tactics

- Send from borrowed email accounts
- Send from short-term officially registered domain
- Use fake sender domains
- Sending spam from botnet machines
- Open mail relays (anyone can do it)
- Open proxies

14.1.4 Business Models behind Spam

- Direct sales business: some recipients actually buy spam advertised products; cheap advertising channel; easy to disguise actual sender; hard to enforce anti-spam laws, low risk of getting caught
- Internet underground business: operators rent out botnets for sending spam: 1million spam emails sent for 250-700$; 10million email addresses for 100euros

14.2. Anti-Spam Techniques

- **Filter at SMTP time**: +cheap and fast -lack info. to decide
- **Whitelists**: allow only emails matching non spam criteria
- **Greylists**: defer 1st email but accept follow-up email (issues with delays and multiple SMTP hosts of larger hosters)
- **Blacklists**: reject email matching spam criteria
  - DNS Blacklisting: can be done either with IP or domain name. URL which redirects to another URL → defeats Blacklists.
  - Blacklists caveats: evasion (botnet,stolen email, short term unpaid domains), denial of service (email on list don't work anymore→blacklists operators have power)
- **Filtering SMTP traffic at FW**: block outgoing SMTP msg not from SMTP server (prevent email-based worms)
- **Heuristical Content Filtering**: look at weird font, strange URL ...)
- **Statistical Content Filtering**:
  $P(spam|words) = \frac{P(words|spam)P(spam)}{P(words)}$
  Misspellings in text → defeats Naive Bayesian filter.
- **Traffic Based Filtering**: Compare flux of emails to recognize bulk mail (DCC = Distributed Checksum Clearinghouse)
- **Image Spam Filtering**: OCR on images before filtering
- **Text Filtering Caveats**: Normalize content before filtering text

14.3. Spam fighting at ETH

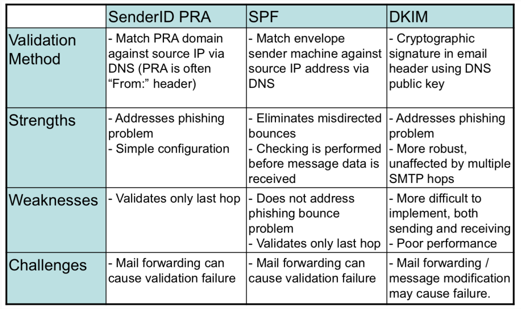
Bayesian filter not used (broad range of messages at University)
Running parallel SMTP sessions with filtering processes : this is a best effort service. They reject confirmed spam email directly. They keep in a "phish trap" suspected spam. They provide tag-only option for spam, and personal blacklist/whitelist
Phishing Countermeasures: (other than Phish trap)

- Set a daily limit on the number of messages that can be sent by a user
- Lock accounts sending spam/phish/malware messages or hosting a phishing web page
- Inform phishing recipients that their account may be "phished"
- Remove phishing messages from user mailboxes
- Redirect a phishing URL to a warning web page OR Block a phishing URL's IP-address

ETH is keeping logs for one year: sender-host/IP; sender/recipient address; subject header (40 characters); attachment file name; delivery/rejection status
For: filter modifications - check for false positives/negatives; message tracking; data mining - identify botnet members, etc.; statistics

14.4. Email Authentication

- **S/MIME**(Secure/MIME): sign and encrypt only msg content (not header)
- **PGP**(Pretty Good Privacy): auth + encry with Web of Trust
- **DKIM**(DomainKeys Identified Mail): Sign mails between servers
- **SPF**(Sender Policy Framework): List of IPs allowed to send emails
- **SenderID**: Match PRA domain against source IP via DNS (PRA is often From: header)



	SenderID PRA	SPF	DKIM
Validation Method	- Match PRA domain against source IP via DNS (PRA is often "From:" header)	- Match envelope sender machine against source IP address via DNS	- Cryptographic signature in email header using DNS public key
Strengths	- Addresses phishing problem - Simple configuration	- Eliminates misdirected bounces - Checking is performed before message data is received	- Addresses phishing problem - More robust, unaffected by multiple SMTP hops
Weaknesses	- Validates only last hop	- Does not address phishing bounce problem - Validates only last hop	- More difficult to implement, both sending and receiving - Poor performance
Challenges	- Mail forwarding can cause validation failure	- Mail forwarding can cause validation failure	- Mail forwarding / message modification may cause failure.

15. Guest Lectures

15.1. Trusted Computing by Swisscom

Trust is based on 3 conditions

1. **Positive Intention** (always positive intentions)
2. **Transparency and predictability**
3. **Reliability and continuity** (long strong trust)

Need to secure hardware first to have secure software :

1. Vendor Flash (MicroCode) initialize TPM
2. Microcode initiate boot sequence of Firmware
3. During boot time BIOS + tboot log every steps in TPM + hash it
4. Boot sequence reported during secure boot sent to attestation server
5. Attestation server checks with previously recorded reference values
6. Integrity confirmed or rejected (contact Whitelist server)
7. Integrity checked; boot or stop ($\simeq$3 minutes)

- **TPM**: Trusted Platform Module
- **TEE**: Trusted Execution Environment (e.g. Trust Zone)
- **TXT**: Trusted Execution Technology (SecureBoot)
- **SE**: Secure Elements (e.g. SIM- Card MobileID)
- **CIT**: Cloud Integrity Technology

15.2. Air Traffic Control by ArmaSuisse

Current Air Traffic Control use 2 systems synchronized:

- **PSR**(Primary Surveillance Radar): Independent surveillance made with ground based radar
- **SSR**(Secondary Surveillance Radar): Dependance surveillance using transponder-based interrogation

Problems: high cost, low accuracy
Future Air Traffic Control → **ADS-B**

- **A**utomatic: Always on (no explicit interrogation)
- **D**ependent: On-board system provides info to other parties
- **S**urveillance: Precise info (altitude, ID, speed)
- **B**roadcast: Send to everyone equipped to receive the data

Problems: No CIA(A), assume everything can be trusted

- Eavesdropping: cheap hardware and not encryption (flightradar24.com)
- Active attacks: message injection/deletion/modification
  - Ghost Aircraft Flooding → DOS on surveillance system
  - Virtual Trajectory Modification
  - ...

Solutions: Large Sensor Deployment to check, ...

- Time Difference of Arrival (TDoA) ⇒ Position validation
- Secure Track Verification (No need for tight time synchronisation)

15.3. Fighting Cybercrime as a CERT by SWITCH

SWITCH = non-profit foundation of Swiss Universities, ISP for Universities + in charge of the .ch + monitors flows to detect bots in Switzerland and fake .ch website (usually hosted outside of Switzerland)
CERT(Computer Emergency Response Team) = support, protect and prevent, international networked
Retefe Malware = Banking Trojan targeting CHE, SWE and JPN

1. Executable in an email installs a new root certificate + reconfigure proxy to re-route traffic for targeted banking websites.
2. When the victim try to reach his bank's website, the malicious DNS server returns the IP address of a rogue web-server containing a copy of the website.
3. No certificate warning from browser because of fake root certificate.
4. Victim types in his username, password and send it to the attacker through the fake website.
5. Then fake website propose to install a smartphone app to be able to forward the bank's mTan to the attacker.
6. Attacker has full control and use another compromised PC in the geographical area as a proxy to connect to the bank account of the victim.

Carbanak Cybergang stole $1bn from 100 banks directly. Malware install on admin computer (targeted attacks). Attackers spies on them to learn how to use the specific money transfer software of each bank.

15.4. Malware Analysis and Prevention by Symantec

The 3 Main Attack Vectors:

1. **Spear Phishing**: Email with link or malicious attachment
2. **Drive-by download attack**: Website uses an exploit to drop malware
3. **Supply-chain hack**: Compromise vendors and supplier software

Malware analysis procedure

1. Get a sample from customers, traps, crawlers...
2. Automated analysis (VirusTotal.com, ThreatExpert...)
3. Blackboxing(Sandboxing): monitor all operating system calls (behavior)
4. Whiteboxing: Disassembling/debugging code to understand it → need to bypass code obfuscation and understand the domain generation algorithm (DGA)
5. write description of the threat and create a signature to block it
6. start again (The analysis hamster wheel)

15.5. SCION Secure Next-Generation Internet Architecture

SCION is an isolation architecture only for the control plane, in the data plane it is a transparency architecture.

15.5.1 Isolation Domain (ISD)

Region that can agree on a common root of trust ⇒ authenticate entities only within each ISD (e.g. domain on geographical regions) political and legal issues arise

15.5.2 SCION Architectural Design Goals

- High availability, even for networks with malicious parties
- Adversary: access to management plane of router
- Communication should be available if adversary-free path exists
- Secure entity authentication that scales to global heterogeneous (dis)trusted environment
- Flexible trust: operate in heterogeneous trust environment
- Transparent operation: Clear what is happening to packets and whom needs to be relied upon for operation ⇒ **Packet-Carried State:** Packets carrying forwarding information
- Balanced control among ISPs, Senders, and Receiver
- Scalability, efficiency, flexibility

Beaconing for Route Discovery: Scalable and secure dissemination of path/topological information from core to edge. Policy-constrained multipath flood to provide multiple paths.

15.5.3 SCION Dangers

- Too many top-level ISDs , ISPs part of ISD core
- Large packet header size
- Too many extensions used
- Higher complexity (Extensions, PKI)
- Extremely high path fluctuations, changes