Software Security

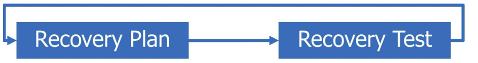
Risks for Companies

Ransom Demand: Lock Data and possibly steal it, faulty recovery, persistent backdoors, target painting, sanctions
Fraud: Using compromised accounts to "act" as the Company and commit fraud
Espionage: IP Theft, can be silent, insider threat, supply chain compromise
Misuse of Computing Ressources: Use compromised systems for other purposes
System Outage: Bring company systems down to cause damage over time
Sabotage: Destruction of physical or digital property, manipulate Data,
Data/Reputation Loss: Expose data, **misuse Brand Image**, **Violation of Regulations**

Threat Actors

State-nexus: Objective: Espionage, disruption, obtain money for their country Characteristics: Well-funded, high motivation, Methods: Advanced, large-scale, long-term operations
Cybercrime: Objective: earn money. Characteristics: opportunistic, provide services e.g., ransomware-as-a-service, Methods: various
Private Sector Offensive: Objective: earn money. Characteristics: normal companies in the cyber-surveillance industry, Methods: develop and sell cyberweapons, zero-day exploits, malicious software
Hacktivists: Objective: Influence politics or social changes, Characteristics: High motivation, Methods: DDoS, various

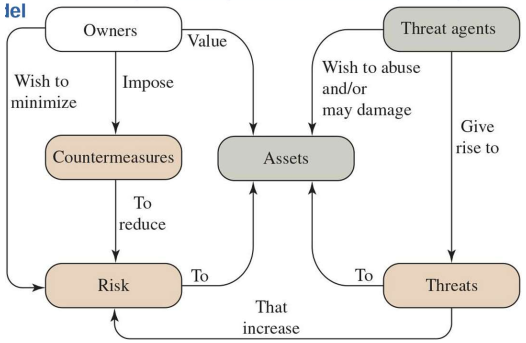
Disaster Recovery



Description of what to do
if something goes wrong
- Roles and responsibilities
- Processes
- Contact details
- Technical instructions

Tests of recovery plans
- Theoretical dry run
- Practical tests
  - Turn of a server or a DC
  - Restore data from a backup

Security Concepts and Relationships

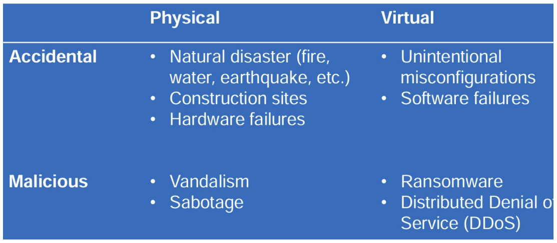


Security Concepts and Relationships

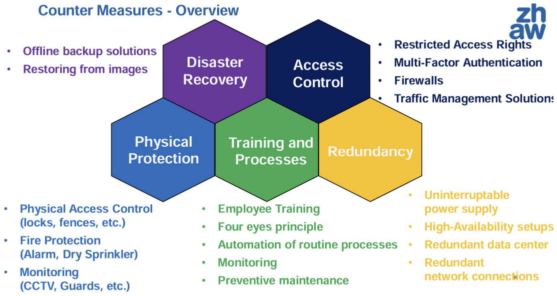
Confidentiality: Protecting information from unauthorized access or disclosure.
Integrity: Ensuring data is accurate, complete, and has not been altered or tampered with.
Authenticity: Verifying the genuine identity of a user, process, or system.
Availability: Ensuring that authorized users have reliable access to data/services when they need
Accountability: The ability to trace actions and changes back to a specific individual or entity.

Weakness => Vulnerability => Risk

Data Availability



	Physical	Virtual
Accidental	• Natural disaster (fire, water, earthquake, etc.) • Construction sites • Hardware failures	• Unintentional misconfigurations • Software failures
Malicious	• Vandalism • Sabotage	• Ransomware • Distributed Denial of Service (DDoS)



Counter Measures - Overview

- Offline backup solutions
- Restoring from images

- Restricted Access Rights
- Multi-Factor Authentication
- Firewalls
- Traffic Management Solutions

- Physical Access Control (locks, fences, etc.)
- Fire Protection (Alarm, Dry Sprinkler)
- Monitoring (CCTV, Guards, etc.)

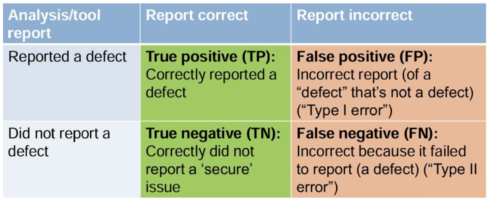
- Employee Training
- Four eyes principle
- Automation of routine processes
- Monitoring
- Preventive maintenance

- Uninterruptable power supply
- High-Availability setups
- Redundant data center
- Redundant network connections

The 7+1 Kingdoms

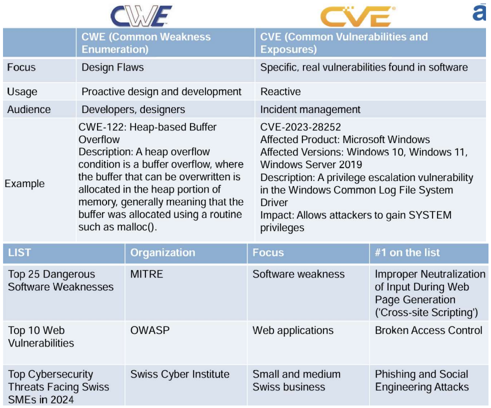
Input Validation and Representation: Sending malformed data, Syntax Validation & Semantic validation, Code injection
API Abuse: Calling APIs out of order or in unexpected ways, leading to unpredictable behavior, like no security check (testing API)
Security Features: Errors in the implementation of security features
Time and State: Race conditions and timing-related vulnerabilities, Time-of-(use=>check)
Errors: Too Much Information in Errors
Code Quality: Poor/complex coding practices that lead to unpredictable behavior.
Encapsulation: Unclear boundaries between trusted/untrusted actions, validated/invalidated, user data, different systems
Environment: deployment environment

Measuring Test Results



Analysis/tool report	Report correct	Report incorrect
Reported a defect	**True positive (TP):** Correctly reported a defect	**False positive (FP):** Incorrect report (of a "defect" that's not a defect) ("Type I error")
Did not report a defect	**True negative (TN):** Correctly did not report a 'secure' issue	**False negative (FN):** Incorrect because it failed to report (a defect) ("Type II error")

List of Weaknesses



	CWE (Common Weakness Enumeration)	CVE (Common Vulnerabilities and Exposures)
Focus	Design Flaws	Specific, real vulnerabilities found in software
Usage	Proactive design and development	Reactive
Audience	Developers, designers	Incident management
Example	CWE-122: Heap-based Buffer Overflow Description: A heap overflow condition is a buffer overflow, where the buffer that can be overwritten is allocated in the heap portion of memory, generally meaning that the buffer was allocated using a routine such as malloc().	CVE-2023-28252 Affected Product: Microsoft Windows Affected Versions: Windows 10, Windows 11, Windows Server 2019 Description: A privilege escalation vulnerability in the Windows Common Log File System Driver Impact: Allows attackers to gain SYSTEM privileges

LIST	Organization	Focus	#1 on the list
Top 25 Dangerous Software Weaknesses	MITRE	Software weakness	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
Top 10 Web Vulnerabilities	OWASP	Web applications	Broken Access Control
Top Cybersecurity Threats Facing Swiss SMEs in 2024	Swiss Cyber Institute	Small and medium Swiss business	Phishing and Social Engineering Attacks

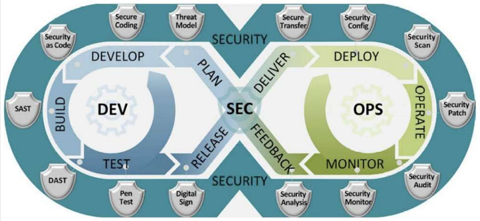
Memory safety

[bugs and vulnerabilities] when dealing with memory access, such as buffer overflows and dangling pointers. Very high % of vulnerabilities are these, especially on mobile. Memory safe languages help, but are not the complete solution, static analysis

Security Testing

Type of tests: Static (Verifying software by scanning text, early), Dynamic (Execute software in specific configuration, Fuzz, later), Hybrid (static and dynamic)
Fuzz testing: Standard (Random without knowledge of implementation), Mutation-based (use a (partial) valid input, called seed, and to mutate it), Greybox (Code coverage optimization=> with info from underlying system)
Human/manual analysis: Good at discerning context & intent, but expensive and can lose concentration, focus on specific issue
Pen testing: Done by pretending to break in, needs rules-of-engagement. Can be noticed internally but not stopped. Automated and manual execution
Red/Blue Team: Hired specifically for pen testing, Red=>Attack. Try to find deep flaws instead of Broad, Blue will try to stop it, meant to be ingenuous, creative attacks

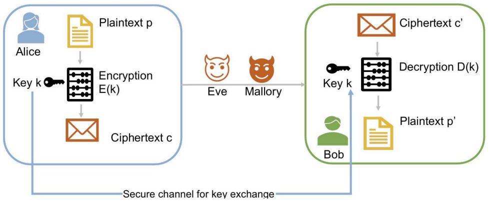
DevSecOps Security Lifecycle



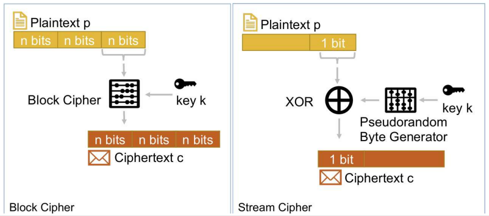
Ensure your entire stack, from the operating system to the web server and database, is patched and configured with security in mind

Encryption Model (Secret Key)

Goals: Confidentiality, Integrity, Authenticity, Freshness, Non-Repudiation



Practical Requirements: Algorithm kown to the public, no errors in design, Work Factor > 128 bits ($\log_2(Work\ Factor)$)



Attack types

Attacker | Encryption

Ciphertext-only: access only to the encrypted data and must try to deduce the key or message without any other reference. passive

Chosen-ciphertext: chooses specific ciphertext and looks at output, active role with system access to try and break encryption

Known plaintext: possesses a sample of encrypted data along with its plaintext. They use this pair to figure out the key.

Chosen plaintext: chooses a specific original message and gets the system to generate the corresponding ciphertext to analyze patterns.

Side Channel: Instead of attacking the math, monitors the physical performance of the system during encryption—such as power consumption, electromagnetic leaks, or processing time—to deduce the key.

Brute Force: Ciphertext-only attack. Calculate entropy of result to see if it worked. Work Factor: Average number of attempts, influenced by Encryption, Key Length and randomness of key

Secure Hash Functions

Variable Input lengths always produce fixed length output, impossible to derive input from output, different inputs => different outputs

Work Factor: output length in bits / 2

Secret Key Algorithms

Work Factor: With n key length: $(2^n + 1)/2$ or approx. for n>128 bits: 2^n

Initialization Vector: Block Ciphers act like math function, being deterministic for I/O. First block of plaintext is mixed with random IV => chain reaction to next blocks. IV is not a secret and is shared publicly, but must be UNIQUE

Authenticated Encryption: Provides Integrity and Authenticity via Message Authentication Code. Plaintext is hashed with the secret key of sender to produce MAC. For Symmetric Key

Mode	Name	Guarantees / Description	Recommendation
ECB	Electronic Code Book	None No IV or MAC is used.	Should not be used, as no real data protection is achieved.
CBC	Cipher Block Chaining	Confidentiality Only an IV is used, but no MAC.	Should not be used, as it does not protect against data modification.
CTR	Counter Mode	Confidentiality Only an IV is used, but no MAC.	Should not be used, as it does not protect against data modification.
CCM	Counter with Cipher Block Chaining	Confidentiality and integrity (authenticity) Both IV and MAC are used.	Can be used, but is slower than GCM and offers no other advantages.
GCM	Galois Counter Mode	Confidentiality and integrity (authenticity) Both IV and MAC are used.	This is the current standard and should be used whenever possible.

Public Key Crypto

Work Factor: Classic=4096 bits = 128-bit, Elliptic Curve=512 bits = 256 bits

Signatures: Hash plaintext with standard method, then encrypt it using private key and send along payload, same theory as MAC

Session Keys: Use Master key to generate a new session key every time => even if key is compromised only single session compromised

Diffie-Hellman Key Exchange (Symmetric):

1. Agree on public large prime number and generator (different specific number)
2. Both come up with **secret** number: a and b
3. Both perform $A = \ g^a\ mod(p)$ (same for b)
4. Both Share A/B with other
5. Both calc secret: $S\ =\ A^b\ mod(p)$ this gives the same S also for B^a

Man in the Middle: Acts as b for a and a for b

Encryption

RSA

Asymmetric mainly for signatures, can also encrypt.

Components: n (Modulus): A large number derived from prime numbers (key length), e (Public), d (Private)

Goal	Operation	Formula
Encryption	Hide data	$c = p^e \pmod n$
Decryption	Reveal data	$p = c^d \pmod n$
Sign	Prove origin	$s = p^d \pmod n$
Verify	Check origin	$p = s^e \pmod n$

Encryption in the Context of Network Protocols

Application Layer	Prepare data for sending	FTP, HTTP	SFTP, HTTPS, SSH
Transport Layer	End to end communication	TCP / UDP	TLS, QUIC
Network Layer	Route packets between networks	IP / ICMP	IPSec
Data Link Layer	Fragment packets for physical network	Ethernet	WPA3, MacSec

Higher layers:

- Easier to deploy (often included in applications)
- Typically provides end-to-end protection
- Less generally applicable (often optimised for an application)
- Examples: S/MIME primarily for e-mail, SSL/TLS only works on top of TCP

Lower layers:

- Difficult to deploy (may require adapting kernels, routers, switches,...)
- If done on layer 2, we only get hop-to-hop protection
- More generally applicable (protects all layers on top of it)
- Examples: IPsec can protect TCP- and UDP-based traffic, WLAN security measures protects all data between the client and the access point

Cryptology

Cryptographic Work Factor

The number of times it takes to come upon the correct key is called (cryptographic) work factor.

- Work factor (WF) = average number of keys to try
- Work factor is usually given in bits: $\log_2(n)$, $n = key\ space\ size$

Entropy

- Entropy is maximal if all outcomes are equally likely

$$H = \sum_{i=1}^{n} p(i) \cdot \log_2\left(\frac{1}{p(i)}\right), \qquad H_{binary} = \sum_{i=1}^{n} \frac{1}{n}\log_2(n) = \log_2(n)$$

Entropy of Cryptographic Keys

Cryptographic keys are typically created with random generators, so they can be considered as elements of a random variable.

What's the entropy of a key with 128-bit?

- Independent with equal probability: $128 \cdot 1 = 128\ bits$
- Dependent with inequal probability: $p(0) = 0.25,\ p(1) = 0.75 \rightarrow 128 \cdot 0.81 \approx 104\ bits$

Relation between entropy and work factor

- Work Factor ≈ entropy, key size = max. entropy ≈ max. work factor

Information-theoretically secure

- Intercepting a ciphertext tells you nothing about the plain

Computationally secure

- Work factor ≈ key entropy

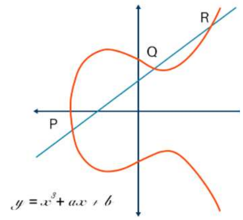
Funktion	Hash Länge	Work Factor
MD5	128 bit	64 bit
SHA-1	160 bit	80 bit
SHA-2	224 - 512 bit	112 - 256 bit
SHA-3	224 - 512 bit	112 - 256 bit

Elliptic Curve DH

Key Share: P-256=> 65 bytes (1 byte format identifier + 32 bytes for X coordinate + 32 bytes for Y coordinate), Modern only X is sent

Generation: Both agree on Curve Equation and base Point G. Both select random private number. Both add G to itself PN times=> Public Keys. Both take other Public Key and multiply by private number = Shared secret

TLS: Curve equation and Starting point are usually hard coded in a standard



Web

Certificates

Types: Domain (PublicK belongs to Domain, non-commercial domains), Organization (PublicK belongs to domain and owner was verified, Webshops), Extended (Additional attributes such as phone numbers were verified, Banking)

Let's Encrypt: free X-509 Certificates for TLS, only Domain Validation, 90 days, Automation of whole process

Domain Validation: Initiation => Challenge => Implementation => Validation => **Certificate Issuance CSR** (Key Generation for key pair(Agent on server)=> CSR Creation with domain name and the new Public Key S), Double Signing (First from agent with private key and authorized private key), Verification (both signature), Issuance of certificate

Certificate Validation: The Chain of Trust (Root=>Intermediate=>End), Additional Verification (Validity Period, Revocation, Domain name correct?, Server in possession of private key?)

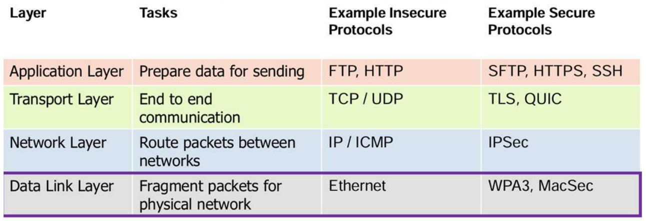
Root: anchor of trust, without no cert validation and no public key authentication, preinstalled in browsers, self-signed

Intermediate: Subject name matches the Issuer name of the End Certificate, verified by root public key

End: Owned by website, verified by intermediate public key

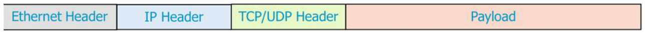
Revocation: Cant query every CA for CRL (Certificate Revocation List) instead Browser vendor summarizes => Download from all CA, compress, push regular updates, local checking. CRL are signed by the CA, so it cannot be revoked itself

TCP/IP Protocol Stack



Layer	Tasks	Example Insecure Protocols	Example Secure Protocols
Application Layer	Prepare data for sending	FTP, HTTP	SFTP, HTTPS, SSH
Transport Layer	End to end communication	TCP / UDP	TLS, QUIC
Network Layer	Route packets between networks	IP / ICMP	IPSec
Data Link Layer	Fragment packets for physical network	Ethernet	WPA3, MacSec

Packet Format:



TLS Client Authentication

Trigger: CertificateRequest during Handshake. Server defines which CA it trusts. Client sends public cert and signs a hash of all previous handshake messages using its **PK** => +1 Packet

Anonymous Client => Only Server is verified by Client; Server does not verify who client is

X.509

X.509 Field	Explanation
Version Number	v1, v2, or v3
Serial Number	Assigned to certificate by the issuer
Signature Algorithm ID	OID e.g. sha256rsa
Issuer Name	The X.500 Distinguished Name (DN) of the Certificate Authority (CA) that issued it
Validity Period	"Not Before" and "Not After"
Subject Name	(DN) of the entity the certificate represents (the subject)
Subject Public Key Info	Contains the OID of the public key algorithm (e.g., rsa) and the actual bytes of the subject's public key.
Extensions (optional)	technically optional, but some are mandatory
Certificate Signature Algorithm	Object identifier (OID) for signature algorithm
Certificate Signature	Bytes that make up the issuer's signature on the cert

Shibboleth (UNI)

Implements SAML, federated(Each Uni identifies its own students, then tells other Unis who they are) **Token based**: SAML Assertion XML contain Subject and key =value list, Signature of idP

Datagram Transport Layer Security, DTLS

Basics of TLS but for UDP. TLS relies on Implicit sequence number. DTLS adds explicit sequence numbers to every packet record. For handshake it adds own reliability functions for Timeouts and Retransmission or Reordering. Optional replay detection => duplicated records are silently discarded

QUIC => HTTP/3

Multistreamed: If one packet is lost other continue, one for JS,CSS,etc

Handshakes: Brand new connection has 1-RTT handshake=> Merge TCP and TLS handshakes, existing even 0-RTT(directly request encrypted data) by storing session ticket=>required!

User space: Currently runs in user space not Kernel

UDP: Reimplement TCP sequence numbers, ACKS, use Connection ID so connection stays, if IP changes

Let's Encrypt

Provide DV only certificates with validity for 90 days, fully automated with **ACME** (Signed Request with Domain=>Challenge=>Authorization=>CSR=>Certificate)

HTTP:LE provides Token, ACME client creates file with Token and account key, LE downloads=>Verified

DNS: LE provides Token, you create TXT entry in DNS

TLS-ALPN: LE provides Token, your server configured to accept ALPN TLS handshake, LE initiates handshake 443

CSR: Public Key, Distinguished Name(Domain, address, etc.), Signature of <. CA(or even yourself) signs this which creates .crt Certificate file

TLS 1.3 (1.2) ~~1.1~~ only for TCP user space

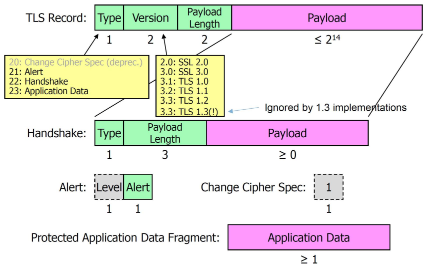
provides authenticated, integrity-protected and confidential data exchange. Secure against replay and deletion of messages

Cipher suite: Protocol, Encryption & Mode, hash for key

Record Protocol: Definition, all data is wrapped in a Record handles fragmentation and compression

Protocols: ^, Handshake, Alert, Application

Message Format:



Handshake: **1**(1.2). Client sends **ClientHello** (Version, Client random 32byte number, available cipher suites=> **ServerHello** (picked Cipher, Server random number) **2. Certificate** Server sends public key inside X.509 cert, client validates this against trusted Root CA **3. ServerKeyExchange** Server generates DH params and signs with PK => **ServerHelloDone** (all from server above sent in same package now) **4. ClientKeyExchange** Client creates its own DH params and sends to server=>Both can calculate Pre-Master Secret **5.** Key calculation PMS is raw and not uniform random => PMS+ the 2 random nr are input into PRF=> Master Secret, used to generated Keyblock, which is again spliced up into 6 different keys **6. ChangeCipherSpec** Client signals to server, everything after this will be encrypted, sends and **Finished** contains a hash (MAC) of all previous handshake messages (all client above in same packet now) **7.** 6 again but from server side. Both try decrypt MAC and verify hash, if it doesn't match=>dropped=> Man-in-the-Middle check

Phases: Handshake, Data Exchange (secure tunnel is established), Connection Teardown (both sides know)

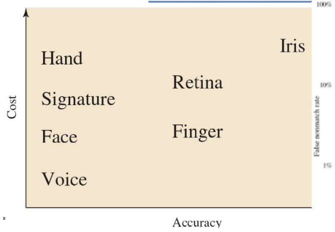
1.3: ClientHello contains **Key Share** for ECDH, ServerHello contains same from Server => After this encryption starts=>1-RTT Wrong "Guess" at start => HelloRetryRequest from Server => Client sends Hello again

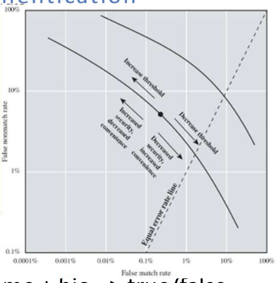
Authentication

Methods & Passwords

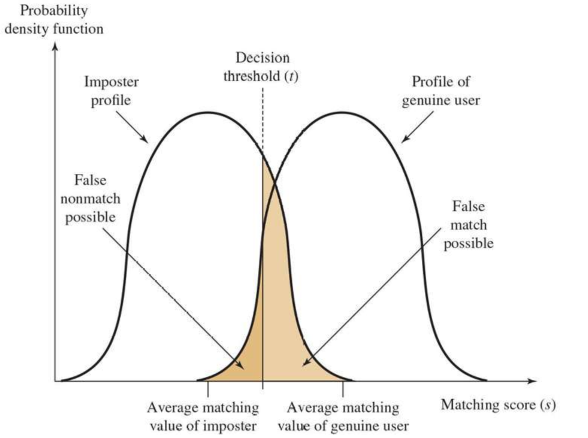
Know: Passwords **Possess**: Token, phone **Be**: Biometrics **Passwords**: Not enough=> Intelligent guessing, trial and error, phishing, leaks, reuse. **Work Factor** in bits $\log_2(possibilites)$. Hash with bcrypt and Argon2 **Storing Passwords**: Only transmit over secure channel. Hash passwords, so server does never store plain text but **Dictionary Attack**: Reverse engineer hash by using table **Salting**: prevent^ Store besides password hash, so hash attacker cannot use precompiled hash table **Pepper**: Additional secret stored separately from passwd **Key stretching**: Use hash multiple times using (some)

Biometric Authentication





Verification: User gives username + bio => true/false **Identification**: Only Bio => identity result **Matching Score**: False Match Rate (FAR) = probability to assess two different people as the same. False Nonmatch Rate (FRR) = probability to assess two samples of the same person as not the same. Accuracy = 1 – (FAR + FRR)



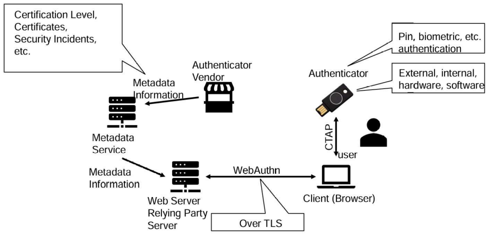
Attacks: Presentation(Impers) => PAD(liveness detection)

MFA

Addition to password, does not prevent inline phishing! **Fatigue**=> spam user with MFA requests, instead require number. If only mobile is used=> can be stolen or hacked => Sim swap.

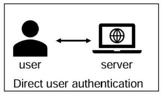
Password-less

A cryptographic key is released by a biometric factor which is then used for the login.

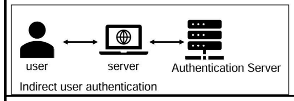

Certification Level, Certificates, Security Incidents, etc.
Authenticator Vendor
Metadata Information
Metadata Service
Metadata Information
Web Server Relying Party Server
WebAuthn
Over TLS
Client (Browser)
CTAP
user
Authenticator
Pin, biometric, etc. authentication
External, internal, hardware, software

Authenticator creates a new keypair for every login => private key never leaves. For Login it signs clientDataHash with that pair. Is secure and requires no password but challenging transition and dependent on working hardware

Direct and indirect user authentication



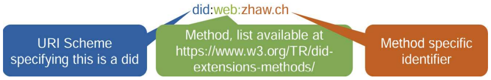
- Configuration of authentication mechanisms and credentials directly on every server
- Doesn't scale well with larger environments
- Do not send any passwords in cleartext!



- Allows to centralize authentication and management of credentials
- Scales with larger environments
- Allows for Single Sign-On

SSI Key Components

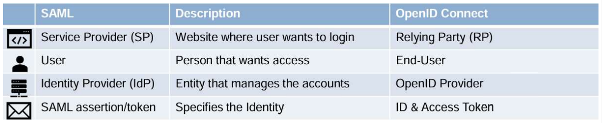
Decentralized Identifiers (DID): For issuers and holders, globally unique URI, public and private key, stored in Verifiable Data Registry (store public keys, revocation list, usually blockchain)


did:web:zhaw.ch
URI Scheme specifying this is a did
Method, list available at https://www.w3.org/TR/did-extensions-methods/
Method specific identifier

Verifiable Credentials (VC): verifiable information (claims) about owner =>certificate that are signed by issuer but can be verified without **Verifiable Presentation (VP)**: Only certain claims, signed by user to prove ownership

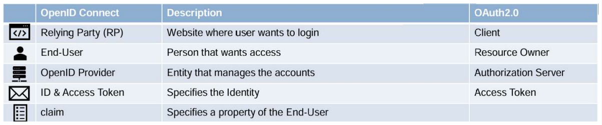
Single Sign On SSO

Kerboros: LAN in a single company with Active Directory **SAML**: Over the Internet different companies, Vendor independent, Web Apps, focus on enterprise, --privacy



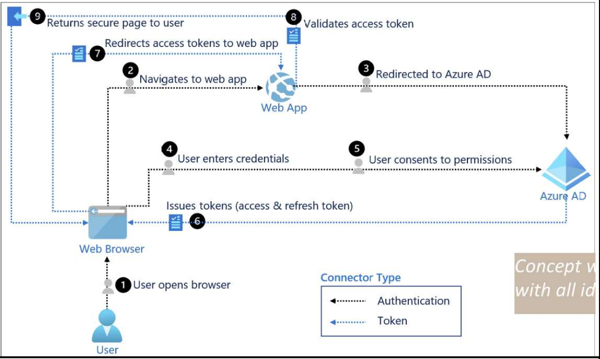
	SAML	Description	OpenID Connect
&lt;/&gt;	Service Provider (SP)	Website where user wants to login	Relying Party (RP)
👤	User	Person that wants access	End-User
☰	Identity Provider (IdP)	Entity that manages the accounts	OpenID Provider
✉	SAML assertion/token	Specifies the Identity	ID & Access Token

OpenID Connect: Over the internet, Mobile Apps, APIs, used by Google etc, **(1)** RP sends «login» request to OP **(2)** OP authenticates End-User and obtains authorization to send data to RP **(3)** OP responds with ID and Access Token to RP **(4)** RP asks OP for user profile information **(5)** OP replies with claims containing user profile information



	OpenID Connect	Description	OAuth2.0
&lt;/&gt;	Relying Party (RP)	Website where user wants to login	Client
👤	End-User	Person that wants access	Resource Owner
☰	OpenID Provider	Entity that manages the accounts	Authorization Server
✉	ID & Access Token	Specifies the Identity	Access Token
☰	claim	Specifies a property of the End-User	

OAuth 2.0: Only Authorization, used by OpenID allow login without sharing password, but too complex **Abstract Protocol Flow**: Protected Resource owner, Resource server, Client Entity, Authorization server



Self-Sovereign Identity (SSI) 10 Principles

Moves control of identity to user, Trust relationship between issuer and verifier, uses wallet, Avoids knowledge gain at issuer **Existence**: right to a digital identity. **Control**: complete **Access**: always **Transparency**: Systems and algorithms must be open and traceable. **Persistence**: long term, no specific providers. **Portability**: between different platforms. **Interoperability**: for Systems **Consent**: explicitly for data. **Minimization**: of data shared. **Protection**: secure and protected from unauthorized access.

Kerberos Single

Authentication Service (AS): User receives a Ticket-Granting Ticket (TGT), Ticket is valid for a specified lifetime, usually used once per day, it proves the users identity to the TGS, it includes Session Key + ID **Principal**: Unique Identity **Ticket-Granting Service (TGS)**: Issues Tickets, which are encrypted with the target servers key => allowing specific access **Walkthrough**: **1.** Alice creates secret Key by using hash on her password **2. AS_REQ**, Alice sends request to AS with username, timestamp which is encrypted by her Key **3.** AS verifies this with its locally stored key **4. AS_REP**, AS generates Session Key encrypted with Alice Key and TGT encrypted with the TGS key **5.** Alice decrypts the Session Key **6. TGS_REQ**, Alice sends TGT + **Authenticator** (Contains ID + timestamp encrypted with session key) to TGS **7.** TGS decrypts TGT for session key, with that decrypt auth and verify **8. TGS_REP** TGS Sends back service Ticket encrypted with Sams Key, containing new Session Key, and new S.K encrypted with old one **9. AP_REQ** Alice sends Sam S.Ticket + new Auth **10.** Sam encrypts Service Ticket and verifies Auth **11. AP_REP** Sam encrypts timestamp with Session Key to Alice=>prove ident.

Authorization & Privacy

Building blocks

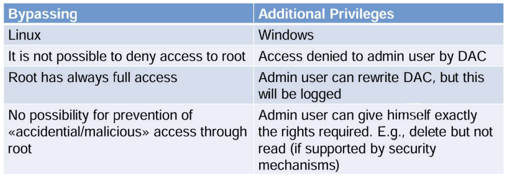
Access control model: Conceptual framework dictating how subjects access

Security policy: defines who is allowed to do what on a system Need to know vs. Need to protect=>. Drivers: business), security (security, integrity, and availability), regulatory. Creep

Security mechanism: Implements policy. Access Matrix or List **Access Management Process**: Request Access => Verification => Providing Rights => Monitoring identity status => Logging and tracking access => Removing/restricting access

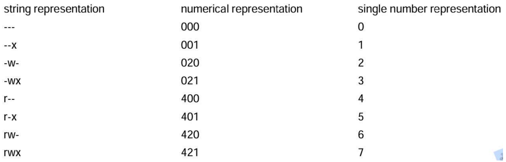
Discretionary Access Control (DAC)

Owner decides who and to what degree has access, typically who created the object, usually also bypass possible (root)



Bypassing	Additional Privileges
Linux	Windows
It is not possible to deny access to root	Access denied to admin user by DAC
Root has always full access	Admin user can rewrite DAC, but this will be logged
No possibility for prevention of «accidential/malicious» access through root	Admin user can give himself exactly the rights required. E.g., delete but not read (if supported by security mechanisms)

POSIX: - No access x Execute r Read w Write always done in order owner, group, others



string representation	numerical representation	single number representation
---	000	0
--x	001	1
-w-	020	2
-wx	021	3
r--	400	4
r-x	401	5
rw-	420	6
rwx	421	7

Privacy Enhancing Technologies

Advanced Encryption: Homomorphic Encryption(Allow computation directly on encrypted data), Secure Multi-Party Computation (join data for computation, without sharing), Trusted Execution Environment(HW modules)

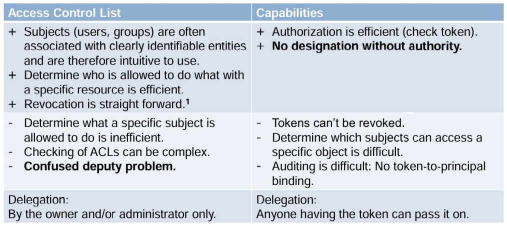
Federated Learning: ML learning but data stays local (mostly) like autocomplete

Obfuscation: Anonymization or Pseudonymization(Instead of Address, Neighbourhood, Birthdate to age)

Differential Privacy: You can't tell if someone is in the dataset or not => Add noise to the data=>Plausible deniability?

Capabilities

Unforgeable token (ticket) owned by a subject that contains the permissions for specific objects. In OS ACLs are used to construct the capability



Access Control List	Capabilities
+ Subjects (users, groups) are often associated with clearly identifiable entities and are therefore intuitive to use. + Determine who is allowed to do what with a specific resource is efficient. + Revocation is straight forward.[1]	+ Authorization is efficient (check token). + **No designation without authority.**
- Determine what a specific subject is allowed to do is inefficient. - Checking of ACLs can be complex. - **Confused deputy problem.**	- Tokens can't be revoked. - Determine which subjects can access a specific object is difficult. - Auditing is difficult: No token-to-principal binding.
Delegation: By the owner and/or administrator only.	Delegation: Anyone having the token can pass it on.

Confused Deputy: Do something and write output to in theory not accessible file. Solution => Provide write capability for operation

Mandatory Access Control (MAC)

Access control is mandated by the system=>system wide policy. User cannot edit only security policy admin => high degree of control **Mandatory Integrity Control (MIC)**: Based on Integrity Levels (IL) assigned to processes and objects

Role-Based Access Control (RBAC)

Core RBAC: min elements to achieve RBAC

Principle of least privilege: Define roles for actual jobs with only actually needed rights

Separation of duties: Mutually exclusive

Data abstraction: Instead of giving perms to individual files, give rights for entire Tasks

Limitations: Not directly supported by OS

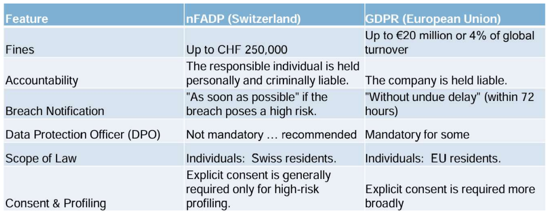
GDPR

For EU Residents, Anonymous data is exempt

Privacy by Design & by Default: Only collect minimum necessary data (data minimization)

Data Subject Rights: easily get a copy of their data(Access), permanently delete user data upon request(Erasure), export/Portability

Security of Processing: Implement encryption, pseudonymization, and robust access controls. Regularly test your security measures and respond to new threats. Treat data security as a core functional requirement, not a feature.



Feature	nFADP (Switzerland)	GDPR (European Union)
Fines	Up to CHF 250,000	Up to €20 million or 4% of global turnover
Accountability	The responsible individual is held personally and criminally liable.	The company is held liable.
Breach Notification	"As soon as possible" if the breach poses a high risk.	"Without undue delay" (within 72 hours)
Data Protection Officer (DPO)	Not mandatory … recommended	Mandatory for some
Scope of Law	Individuals: Swiss residents.	Individuals: EU residents.
Consent & Profiling	Explicit consent is generally required only for high-risk profiling.	Explicit consent is required more broadly

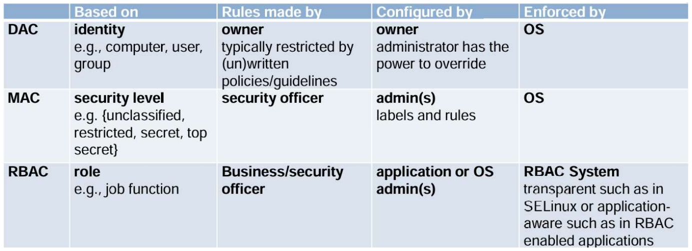
Fundamental: Data minimization, Automated data deletion, Data inventory, Access controls, Notice and Choice (Like Cookies request)

Privacy vs Security vs Data Protection

Privacy "Identified or identifiable natural person." =< human rights, dignity, and the individual's control over their personal info. **Security** "Company data." It focuses on the protection of assets and information (Confidentiality, Integrity, and Availability) belonging to an organization. Privacy needs Security. **Data protection** technical measures to protect privacy of data

Attribute Based Access Control

Definition of Roles + Attributes that enrich that role. Used in many layers OS=>Application=>DB>Frameworks



	Based on	Rules made by	Configured by	Enforced by
DAC	**identity** e.g., computer, user, group	**owner** typically restricted by (un)written policies/guidelines	**owner** administrator has the power to override	**OS**
MAC	**security level** e.g. {unclassified, restricted, secret, top secret}	**security officer**	**admin(s)** labels and rules	**OS**
RBAC	**role** e.g., job function	**Business/security officer**	**application or OS admin(s)**	**RBAC System** transparent such as in SELinux or application-aware such as in RBAC enabled applications

Anonymize Data

Linkage Attack: you never know exactly what outside(Background) information an attacker already possesses **Quasi-Identifiers**: Linking together multiple "common" attributes **Dimensionality**: In high dimensions(features), almost every record is unique. => **Data Utility**: more removal=>less useful

Kerberos Realms

Realm: A logical network or trust boundary (e.g., Wonderland or ZHAW) under the control of a specific KDC

KDC (Key Distribution Center): Central authority that stores secret keys for all users and services=> Split up in AS+TGS

Security Analysis: v5 is considered secure, timestamps against replay but user key is derived from password so offline attack possible. It needs synchronized time among users=>define timeframe (1m)

Across Realms: Realms KDC need to share a secret key. **1.** Alice performs local AS_REQ + AS_REP to receive TGT from local TGS **2.** TGS_REQ, Alice Asks local TGS for ticket to foreign TGS **3.** TGS_REP, Local TGS issues inter-realm ticket encrypted with shared key **4.** TGS_REQ Alice contacts foreign TGS with this ticket and requests access to specific server **5.** TGS_REP, The foreign TGS verifies the inter-realm ticket and issues a service ticket for Server Sam **6.** AP_REQ + AP_REP for foreign Sam

SSI Walkthrough

Definitions: Holder (Identity owner), Issuer (Attest claims about user and issues VC), Verifier (Requested proof of user claim), Verifiable Presentation (Subset of claim, signed by Holder), Registry (Blockchain)

Registration: User generates a DID and a corresponding public/private key pair and stores it in the Verifiable Data Registry

Issuance: User Requests Credential from Issuer, Issuer verifies in RL and creates a VC, user stores this in personal wallet

Presentation: User selects which claims to share, wraps these in VP=>send to Verifier

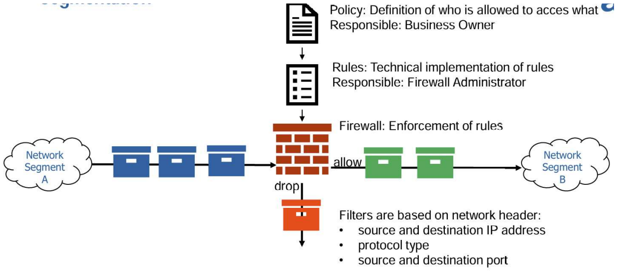
Verification: Verifier retrieves public key of Issuer from registry and verifies, same for Holder, if both matches => **Trust decision** =>Verifier must trust I&H secured their PK

Network

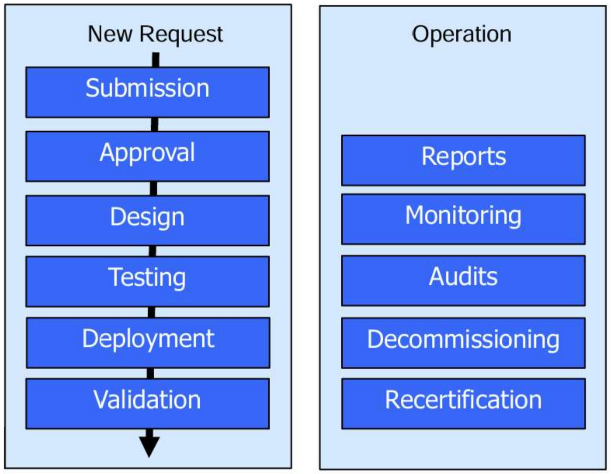
Segmentation & Firewall

Within each segment, machine trusts each other. Improve performance by “hiding” traffic in logical networks,

Packet Filtering Firewalls: Achieve Segmentation


Policy: Definition of who is allowed to acces what
Responsible: Business Owner
Rules: Technical implementation of rules
Responsible: Firewall Administrator
Firewall: Enforcement of rules
Network Segment A
allow
drop
Network Segment B
Filters are based on network header:
source and destination IP address
protocol type
source and destination port

Firewall Rule Management Processes: Otherwise, it grows and gets too complex, reassess existing rules



New Request

Submission

Approval

Design

Testing

Deployment

Validation

Operation

Reports

Monitoring

Audits

Decommissioning

Recertification

Firewall Rule Management Policy

Benefits: Blocks a lot of unwanted traffic, controls access at centralized point, hide internal network structure

Limitations: Only against outside attacks or application spe

Next Generation Firewall: Deep Packet Inspection, Intrusion Prevention, Application Awareness, Antivirus, Sandboxing

Microsegmentation: as small as one machine or a «workload» in cloud setups Tradeoff: Security vs. Manageability

Endpoint Detection and Response (EDR): Protection (Host local firewall, Antivirus, Antimalware), Monitoring (Anomaly detection, Integrity Checks), Investigation and Response (Isolation of devices, Collection of evidence, Rollback)

Zero Trust

Least privilege access is enforced, Security monitoring is implemented

Function	Traditional	Initial	Advanced	Optimal
Traffic Encryption (Formerly Encryption)	Agency encrypts minimal traffic and relies on manual or ad hoc processes to manage and secure encryption keys.	Agency begins to encrypt all traffic to internal applications, to prefer encryption for traffic to external applications[27], to formalize key management policies, and to secure server/service encryption keys.	Agency ensures encryption for all applicable internal and external traffic protocols,[28] manages issuance and rotation of keys and certificates, and begins to incorporate best practices for cryptographic agility.[29]	Agency continues to encrypt traffic as appropriate, enforces least privilege principles for secure key management enterprise-wide, and incorporates best practices for cryptographic agility as widely as possible.

Single Point of Failure: at the Policy Decision and Enforcement Point or misconfig

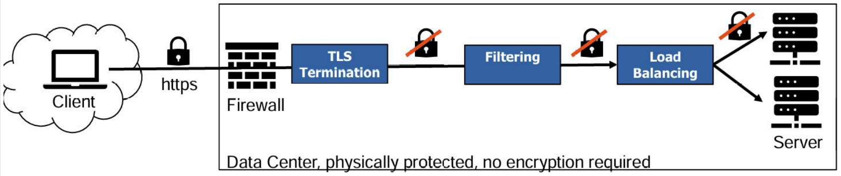
Misused credentials: No protection against phishing and malicious insider.

Lack of network visibility: due to encrypted traffic **Access of monitoring data**

Misuse of agents: (unpersonal management accounts) through attacker

Application-Level Firewalls

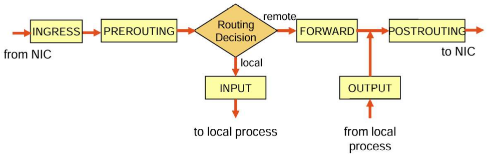
Understands higher layer protocols => look into packets. **Web Application FW:**



FW must deny all direct requests, DNS is configured to point to TLS termination, Clients must be configured with WAF certificate to avoid certificate warnings

Netfilter and the Linux kernel

Netfilter allows to access packets in the network to analyze extract and delete them. **nftables** is a packet classification and mangling framework that runs on rulesets that are applied to the packets.



Policies: accept (continue to process the packet,) drop (stop processing the packet), reject (stop processing the packet and tell the sender), jump (continue processing elsewhere)

```
chain tcp-traffic {
  type filter hook input priority 0; policy drop;
  tcp dport { https, http } jump http-traffic
}
```

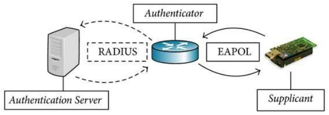
IEEE 802.1x

Supplicant: the client device

Authenticator: the network device (switch or Wi-Fi access point)

Authentication Server: usually a **RADIUS** server

At start Port is in a blocked state=> only EAPOL traffic is allowed. After Auth port is in authorized state. This allows flexible checking of who and what is connecting but requires that the end device supports the framework.



Secure Web Gateway (SWG)

(Forward-) Proxy with URL Filtering, Data Leakage Prevention (Blocking sending of sensitive info=>requires tagging), TLS Inspection

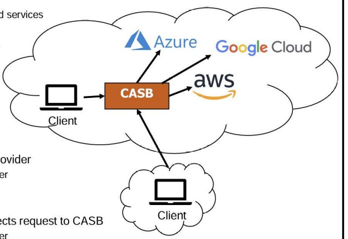
Cloud Access Security Broker (CASB)

Capabilities

- Shadow IT discovery
  - Generation of reports of used cloud services
- Cloud usage Control
  - Set access rights to cloud services
- Data leakage prevention
  - Set policies for data sharing
- Anomaly detection
  - Alerts on unusual behavior
- Etc.

Implementation

- API Scanning: Direct API to cloud provider
  - Only works for known cloud provider
- Forward Proxy: similar to SWG
  - Only works for managed devices
- Reverse Proxy: cloud provider redirects request to CASB
  - Only works for known cloud provider

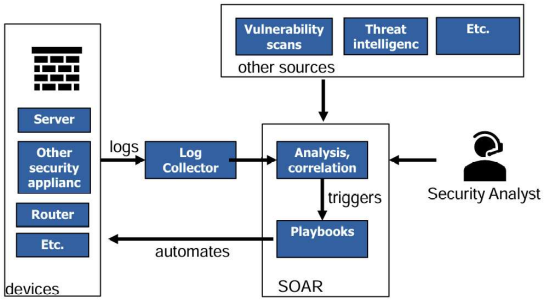


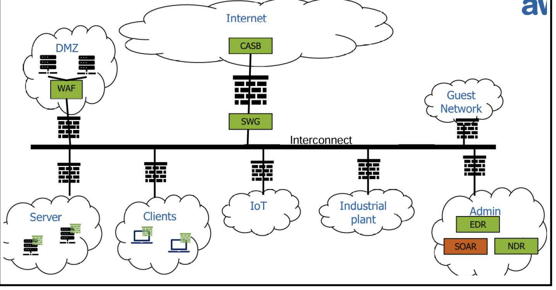
Network Detection and Response

Continuous monitoring of network traffic generates baseline => detect anomalies => automatic remediation of incidents (Changing Firewall configuration, isolating an infected device from the network.)

SIEM – Security Information and Event Management: Correlation of events provided by different log sources to find malicious activity in single dashboard

SOAR (Security Orchestration, Automation and Response): Extended SIEM and automate response





AI & Lower Level

Virtual Private Networks

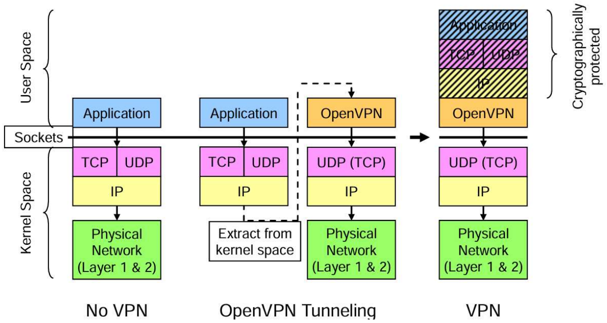
outsiders can neither read nor modify the data transmitted between participants. Virtual means that the privacy (protection) is not achieved by dedicated network links **VPN gateways** are the endpoints of the secure channel and apply and remove the cryptographic protection. The **secure channel** between VPN endpoints is often identified as **secure tunnel**. TLS is only on top of TCP => connection between App. VPN works between hosts

Usecases: public VPN, tunnel between networks, tunnel to company, Connecting Supplier to Internal Network (Security of their devices not certain, do we have a contract?)

Internet Protocol Security (IPSec): enable secure end-to-end at IP layer, it has different modes but only VPN is really used, implemented in kernel. Internet Key Exchange (IKE)

Encapsulating Security Payload (ESP): Hides internal IP addresses => private addresses. Does not work with NAT as no ports are used (can be solved with NAT-T)

OpenVPN: application-layer tunnel that runs on top of UDP (or TCP but bad performance). Uses TLS handshake with simple ACK-based mechanism for reliability. **Packet** consists of header and payload, opcode in header first 5 bits message type, last 3 key_id for handshake keyset. Payload+Seq. Nr+Padding to achieve cipher block, encrypt, compute and prepend HMAC



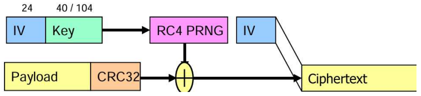
Wireguard: Layer 3. Not much configurable => small attack surface. Has a 1-RTT handshake(low cost) and some Dos protection. Runs in kernel(instead of extract directly to physical above) und UDP only => good performance

Wireless

Packet Sniffing in range and unauthorized use => Encryption and Authentication at Layer 2 required. **WEP** all clients and AP use same pre-configured long-term key to encrypt frame. IV only 24 bits=> wait until IV is reused, then XOR to get plain text, then keystream. Afte a while all streams are known even though the key itself is never discovered. weakness of RC4.

Why? driven by performance optimizations

The attacker knows PL_A → he also knows $CRC(PL_A)$
This allows to get the keystream: $KS = C \oplus PL_A||CRC(PL_A)$
The attacker knows PL_B → he also knows $CRC(PL_B)$
This allows to create the modified ciphertext: $C' = PL_B||CRC(PL_B) \oplus KS$
After decryption, the recipient gets $PL_B||CRC(PL_B)$, which will be accepted

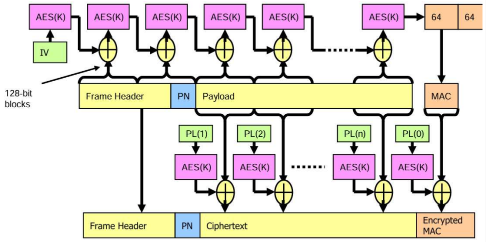


C1 xor C2 = P1 xor P2; C1= K XOR P1

WPA: Enterprise with port-based access or again with PSK. Periodic re-keying, typically after one hour (avoid IV wrap around). Each Client uses its own key material => Users cannot read the unicast data of other users

Temporal Key Integrity Protocol (TKIP): HW compatibility with WEP. For each frame, generates an individual encryption key. This key is used to initialize RC4, which is used to encrypt the frame. A MAC using the Michael algorithm is appended

Counter Mode CBC-MAC Protocol (CCMP): Available in WPA2 and WPA3 (as of 2014, also in WPA). Based on the Advanced Encryption Standard (AES). Guarantees confidentiality and authenticity/integrity



Definitions

Features: Novel (still learning), Easy to use (incompetent), Training data (bad parts also, Complex and large, Security issue, non-deterministic LLMs optimize for plausibility, not correctness

Prompt Injection: Direct (the prompt) or Indirect (external data). Can be solved my adding more LLMs or meta prompt (factory settings)

Language Bias: Non-English usually weaker =>unable to detect phishing or other risks

Data and Model poisoning: Attackers plant malicious patterns in public or training data so the model learns harmful behavior; it is hard to detect because the poisoning is mixed with vast legitimate data.

System Prompts: They define how the model behaves and what it knows; leaking them allows attackers to bypass controls or extract sensitive information.

Prompt techniques: Root (give n-shot number of examples), Refinement-based (RCI (Recursive Criticism and Improvement), Self-refine (the model itself), and Progressive hint (giving hints step-by-step)), Decomposition (Least-to-most (solving simpler sub-problems first) and Self-planning (model generate a plan before writing code)), Reasoning (model to articulate its logic and thought process), Priming (Persona pattern (assigning a role) and Memetic Proxy (using cultural references or proxies to guide behavior))

AI in security engineering

Cyber wage burnout: Each new platform (cloud, SaaS, IoT, AI, endpoints) adds new inputs

Detection engineering: High-quality detections drive everything else — triage, investigation, response, automation, and posture improvement all fail if alerts are noisy or wrong

Triple-A: Applicable means it targets real threats, Actionable means it gives enough context to respond, and Accurate means it minimizes false positives.

DDoS

Attacks: Network bandwidth (Send huge amount of network traffic.), System resources (Send specific types of packets that consume the resources or crash system.), Application resources (Send many valid requests that consume available resources.)

DNS Amplification Attack: Request data from DNS Resolver with a spoofed IP source address

SYN Flood Attack: Send spoofed TCP Syn packets, target opens all available ports => no more

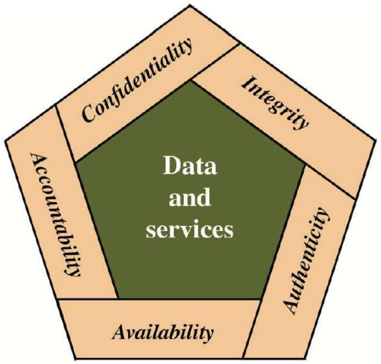
HTTP Flood: REST overload

Solution: Overprovisioning or Hardening, GeoIP Blocking, WAF

ARP Poisoning: Attacker has access to network. Forged ARP tells 2 victims that the attackers MAC is the router

Data and services

Often reduced to only Confidentiality, Integrity and Availability => **CIA**

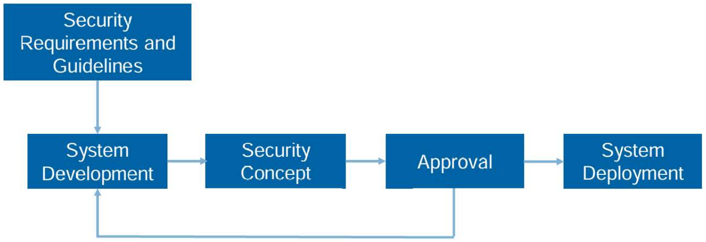


Agentic AI

AI agents can reason, plan, use tools, and adapt autonomously, while SOAR only executes predefined playbooks. The AI agent analyzes, correlates, plans, and executes containment in minutes, but humans require long triage and investigation cycles

Security Concept

Processes



Security Requirements and Guidelines

System Development → Security Concept → Approval → System Deployment

General Content: System Requirements, System Architecture, Data, Security Controls, Access Control, Logging and Monitoring, Backup and Recovery, Encryption, Software Development and Operation, Threat and Risk Management

Data Inventory:

Data	Description	Criticality
User Profiles	User profiles as well as the weather data that is not shared	medium
Weather Data	Collected data that is publicly available	low
Public Data	Website Content Product Documentation	low
Product Specification	Required for Production of Devices	high
Source Code	For Sensors, Processing as well as Webpage	high
Contracts	With suppliers and large customers	high

Role Based Access Control: Also Authentication table for Roles

	Public sensor data	Private sensor data	Product specification	sensor	Etc.
Anonymous	read	-	read (via webpage)	-	
Owner (logged in)	read	read (own data)		-	
Application on sensor	write (own data)	write (own data)		-	
Product developer	read	-	write read	-	
Sales personnel	read	-	read (direct)	-	
Update server	-	-	-	write	
Etc.					

Transparent Data Encryption: is a security technology used primarily by database management systems (like SQL Server, Oracle, and Azure SQL) to encrypt **data only at rest**. Its primary goal is to protect the physical files of the database so that if a hard drive or backup tape were stolen, the data would be unreadable to the thief

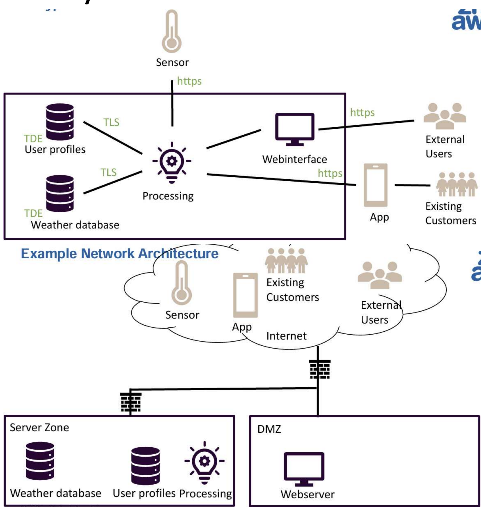
Threat modeling: identifies possible attacks, and risk assessment ranks them by likelihood and impact so resources are spent on the most dangerous risks.

Basics

Written concept ensures that all stakeholders agree on security goals, responsibilities, and controls, and that security is systematically applied instead of ad hoc. **Requirements** define what the system must do, while **architecture** defines how components interact; **security controls** must be chosen to protect these specific data flows and components. **RBAC** ensures that each role (sensor, owner, developer) only has the minimum permissions needed, preventing unauthorized writes or data leaks.

Example: Sensor Data Aggregation

System Requirements: Sensors shall measure the temperature, humidity and noise; The measurements shall be stored centrally; As much measurements as possible shall be shared with the public; Owners shall have a private view of their measurements and the possibility to make all their data private; All measurements shall be used to develop a weather forecast that is accessible by the Public **System & Network Architecture:**



Backup and Recovery

System	RTO	RPO	MAO	Amount of data	Backup Frequency
User Database	8h	1h	72h	300 MB	hourly
Measurement Database	8h	72h	72h	80 GB	daily

RTO: Recovery Time Objective: defines the maximum tolerable downtime for a system or process after a disruption, setting a target for how quickly operations must be restored to avoid unacceptable business impact.

RPO: Recovery Point Objective: the maximum amount of data an organization can afford to lose after a disruption from the last good backup to the failure point.

MAO: Maximum Acceptable Outage: The longest period an organization can sustain an outage before severe business impact or failure.

Example Backup Policy: 3 – 2 – 1 – 1 Rule=> 3: Maintain three copies of data 2: Use two different types of media for storage 1: Keep at least one copy off-site 1: Keep at least one copy read only

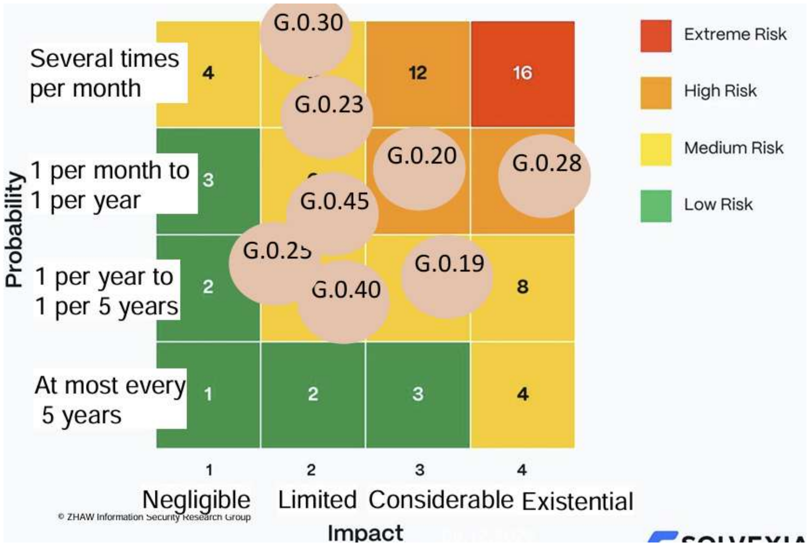
Data Encryption: The backup does not need to be encrypted, as the original data is encrypted

Admin: There is a dedicated Backup admin, the normal Admin has no access to the Backup system. Admins access is only temporal, has MFA enabled and all access is logged

Monitoring: All Backup Jobs are monitored. If failed, an alert is generated. Backup Logs are evaluated weekly

Recovery: Data recovery tested once a year. Emergency recovery only with explicit permission of incident manage

Example Risk Assessment:


Several times per month
1 per month to 1 per year
1 per year to 1 per 5 years
At most every 5 years
Probability
G.0.30
G.0.23
G.0.20
G.0.28
G.0.45
G.0.25
G.0.40
G.0.19
Negligible
Limited
Considerable
Existential
Impact
🟥 Extreme Risk
🟧 High Risk
🟨 Medium Risk
🟩 Low Risk
© ZHAW Information Security Research Group
SOLVEXIA