Cyber Security | CySec

Zusammenfassung

1. CYBER SECURITY FOUNDATIONS

1.1. THE ASSETS OF AN ORGANISATION

An asset is anything within an environment that should be protected. The loss or disclosure of an asset could result in:

- An overall security compromise
- Loss of productivity
- Reduction in profits
- Additional expenditures
- Discontinuation of the organization
- Numerous intangible consequences

1.1.1. Different Types of assets of an organization

- ***Information:*** includes all its data
- ***Systems:*** includes any information technology (IT) systems that provide one or more services
- ***Devices:*** refers to any computing system, including servers, desktop computers, portable laptop computers, tablets, smartphones, and external devices such as printers
- ***Facilities:*** includes any physical location that it owns or rents
- ***Personnel:*** People working for an organization are also a valuable asset

1.1.2. Intellectual Property

Assets that are ***intangible*** are collectively referred to as ***intellectual property***. The most valuable assets of large companies are the ***brand names***. Publishing companies, movie producers, and artists depend on their ***creative output*** to earn their livelihood. Many products depend on ***secret recipes*** or production techniques.

1.1.2.1. How to protect intellectual property

Copyright Guarantees the creator of ***«original works of authorship»*** protection ***against unauthorized duplication*** of their work. There are 8 categories: Literary, musical, dramatic choreographic, graphical/sculptural, audiovisual, sound recordings, architectural works.

Computer software falls under literary works. ***Protects only the actual source code***. Does ***not protect the idea*** behind the software. Does not prohibit others from rewriting your code in a different form.

Officially ***registering a copyright is not a prerequisite*** for copyright enforcement. The Creator of a work has an ***automatic copyright from the instant the work is created***. If you can prove in court that you were the creator of a work, you will be protected under copyright law.

There is a ***formal procedure to obtain a copyright*** that involves sending copies of the protected work along with an appropriate registration fee to the Swiss Federal Institute of Intellectual Property.

You can mark your work with the copyright symbol (©) to protect it. Works by one or more authors are protected until 70 years after the death of the last surviving author. The exceptions to this policy are works for hire. A work is considered «for hire» when it is made for an employer during the normal course of an employee's workday.

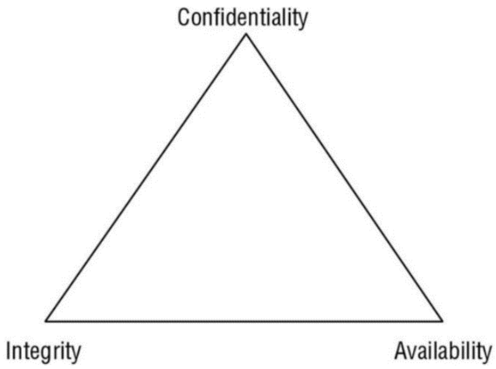
Trademarks Trademarks are ***words, slogans, and logos*** used to identify a company and its products or services. Trademarks do ***not need to be officially registered*** to gain protection under the law. If you use a trademark during your public activities, you are automatically protected under any relevant trademark law and can use the ø symbol to show that you intend to protect words or slogans as trademarks.
If you want ***official recognition*** of your trademark, you ***can register it*** with the Swiss Federal Institute of Intellectual Property. Once you've received your registration certificate, you can denote your mark as a registered trademark with the ® symbol.

Patents Patents protect the ***intellectual property rights*** of inventors. They provide a period of ***20 years*** (from the date of initial application) during which the inventor is granted ***exclusive rights*** to use the invention. After the 20 years end, the invention is in the public domain available for anyone to use.
Patents have three main requirements: The invention must be ***new***, ***useful*** and ***not obvious***. Patent law does ***not provide adequate protection for computer software***.

Trade Secrets Copyright and patents could be used to protect this type of information, but with ***two major disadvantages***:
Filing a copyright or patent application requires that ***you publicly disclose the detail of your work*** or invention. This automatically removes the «secret» nature of your property and may allow unscrupulous competitors to copy your property in violation of the laws.
Copyright and patents both provide protection for a limited period of time. Once the protection expires, other firms are free to use your work and they have all the details from the public disclosure you made during the application process.
Trade secret protection is one of the best ways to protect computer software. This is the technique used by large software development companies to protect their core base of intellectual property.

1.2. THE CIA TRIAD

The primary goal and objective of a security infrastructure is the ***protection*** of the ***confidentiality***, ***integrity***, and ***availability*** of the ***assets*** of a company.



1.2.1. Confidentiality

The goal is to ***prevent or minimize unauthorized access to data***. For confidentiality to be maintained on a network, ***data must be protected while in storage, in process and in transit.***

Example of security controls: ***encryption & access controls***.

1.2.2. Integrity

The concept of ***protecting the reliability and correctness of data***. ***Prevents unauthorized alterations*** of data. ***Alterations should not occur while the object is in storage, in process and in transit.***

Data integrity implies information is known to be good, and that the ***information can be trusted*** as being ***complete***, ***consistent*** and ***accurate***. System integrity implies that a system will work as it is intended to.

Example of security controls: ***intrusion detection systems & hash verifications***.

1.2.3. Availability

Authorized subjects are granted ***timely*** and ***uninterrupted access*** to objects. An acceptable level of performance should be guaranteed, and interruptions should be handled quickly.

Example of security controls: ***Redundancy and scalability, maintain reliable backups & prevent data loss or destruction***.

1.2.4. Nonrepudiation & Accountability

Nonrepudiation ensures that the subject of an activity or who caused an event ***is not able to deny*** having performed an action or cannot deny that the event occurred.

Accountability means ***being responsible or obligated for actions and results***. Nonrepudiation is an essential part of accountability. A suspect cannot be held accountable if they can repudiate the claim against them.

Example of security controls: ***Digital certificates, session identifiers, transaction logs***.

1.3. DATA CLASSIFICATION

Data classification is used to determine ***how much effort, money and resources are allocated to protect the data and control access to it***. It is ***inefficient to treat all data the same way*** when designing and implementing a security system because some data items need more security than others.

- Securing everything at a ***low security*** level means sensitive data is ***easily accessible***.
- Securing everything at a ***high security*** level is ***too expensive*** and ***restricts access to unclassified***, ***noncritical data***.

1.3.1. Data states

- ***Data at Rest:*** Any data stored on media such as system hard drives, USB drives, storage area networks and backup tapes.
- ***Data in Transit (Data in Motion):*** Any data transmitted over a network. Incudes data transmitted over an internal network and data transmitted over public networks.
- ***Data in Use:*** Because an application can't process encrypted data, it must decrypt it in memory. Data in use refers to data in memory or temporary storage buffer, while an application is using it.

1.3.2. Defining sensitive data

1.3.2.1. Personally Identifiable Information (PII)

Any information that can ***identify an individual***.

National Institute of Standards and Technology (NIST) Special Publication definition: Any information that can be used to distinguish or trace an individual's identity, such as name, social security number, date and place of birth, mother's maiden name, or biometric records. Any other information that is linked or linkable to an individual, such as medical, educational, financial, and employment information.

Organizations have a responsibility to protect PII. This includes PII related to employees and customers. Many laws require organizations to notify individuals if a data breach results in a compromise of PII.

1.3.2.2. Protected Health Information (PHI)

Any ***health-related information*** that can be ***related to a specific person***.

In the United States, the Health Insurance Portability and Accountability Act (HIPAA) mandates the protection of PHI. HIPAA provides a more formal definition: Health information means ***any information***, whether oral or recorded ***in any form or medium***, that is ***created or received by a health care provider***, ***health plan, public health authority, employer, life insurer, school or university, or health care***

clearinghouse. Information that relates to the ***past, present or future physical or mental health*** or condition of any individual, the ***provision of health care to an individual***, or the ***past, present, or future payment for the provision of health care*** to an individual.

The Federal Law on Swiss Patient Records (Gesetzgebung Elektronisches Patientendossier (EPDG)) came into effect in April 2017 and ***standardizes data protection*** and data security relating to the exchange of digital health data.

1.3.2.3. Proprietary Data

Refers to any data that ***helps an organization maintain a competitive edge***.

Could be software code it developed, technical plans for products, internal processes, intellectual property or trade secrets. ***If competitors can access the proprietary data, it can seriously affect the primary mission of an organization.***
Although copyrights, patents and trade secret ***laws provide a level of protection*** for proprietary data, this ***isn't always enough***. Many criminals don't pay attention to copyrights, patents and laws.

1.3.3. Government/military classification

The term classified is generally used to refer to any data that is ranked above the unclassified level. Revealing the actual classification of data to unauthorized individuals is a violation of that data.

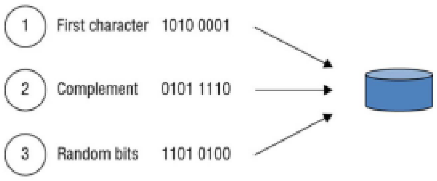
High	***Top secret***	The unauthorized disclosure of top-secret data will have ***drastic effects*** and cause ***grave damage*** to national security.
	Secret	The unauthorized disclosure of data classified as secret will have ***significant effects*** and cause ***critical damage*** to national security
	Confidential	The unauthorized disclosure of data classified as confidential will have ***noticeable effects*** and cause ***serious damage*** to national security.
	Sensitive but unclassified	Sensitive but unclassified is used for data that is for internal use.
Low	***Unclassified***	

1.3.4. Commercial business/private sector classification

High	***Confidential / Private***	***Confidential*** is used for data that is ***extremely sensitive*** and for internal use only. If proprietary data is disclosed, it can have ***drastic effects*** on the ***competitive edge*** of an organization. ***Private*** is used for data that is of a ***private or personal nature*** and intended for ***internal use only***.
	Sensitive	Sensitive is used for data that is ***more classified than public data***.
Low	***Public***	This is used for all data that ***does not fit in one of the higher classifications***.

1.3.5. Destroying sensitive data

- ***Erasing:*** In most cases, the deletion or removal process removes only the directory or catalog link to the data, like removing an entry in a phone book. The actual data remains on the drive. Anyone can typically retrieve the data using widely available undelete tools.
- ***Clearing:*** Clearing or overwriting is a process of preparing media for reuse and ensuring that the cleared data cannot be recovered using traditional recovery tools. When media is cleared, unclassified data is written over all addressable locations on the media. The method writes a single character over the entire media, writes the characters complement over the entire media, and finishes by writing random bits over the entire media.



- ***Purging:*** More intense form of clearing that prepares media for reuse in less secure environments. Provides a level of assurance that the original data is not recoverable using any known methods. Will repeat the clearing process multiple times and may combine it with another method such as degaussing to completely remove the data.
- ***Degaussing:*** A degausser creates a strong magnetic field that erases data on some media. Degaussing works only on magnetic storage media, such as HDDs, floppy disks and magnetic tapes. It does not affect optical CDs, DVDs, or SSDs.
- ***Destruction:*** When destroying media, it's important to ensure that the media cannot be reused or repaired, and that data cannot be extracted from the destroyed media. Methods of destruction include incineration, crushing, shredding, disintegration and dissolving using caustic or acidic chemicals.

1.3.6. Tracing or hiding sensitive data

- ***Steganography:*** The practice of embedding a message within a file.
- ***Watermarking:*** Embedding an image or pattern in paper that isn't readily perceivable. Is often used with currency to thwart counterfeiting attempts.
- ***Digital Watermark:*** Secretly embedded marker in a digital file. For example, some movie studios digitally mark copies of movies sent to different distributors to be able to identify the culprit if pirated copies get released.

1.4. THREAT

A threat is ***any potential danger to an asset***. Threats are any action or inaction that could cause damage, destruction, alteration, loss or disclosure of assets that could block access to or prevent maintenance of assets. ***They can be intentional or accidental***.

- ***Threat actor or agent:*** intentionally exploits vulnerabilities: Script kiddies, organized crime groups, state sponsors and governments, hacktivists, terrorist groups
- ***Threat intelligence:*** The knowledge about an existing or emerging threat to assets, including networks and systems.
- ***Threat event:*** Accidental or intentional exploitations of vulnerabilities. They can be natural or man-made. Include fire, earthquake, flood, system failure, human error and power outage.

1.4.1. The STRIDE threat model

S	**Spoofing != Authenticity** An attack with the ***goal of gaining access*** to a target system using a ***falsified identity***.
T	**Tampering != Integrity** Any action resulting in ***unauthorized changes or manipulation of data***, whether in transit or in storage. Is used to ***falsify communications or alter static information***.
R	**Repudiation != Non-repudiation** The ability of a user or attacker to ***deny having performed*** an action or activity. Repudiation attacks can also result in innocent third parties being blamed for security violations.
I	**Information disclosure != Confidentiality** The ***revelation or distribution*** of private, confidential or controlled information to external or unauthorized entities.
D	**Denial of service (DoS) != Availability** An attack that attempts ***to prevent authorized use*** of a resource. A DoS attack does not necessarily result in full interruption to a resource, it could instead reduce throughput or introduce latency in order to hamper productive use of a resource.
E	**Elevation of privilege != Authorization** An attack where a limited user account is ***transformed into an account with greater privileges***, powers, and access.

1.5. VULNERABILITY

The ***weakness in an asset***. If a vulnerability is exploited, loss or damage to assets can occur.

A ***Common Vulnerabilities and Exposures (CVE)*** is an industry-wide standard identification number for vulnerabilities. CVE Entries are used in numerous cybersecurity products and services from around the world, including the U.S. National Vulnerability Database (NVD).

The ***Common Vulnerability Scoring System (CVSS)*** uses the CIA triad principles within the metrics used to calculate the CVVS base score and assigns severity scores to a vulnerability.

CVE → Identifies a vulnerability, CVSS → Rates how severe a vulnerability is

1.5.1. Exploit

An exploit is a ***software*** or a sequence of commands that ***takes advantage of a vulnerability*** in order to ***cause harm to a system or network***. An ***exploit kit*** is a ***compilation of exploits*** that are often designed to be served from webservers.

1.6. RISK MANAGEMENT

The overall process of risk management is used to ***develop and implement information security strategies***. The goal of these strategies is to ***reduce risk*** and to ***support the mission*** of the organization.

It is ***impossible*** to design and deploy a ***risk-free environment***. Significant risk reduction is possible, often with little effort.

Risk management is a detailed process of:

- ***Identifying*** factors that could damage or disclose data
- ***Evaluating*** those factors considering data value and countermeasure cost
- ***Implementing cost-effective solutions*** for mitigating or reducing risk

1.6.1. Countermeasure

Countermeasure is any action or product that ***reduces risk*** through the ***elimination or lessening of a threat*** or a vulnerability anywhere within an organization.

1.6.2. Risk analysis

The process by ***which the goals of risk management are achieved***.

- ***Evaluation***, ***assessment*** and the ***assignment of value*** for all assets within the organization.
- ***Examining*** an environment for risks.
- ***Evaluating each threat*** event as to its ***likelihood*** of occurring and the ***cost of the damage*** it would cause if it did occur
- ***Assessing the cost of various countermeasures*** for each risk and creating a cost/benefit report for safeguards to present to upper management

1.6.3. Asset Valuation

Asset valuation is a ***dollar value*** assigned to an asset based on actual cost and nonmonetary expenses. These can include costs to develop, maintain, administer, advertise, support, repair and replace an asset. They ***can also include more elusive values***, such as public confidence, industry support, productivity enhancement, knowledge equity, and ownership benefits.

1.6.4. Exposure

Exposure means that there is a ***possibility for an asset loss because of a threat***. The quantitative risk analysis value of ***exposure factor (EF)*** is a calculation how extensive or serious that harm might be. It is expressed as a percentage value.

1.6.5. Risk

The ***possibility that something could happen*** to damage, destroy, or disclose data or other resources is known as risk. It is an assessment of probability, possibility, or chance that a threat will exploit a vulnerability to cause harm to an asset. ***The more likely it is that a threat will occur, the greater the risk***.

A realized risk

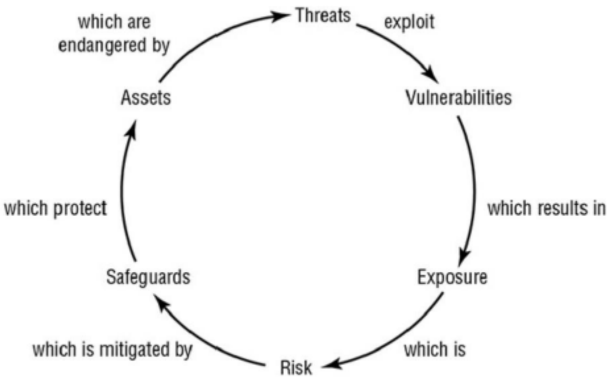
When a risk is realized, a threat agent***, a threat actor or a threat event has taken advantage of a vulnerability and caused harm*** to or disclosure of one or more assets. The whole purpose of security is to prevent risks from becoming realized. Realized Risk → Risk that happened.

1.6.6. Attack

An attack is the ***exploitation of a vulnerability by a threat agent***. It is any intentional attempt to exploit a vulnerability of an organization's security infrastructure to cause damage, loss, or disclosure of assets.

1.6.7. Breach

A breach is the ***occurrence of a security mechanism being bypassed or thwarted by a threat agent***. When a breach is combined with an attack, a penetration, or intrusion can result. A penetration is the condition in which a threat agent has gained access to an organization's infrastructure through the circumvention of security controls and is able to directly imperil assets.



1.6.8. Risk assessment/analysis

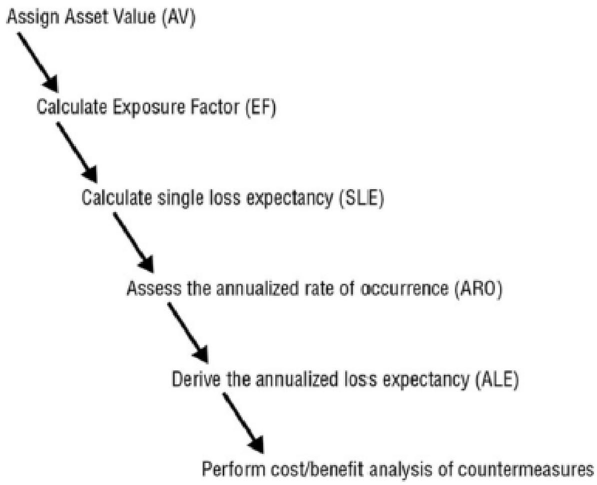
All IT systems have risk. Upper management must ***decide which risks are acceptable and which are not***. Determining which risks are acceptable requires detailed and complex ***asset and risk assessments***. Once you develop a list of threats, you must ***individually evaluate each threat*** and its related risk.

There are two risk assessment methodologies:

- ***Quantitative risk analysis*** assigns real dollar figures to the loss of an asset.
- ***Qualitative risk analysis*** assigns subjective and intangible values to the loss of an asset.

Both methods are necessary for a complete risk analysis. Most environments employ a hybrid of both risk assessment methodologies in order ***to gain a balanced view*** of their security concerns.

1.6.9. Quantitative risk analysis


Assign Asset Value (AV)
Calculate Exposure Factor (EF)
Calculate single loss expectancy (SLE)
Assess the annualized rate of occurrence (ARO)
Derive the annualized loss expectancy (ALE)
Perform cost/benefit analysis of countermeasures

- ***Asset Value (AV):*** The value of the asset
- ***Exposure factor (EF):*** represents the percentage of loss that an organization would experience if a specific asset is violated by a realized risk. In most cases, a realized risk does not result in the total loss of an asset. The EF indicates the expected overall asset value loss because of a single realized risk. If the value of a building would be reduced from $1’000’000 to $250’000 by a fire, the exposure factor for the risk of fire to the building would be 75%.
- ***Single Loss Expectancy (SLE):*** The cost associated with a single realized risk against a specific asset. Indicates the exact amount of loss an organization would experience if an asset were harmed by a specific threat occurring. $SLE = AV * EF$. An Asset has a value (AV) of $200’000 and an EF of 45%, the SLE would be $200'000\$ * 0.45 = 90'000\$$
- ***Annualized Rate of Occurrence (ARO):*** The expected frequency with which a specific threat or risk will occur withing a single year. For example, the ARO of an earthquake of a 6.7+ magnitude in Paris may be .00001, whereas the ARO of an earthquake in San Francisco may be .03
- ***Annualized Loss Expectancy (ALE):*** The possible yearly cost of all instances of a specific realized threat against a specific asset. $ALE = SLE * ARO$. If the SLE of an asset is $90,000 and the ARO for a specific threat (such as total power loss) is .5, then the ALE is $90'000\$ * 0.5 = 45'000\$$. On the other hand, if the ARO for a specific threat (such as compromised user account) is 15, then the ALE would be $90'000\$ * 15.0 = 1'350'000\$$.
- ***Annualized Loss Expectancy with a Safeguard:*** You must calculate the ALE for the asset if the safeguard is implemented. This requires a new EF and ARO specific to the safeguard. The whole point of a safeguard is to reduce the ARO. The EF often remains the same even with an applied safeguard because if the safeguard fails, the loss on the asset is usually the same as when there is no safeguard.
- ***Safeguard Costs:*** You must compile a list of safeguards for each threat. Then you assign each safeguard a deployment value ***ACS (Annual cost of the safeguard)***

Calculating Safeguard Cost/Benefit: $ALE\ before\ safeguard - ALE\ with\ safeguard - ACS = Value\ of\ Safeguard$

If the result is ***negative***, the safeguard is ***not a financially responsible choice***.

If the result is ***positive***, then that value is the annual savings your organization may reap by deploying the safeguard. The countermeasure with the greatest resulting value from the cost/benefit formular makes the most economic sense.

Security should be cost effective. Thus it is not prudent to spend more protecting an asset than it is worth to the organization.

If the cost of the countermeasure is greater than the value of the asset or the cost of the risk, then you should accept the risk.

1.6.10. Risk handling

Risk Mitigation
The ***implementation of safeguards*** and countermeasures to eliminate vulnerabilities or block threats.

Risk Assignment
The ***placement*** of the cost of loss a risk represents ***onto another entity*** or organization, like ***purchasing insurance*** and outsourcing.

Risk Acceptance
If the cost/benefit analysis shows that ***countermeasure costs would outweigh the possible cost of loss*** due to a risk, the risk can be ***accepted***. In most cases, this requires a clearly written statement that indicates why a safeguard was not implemented, who is responsible for the decision and who will be responsible for the loss if the risk is realized. Usually in the form of a sign-off letter.

Risk Deterrence
The ***process of implementing deterrents*** to would-be violators of security and policy. Examples: Implementation of auditing, ***security cameras or guards***, warning banners, motion detectors, strong authentication.

Risk Avoidance
The process of ***selecting alternate options*** or activities that have ***less associated risk*** than the default option. The risk is avoided by eliminating the risk cause. Example: Removing FTP protocol from a server to avoid FTP attacks, ***move to an inland location to avoid risks from hurricanes***.

Risk Rejection
A Final but ***unacceptable*** possible response to risk is to reject or ***ignore risk***.

1.6.11. Residual risk

Once ***countermeasures are implemented***, the ***risk that remains*** is known as residual risk. This is the risk that management has chosen to accept rather than mitigate.

1.7. PRIVACY

Privacy is the ***right of an individual to control their personal data***.

- Data ***collection*** should be ***restricted***.
- Data owners have a ***responsibility*** to respect and enforce privacy principles.
- Data processes should ***ensure enforcement*** of privacy and data integrity.
- Data remanence techniques should be used to ***permanently delete*** data.

1.7.1. USA PATRIOT Act of 2001

Allows authorities to obtain a ***blanket authorization*** for a person and then ***monitor all communications*** to or from that person under the single warrant. Internet service providers may have to provide the government with a large range of information.

1.7.2. European Union General Data Protection Regulation (GDPR /DSGVO)

Companies have to inform authorities of serious data breaches withing 24 hours. Provisions that individuals will have access to their own data. The ***«right to be forgotten»*** that allows people to require companies to delete their information if it is no longer needed. Violations of the GDPR can lead to heavy fines, up to 20% of global income of the company.

1.7.3. Pseudonymization

One key security control in the GDPR. The process of ***replacing some data elements with pseudonyms***. This makes it more difficult to identify individuals. ***Can be reversed***.

1.7.4. Anonymization

The process of ***removing all relevant data*** so that it is ***impossible to identify*** the original subject or person.

2. IDENTITY AND ACCESS MANAGEMENT (IAM)

2.1. CONTROLLING ACCESS TO ASSETS

An access control is any hardware, software, or administrative policy or procedure that controls access to resources. The goal is to provide access to authorized subjects and prevent unauthorized access attempts.

- ***Subject:*** An ***active entity*** that ***accesses a passive object*** to receive information from, or data about, an object. *Examples: Users, programs, processes, services, computers or anything else that can access a resource.*
- ***Object:*** A ***passive entity*** that ***provides information*** to active subjects. *Examples: Files, Databases, computers, programs, processes, services, printers, and storage media.*

The ***management of the relationship between subjects and objects*** is known as access control.

2.1.1. The primary access control types

Preventive Access Control: Attempts to thwart or stop unauthorized activity from occurring.
Examples: Fences, Locks, Alarm systems, data classification, penetration testing, training

Detective Access Control: Attempts to discover unauthorized activity after it has occurred.
Examples: Security Guards, Motion detectors, Cameras, Honeypots, Incident investigations

Corrective Access Control: Attempt to correct any problems that occurred because of a security incident.
Examples: Rebooting a system, antivirus solutions, backup and restore plans

2.1.2. Other access control types

Deterrent Access Control: Discourage violation of security policies. Depends on choices of individuals.
Examples: Policies, Training, Locks, Fences, Guards, Cameras, Security Badges

Compensation Control: Provides various options to aid in enforcement and support of security policies.
Examples: If data that should be encrypted isn't during transit, a compensation control can be added to protect the data.

Directive Control: Directs, confines or controls the action of subjects to force or encourage compliance.
Examples: Security policy requirements or criteria, posted notifications, escape route exit signs, monitoring

Recovery Access Control: Extension of Corrective Control, but more advanced.
Examples: Backup and restores, fault-tolerant drive systems, antivirus software, multisite solutions

2.1.3. Controlling access to assets

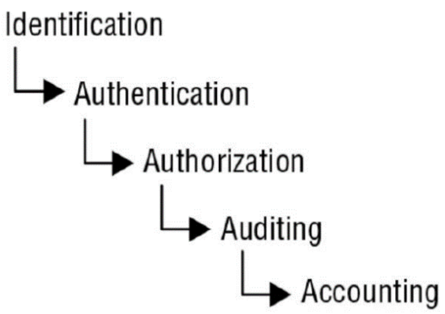
Access controls are also categorized by how they are implemented.

Physical controls: prevent, monitor or detect contact with systems or areas within a facility. Touchable.
Examples: Guards, Fences, locked doors, lights, laptop locks, cameras, alarms

Technical or logical controls: hardware or software used to manage access for resources. Uses technology.
Examples: Authentication methods, encryption, lists, protocols, firewalls, routers, intrusion detection systems

Administrative / management controls: Policies defined by an organizations security policy.
Examples: Procedures, hiring practices, background checks, security awareness and training efforts, reports, testing

2.2. THE STEPS OF ACCESS CONTROL



2.2.1. Identification

The process of a subject claiming an identity. Might entail ***typing a username***, swiping a smartcard, speaking a phrase, or positioning a body part in front of a scanning device.

All subjects must have ***unique identities***. IT systems track activity by identities, not by the subjects themselves (i.e., a person can have multiple identities/accounts). A subject's identity is ***public information***.

A subject must ***provide an identity to a system to start the authentication***. The identity must be ***proven or*** verified before access is allowed.

2.2.2. Authentication

Requires the subject to ***provide additional information that corresponds to the identity they are claiming***. The most common form: using a password. Identification and authentication are often used together as a single two-step process.

Authentication information used to verify identity is ***private information*** and needs to be protected.
The process of ***verifying or testing that the claimed identity is valid is authentication***.

Password / Basic Authentication

Passwords are typically ***static***. They are the ***weakest form of authentication***. Users often choose easy passwords, write them down, share or forget them.
Passwords should not be stored in plaintext. Instead, they are hashed with a ***hashing algorithm***. Systems store the hash, not the password. When a user authenticates, the system hashes the supplied password and compares it to the stored value. If the hashes are the same, the password is correct.

Weaknesses: basic authentication is prone to passive sniffing attack thus directly revealing the plaintext password irrespective of password strength. Basic authentication must never be used over insecure communication links.

Password Phrase

It is ***easier to remember*** than a password and encourages the user to create a longer password.
Example: 1P@ssedTheCySecEx@m

Cognitive password

A ***series of questions*** that only the subject should know. Often used to assist with password reset systems. The best cognitive password systems allow users to create their own questions and answers. This makes the attacker's job much more difficult. *Examples: Mothers maiden name, name of first pet, name of favorite restaurant, first car model etc.*

Smartcard

Credit card-sized ID or badge that has an integrated circuit chip embedded in it. Contain information about the authorized user that is ***used for identification and/or authentication***. Most current smartcards include a ***microprocessor*** and one or more ***certificates***. The certificates are used for asymmetric cryptography such as encrypting data or digitally signing email. Smartcards are ***tamper-resistant*** and provide users with an easy way to carry and use complex encryption keys.

Tokens

A password-generating device that users can carry with them. An ***authentication server*** stores the details of the token, so at any moment, the server knows what number is displayed on the user's token.

Synchronous Dynamic Password Token: Hardware token that create synchronous dynamic passwords are time-based and synchronized with an authentication server. They ***generate a new password periodically***, such as every 30 seconds. This does require the token and the server to ***have accurate time***.

Asynchronous Dynamic Password Tokens: Does ***not use a clock***. The hardware token generates passwords based on an algorithm and an incrementing counter. When using an incrementing counter, it creates a dynamic onetime password that stays the same until used for authentication.

Onetime Password Generators

Dynamic passwords that change every time they are used. Onetime password generators are token devices that create passwords. The PIN can be provided via a software application running on the user's device (smartphone).

TOTP (Time-based One-Time Password): Uses a ***timestamp*** and remains valid for a certain timeframe, such as 30 seconds. Similar to synchronous dynamic passwords used by tokens. Used by most websites using 2 Factor Authentication and Authenticator apps.

HOTP (HMAC-based One-Time Password): Includes a hash function to create onetime passwords which ***remain valid until used***. It creates HOTP values of six to eight numbers. Similar to asynchronous dynamic passwords used by tokens.

- ***Strengths of OTPs:*** An efficient means against passive sniffing attacks since a captured password is of no use for an attacker.
- ***Weaknesses:*** OTPs do not offer protection against active Man-In-The-Middle (MITM) attacks since the attacker can just forward the OTP and then hijack the current session.

Challenge / Response

Response is a function of password and one-time challenge

Example: Server sends Client a value, the client then hashes their password with this value attached and sends it back. The server does the same hashing with the stored password (hash) of the client and if they match, the challenge is solved.

- ***Strengths:*** Resistant against passive sniffing attacks if the password used has a sufficient strength
- ***Weaknesses:*** If the password is weak then passive attackers can retrieve the password via an offline dictionary or brute force attack. Therefore, in practice challenge/response protocols are not used over an insecure communication channel. They do not offer protection against active MITM attacks.

Anonymous Key Exchange

Exchange credentials over unauthenticated secure channel

- ***Strengths:*** Key exchange algorithms like Diffie-Hellman allow the derivation of a common secret between two anonymous endpoints. The secret can then be used to encrypt the communication between the endpoints thus preventing passive sniffing attacks. Within the encrypted tunnel the vulnerable authentication schemes can be used.
- ***Weaknesses:*** Anonymous key exchange is vulnerable to MITM attacks because the endpoints do not authenticate themselves. The attacker does a separate DH exchange with both client and server so that the plaintext communications traffic becomes visible to the attacker.

Server Certificates plus User Authentication

Transmit user password over unilaterally authenticated secure channel

- ***Strengths:*** The RSA or ECDSA public key signature computed by the server over the hash of the DH exchange binds the shared secret jointly derived by server and client to the server certificate thus effectively preventing MITM attacks. Within the encrypted and authenticated tunnel, the vulnerable authentication schemes can be used.
- ***Weaknesses:*** MITM attacks are still possible if the X.509 trust chain validation is broken. The database of password hashes or the OTP seeds residing on the server could be stolen by an attacker so that all user credentials become compromised.

Mutual Public Key Authentication

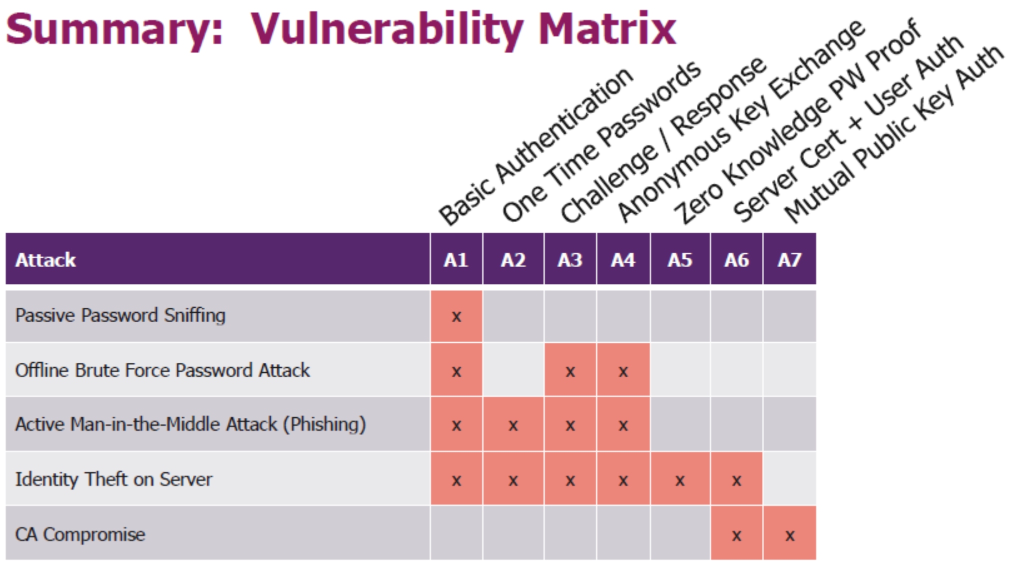
Bilateral use of public key signatures

- ***Strengths:*** The RSA or ECDSA public key signatures computed by both server and client over the hash of the DH exchange binds the shared secret jointly derived by server and client to the end entity certificates thus effectively preventing MITM attacks. The use of client certificates removes the need to store user credentials on the server thus minimizing the risk of large scale identity theft. Client certificates stored on a smart card and protected by a PIN guarantees a strong two-factor authentication.
- ***Weaknesses:*** None besides the worst case scenario of a root CA compromise.

Zero Knowledge Password Proofs

A password-based authentication protocol that allows a claimant to authenticate to a verifier without revealing the password to the verifier. *See example in class with the different colored balls and figuring out which ball is shown.*

- ***Strengths:*** Since public DH factors are totally random, they can be encrypted with a symmetric key which can be relatively weak. With the exception of trivial keys such as 00000000 or 12345678 a brute force attack on the weak password is impossible because there is no measurable dip in the entropy of the decrypted data when the correct key is applied.
- ***Weaknesses:*** The password database residing on the server could be stolen by an attacker so that all user credentials become compromised.



Summary: Vulnerability Matrix

	Basic Authentication	One Time Passwords	Challenge / Response	Anonymous Key Exchange	Zero Knowledge PW Proof	Server Cert + User Auth	Mutual Public Key Auth
Attack	**A1**	**A2**	**A3**	**A4**	**A5**	**A6**	**A7**
Passive Password Sniffing	x						
Offline Brute Force Password Attack	x		x	x			
Active Man-in-the-Middle Attack (Phishing)	x	x	x	x			
Identity Theft on Server	x	x	x	x	x	x	
CA Compromise						x	x

Authentication Factors

Type 1 (weakest): Something you know.
Example: Password, PIN (Personal identification number), Passphrases, Cognitive passwords

Type 2: Something you have, like a physical device that a user possesses
Example: Smartcard, Hardware token, Memory card, USB (Universal Serial Bus) drive, Authenticator app on your phone

Type 3 (strongest): Something you are / you do, like physical characteristic / biometrics
Example: Fingerprint, voice print, retina patterns, face shape, hand geometry, heart/pulse pattern, keystroke pattern

Multifactor authentication

Any authentication using ***two or more factors***.
When two authentication methods of the ***same factor*** are used together, the authentication ***does not get stronger***.

Secondary authentication factors

Somewhere You Are: Location based.
Example: IP Address, Caller ID, specific computer

Somewhere You Aren't: To identify suspicious activity.
Example: Netflix sends an email if a device logs in from a unusual location

2.2.3. Authorization

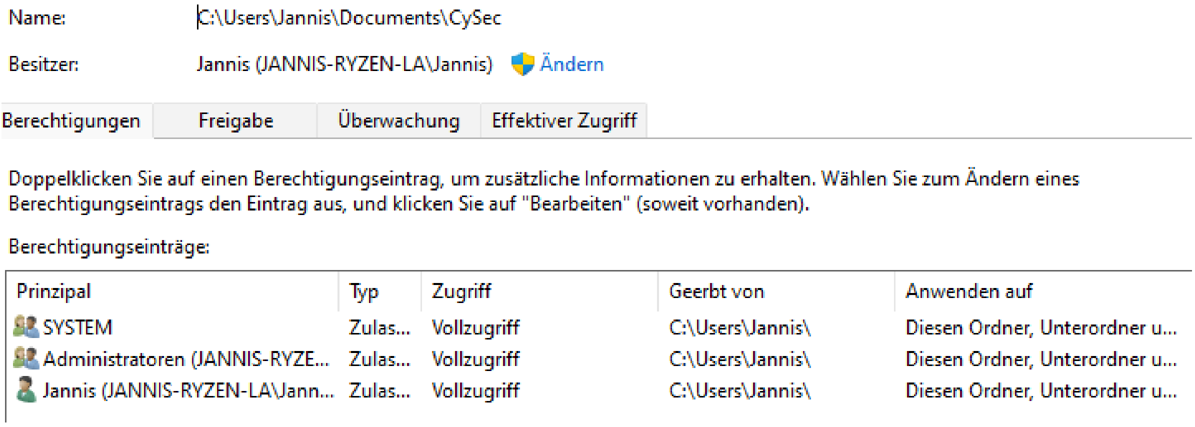
Just because a subject has been ***identified and authenticated does not mean they have been authorized*** to perform any function or access all resources within the controlled environment.

Identification and authentication are all-or-nothing aspects of access control. Authorization has a ***wide range of variations*** between all or nothing ***for each object*** within the environment.

The process of authorization ensures that the ***requested activity or access to an object is possible*** given the ***rights and privileges assigned*** to the authenticated identity → Check if you're allowed to do what you want to do.

Access control models

- ***Discretionary Access Control (DAC):*** Every object has an owner which can grant or deny access to other subjects. The New Technology File System (NTFS) used on Windows, uses this mode.



Name: C:\Users\Jannis\Documents\CySec

Besitzer: Jannis (JANNIS-RYZEN-LA\Jannis)

Doppelklicken Sie auf einen Berechtigungseintrag, um zusätzliche Informationen zu erhalten. Wählen Sie zum Ändern eines Berechtigungseintrags den Eintrag aus, und klicken Sie auf "Bearbeiten" (soweit vorhanden).

Berechtigungseinträge:

Prinzipal	Typ	Zugriff	Geerbt von	Anwenden auf
SYSTEM	Zulas...	Vollzugriff	C:\Users\Jannis\	Diesen Ordner, Unterordner u...
Administratoren (JANNIS-RYZE...	Zulas...	Vollzugriff	C:\Users\Jannis\	Diesen Ordner, Unterordner u...
Jannis (JANNIS-RYZEN-LA\Jann...	Zulas...	Vollzugriff	C:\Users\Jannis\	Diesen Ordner, Unterordner u...

- ***Role Based Access Control (RBAC):*** Permissions are not assigned directly to users. Users are placed in roles which have been assigned privileges. Roles are typically identified by job functions. *Example: Different groups for the IT, HR & Marketing departments with different access rights.*
- ***Rule-based access control (RuBAC):*** Global rules that are applied to all subjects. A firewall uses rules that allow or block traffic to all users equally. Rules within the RuBAC are sometimes referred to as restrictions or filters. *Example: No user should be allowed to access blick.ch*

- ***Attribute Based Access Control (ABAC):*** Rules can include multiple attributes. More flexible than a model that applies the rules to all subjects equally. *Example: No user should be allowed to access digitec.ch, except if they are a part of the "IT department" or "R&D" group*
- ***Mandatory Access Control (MAC):*** Use of labels applied to both subjects and objects. If a user has a label of top-secret, he can be granted access to a top-secret document.

Authorization Mechanisms

- ***Implicit Deny:*** Most mechanism use it. Ensures that access to an object is denied unless access has been explicitly granted (Deny/Block by default).
- ***Constrained Interface:*** Restrict what users can do or see based on their privileges. Users with full privileges have access to all the capabilities. A common method is to hide the capability if the user doesn't have permission to use it. *Example: Hide advanced settings if user isn't in admin group.*
- ***Access Control Matrix (ACL):*** Table that includes subjects, objects and assigned privileges. The system checks the matrix to determine if the subject has the needed privileges for an action. ACLs are object focused and identify access granted to subjects for any specific object. *Example: Read/Write/Execute permissions on files/folders per user/group*
- ***Capability Tables:*** Also used to identify privileges assigned to subjects. Are subject focused and identify the objects that subjects can access.
- ***Content-Dependent Control:*** Restrict access to data based on the content within an object. *Example: A database view is a content-dependent control. A view retrieves specific columns from one or more tables, creating a virtual table.*
- ***Context-Dependent Control:*** Require specific activity before granting users access. *Example: A download page in an online shop is only visible if a user goes through the purchase process first.*

Principles

- ***Need to Know:*** Subjects are granted access only to what they need to know for their work tasks. Subjects may have clearance to access classified or restricted data but are not granted authorization to the data unless they actually need it to perform a job. → "So viel wie nötig"
- ***Least Privilege:*** Subjects are granted only the privileges they need to perform their work tasks. Relies on the assumption that all users have a well-defined job description. Without that, it is not possible to know what privileges users need. → "So wenig wie möglich"
- ***Separation of Duties and Responsibilities:*** No single person has total control over a critical function or system. This ensures that no single person can compromise the system.

2.2.4. Auditing

The programmatic means by which a subject's action are ***tracked*** and ***recorded***. The purpose is to ***hold the subjects accountable*** for their actions while authenticated on a system. → Logging of user activity

2.2.5. Accounting

Effective accountability relies on the ***capability to prove a subject's identity and track their activities***. Is established by ***linking a human to the activities of an online identity*** through the mechanisms of auditing, authorization, authentication and identification.
To have viable accountability, you should be able to ***support your security decisions in a court of law***.

2.3. THE COMMON ACCESS CONTROL ATTACKS

2.3.1. Access Aggregation Attacks (passive)

Collecting multiple pieces of non sensitive information and aggregating them to learn sensitive information. Reconnaissance attacks are access aggregation attacks that combine multiple tools to identify multiple elements of a system, such as IP addresses, open ports, running services, operating systems.

2.3.2. Password Attacks (Brute-force)

Online: Attacks against online ***accounts***. Offline: ***steal an account database*** and then crack the passwords.

2.3.3. Dictionary Attacks (Brute-force)

Attempt to discover passwords by ***using every possible password*** in a predefined database of common passwords and words from the dictionary (also called a ***password-cracking dictionary***). Often also scan for ***one-upped-constructed*** passwords like `password1`.

2.3.4. Birthday Attacks (Brute-force)

Focuses on ***finding collisions in hash functions***. You can reduce the success of birthday attacks by using ***hashing algorithms*** with enough bits to make collisions computationally infeasible and by using salts. MD5 is not collision free. ***SHA-3*** is considered safe for now. The name comes from the Birthday Problem / Paradox.

2.3.5. Rainbow Table Attacks

A rainbow table is a ***large database of precomputed hashes*** which can be compared to the hashes in a stolen password database file. Can ***significantly reduce the time*** it takes to crack a password.

2.3.6. Sniffer Attacks

Occurs when an attacker uses a sniffer (software application) to ***capture information transmitted over a network.*** Prevent sniffing attacks: ***Encrypt*** all sensitive data sent over a network, use ***onetime passwords***, ***protect network devices*** with physical security to hinder the installation of sniffers on devices.

2.3.7. Spoofing / Masquerading Attacks

Pretending to be something or someone else. Attackers ***replace a valid IP address, email address or phone number with a false one*** to hide their identity or to ***impersonate*** a trusted system.

2.3.8. Social Engineering Attacks

Sometimes the easiest way to get a password is to ***ask for it***. Attacker attempts to ***gain the trust*** of someone and ***trick people into revealing information***. Is often used to gain access to the IT infrastructure or the physical facility. Educating employees reduces the effectiveness of these attacks.

2.3.9. Shoulder surfing

Trying to look over the shoulder to ***read information on the computer screen*** or ***watch the keyboard*** as a user types. Screen filters placed over a monitor can restrict the attacker's view.

2.3.10. Phishing

Form of social engineering that attempts to trick users into giving up sensitive information, opening an attachment or clicking a link. Often tries to obtain user credentials or PII like usernames, passwords, credit card details by masquerading as a legitimate company. Phishing emails inform the user of a bogus problem and say that if the user doesn't take action, the user's account will be locked.

More sophisticated phishing

- A link to a bogus website ***that looks legitimate***
- The message includes an ***infected file*** and encourages the user to open it
- The message includes a link to a website that installs a malicious ***drive-by download (automatic download without the users consent)***.
- Attackers often use ***social media to identify friendships*** between people when crafting phishing mails.

Spear Phishing

Targeted to a ***specific group of users***, such as employees within a specific organization.

Whaling

Targets ***senior or high-level executives*** within a company.

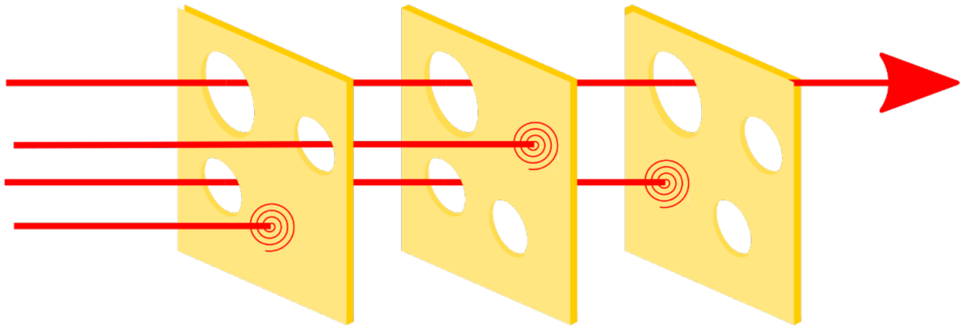
Vishing

Phishing attack ***via instant messaging*** (IM) and VoIP.

2.4. PROTECTION MECHANISMS

2.4.1. Layering (defense in depth)

Use of multiple controls in a series. When security solutions are designed in layers, ***a failed control should not result in exposure of systems or data***. See also: Swiss cheese model – Multiple layers should prevent a single point of failure, if one layer fails to catch something, the other layers should prevent them



2.4.2. Abstraction

Used when classifying objects or assigning roles to subjects. Simplifies security by ***enabling to assign security controls to a group of objects*** collected by type or function.

2.4.3. Data Hiding

Preventing data from being discovered or accessed by a subject by ***positioning it so that it is not accessible or seen*** by an unauthorized subject. *Example: private variables in code can't be accessed by other classes*

2.4.4. Security through obscurity

Not informing a subject about an object being present and ***hoping that the subject will not discover*** the object. Does ***not*** implement any form of protection. *Example: Storing a hard coded password in the source code*

2.4.5. Encryption

Hiding the meaning or intent of a communication from unintended recipients. Weak or poor encryption can be considered nothing more than security through obscurity.

3. LINUX FOR CYBERSECURITY PROFESSIONALS

3.1. CYBERSECURITY RED TEAM VERSUS BLUE TEAM

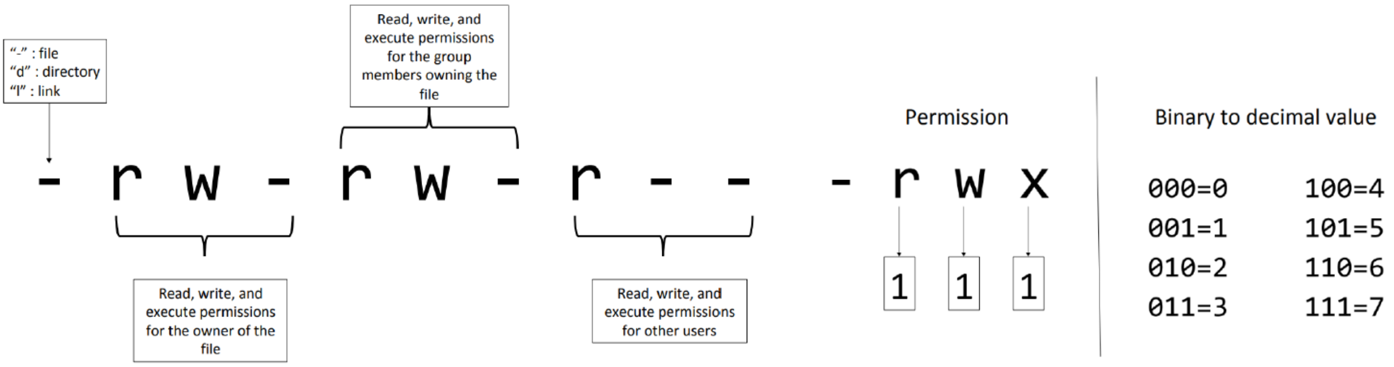
Red Team (offensive)	***Blue Team (defensive)***
Focused on penetration testing Assume the role of an attacker They show organizations what could be backdoors or exploitable vulnerabilities that pose a threat.	Assessment of network security and identification of possible vulnerabilities. Find ways to defend mechanisms to make incident response stronger.
- Think outside the box - Deep knowledge of systems - Software development - Penetration testing - Social engineering	- Organized and detail-oriented - Cybersecurity analysis and threat profile - Hardening techniques - Knowledge of detection systems - SIEM (Security Information and Event Management)

3.2. LINUX USER AND GROUP ACCOUNTS

Users should be able to accomplish tasks that require privileges. For example, when installing a program or adding another user. This is why sudo exists.
To be able to run command in sudo, a user has to be added to the sudo group.

- `root@kali# useradd -m superman`
- `root@kali# chsh -s /bin/bash superman`
- `root@kali# passwd superman`
- `root@kali# usermod -a -G sudo superman`
- `root@kali# sudo groupadd -g 1000 engineering`
- `root#kali# usermod -a -G engineering superman`

3.3. LINUX FILE PERMISSIONS


"-" : file
"d" : directory
"l" : link
Read, write, and execute permissions for the group members owning the file
- r w - r w - r - -
Read, write, and execute permissions for the owner of the file
Read, write, and execute permissions for other users
Permission
- r w x
1 1 1
Binary to decimal value

000=0	100=4
001=1	101=5
010=2	110=6
011=3	111=7

SELinux

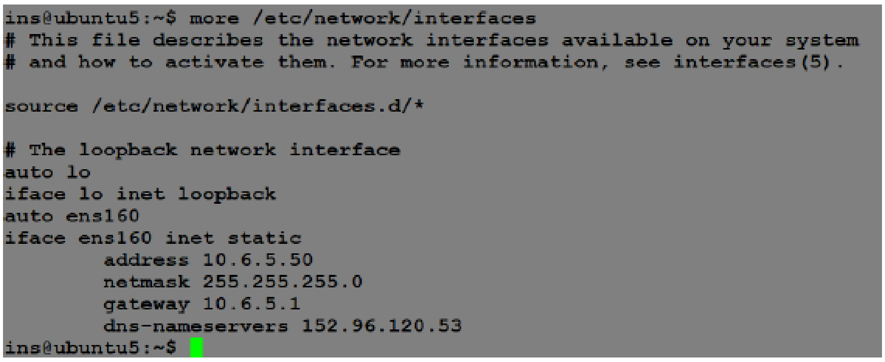
Security Enhanced Linux was developed by the United States National Security Agency (NSA) as an implementation on the mandatory access control instead of standard Unix model of discretionary access control. Files and processes have different "levels" similar to the Government/military classification. A lower-level process can't read a file created by a higher-level process.

Sticky Bit

The restricted deletion flag or sticky bit is a ***single bit*** whose interpretation depends on the file type. If the sticky bit is set on a ***directory***, ***files inside*** the directory may be ***renamed*** or ***removed*** only by the ***owner*** of the file, the owner of the directory or the ***superuser***. Read/write permissions of the files are not affected by the sticky bit. It is commonly found on world-writable directories such as /tmp. The sticky bit is obsolete with files.

3.4. LINUX NETWORKING

- ***Setting up the physical network:*** `ip link, sudo ip link set dev ens160 up`
- ***Manually configuring Ipv4:*** `sudo ifup ens160, sudo ip addr add dev ens160 10.6.5.50/24`
- ***Configuring Ipv4 permanently:*** `more /etc/network/interfaces`



```
ins@ubuntu5:~$ more /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback
auto ens160
iface ens160 inet static
        address 10.6.5.50
        netmask 255.255.255.0
        gateway 10.6.5.1
        dns-nameservers 152.96.120.53
ins@ubuntu5:~$
```

Dig

Useful for performing DNS Queries on specific DNS servers.

DNS Zone Transfer (red team)

Process, where a ***DNS server passes a copy of part of its database to another DNS server***. This is how you can have more than one DNS server able to answer queries about a particular zone. There is a Master DNS server, and one or more Slave DNS servers, and the slaves ask the master for a copy of the records for that zone. In a basic ***DNS Zone Transfer Attack, you pretend to be a slave*** and ask the master for a copy of the zone records.

Enabling SSH Access (blue team)

Secure Shell is designed to ***replace unsecure remote communication*** operations, such as the `telnet`, `ftp`, `rlogin`, `rsh`, `rcp` and `rexec` commands/protocols.

Using a ***nonstandard port*** makes it more difficult for a hacker to discover a service. → But this is Security through Obscurity, does not provide actual protection
Authentication Methods for SSH: ***Password*** Authentication or ***Pubkey*** Authentication.

3.5. LINUX PROCESSES AND LOGS (BLUE TEAM)

3.5.1. Processes

The Linux kernel assigns a ***unique process ID to each process sequentially***, as the processes are created. The PID of the process is more important than the name.
In Linux, there are two methods for starting a process: starting in the ***foreground*** or in the ***background***.

Looking for suspicious processes is one of the tasks in Linux forensics. If a process runs with an open network socket that does not show up on a similar system, there may be something suspicious.

Used commands: `ps, top, htop, atop`

3.5.2. Logs

Linux uses a daemon called `syslogd` to automatically log events on your computer. In the logs, you can search via `grep` or `awk`, search for invalid users.
Several variations of `syslog`, like `rsyslog` and `syslogng` are used on different distributions of Linux.

Syslog also has a log rotate feature, where after a certain condition is met (file size exceeded, start of new month), the current file is renamed and a new log is started.

rsyslog.conf Logging rules

The facility keyword references the program, such as mail or kernel whose messages are being logged. The priority tells the system what kinds of messages to log. Codes are listed from lowest priority, starting at debug, to highest priority, ending at panic.

The shred command

-f: force permission, -n: number of times to overwrite, *: also shred all logrotate files (auth.log.1, ..2 etc)

```
root@kali:# shred -f -n 15 /var/log/auth.log*
```

Different tools

- ***awk:*** similar to grep, but can also replace content in a file
- ***dd:*** for bit-by-bit copy
- ***cron:*** scheduled tasks
- ***init.d:*** service startup scripts
- ***cat .bash_history:*** Command history
- ***ls-lasrt:*** Show recently modified files

What else to look for?

- Rogue «set-UID» files
- Directories with names that start with «.» (hidden)
- Regular files under /dev directory
- Recently modified files

3.6. LINUX FIREWALLS AND INTRUSION PREVENTION SYSTEMS (BLUE TEAM)

3.6.1. Iptables

Customize the behavior of `netfilter` to do various tasks and improve security. Common commands:

Delete (flush) existing rules:

```
iptables -F
```

Block all Traffic:

```
iptables -P INPUT DROP, iptables -P FORWARD DROP, iptables -P OUTPUT DROP
```

This will kill all communications in your system

Allowing SSH Traffic from specific Sources:

```
iptables -A INPUT -p tcp -s 192.168.78.0/24 –dport 22 -m state –state NEW, ESTABLISHED
-j ACCEPT
iptables -A OUTPUT -p tcp –sport 22 -m state –state ESTABLISHED -j ACCEPT
```

Allowing DNS

```
iptables -A INPUT -p udp -i eth0 –sport 53 -j ACCEPT

iptables -A OUTPUT -p udp -o eth0 –dport 53 -j ACCEPT
```

Allowing Web Traffic

```
iptables -A INPUT -p tcp --dport 80 -m state --state NEW, ESTABLISHED -j ACCEPT
iptables -A OUTPUT -p tcp --sport 80 -m state --state ESTABLISHED -j ACCEPT
iptables -A INPUT -p tcp --dport 443 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -p tcp --sport 443 -m state --state ESTABLISHED -j ACCEPT
```

Allowing Ping

```
iptables -A OUTPUT -p icmp --icmp-type echo-request -j ACCEPT
iptables -A INPUT -p icmp --icmp-type echo-reply -j ACCEPT
```

Showing status of the firewall

```
iptables -L -n -v --line-numbers
```

3.6.2. nftables

iptables replacement. Better in Performance and Flexibility.

3.6.3. Other Tools

There are also Tools like Snort, Security Onion or RedHunt Linux.

3.7. WRITING A NETWORK SCANNER USING PYTHON (RED TEAM)

Python-nmap is a python library which helps in using nmap port scanner and create your own tools. It allows to easily manipulate nmap scan results and is great if you want to automate scanning tasks and reports.

3.8. REVERSE AND BIND SHELLS WITH NETCAT (RED TEAM)

- ***Bind Shell:*** Attacker connects to victim on listening port
- ***Reverse Shell:*** Victim connects to attacker on listening port

Netcat can be used for this.

3.9. SCAPY

Scapy is a python utility that allows to send, sniff, dissect and forge IP packets. Is also used to create attack signature for IDS/IPS systems.

4. SYMMETRIC ENCRYPTION AND KEY EXCHANGE

When to use encryption: User/Company Database Records, cloud storage, password storage, compliance, in transit: Network communication, Authentication data, Payment and money transfers, …

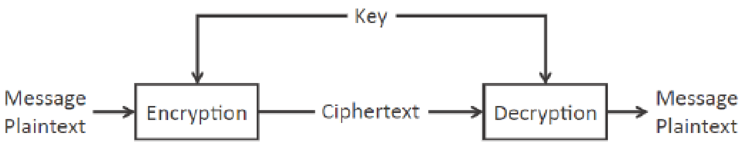
4.1. CRYPTOGRAPHY CONCEPTS

Message or plaintext

Before a message is put into a ***coded form***, it is known as a plaintext message. → Unencrypted Text

Ciphertext

The sender of a message uses a ***cryptographic algorithm*** to ***encrypt the plaintext message*** and produce a ***ciphertext*** message. → Encrypted Text



Cipher

The ***encryption algorithm*** is also called ***cipher***. An algorithm is a set of rules, usually mathematical, that dictates how enciphering and deciphering processes are to take place.

Cryptographic key

A key is nothing more than ***a (very large) number***. Every algorithm has a specific key space, A key space is defined by its bit size. The key space is the range of number from 0 to 2^n, where n is the bit size of the key. A 128-bit key can have a value from 0 to 2^{128}. It is absolutely critical to protect the security of secret keys.

One-Way functions

A one-way function is a mathematical operation that ***easily produces output values*** for each possible combination of inputs but makes it ***impossible to retrieve the input values***. In practice, it has never been proven that any specific known function is truly one-way. Cryptographers rely on functions that the believe are one-way. It is always possible that they might be broken by future cryptanalysts.

Reversibility

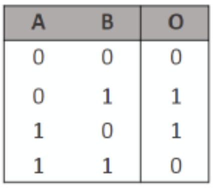
A very important property in cryptography, because we should be ***able to undo the operation of encryption***.

Nonce

The nonce must be a ***unique number*** each time it is used (could be a ***counter***). It is used to make sure that a key is not re-used twice. The nonce is public, whereas the (shared) key is private.

XOR

Binary operator between two values that ***returns true if either input or the other is true but not both***. Useful in Cryptography. B is deciding whether A will change. Anytime B is 1, the output is the opposite of A. Anytime B is 0, A passed through unchanged. Applying XOR twice, it reverses its effect. B is the message and A is the key. Think of A as encrypting B and then decrypting it again. $A \oplus B \oplus A = B$



A	B	O
0	0	0
0	1	1
1	0	1
1	1	0

Initialization vector (IV)

A ***random bit string***. Same length as the block size and is XORed with the message. IVs are used to ***create unique ciphertext*** every time the same message is encrypted using the same key.

Confusion

Occurs when the ***relationship between*** the ***plaintext*** and the ***key*** is so ***complicated*** that an attacker can't merely continue altering the plaintext and analyzing the resulting ciphertext to determine the key.

The mapping between input and output is very confusing. Substitution of bytes adds confusion. Example: Enigma machine: only confusion, no diffusion.

Diffusion

Occurs when ***a change in the plaintext results in multiple changes spread throughout the ciphertext***. A small change in the input leads to a big change on the output. Permutation of bytes adds diffusion.

The Kerckhoff's principle («the enemy knows the system»)

A cryptographic system should be ***secure*** even if ***everything about the system, except the key, is public knowledge.*** Public exposure may expose weaknesses more quickly, leading to the abandonment of insufficiently strong algorithms and quicker adoption of suitable ones.

SP-Network

Algorithm that uses repeated substitution and permutation operations.

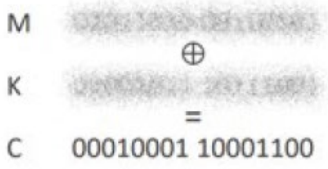
- ***Substitution:*** Replacing bytes with others
- ***Permutation:*** Swapping bytes around
- The substitutions and permutations are combined into a round
- Rounds are then repeated many times

Caesar cipher / ROT3 cipher

One of the earliest known cipher systems was used by Julius Caesar to communicate with Cicero in Rome while he was conquering Europe. To encrypt a message, you simply ***shift each letter of the alphabet three places to the right***.

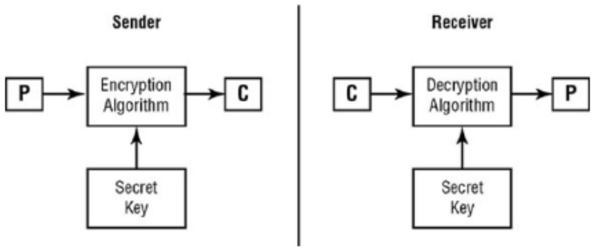
One Time pad

Cipher that uses XOR to encrypt and decrypt a message. Uses a ***key*** that's the ***same length as the message***, ***XOR*** each message bit with each key bit. If you take away the key, there is no way to find the message because there is no statistical mapping between input and output. ***It is not possible to break that cipher,*** but it is ***not practical with large amounts of data***. A 1GB file would need a 1GB key. And if you ever reuse a key, the entire cipher is broken.



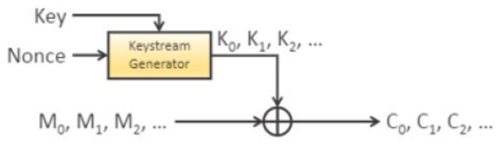
4.2. SYMMETRIC CRYPTOGRAPHY

Symmetric key algorithms rely on a ***«shared secret» key*** that is distributed to all members who participate in the communications. This ***key is used by all parties to both encrypt and decrypt messages***. When large-sized keys are used, symmetric encryption is ***very difficult to break***. It provides only the security service of ***confidentiality***.



Stream Cipher

We can approximate a one-time pad by generating an ***infinite pseudo-random keystream***. Stream ciphers work on messages of any length.



+ Encryption of ***long continuous streams***, possible of unknown length

+ ***Extremely fast*** with low memory footprint, ideal for low-power devices

+ If designed well, it can ***seek any location*** in the stream

- The keystream must appear ***statistically random***
- You must ***never reuse*** a key + nonce
- Stream ciphers ***do not protect the ciphertext*** (no guaranteed integrity)

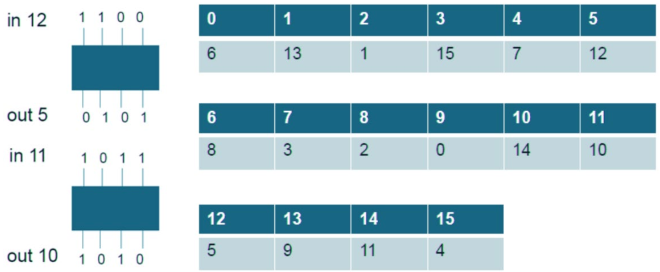
Block Cipher

Block ciphers ***take an input of a fixed size and return an output of the same size***. They attempt to hide the transformation from message to ciphertext through ***confusion*** and ***diffusion***. Most block ciphers are SP-Networks.

The ***Advanced Encryption Standard (AES)*** is an SP-Network. Almost everything uses AES.

Basic substitution box

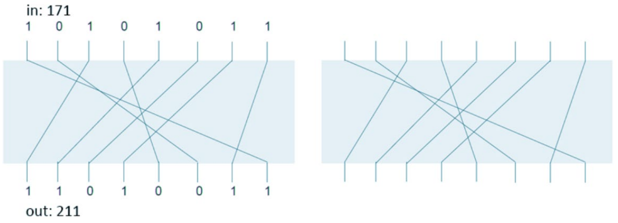
Every input has a different predefined output



0	1	2	3	4	5
6	13	1	15	7	12
6	**7**	**8**	**9**	**10**	**11**
8	3	2	0	14	10
12	**13**	**14**	**15**		
5	9	11	4		

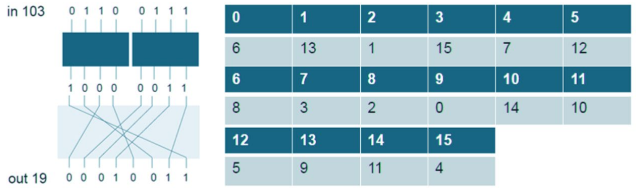
Basic permutation box

Bit position gets changed in a predetermined way



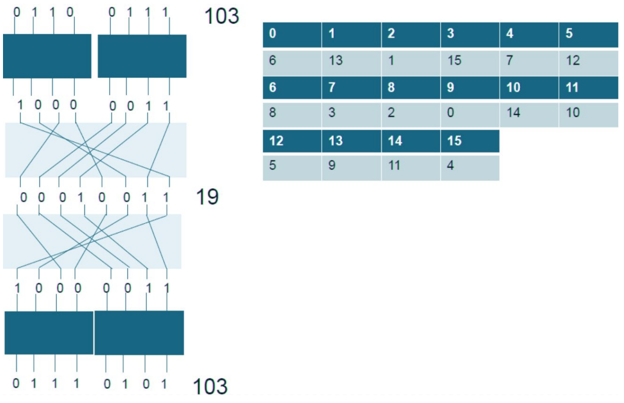
Basic SP-Network

Combination of substitution & permutation



0	1	2	3	4	5
6	13	1	15	7	12
6	**7**	**8**	**9**	**10**	**11**
8	3	2	0	14	10
12	**13**	**14**	**15**		
5	9	11	4		

Basic SP-Network: Decryption and encryption



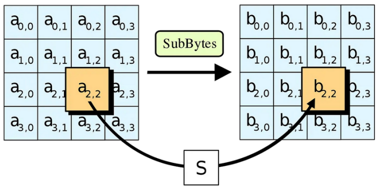
0	1	2	3	4	5
6	13	1	15	7	12
6	**7**	**8**	**9**	**10**	**11**
8	3	2	0	14	10
12	**13**	**14**	**15**		
5	9	11	4		

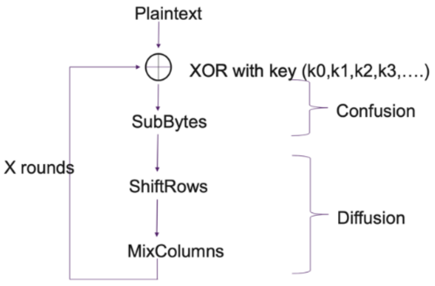
Advances Encryption Standard (AES)

A standard built around the Rijndael algorithm. Superseded DES as a standard in 2002. SP-Network with a 128-bit block size. Key length of 128, 192 or 256-bits. 10, 12 or 14 rounds.

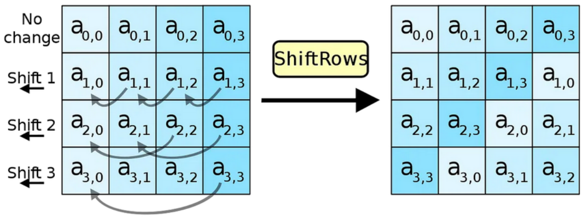
Each round:

- ***SubBytes:*** Each cell of 8 bits gets substituted with another 8 bits. Those 8 bits will be chosen from a predefined loop up table.


$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
SubBytes
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$
S



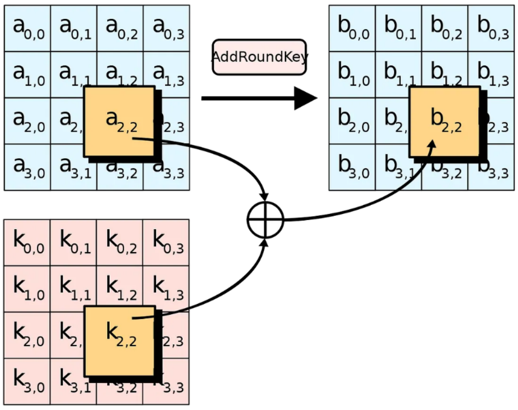
- ***ShiftRows:*** Shifts each row of the matrix to the left. First row: no changes, Second row: 1 to the left, Third row: 2 to the left, …


No change
Shift 1
Shift 2
Shift 3
ShiftRows
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,1}$ $a_{1,2}$ $a_{1,3}$ $a_{1,0}$
$a_{2,2}$ $a_{2,3}$ $a_{2,0}$ $a_{2,1}$
$a_{3,3}$ $a_{3,0}$ $a_{3,1}$ $a_{3,2}$

- ***MixColumns:*** Perform matrix multiplication between the current matrix and a predefined given matrix. The sum operation is substituted by *xor* and multiplication for *and*.

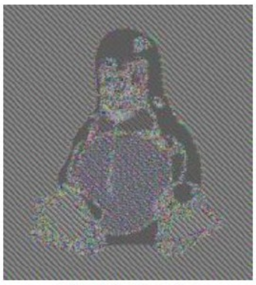
$$\begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} \times \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} = \begin{bmatrix} b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} \\ b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} \end{bmatrix}$$

- ***Key Addition:*** After all the other steps, the information is not yet encrypted. In this step, the matrix gets XORed with the key matrix.



Mode of operations for block ciphers

Messages of exactly 128-bits are unlikely. We need to be able to encrypt messages that are longer or shorter. A mode of operation is the combination of multiple instances of block encryption into a usable protocol. There are 3 modes of operations:


The ECB Penguin

- ***Electronic Code Book (ECB):*** Encrypts each block one after another. The same input creates the same output, so it is weak to redundant data divulging patterns. Not recommended
- ***Cipher Block Chaining (CBC):*** XOR the output of each cipher block with the next input. Not parallelizable, better than ECB but not perfect.
- ***Counter Mode (CTR):*** Encrypting a counter (Nonce + n) to produce a stream cipher. Can be parallelized. Converts a block cipher into a stream. Does not encrypt the message, but a random number and use that to XOR the message. Standard mode for all type of encryption cipher (AES)

Key distribution

Parties must have a ***secure method of exchanging the secret key*** before establishing communications with a symmetric key protocol. Symmetric key cryptography does ***not implement nonrepudiation***. Because any communicating party can encrypt and decrypt messages with the shared secret key, there is no way to prove where a given message originated.
Symmetric key cryptography does ***not implement message integrity***, we do not know if a message has been modified during transit.

The major strength of symmetric key cryptography is the great ***speed*** at which it can operate. 1'000 to 10'000 times faster than asymmetric algorithms.

4.3. DIFFIE-HELLMANN KEY EXCHANGE

How do we establish a shared secret key?

Every communication handshake in the internet is powered by Diffie Hellman handshake. TLS is relying heavily on DH. ***Two parties can jointly agree a shared secret over an insecure channel***. Not really a key exchange, but a exchange of some parts of the mathematical key so each party can create the key by themselves.

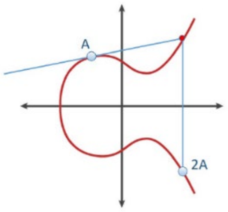
Uses ***prime numbers***, ***modulo*** and ***logarithms***.
g and p are prime numbers. Only $\boldsymbol{g}, \boldsymbol{p}, \boldsymbol{g}^{\boldsymbol{a}} \boldsymbol{mod}\ \boldsymbol{p}$ and $\boldsymbol{g}^{\boldsymbol{b}} \boldsymbol{mod}\ \boldsymbol{p}$ are public. $\boldsymbol{a}$ and $\boldsymbol{b}$ are private.

A and B agree on $\boldsymbol{g = 3}$ and $\boldsymbol{p = 29}$

A chooses $\boldsymbol{a = 23}, 1 \leq \boldsymbol{a} \leq \boldsymbol{p}$ A calculates $\boldsymbol{g}^{\boldsymbol{a}} \boldsymbol{mod}\ \boldsymbol{p} = \boldsymbol{3}^{\boldsymbol{23}} \boldsymbol{mod}\ \boldsymbol{29} = \boldsymbol{8}$	B chooses $\boldsymbol{b = 12}, 1 \leq \boldsymbol{b} \leq \boldsymbol{p}$ B calculates $\boldsymbol{g}^{\boldsymbol{b}} \boldsymbol{mod}\ \boldsymbol{p} = \boldsymbol{3}^{\boldsymbol{12}} \boldsymbol{mod}\ \boldsymbol{29} = \boldsymbol{16}$
A calculates $\left(\boldsymbol{g}^{\boldsymbol{b}}\right)^{\boldsymbol{a}} \boldsymbol{mod}\ \boldsymbol{29} = \boldsymbol{16}^{\boldsymbol{23}} \boldsymbol{mod}\ \boldsymbol{29} = \boldsymbol{24}$	B calculates $(\boldsymbol{g}^{\boldsymbol{a}})^{\boldsymbol{b}} \boldsymbol{mod}\ \boldsymbol{29} = \boldsymbol{8}^{\boldsymbol{12}} \boldsymbol{mod}\ \boldsymbol{29} = \boldsymbol{24}$

The shared secret is **24**

4.4. ELLIPTIC CURVE CRYPTOGRAPHY



Replacement for the mathematics underpinning regular ***Diffie-Hellman***. Is becoming the standard. Elliptic curve is a ***two dimensional curve***. $y^2 = x^3 + ax + b$ The private key is a number, the public key is composed of two numbers (X, Y). The elliptic curve discrete logarithm problem (ECDLP) is ***more difficult to solve*** than the discrete logarithm. Much stronger than other schemes for same key length.

Ephemeral Mode (Perfect forward secrecy): New key exchange for every new session

5. ASYMMETRIC CRYPTOGRAHPY AND HASH FUNCTIONS

5.1. RSA (RIVEST-SHAMIR-ADLEMAN)

Public-key cryptosystem. Widely used. The keys are reversible, either can be used for encryption or decryption.

Use Cases:

- Encryption that only the owner of the public key can read
- Signing that must have been performed by the owner of the private key

5.1.1. Introduction

Public key (e, n)

Public, is sent to users for communication. Has two parts: e (small number) and n (two large prime numbers multiplied together)

Private key (d)

Private, used for signing

Prime factorization

Every number has exactly one prime factorization. ($n = 30 \Rightarrow\ 5 * 3 * 2, n = 589 \Rightarrow 19 * 31$). If n is very big, it is difficult to find the factorization. This is why the factorization is a one-way trapdoor function. There is no way to undo the encryption function unless you know the trapdoor (Knowing the factors).

Generating RSA key pairs is time-consuming and should be done rarely.

5.1.2. Ablauf

1. ***Zwei Primzahlen (*****p**,**q*****) multiplizieren*** zum Produkt **n***:* $\boldsymbol{n} = \boldsymbol{p} * \boldsymbol{q} = \mathbf{3} * \mathbf{11} = \mathbf{33}$

2. Die Eulersche ***φ-Funktion von n berechnen.*** $\varphi(\boldsymbol{p} * \boldsymbol{q})$ ergibt dasselbe Resultat und da beide Zahlen Primzahlen sind, kann einfach $(\boldsymbol{p} - 1) * (\boldsymbol{q} - 1)$gerechnet werden.
$\boldsymbol{\varphi}(\boldsymbol{n}) =\ \varphi(\boldsymbol{p} * \boldsymbol{q}) =\ \varphi(\mathbf{3}) * \varphi(\mathbf{11}) = (\mathbf{3} - 1) * (\mathbf{11} - 1) = 2 * 10 = \mathbf{20}$

3. Eine ***beliebige Zahl a zwischen 1 und*** $\varphi(\mathrm{n})$ ***auswählen***, die mit $\boldsymbol{\varphi}(\boldsymbol{n})$teilerfrend ist:
z.B. **a** = **3**, Test: ggt(20,3)=1

4. Das ***multiplikative Inverse b*** von a in $\mathbb{Z}_{\boldsymbol{\varphi}(\boldsymbol{n})}$ berechnen $a * b\ \equiv 1\, mod\ \boldsymbol{\varphi}(\boldsymbol{n})$
$\mathbf{3} * \boldsymbol{b}\ \equiv 1\, mod\ \mathbf{20}$
Kleinstmögliche Zahl $\mathbf{3} * \mathbf{7} = 21, \frac{21-1}{\mathbf{20}} = 1$ => durch **20** teilbar, also $\boldsymbol{b} = \mathbf{7}$

5. Nun haben wir den ***Public Key (b & n)*** und den ***Private Key(a)*** zur Verschlüsselung:
$\boldsymbol{b} = \mathbf{7}, \boldsymbol{n} = \mathbf{33}\ und\ \boldsymbol{a} = \mathbf{3}$

Text verschlüsseln

Zu sendende Zahlen mit **b** ***potenzieren*** und ***Modulo*** **n** rechnen:
$20^{\mathbf{7}}\ mod\ \mathbf{33}\ \equiv 26,\ 5^{\mathbf{7}} mod\ \mathbf{33}\ \equiv 14,\ 19^{\mathbf{7}} mod\ \mathbf{33} \equiv 13$

Text entschlüsseln

Die empfangene Nachricht mit $\boldsymbol{a}$ ***potenzieren*** und ***Modulo*** **n** rechnen:
$26^{\mathbf{3}}\ mod\ \mathbf{33}\ \equiv 20,\ \ 14^{\mathbf{3}}\ mod\ \mathbf{33}\ \equiv 5\ \ \ 13^{\mathbf{3}}\ mod\ \mathbf{33} \equiv\ 19$

5.1.3. Encryption using RSA

RSA is ***very weak for short messages***. It is not common to see encryption done using RSA (***Replaced by DH***). Is 1000x ***slower*** than symmetric crypto systems.

5.1.4. Signing using RSA

Signing is ***encrypting with the private key***. Is mostly for the ***verification of the integrity of the message*** (Not changed on the way). The message is first ***hashed***. The hash is ***signed*** and ***sent with the message***. The ***receiver hashes the message***, decrypts the signature and ***checks if the hashes match***. The hashes cannot be decrypted.

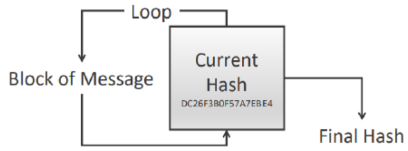
Digital signature is often sent as a ***challenge***. We want the server to ***prove its identity***. The client sends a message to the server that it has to sign. The ***server*** is going to use PSS (probabilistic signature scheme) padding and ***sign*** it with its ***private key***. The ***client*** decrypts the signed message using the ***public key***. If the message ***matches*** the original message, the server has successfully proved itself. This construct (***challenge-response***) is a core part of Transport layer security (TLS).

5.1.5. DSA (Digital Signature Algorithm)

Within a few years, ***RSA is going to become too slow*** because ever bigger keys will have to be used to offset advancements in processing power. The main alternative is DSA. ***Much faster***, but can’t be used for encryption, ***only for signing***.

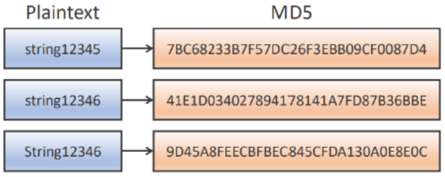
5.2. HASH FUNCTION

Takes a message of any length and returns a pseudorandom hash of fixed length. This hash is non-reversable, meaning you cannot restore the original message from the hash. ***Used for authentication, integrity, passwords etc.*** A good hash algorithm should perform ***quickly but not too quickly*** because then it is easy to break.



5.2.1. Functions

Hash functions ***iteratively jumble blocks of a message after another***. ***One-way function***. The message is processed in blocks, every round there is a new current hash. We loop for every block of the message. When we run out of message, we use the current hash as the final hash.



Plaintext	MD5
string12345	7BC68233B7F57DC26F3EBB09CF0087D4
string12346	41E1D034027894178141A7FD87B36BBE
String12346	9D45A8FEECBFBEC845CFDA130A0E8E0C

The output must be ***indistinguishable from random noise***. Bit changes must ***diffuse*** through the entire output i.e., when one bit changes, the whole hash changes completely. This is called the avalanche effect.

5.2.2. Passwort storage

Hash functions are not good to store passwords because they are ***too fast***. Vulnerable to brute-force attacks. ***PBKDF2*** (Password-Based Key Derivation Function 2) uses a hash similar to SHA-2 but runs it in a loop 5000 times so it is ***slower***. ***Bcrypt*** is an alternative based on a completly different function.

5.2.3. Hash collision

When ***two different inputs get the same hash***, the hash function is ***broken*** and can be exploited. That happened with MD5 – should not be used anymore!

5.2.4. Current standard

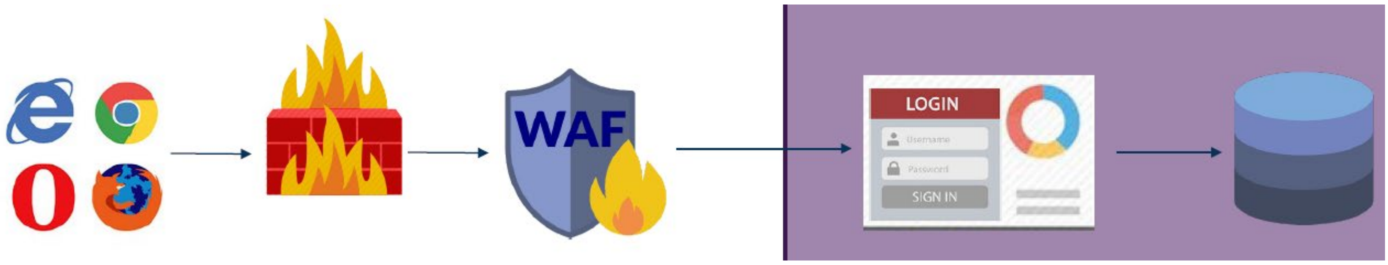
The current standard is ***SHA-2 256 bits and 512 bits***. SHA-1 is a lot better than MD5, but relatively weak. SHA-2 is the same function as SHA-1 but the output length is bigger. SHA-3 is not better or worse than SHA-2, just a different function. Was designed as a stand-in in case something happens to SHA-2.

5.2.5. HMAC (hash message authentication code)

HMAC ***splits a key in two*** and hashes twice. This way, it is ***not vulnerable*** to ***length extension attacks***.

6. WEB SECURITY

6.1. WEB ARCHITECTURE



6.2. HTTP PROTOCOL BASICS

http protocol itself is stateless. A request is sent to the server, the server immediately processes the requests, a response with the process output is returned.

6.2.1. Requests

Structure of a HTTP *GET* Request:

GET /index.html?**showprofile=1** HTTP/1.1	**Method**, **URI**, **Protocol**
User-Agent: curl/7.10.3 (i686-pc-linux-gnu) … **Host:** www.xy.com **Pragma:** no-cache **Accept:** image/gif, image/jpeg, */* **Referrer:** http://www.portal.org/	Header
	Body (empty)

Structure of a HTTP *POST* Request:

POST /index.html HTTP/1.1	**Method**, **URI**, **Protocol**
User-Agent: curl/7.10.3 (i686-pc-linux-gnu) … **Host:** www.xy.com **Pragma:** no-cache **Accept:** image/gif, image/jpeg, */*	Header
showprofile=1	Body

HTTP Request Methods Overview

- ***GET:*** normal use `GET /index.html HTTP/1.1`
- ***POST:*** submit data (login, forms) `POST /webapp/doLogin HTTP/1.1`
- ***HEAD:*** search engines `HEAD /index.html HTTP/1.1`
- ***PUT:*** upload files (webdav)
- ***OPTIONS:*** list available methods in webserver
- ***TRACE:*** debug method in webservers

6.2.2. Response

Structure of a HTTP Response

HTTP/1.x 200 ok	Response Status
Cache-Control: private Content-Type: text/html Content-Encoding: gzip **Server:** GWS/2.1 Content-Length: 1609 **Date:** Wed, 08 Mar 2006 08:43:06 GMT **X-Cache:** MISS from horus.csnc.ch Proxy-Connection: keep-alive	Response Header
<html><head><meta http-equiv="content-type" content="text/html; charset=UTF-8"><title>Google</title>…	HTML Content

6.3. SESSION HANDLING SERVER SIDE

Server: Create server-side memory structure & ***return a key*** to the structure = ***SessionID*** to Client
Client is now ***known to the server*** because he sends the session (token) in every following request.

Where is the SessionID?

`GET or POST /index.html?session=123 HTTP/1.1`	Method, URI, Protocol
User-Agent: curl/7.10.3 (i686-pc-linux-gnu)… **Host:** www.xy.com **Pragma:** no-cache **Accept:** image/gif, image/jpeg, */* **Cookie:** Session=123	Request Header
Session=123	Body

Session is embedded as a hidden input field in the HTML page:

```
<input type=”hidden” name=”session” value=”123”>
```

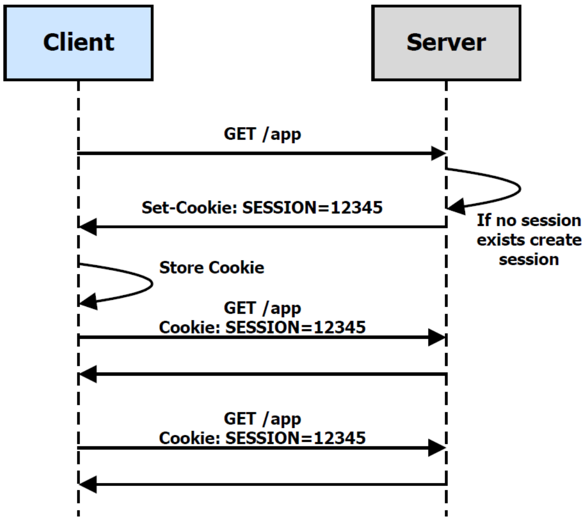
6.4. SESSION HANDLING CLIENT SIDE

- ***Server is stateless:*** Gets session from the client. Server must check the session that is being sent by a client (expiration, validity, authorization)
- ***Everything is held within the client:*** Client gets the session from the Identity Provider. Session is signed, encrypted
- ***The client is not allowed to change the content:*** That’s why the session is signed

Advantage: Scaling, Client is not “sticky” with the same service. Server must be able to check the client state.

6.5. COOKIES

Cookie creation Process: see image



How does the client receive the session-id? Via «Set-Cookie» HTTP Response header from the server.

When does the client insert the cookie to a HTTP request header? Depends on the cookie attributes.

Cookie Attributes

Set-Cookie HTTP header consists of:

- ***NAME*****=CONTENT (key, value pair)**
  `Set-Cookie: jsessionid=1234`
- ***Domain:*** **place where the cookie can be sent to.**
  Default: No domain. Cookie is sent to originating server only.
  `Set-Cookie: jsessionid=123; domain=.hacking-lab.com`
- ***Path:*** **URL-paths where the cookie can be sent to.**
  Default: No path. Cookie is sent to originating path only.
  `Set-Cookie: jsessionid=123; path=/private/`
- ***Secure:*** **Cookie is sent over HTTPS only.**
  Default: insecure
  `Set-Cookie: jsessionid=123; secure`
- ***Expire:*** **Validity period of cookie.**
  Default: No date: Cookie is not persistent on disk
  `Set-Cookie: jsessionid=123; expire=Sun, 17-Jan-2038 19:14_04 GMT`

- ***HttpOnly:*** **Cookie cannot be accessed with JavaScript.**
  ASP.NET 1.x: Flag needs to be added manually. ASP.NET 2.0: HttpOnly property available. Useful for sensitive information, as this prevents the information being read by a XSS attack.
  `Set-Cookie: jsessionid=123; HttpOnly`
- ***SameSite:*** **Prevents the browser from sending this cookie along with cross-site requests.**
  The main goal is to mitigate the risk of cross-origin information leakage. Also provides protection against cross-site request forgery attacks (CSRF).
  `Set-Cookie: jsessionid=123; secure; HttpOnly; SameSite=Strict`

  strict: ***Prevents*** the cookie from being sent by the browser to the target site in ***all*** cross-site browsing context, even when following a regular link. For example, for a GitHub-like website this would mean that if a logged-in user follows a link to a private GitHub project posted on a corporate discussion forum or email, GitHub will not receive the session cookie and the user will not be able to access the project. A ***bank website*** however most likely doesn't want to allow any transactional pages to be linked from external sites so the strict flag is used.

  lax: Provides a reasonable ***balance between security and usability*** for websites that want to maintain user's logged-in session after the user arrives from an external link.

6.6. JAVASCRIPT

Code that is interpreted by the browser. Can access the cookies of the site where the script came from. Is executed withing the web browser, always runs on the client.

```
var cookie = document.cookie
```

6.7. SECURE SOFTWARE

+ Forensic Readiness
+ Secure Programming
+ Patching, Updating
+ Hardening

- Weakness in monitoring
- Weakness in the application
- Weakness in Libraries
- Poorly configured application

6.8. CLASSIFICATION VULNERABILITIES

- ***CVE (Common Vulnerabilities and Exposures):*** List of common identifiers for publicly known cybersecurity vulnerabilities. → Actual security problems found in software
- ***CWE (Common Weakness Enumeration):*** Community-developed list of software and hardware weakness types. → Types of exploits that software should not contain
- ***ASVS (The OWASP Application Security Verification Standards):*** Provides a basis for testing web application technical security controls and provides developers with a list of requirements for secure development.

Vulnerability Types:

- Use of hard-coded credentials
- Information leak through log files
- Use of insufficiently random values
- Information exposure
- Failure to constrain operations within the bounds of a memory buffer
- Divide by zero
- Cleartext storage of sensitive information

6.9. OWASP APPLICATION SECURITY VERIFICATION STANDARD (ASVS)

First application security standard by developers, for developers. Each item is a single concept, each item is testable, completely open source. Adopted by governments, large corporations, secure supply chain procurement agreements, and referenced by relevant standards.

Why not just use the OWASP Top 10?

- Awareness only
- Insufficient for secure software
- Contains ~50 CWEs and categories instead of 10
- Not written for developers
- Not easily testable
- Doesn’t tell you what to do, but rather what not to do.

6.9.1. Level 1: Just basics

Considered the «minimum» level, designed for lower assurance levels, ~136 requirements

6.9.2. Level 2: Recommended Level

Considered the «standard» level, for applications that contain sensitive data, provides additional protections, recommended for most apps, ~130 additional requirements

6.9.3. Level 3: For apps that can kill you or destroy the economy

Considered the «advanced» level, for the most critical applications, high value transactions. ~21 additional requirements, 286 in total.

6.10. OWASP TOP 10

The Open Web Application Security Project is a nonprofit foundation that works to improve the security of (web) software. (2021 version)

- ***A01:*** Broken Access Control
- ***A02:*** Cryptographic Failures
- ***A03:*** Injection
- ***A04:*** Insecure Design
- ***A05:*** Security Misconfiguration
- ***A06:*** Vulnerable and Outdated Components
- ***A07:*** Identification and Authentication Failures
- ***A08:*** Software and Data Integrity Failures
- ***A09:*** Security Logging and Monitoring Failures
- ***A10:*** Server-Side Request Forgery

6.11. A03: SQL INJECTION

Injection flaws occur when an application sends untrusted data to an interpreter. Injection flaws are very prevalent, often found in SQL queries, LDAP queries, Xpathqueries, OS commands, program arguments etc. Easy to discover when examining code, but more difficult via testing.

Modification of SQL query. Example:

```
SELECT Username FROM Users WHERE Username =‘meier’ AND Password =’’ OR 1=1 #
```

The WHERE clause evaluates to TRUE and the user gets authenticated.

6.11.1. Countermeasures

- ***Use prepared Statements:*** Statement gets precompiled, parameters are separate from the SQL Statement. Much faster and save against SQL injection attacks (If used correctly).

```
PreparedStatement updateSales = dbCon.prepareStatement("UPDATE COFFEES SET" + "SALES=?
WHERE COF_NAME LIKE ?");

updateSales.setInt(1,75); //correct

updateSales.setString(2, "Colombian"); //usage

updateSales.executeUpdate();
```

- DB least Privileges
- ***WAF:*** Web Application Firewall
- ***Don't disclose SQL errors to the user:*** Anonymous error messages instead

6.12. A03: XSS INJECTION (CROSS SITE SCRIPTING)

XSS is the most prevalent web application security flaw. XSS flaws occur when an application includes user supplied data in a page sent to the browser without properly validating or escaping that content.

- ***Stored XSS:*** Malicious code permanently stored on a website/application, executed when accessed. *Typical example: message board, comment section*
- ***Reflected XSS:*** Data provided by a web client is used immediately by server-side code to generate a page of results for the user. *Typical example: User inputs a <script> tag into a search form, the script gets executed when the search is started.*
- ***DOM-based XSS:*** Does not require the web server to receive the malicious XSS payload. Instead, the attack payload is embedded in the DOM object in the victim's browser used by the original client side script, so that the client side code runs in an «unexpected» manner. *Example: A URL parameter can be used to place a script tag, which then gets executed by a poorly coded website: http://www.some.site/page.html?default=<script>alert(document.cookie)</script>*

6.12.1. XSS Problem

Failure of the application to properly sanitize output to the user's browser. Improper trust of user supplied data. Bad example, because variable a gets unfiltered data from the request:

```
$a = $_GET['search'];

Print 'Your search results ' . $a;
```

6.12.2. Effects of XSS

- Theft of session cookies / Session Hijacking
- Arbitrary HTML or JavaScript injection
- Exploit injection
- Keystroke Logging
- BeEF & Metasploit can be used to show effects of XSS

JavaScript from Malware Site is generally denied access to session cookie because of the same origin policy. *Example: evil.com cannot request cookies from ubs.ch because the same origin policy requires cookie requests to come from the same domain.*

6.12.3. Countermeasures

- ***Secure Coding:*** Convert dangerous characters (>, <, ", ') into HTML entities `htmlentities(..);`
- ***CSP (Content Security Policy):*** Denies external JavaScript access to session cookies.
- Cookie with ***HttpOnly***, Secure, SameSite=Strict attributes
- ***X-XSS Protection Header*** (Renders a blank page if a reflected XSS is detected by the browser)
- ***WAF:*** Web application firewall

6.13. A01: BROKEN ACCESS CONTROL

Access control enforces policy such that users cannot act outside of their intended permissions. Failures typically lead to unauthorized information disclosure, modification or destruction of all data or performing a business function outside the user's limits. Common access control vulnerabilities include:

- Violation of the principle of least privilege or deny by default
- Bypassing access control checks by modifying the URL (parameter tampering) or modifying API requests
- Permitting viewing or editing someone else's account
- Accessing API with missing access controls for POST, PUT and DELETE
- Elevation of privilege
- Metadata manipulation like tampering or replaying JSON Web Token access control token, or a cookie or hidden field manipulated
- CORS (Cross-Origin Resource Sharing, the thing prevented by Same-Origin-Policy) misconfiguration allows API access from unauthorized origins

6.13.1. Login Service Attacks

User Enumeration

Verbose login related error messages can lead to user enumeration (Like «Password incorrect» or «User unknown», this tells attackers that the specified account exists, but the password is wrong). Login error messages must be neutral («Username or Password incorrect»).

Remedy Brute-force Attacks

User accounts should be blocked after a certain number of tries. Locked accounts should be unlocked after a certain time. Monitor failed login attempts and detect Brute-force attacks and block the attacking IP addresses on the firewall.

6.13.2. Session Fixation Attack

Special form of session hijacking. Hacker tricks the victim to use a session known to the hacker.

6.13.3. Countermeasures – Broken Authentication and Session Management

- Anonymize error messages
- Enforce strong authentication
- Block Password Brute-Forcing, Time-based lock-out, Captcha, OTP
- Monitor attacks

6.14. A07: IDENTIFICATION AND AUTHENTICATION FAILURES

For direct references to restricted resources, the application needs to verify the user is authorized to access the exact resource they have requested. If the reference is an indirect reference, the mapping to the direct reference must be secured. *Example: If your bill can be accessed by swisscom.ch/bills?id=12345, can you also access other bills by changing the ID?*

6.14.1. Countermeasures

Authorization is the function of specifying access rights to resources related to information security and computer security in general and to access control in particular.

6.15. A05: SECURITY MISCONFIGURATION

Developers and network administrators need to work together to ensure that the entire stack is configured properly.

- ***Can happen at any level of an application stack:*** Operating system, Webserver, Application server, Framework, Custom Code

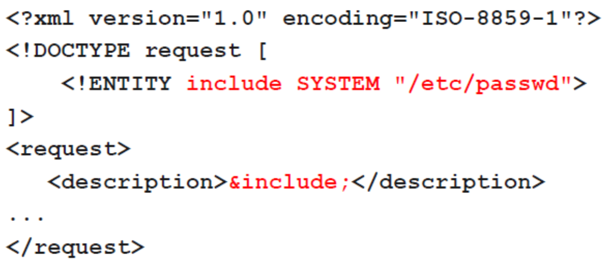
- ***Can be detected by automated scanners:*** Missing patches, Directory Listings, Default accounts
- ***Is often caused by non-existent or incomplete (administrative) processes***

6.15.1. Countermeasures

- ***Hardening:*** Make it very difficult for the attackers to work on a vulnerable server. Process privileges, file privileges, samples, admin interfaces, unused services and components, hardening code components
- ***Jailing/Isolation:*** chroot (programs can only access files within a specified folder), docker, VM

6.15.2. XML External Entity Attack (XXE)

Attack range: DoS, Inclusion of local files, Port scanning, Overloading of XML-Schema, NTLM authentication material theft by accessing files on a network share.



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE request [
    <!ENTITY include SYSTEM "/etc/passwd">
]>
<request>
   <description>&include;</description>
...
</request>
```

Example: A new variable named `include` is created with the `!ENTITY` command. With the `SYSTEM` parameter, one can read external files and store them as a value, like here with the file `/etc/passwd`. which is subsequently printed on the website in the <description> tag.

Countermeasures

Xerces Hardening: Hardening of the XML parser. Validate schemas features, Avoid external entity attacks, avoid resolving of external XML schema locations. Check XML against local server-side schemas and DTDs (Document Type Definitions, structure of an XML file, like a SQL Table definition).

6.16. A02: CRYPTOGRAPHIC FAILURES

- ***Sensitive Data Exposure:*** Passwords, credit card numbers, health records etc. should be protected. Encrypt with current standards, ensure proper private key management
- ***Hashed and Salted User Passwords:*** Do not store passwords in plain-text
- ***Unprotected APIs:*** Carefully choose a strategy to test all defenses that matter

6.16.1. Countermeasures

- ***Bind attribute:*** Will let you specify the exact properties a model binder should include (Whitelist)
  Example: A user has a FirstName and a IsAdmin attribute. Users can change their name via a GET request. An attacker can add a IsAdmin parameter to the URL and receive Admin rights. Through the bind property, the valid fields that can change with a request can be limited
- ***SSL + TLS:*** Use the secure & HttpOnly attributes when creating cookies
- ***Reverse Proxy:*** Entry Server, Reverse Proxy, Secure Gateway

6.17. A10: SERVER-SIDE REQUEST FORGERY / CROSS SITE REQUEST FORGERY

The easiest way to check whether an application is vulnerable is to see if each link and form contains an unpredictable token for each user. Without such an unpredictable token, attackers can forge malicious requests. Focus on the links and forms that invoke state-changing functions, since those are the most important CSRF targets.

XSRF != XSS

XSS exploits the trust the client has for the website. XSRF exploits the trust a website has for the user.

6.17.1. Countermeasures

Cookie Attribute: SameSite=Strict or SameSite=Lax

6.18. A03: INJECTION

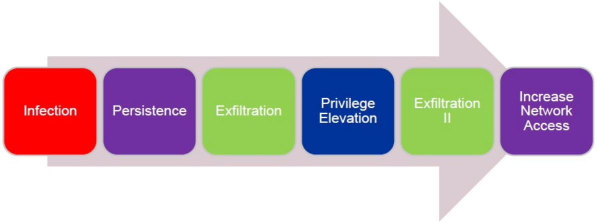
Insecure Deserialization – WAF does not know how to deserialize and check and is useless

6.19. A06: SECURITY LOGGING AND MONITORING FAILURES

Insufficient Logging, Fraud detecting system on critical user actions (i.e., money transfer)

7. DETECTION & RESPONSE

7.1. APT (ADVANCED PERSISTENT THREATS)



Advanced Persistent Threats (APTs) are ***sophisticated attacks*** in which a specific target is attacked over a ***longer period of time*** (years). They are used by ***skilled attackers*** to penetrate networks undetected, steal data or conduct ***espionage***. APTs are ***difficult to detect*** and use ***advanced techniques*** to circumvent security measures. Comprehensive protection and proactive security measures are required to defend against APTs. Often associated with ***zero-day exploits***.

7.2. CYBER DEFENSE FRAMEWORKS

Documenting Attacks and Adversaries: Documentation is still text based. Framework needed to have a common description language.

- ***«Cyber Kill Chain»*** Has eight phases:
  - Reconnaissance *The observation stage: attackers typically assess the situation from the outside-in, in order to identify both targets and tactics for the attack.*
  - Intrusion *Based on what the attackers discovered in the reconnaissance phase, they're able to get into your systems: often leveraging malware or security vulnerabilities.*
  - Exploitation *The act of exploiting vulnerabilities, and delivering malicious code onto the system, in order to get a better foothold.*
  - Privilege Escalation *Attackers often need more privileges on a system to get access to more data and permissions: for this, they need to escalate their privileges often to an Admin.*
  - Lateral Movement *Once they're in the system, attackers can move laterally to other systems and accounts in order to gain more leverage: whether that's higher permissions, more data, or greater access to systems.*
  - Obfuscation *In order to successfully pull off a cyberattack, attackers need to cover their tracks, and in this stage they often lay false trails, compromise data, and clear logs to confuse and/or slow down any forensics team.*
  - Denial of Service *Disruption of normal access for users and systems, in order to stop the attack from being monitored, tracked, or blocked*
  - Exfiltration *The extraction stage: getting data out of the compromised system.*
- ***«Diamond Model»*** Consists of four basic components:
  - Adversary *Name, alias, origin, motivation, description*
  - Infrastructure *IP addresses, malware used, Email addresses*
  - Victim *Location, vertical, goal, person, o35rganization*
  - Capability *Attack methods, targets, operation manual, malware*
- ***STIX:*** Structured Threat Information eXpression, describes cyber threat information in 4 dimensions: Motivation, Abilities, Capabilities & Response. JSON based document
- ***TAXII:*** Trusted Automated eXchange of Intelligence Information, standardized language
- ***MISP:*** Threat intelligence platform that facilitates the exchange and sharing of threat intelligence, Indicators of Compromise, Targeted malware and attacks, financial fraud…
- ***MITRE «ATT&CK»:*** Adversarial Tactics, Techniques & Common Knowledge. Framework to document common tactics, techniques and procedures that APT use against Windows enterprise networks.

7.2.1. SIEM (Security Information and Event Management)

Responsible for ***collecting log*** and ***event data*** from various sources such as network, servers and applications and ***aggregating***, ***identifying***, ***categorizing,*** and ***analyzing*** it in real time. With a SIEM solution, security problems should be ***detected automatically*** as well as the ability ***to send an alert***.

Enables ***pattern search in log data*** for indicators of a cyberattack (IOC), enables ***correlation of event informatio***n and ***identifies abnormal activity***, alerts according to defined alert rules.

7.2.2. SOAR (Security Orchestration, Automation and Response Solutions)

SOAR also ***collects data from various sources*** similar to SIEM, but SOAR takes a step further and ***supports the incident responder*** in managing the crisis. SOAR enables ***automated intervention*** when a security incident occurs. A SOAR system also supports the incident responder in ***rolling out security countermeasures***. Alert Investigation, Orchestration, Automation workflow.

7.2.3. EDR (Endpoint Protection and Response)

Also known as Endpoint Threat detection and response (ETDR), is an ***integrated endpoint security solution*** that combines ***real-time continuous monitoring*** and ***collection of endpoint data*** with rules-based automated response and analysis capabilities.

7.2.4. CSIRT

A CSIRT is a ***team of IT security experts*** whose main business is to ***respond to computer security incidents***. It provides the necessary services to handle them and support their constituents to recover from breaches.

7.2.5. YARA

Powerful ***keyword scanner***, uses rules designed to identify binary patterns in bulk data. Is optimized to scan for many rules simultaneously.

8. INTRODUCTION TO ETHICAL HACKING AND PENETRATION TESTING

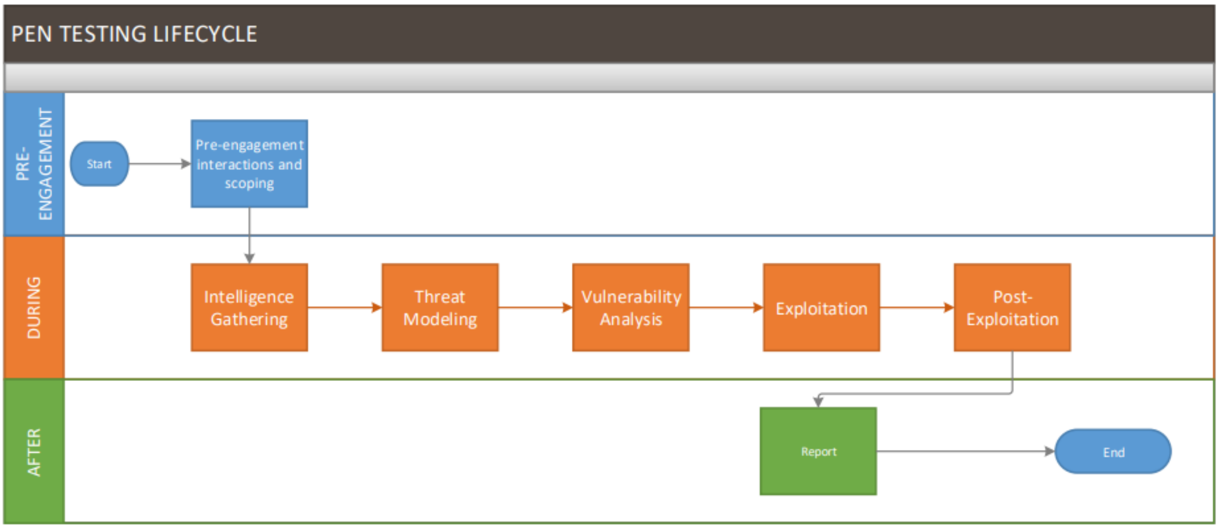
- ***Hacking:*** Exploiting vulnerabilities in systems and / or software to gain unauthorized access
- ***Ethical Hacking:*** Using tools and techniques to validate, audit and report on system / software vulnerabilities

Hacker Types

- ***Black Hat:*** Malicious, destructive hacker that usually remains anonymous
- ***Grey Hat:*** Those possessing black hat skills who focus on both offense and defense
- ***White Hat:*** Those possessing black hat skills who primarily focus on defense
- ***Script Kiddie:*** Individuals that use tools without understanding what they are doing
- ***Cyber Terrorist:*** Skilled attacker whose purpose it is to further an ideology
- ***State Sponsored:*** Hackers employed by the government for both offensive and defensive activities
- ***Hacktivists:*** A hacker whose activity is aimed at promoting a cause

Penetration testing

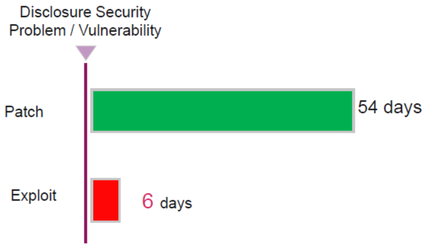
For Pen testing, a contract, statement of work and non disclosure agreement is drawn up.



8.1. MALICIOUS CODE

Malicious code objects are ***computer security threats that exploit*** various network, operating system, and software vulnerabilities to ***spread malicious payloads*** to computer systems. Most of the ***computer viruses*** and ***Trojan horses depend on irresponsible computer use by humans*** to spread from system to system. ***Worms spread rapidly*** among vulnerable systems ***under their own power***.

8.1.1. Zero-Day attack



Exploits a Zero-Day vulnerability which are ***security flaws discovered by hackers*** that have not been thoroughly addressed by the security community.
A Zero-Day exploit is ***not known to the software vendors***. The ***delay*** between the ***discovery*** of a new type of malicious code and the issuance of ***patches*** and antivirus updates is known as the ***window of vulnerability***. Lots of system are long vulnerable to Zero-day attacks because of the slowness in applying updates on the part of system administrators.

8.1.2. Source of Malicious Code

- ***Script Kiddie:*** Anyone with a minimal level of technical expertise can create a virus with freely available tools and unleash it upon the internet.
- ***Drive-by download:*** Unintentional download of malicious code to your computer or mobile device that leaves you open to a cyberattack. Takes advantage of an app, operation system or web browser that contains security flaws due to unsuccessful updates or lack of updates. Does not rely on the user to do anything to actively enable the attack.
- ***APT (Advanced persistent threat):*** Sophisticated adversaries with advanced technical skills and significant financial resources. Military units, Intelligence agencies or other shadowy groups. Often have access to zero-day exploits. Malware built by APTs is highly targeted, designated to impact only a small number of adversary systems and difficult to defeat.
- ***Viruses:*** Has two main functions, propagation and destruction. The propagation function defines how the virus will spread from system to system. The destructive power is delivered by the virus's payload by implementing whatever malicious activity the virus writer had in mind.

8.2. VIRUS PROPAGATION TECHNIQUES

By definition, a virus must contain technology that enables it to ***spread*** from system to system. Once the virus has «touched» a new system, they use one of several propagation techniques to ***infect the new victim*** and expand their reach:

- ***Master boot record (MBR) infection:*** attack the portion of bootable media that the computer uses to load the operation system during the boot process. Doesn't contain all the code required to implement the virus's propagation and destructive function and is extremely small. Store most of their code on another portion of the storage media, which will be loaded later. Rarely used today.
- ***File infection:*** Infect executable files. Often self-contained executable files that escape detection by using a filename like a legitimate operating system file. Often easily detected.
- ***Macro infection:*** Virus that is written in a macro language, inside a software application / document. Start automatically when opening the infected file. Common in MS Office files.
- ***Service injection:*** Inject themselves into trusted runtime processes of the operation system. The malicious code can bypass detection by any antivirus software running because those processes are trusted.

8.3. MALWARE TECHNOLOGIES

- ***Multipartite Viruses:*** Use more than one propagation technique to penetrate systems that defend against only one method.
- ***Stealth Viruses:*** Hide themselves and fool antivirus packages into thinking that everything is functioning normally.
- ***Polymorphic Viruses:*** Modify their own code as they travel from system to system. Propagation and destruction techniques remain the same, the signature is somewhat different on each system. The constantly changing signature will render signature-based antivirus packages useless.
- ***Encrypted Viruses:*** Use cryptographic techniques to avoid detection. A very short segment of code known as the virus decryption routine contains the information necessary to load and decrypt the main virus code stored elsewhere on the disk. Each infection utilizes a different cryptographic key, causing the main code to appear completely different on each system.
- ***Logic Bombs:*** Lie dormant until they are triggered by the occurrence of one or more conditions such as time, program launch, website logon.
- ***Trojan Horses:*** Software program that appears «kind» but carries a malicious, behind-the-scenes payload. Example: a rogue antivirus software.
- ***Keystroke logging/Keylogger:*** The action of recording the keys struck on a keyboard. Data can be retrieved by the person operating the logging program. Used for stealing passwords or other information
- ***Ransomware:*** Infects a target machine and then uses encryption technology to encrypt files stored on the system with a key known only to the malware creator. The user must pay ransom (often in Cryptocurrency) to regain access to their files.
- ***Worms:*** Malicious code objects that propagate themselves without requiring any human intervention (mostly through internal networks, sometimes over the internet).
- ***Spyware:*** Monitors action and transmits important details to a remote system that spies on you.
- ***Adware:*** Displays advertisements on infected computers.

8.4. ANTIVIRUS

An Antivirus software is a computer program used to prevent, detect and remove malware. If possible, the antivirus package eradicates the virus, disinfects the affected files and restores the machine to a safe condition. If not possible, the files can be quarantined or deleted to preserve system integrity. There are different ways to detect a virus:

- ***Signature-based:*** The antivirus package maintains a database that contains the characteristics of all known viruses. Newly created viruses are not detectable because they are not in the database yet.
- ***Heuristic-based:*** The antivirus package analyzes the behavior of software, looking for the signs of virus activity, such as attempts to elevate privilege level, coverage of electronic tracks, and alteration

unrelated or operating system files. If the software behaves suspiciously in that environment, it is added to blacklists throughout the organization, rapidly updating antivirus signatures.

- ***Data integrity:*** Designated to alert administrators to unauthorized file modifications. These systems work by maintaining a database of hash values for all files stored on the system. These archived hash values are then compared to current computed values to detect any files that were modified between the two periods.

8.5. APPLICATION ATTACKS

Buffer Overflows

Buffer overflow vulnerabilities exist when a developer does ***not properly validate user input*** to ensure that it is of an ***appropriate size***. Input that is too large can ***«overflow»*** a data structure to ***affect other data*** stored in the computers memory. *Example: Trying to store a 100-char string in a 64 byte array*

Time of Check to Time of Use (TOCTTOU or TOC/TOU)

Timing vulnerability that occurs when a program ***checks access permissions too far in advance*** of a resource request. If an operating system ***builds a comprehensive list of access permissions*** for a user upon logon and then ***consults that list throughout the logon session***, a TOCTTOU vulnerability exists. If the system administrator revokes a particular permission, that restriction would not be applied to the user until the next time they log on. If a user is logged on when the access revocation takes place, they will have access to the resource indefinitely.

Back Doors

Undocumented command sequences that allow individuals with knowledge of the back door to ***bypass normal access restrictions***. Can be ***created*** by ***malicious code*** on infected systems or by ***software developers***. They are used during the development and ***debugging*** process to ***speed up the workflow*** and avoid to continuously authenticate to the system.

Escalation of Privilege and Rootkits

An escalation of privilege attack occurs when an attacker ***expands his access from a normal user account to mor administrative access***. A ***rootkit*** is a ***way to perform an escalation of privilege*** attack. They exploit known vulnerabilities in various operation systems and provide the hacker the possibility to increase his access to the root level.

8.6. WEB APPLICATIONS ATTACKS

8.6.1. Cross-Site Scripting (XSS)

Occur when web applications perform reflection. ***Scripts*** can be ***embedded*** in web pages by using the HTML tags `<script></script>`. A successful attack can allow the attacker to execute JavaScript in the victims browser.

It can be found in search fields that echo a search string back to the user, input fields that echo user data, error messages that return user supplied text, hidden fields that contain user supplied data, ***any page that displays user supplied data***.

Example: Consider a web application that contains a text box asking the user to enter his name. When the user clicks submit, the web application loads a new page that says «Hello, Name». If a script is entered, like `<script>alert("hello")</script>`, the web application reflects the input on the web page and the browser processes it and executes the script.

8.6.2. Cross-Site Request Forgery (XSRF or CSRF)

Attackers ***embed code in one website that sends a command to a second website***. When the user clicks the link on the first site, he or she is unknowingly sending a command to the second site. If the user happens to be logged into that second site, the command may succeed.

Similar to cross-site scripting but ***exploit a different trust relationship***. ***XSS*** attacks ***exploit*** the ***trust*** that a

user has in a website to execute code on the user's computer. ***XSRF*** attacks exploit the trust that ***remote sites have in a user's system*** to execute commands on the user's behalf.

Developers should ***protect*** their web applications against XSRF attacks with ***secure tokens*** and ***checking*** the ***referring URL*** in requests.

Example: *An attacker wants to steal funds from user accounts of an online banking site. The attacker goes to an online forum and posts a message containing a link. The link is actually a link directly into the money transfer site that issues a command to transfer funds to the attackers account. If the user who clicks on the link happens to be logged into the banking site, the transfer succeeds.*

8.6.3. Dynamic Web Applications

The web server is in a separate network zone called a ***demilitarized zone (DMZ)***. It is ***publicly accessible***.
The ***database*** server is in the ***internal*** network, it is ***not*** meant for ***public*** access.
The firewall administrator must create a ***rule allowing access*** from the web server to the database server. This rule creates a ***potential path*** for internet users to ***gain access*** to the database server.

Web applications use a database to create content on demand when the user makes a request. If there is a flaw in the web application, it may allow individuals to attack the database by using ***SQL injection*** attacks.

8.6.4. SQL Injection Attacks

Allow a malicious individual to directly perform ***SQL transactions*** against the underlying database.
Unexpected input into a web application allow an attacker to gain unauthorized access to a database. The user may be able to ***insert his own SQL code into the statement executed by the web server***. ***Databases will process multiple SQL statements at the same time***, provided that you end each one with a semicolon.

Example: *A bank customer enters an account number to gain access to a dynamic web application that retrieves current account details. The web application uses a SQL query to obtain that information:* `SELECT * FROM transactions WHERE account_number = "<number>"`*. If the user's account number is 145249, it is possible to enter the following into the <number> field:* `"145249"; DELETE * FROM transactions WHERE "a" = "a"`*. This is a valid SQL transaction containing two statements, the second of which deletes all the records stored in the database.*

Protection against SQL Injection Attacks

- ***Use prepared statements:*** Store the SQL prepared statements on the database server. Web applications calling the prepared statements may pass parameters to it but cannot alter the statement.
- ***Limit Account Privileges:*** The database account used by the web server should have the smallest set of privileges possible. Limit the web application to only retrieve data.
- ***Perform Input Validation:*** Input validation allows you to limit the types of data a user provides in a form. Removing the single quote character from the input can prevent a lot of attacks. The safest form of input validation is whitelist validation, where the developer specifies the exact nature of the expected input and the code verifies that user-supplied input matches the expected pattern before submitting it to the database.

8.7. NETWORK SECURITY

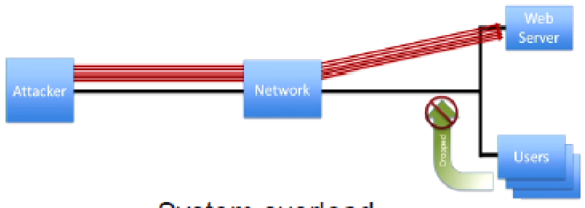
Why are network devices getting hacked?

- Used as stepping stones
- Sometimes not monitored as closely as your hosts
- Longer system lifecycle
- No malware detection
- Sometimes running protocols designed back in the 80ies
- Take advantage of features like port mirroring, tunneling to infiltrate and exfiltrate data.
- Protocol authentication only supports MD5

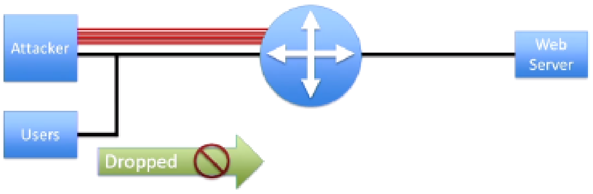
8.7.1. Denial of Service (DoS) Attack

Resource consumption attack that has the primary goal of ***preventing legitimate activity*** on a victimized system. DoS attacks occur between one attacker and one victim. A DoS attack renders the target unable to respond to legitimate traffic.

System overload and Link overload: Attacks that flood the victim's communication pipeline with garbage network traffic.



System overload



Link overload

- ***SYN Flooding:*** Send TCP SYN segments to open ports with a spoofed source IP address. Server replies with SYN/ACK to spoofed source. Leads to a lot of half-open connections until the server cannot accept new connections.
- ***Service Request Floods:*** Create many connections to a service. Request service from the application with the goal to exhaust server resources.
- ***Application level DoS:*** Attacks exploiting a vulnerability in hardware or software. Ist is not about the quantity but the quality. This exploitation of a weakness or error causes a system to crash, hang, freez or consume all system resources.
- ***Permanent DoS:*** Installing compromised hardware updates that render the hardware useless
- ***Botnets:*** A botnet is a logical collection of Internet-connected devices such as computers, smartphones or IoT devices whose security have been breached and control ceded to a third party. Each compromised device, known as a bot or zombie is added to the botnet when a device is penetrated by a malicious software. The attacker installs remote-control tools onto these zombies. A botnet is under the control of an attacker known as the botmaster. Zombies are called secondary victims because they don't know that they have become part of a botnet. Botnets are increasingly rented out by cyber criminals as commodities. Botnet software can propagate to grow the network in an automated fashion.
- ***Distributed DoS (DDoS):*** The controller of a botnet is able to direct the activites of these compromised computers. In response to a launch command from the botmaster, each Zombie conducted a DoS attack against the victim. The victim may be able to discover zombie systems that are causing the DoS attack but won't be able to track down the actual attacker.

8.7.2. Man-in-the-Middle Attack

The attacker inserts itself ***in-between the client and the server***. The ***traffic*** is ***forced through the attacker*** machine. The attacker can now ***view*** and ***control*** all network traffic

Man-in-the-Browser Attack: Trojan horse program installed as a browser plugin. Capture form data, such as usernames and passwords. Inject JavaScript into webpages, Hijack authentication sessions.

8.7.3. Eavesdropping

Listening to communication traffic for the purpose of duplicating it. Eavesdropping is often facilitated using a network traffic capture or monitoring program or a protocol analyzer system (often called a

sniffer). Eavesdropping devices and software are usually ***difficult to detect*** because they are used in ***passive attacks***.

8.7.4. Impersonation / Masquerading

The act of ***pretending to be someone*** or something you are not to ***gain unauthorized access*** to a system. This usually implies that authentication credentials have been stolen or falsified in order to satisfy the authentication mechanisms.

8.7.5. Spoofing

Putting forth a ***false identity but without any proof*** (unlike masquerading where you supply proof). Includes using a false IP address, MAC addresses, email address, system name, domain name etc).

8.7.6. Replay Attacks

Attempt to ***reestablish a communication session*** by ***replaying captured traffic*** against a system. Are made possible through capturing network traffic via eavesdropping.

8.7.7. Modification Attacks

Captured packets are altered and then played against a system. Modified packets are designed to bypass the restrictions of improved authentication mechanisms and session sequencing.

8.7.8. Session Hijacking

Session IDs are used to establish a «stateful» connection between web client and web server. Stored in cookies, passed in URL or stored in «hidden fields». Sometimes also used for authentication and authorization to avoid re-entering the password. If compromised, attacker has access to the session.

- ***With Session Predicting:*** The attacker watches session IDs for pattern and tries to predict. Possibility of a session brute force attack.
- ***With Session Fixation:*** The attacker uses an established connection and tries to get the victim to use the connection. Once the client has established the session, the session can be hijacked by the attacker.
- ***With Cross-Site Scripting:*** The attacker tells the client with JavaScript with which session he should log in to the server. Once the client has established the session, the session can be hijacked.
- ***With Cross-Site Request Forgery:*** The attacker provides a fake website through which he can intercept the session and exploit it at the target server.
- ***With TCP/IP Hijacking:*** The hijack takes place after step 2 of a 3-way handshake. The attacker responds quicker to the server than the client.

9. COMPLETE CRYPTOGRAPHIC SYSTEMS

9.1. PUBLIC CRYPTOGRAPHY

- ***Digital Signatures:*** Hashes are used with RSA and DAS to create digital signatures. They prove the authenticity of the sender. The server only proves that it has the private key, it doesn't prove that the server can be trusted yet.
- ***Digital Certificates:*** Are a mechanism through which we can verify the ownership of a public key. This verification uses a trusted third party. Usually managed through a Public Key Infrastructure.

9.1.1. Certificate Issuance

The server server.com generates an RSA (or DSA) private and public key. The ***server has a public key*** that it wants the ***clients to trust***. It creates a ***Certificate Signing Request*** (CSR). A CSR is a certificate that has yet

to be verified. This CSR is ***sent off to a Certification Authority*** (CA).
Example: *Geotrust, Globalsign, Digicert, Lets encrypt.*

The certificate Authority (CA) does some ***identification checks***, not necessarily in person, They probably will charge you except for «letsencrypt» that is a free CA. The CA ***creates and signs the certificate*** with its private key and sends it back to the server. The server has now a Certificate signed by a CA that it can use to prove its identity.

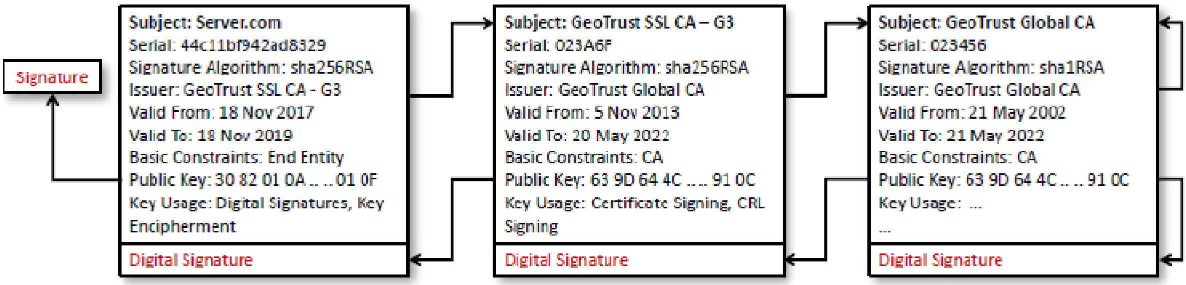
9.1.2. Certificate Use

During a ***TLS handshake***, the server supplies to the client a signature backed up by a ***certificate***. The client has to use the public key in the certificate server.com to ***decrypt the signature***. In order to ***trust*** the public key in the server.com certificate, the server has to ***check*** that the digital signature at the bottom of the certificate is ***valid***. The only way to ***check the validity*** of the signature at the bottom of the certificate is to use the public key of the issuer of the certificate to decrypt the digital signature. That is called the ***chain of trust***.

9.1.3. Chains of Trust

To verify the trust in the server.com certificate, we need to examine the digital certificate of the issuer of the certificate that signed it. In many cases, the ***chain involves multiple certificates***. Chains always ***end*** in a ***root certificate that is located on the client's machine***. A root certificate is ***self-signed*** and is trusted because it is built into the operating system, the browser or the phone. If the root certificate is not on the machine, the connection is rejected. This verification of the chain of trust happens with TLS every time that a client connects to a website.

The two assumptions that have to be true for the public infrastructure to be secure: The ***private key*** of a server is ***kept secret***, and the ***trust store or the root certificate store on the client hasn't been manipulated***. How can a client trust the issuer of a Certificate? By auditing the best practices of the issuer.



Subject: Server.com
Serial: 44c11bf942ad8329
Signature Algorithm: sha256RSA
Issuer: GeoTrust SSL CA - G3
Valid From: 18 Nov 2017
Valid To: 18 Nov 2019
Basic Constraints: End Entity
Public Key: 30 82 01 0A 01 0F
Key Usage: Digital Signatures, Key Encipherment
Digital Signature

Subject: GeoTrust SSL CA – G3
Serial: 023A6F
Signature Algorithm: sha256RSA
Issuer: GeoTrust Global CA
Valid From: 5 Nov 2013
Valid To: 20 May 2022
Basic Constraints: CA
Public Key: 63 9D 64 4C 91 0C
Key Usage: Certificate Signing, CRL Signing
Digital Signature

Subject: GeoTrust Global CA
Serial: 023456
Signature Algorithm: sha1RSA
Issuer: GeoTrust Global CA
Valid From: 21 May 2002
Valid To: 21 May 2022
Basic Constraints: CA
Public Key: 63 9D 64 4C 91 0C
Key Usage: ...
...
Digital Signature

9.2. COMPLETE CRYPTOGRAPHIC SYSTEMS

The building blocks of cryptography need to be carefully assembled into protocols that keep our systems secure. There are numerous well-established protocols out there.

9.2.1. Authenticated Encryption (AE)

Authenticated Encryption (AE) is an encryption scheme which simultaneously assures the data ***confidentiality*** and ***authenticity***.

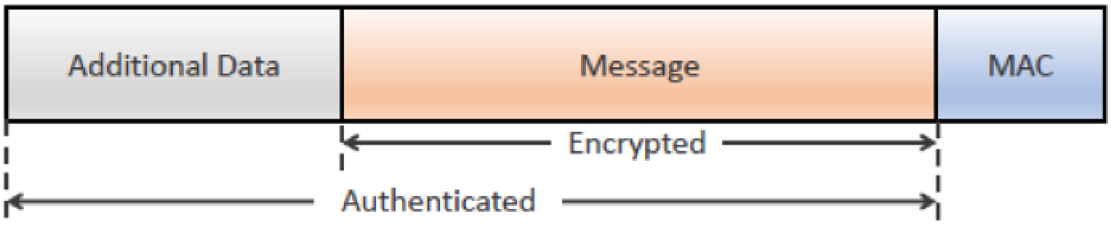
- ***Encrypt-then-MAC (EtM):*** The plaintext is first encrypted, then a MAC is produced based on the resulting ciphertext. The ciphertext and its MAC are sent together.
- ***Encrypt-and-MAC (E&M):*** A MAC is produced based on the plaintext, and the plaintext is encrypted without the MAC. The plaintext's MAC and the ciphertext are sent together.

- ***MAC-then-Encrypt (MtE):*** A MAC is produced based on the plaintext, then the plaintext and MAC are together encrypted to produce a ciphertext based on both. The ciphertext (containing an encrypted MAC) is sent.

9.2.2. Authenticated Encryption with Associated Data (AEAD)

Variant of AE. Allows recipient to ***check integrity of both encrypted and unencrypted information***. Binds associated data (AD) to the ciphertext and to context. A modern cryptography algorithm must guarantee the confidentiality as well as the integrity. A message authentication code (MAC) is often attached to the end of the ciphertext.

It is required, for example, by network packets or frames where the header needs visibility, the payload needs confidentiality, and both need integrity and authenticity.



9.2.2.1. Example: AES Galois Counter Mode

AES is a Block cipher in CTR mode and GCM is the message authentication code (MAC). GCM computes a Galois Message Authentication Code (GMAC) over the ciphertext and the additional data. Provides confidentiality and integrity. Used by IPSec tunnels over the Internet.

9.2.2.2. Example: ChaCha20_Poly1305

Chacha20 is a stream cipher and Poly1305 is a Message Authentication Code (MAC). Is used on mobile phones, especially between Android phones and connection to Android servers for speed purposes. As a stream cipher, it needs a nonce. The nonce is 12 bytes = 96 bits. ChaCha20 is one of the only two cipher that is allowed in TLS 1.3 alongside AES.

9.2.3. Protocol handshakes

- ***The handshake Phase:*** Agree on a set of cryptographic protocols, Perform a key exchange, verify any authenticity using the public key
- ***Transport/Record Phase:*** Packets encrypted using the agreed cipher, Includes a MAC or similar to verify integrity.

9.2.4. Common Protocol Issues

- ***Ciphertexts that aren't secured with a MAC:*** vulnerable in transit, less vulnerable on disk
- ***Messages that don't include a time-stamp or counter:*** Vulnerable to a replay attack
- ***Protocols that don't use public key for authenticity:*** vulnerable to man in the middle attacks
- ***Reuse of Nonces or Ivs***

9.2.5. Do's and Don'ts

DO's	***Don'ts***
- Use cryptographically strong random numbers. - Use «recipe» layers where possible. - Use ephemeral session keys for communication, and long-term public keys for authentication - Nonces are numbers only used once - Ivs must be unpredictable	- Implementing your own algorithms is a bad idea - Never use hard-coded keys - Don't use ECB mode - Don't use small public key sizes

9.2.6. Tips

Network Cryptography Tips:

- For general end-to-end communication, consider using TLS
- TLS supports client and server authentication using two certificates – useful for many setups.
- Restrict what cipher suites are allowed to avoid protocol downgrades
- If you don't use TLS, always use AEAD or another authenticated encryption mechanism.

Storage and DP Tips:

- Use password derivation functions not hashing functions for password storage. E.g. PBKDF2
- Encrypt data where possible using keys that you change fairly often.
- Encrypt the keys, adding a layer of indirection for an attacker.

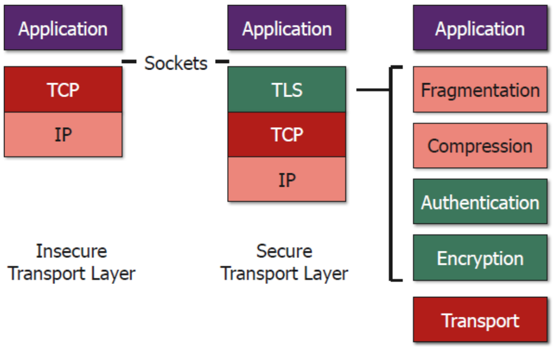
10. TRANSPORT LAYER SECURITY

10.1. INTRODUCTION

SSL (Secure Socket Layer) and TLS (Transport Layer Security) are ***network security protocols*** for ***setting up authenticated and encrypted connections*** and exchanging data.

SSL was the original name, which became after version 3.0 TLS, which currently exists as version 1.3. People still use SSL as a term, even though technically it is now TLS.

TLS is the primary mechanism for encryption for HTTP communication, that is called HTTPS.



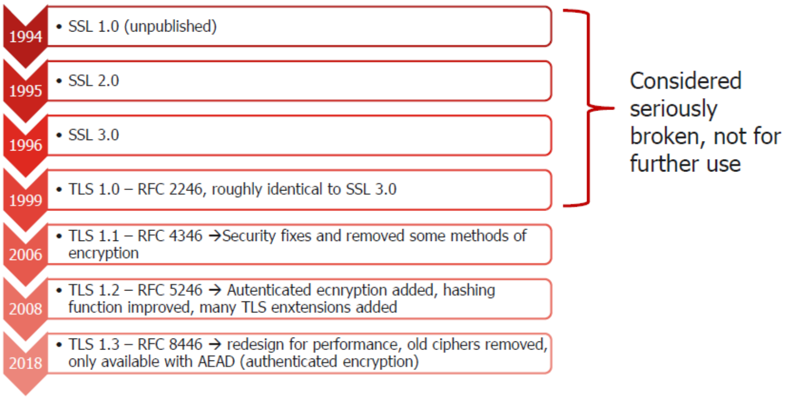
TLS is perfectly usable to ***encrypt communication over the internet*** with any other protocol than HTTP. TLS could be provided as part of the underlying protocol suite and therefore be transparent to applications.

Alternatively, TLS can be ***embedded in specific packages***.

10.2. TLS ARCHITECTURE

- ***Connection:*** It is a transport that provides a suitable type of service. Such connections are peer-to-peer relationships. Connections are transient, every connection is associated with one session.
- ***Session:*** Is an association between a client and a server created by the Handshake Protocol. Defines a set of cryptographic security parameters which can be shared among multiple connections. Sessions are used to avoid the expensive negotiation of new security parameters for each connection.

10.3. TLS HISTORY



1994 • SSL 1.0 (unpublished)

1995 • SSL 2.0

1996 • SSL 3.0

1999 • TLS 1.0 – RFC 2246, roughly identical to SSL 3.0

Considered seriously broken, not for further use

2006 • TLS 1.1 – RFC 4346 →Security fixes and removed some methods of encryption

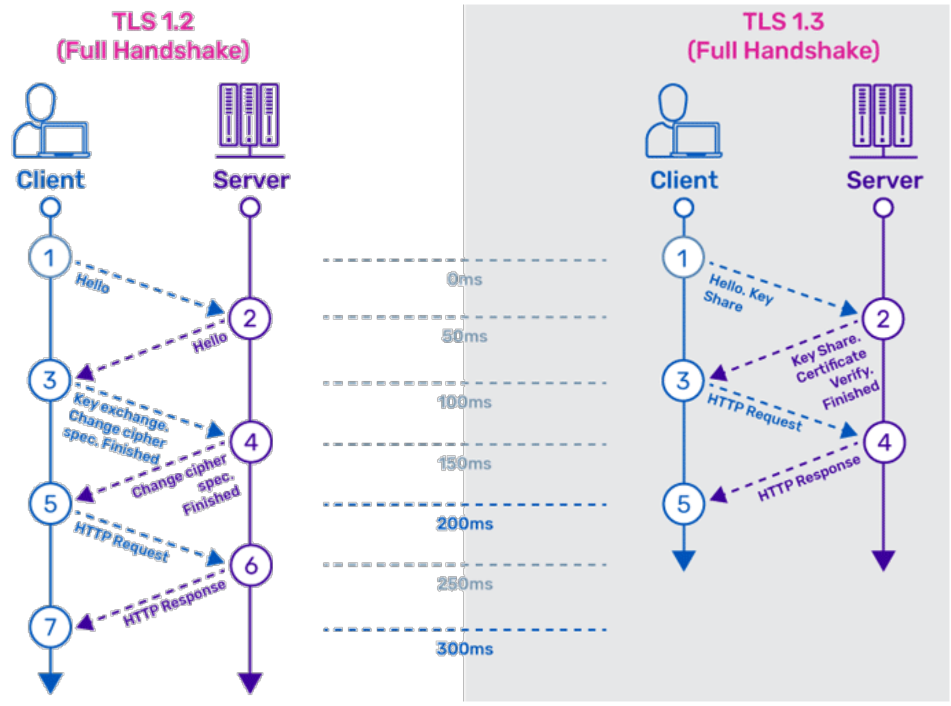
2008 • TLS 1.2 – RFC 5246 → Autenticated ecnryption added, hashing function improved, many TLS enxtensions added

2018 • TLS 1.3 – RFC 8446 → redesign for performance, old ciphers removed, only available with AEAD (authenticated encryption)

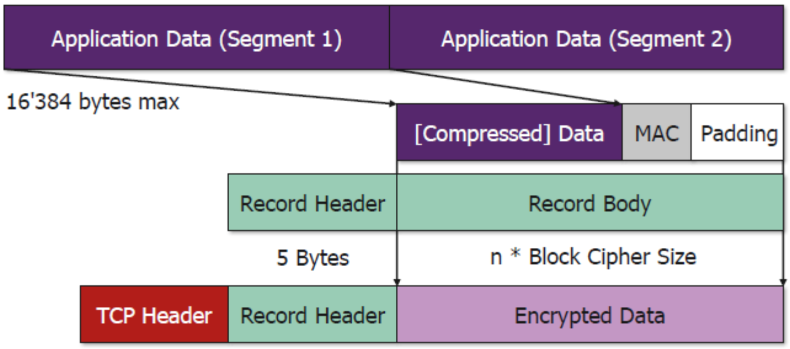
10.3.1. TLS 1.3: Key Improvements

- ***Clean up:*** remove unsafe or unused features
- ***Security:*** Improve security with modern techniques
- ***Privacy:*** Encrypt more of the protocol
- ***Continuity:*** Backwards compatibility

The Handshake got shorter from TLS 1.2 to TLS 1.3: The Hello and the key exchange got put together.



10.4. TLS RECORD STRUCTURE



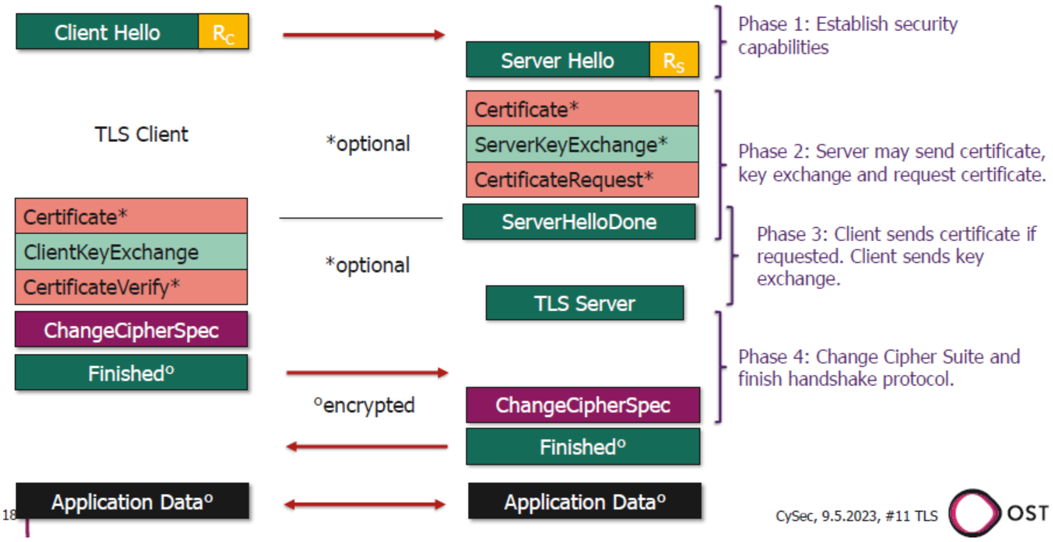
Content Types for HTTPS

- ***Handshake Protocol (22):*** ClientHello, ServerHello, Certificate, ServerHelloDone
- ***Change Cipher Spec Protocol (20)***
- ***Alert Protocol (21):*** Warning, Fatal (Session is closed immediately)
- ***Application Protocol (23):*** Transmission of (encrypted) payload

10.5. TLS HANDSHAKE

Is used before any application data are transmitted. It allows ***server*** and ***client*** to ***authenticate*** each other, ***negotiate encryption*** and ***MAC algorithms*** and ***negotiate cryptographic keys*** to be used.

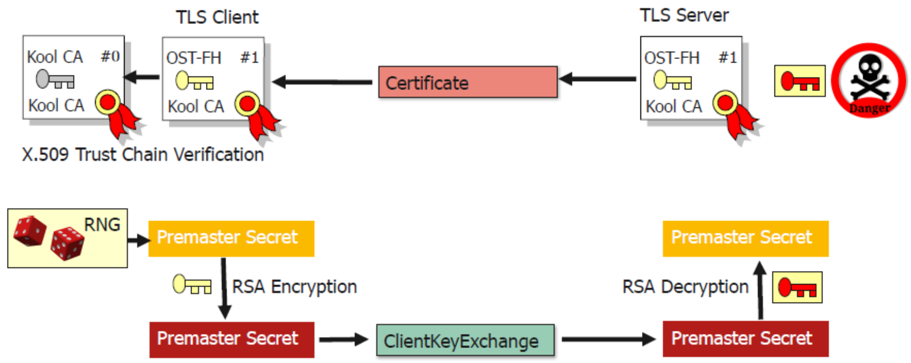
It compromises a ***series of messages exchanged*** by client and server. Its exchange has ***four phases***.



The keys for encryption can be shared in two ways: RSA or Diffie Hellmann Key Exchange.

10.6. RSA PUBLIC KEY ENCRPYTION

No perfect forward secrecy

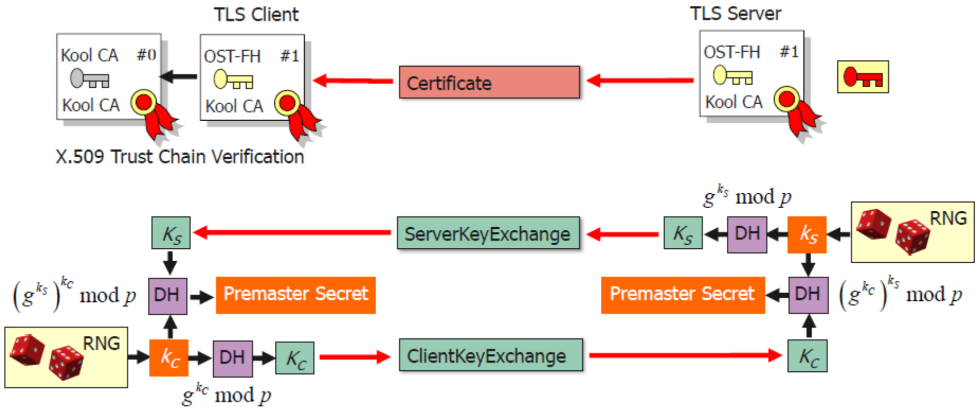


Client rolls a premaster secret. This is encrypted with the server's public key and sent to the server. The server can then decrypt the premaster secret with its private key and thus has the premaster secret. This is not optimal because the Premaster Secret is transmitted. Better would be:

10.7. EPHEMERAL DIFFIE-HELLMAN KEY EXCHANGE

With perfect forward secrecy

New key for every session. The Premaster Secret is not transmitted. Better than RSA for this reason.



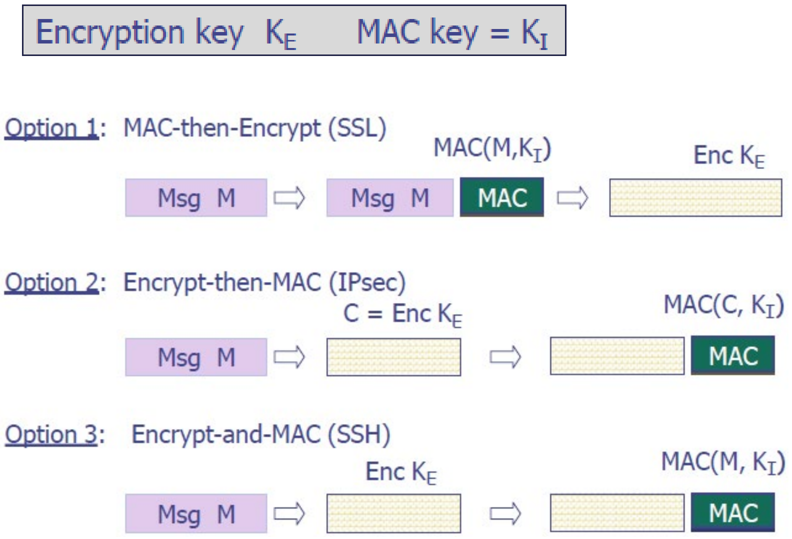
10.8. FORWARD SECRECY (FS)

In cryptography, forward secrecy (FS), also known as perfect forward secrecy (PFS), is a feature of specific key agreement protocols that ***gives assurances that session keys will not be compromised even if long-***

term secrets used in the session key exchange are compromised. For HTTPS, the long-term secret is typically the ***Private signing key of the server***. Forward secrecy protects past sessions against future compromises of keys or passwords. By generating a ***unique session key*** for every session a user initiates, the compromise of a single session key will not affect any data other than that exchanged in the specific session protected by that particular key. This by itself is not sufficient for forward secrecy which additionally requires that a long-term secret compromise does not affect the security of past session keys.

10.9. AUTHENTICATED ENCRYPTION OPTIONS

The MAC (Message Authentication Code) is a hashed value that ensures integrity during transport. There are three ways to transmit it with the data:



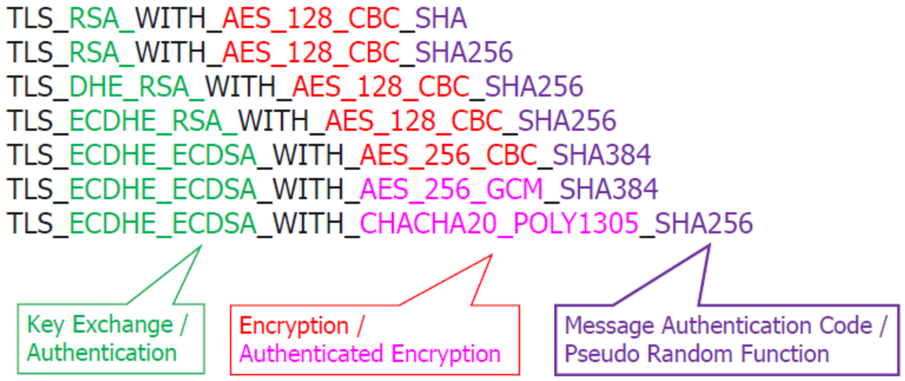
Encryption key K_E MAC key = K_I

Option 1: MAC-then-Encrypt (SSL)

Option 2: Encrypt-then-MAC (IPsec)

Option 3: Encrypt-and-MAC (SSH)

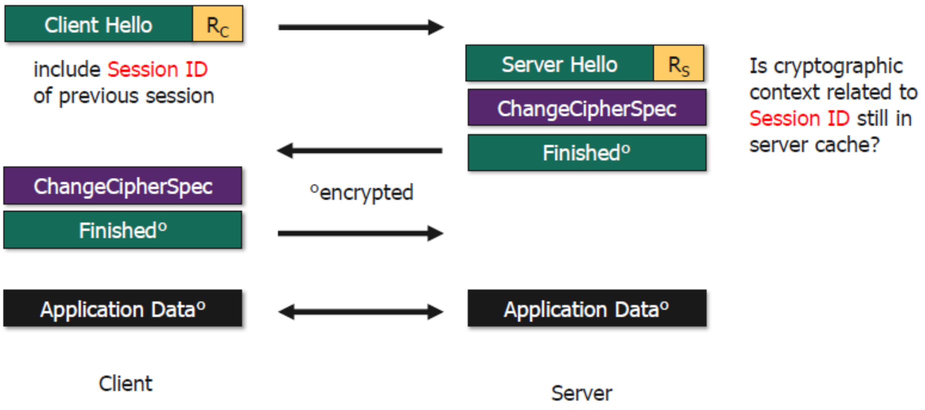
10.10. TLS CIPHER SUITES



TLS_RSA_WITH_AES_128_CBC_SHA
TLS_RSA_WITH_AES_128_CBC_SHA256
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256

Key Exchange / Authentication
Encryption / Authenticated Encryption
Message Authentication Code / Pseudo Random Function

10.11. TLS SESSION RESUMPTION



10.12. HEARTBEAT PROTOCOL

A ***periodic signal*** generated by hardware or software to ***indicate normal operation*** or to ***synchronize*** other parts of a system. Typically used to ***monitor the availability*** of a protocol entity. Runs on top of the TLS Record Protocol, use is ***established during Phase 1*** of the Handshake Protocol. Each peer indicates whether it supports heartbeats.

This serves ***two purposes***:

- Assures the sender that the ***recipient is still alive***
- ***Generates activity*** across the connection during idle periods

10.13. TLS ATTACKS

There are four categories of attacks in TLS:

- Attacks on the Handshake Protocol
- Attacks on the record and application data protocols
- Attacks on the PKI
- Other attacks

11. PUBLIC KEY INFRASTRUCTURE (PKI) AND ADVANCED TLS

11.1. PUBLIC KEY INFRASTRUCTURE (PKI)

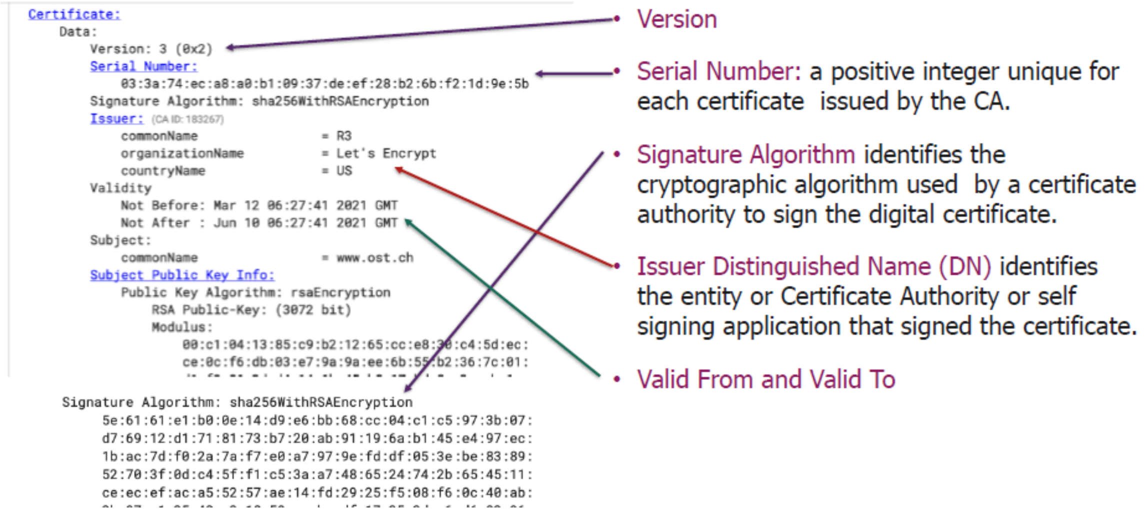
A set of roles, policies, hardware, software and procedures needed to create, manage, distribute, use, store and revoke ***digital certificates and manage public-key encryption***.

A PKI is used to ***bind a public key to an identity*** of a person or an organization. The binding is performed with a registration process by a ***Registration authority (RA)*** and an issuance of a certificate by a ***Certificate Authority (CA)***. The CA itself can be validated to be able to perform this service by an independent ***Validation Authority (VA)***.

11.2. X.509 CERTIFICATES

- ***X.500*** is a set of networking standards covering electronic directory services, developed by the Telecommunication Standardization Sector of the International Telecommunications Union (ITU-T), approved 1988.
- ***X.509*** is the standard for the format of public key certificates, first version published in July 1988.
- ***X.509v3*** (Version 3) profiles are the most used one. They are also the one of relevance for SSL/TLS and have been standardized by the IETF on RFC 3280 which was obsoleted in May 2008 by RFC 5280.

A X.509 certificate has an unique serial number, an issuer, validity, subject and a signature algorithm.



```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            03:3a:74:ec:a8:a0:b1:09:37:de:ef:28:b2:6b:f2:1d:9e:5b
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: (CA ID: 183267)
            commonName                = R3
            organizationName          = Let's Encrypt
            countryName               = US
        Validity
            Not Before: Mar 12 06:27:41 2021 GMT
            Not After : Jun 10 06:27:41 2021 GMT
        Subject:
            commonName                = www.ost.ch
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (3072 bit)
                Modulus:
                    00:c1:04:13:85:c9:b2:12:65:cc:e8:30:c4:5d:ec:
                    ce:0c:f6:db:03:e7:9a:9a:ee:6b:55:b2:36:7c:01:

    Signature Algorithm: sha256WithRSAEncryption
         5e:61:61:e1:b0:0e:14:d9:e6:bb:68:cc:04:c1:c5:97:3b:07:
         d7:69:12:d1:71:81:73:b7:20:ab:91:19:6a:b1:45:e4:97:ec:
         1b:ac:7d:f0:2a:7a:f7:e0:a7:97:9e:fd:df:05:3e:be:83:89:
         52:70:3f:0d:c4:5f:f1:c5:3a:a7:48:65:24:74:2b:65:45:11:
         ce:ec:ef:ac:a5:52:57:ae:14:fd:29:25:f5:08:f6:0c:40:ab:
```

- Version
- Serial Number: a positive integer unique for each certificate issued by the CA.
- Signature Algorithm identifies the cryptographic algorithm used by a certificate authority to sign the digital certificate.
- Issuer Distinguished Name (DN) identifies the entity or Certificate Authority or self signing application that signed the certificate.
- Valid From and Valid To

11.2.1. Public Key Certificate

Just a set of data

It is an ***electronic document proving the ownership of a public key***. It includes ***information about the key***, ***information about the identity of its owner*** (called the subject) and ***information of an entity that has verified the certificates contents*** (called the issuer).

11.2.2. Quality of a certificate

There exist four categories depending on the type of checks done during registration and the use of the TLS X.509 certificate:

- ***Domain Validated (DV) Certificate:*** A Domain Validated certificate is issued after the proof that the owner has the right to use their domain is established.
- ***Organization Validated (OV) Certificate:*** For OV certificates, CSs must validate the company name, domain name and other information using public databases.
- ***Extended Validation (EV) Certificate:*** EV certificates are only issues once an entity passes a strict authentication procedure.
- ***Qualified Website Authentication Certificate (QWAC):*** A qualified certificate under the trust services defined in the eIDAS Regulation (main current purpose is conformance to PSD2 regulation).

Depending on the «quality» the entries in the Distinguished Name (DN) vary to indicate what has been checked or validated.

Where to get validity information of the certificate

- ***Certificate Revocation List (CRL):*** CRL Distribution Point included in the certificate provides the location of this CRL. Most web browsers perform this check.
- ***Online Certificate Status Protocol (OCSP):*** The Authorative Information Access (AIA) field in the certificate provides information about how to access information about a CA, such as OCSP validation and CA policy data.

- ***Certificate Policies:*** This gives the link to the governance rule of the CA that are valid for a particular certificates usage. Each CA has their own set of policies.
- ***Authority Key Identifier (AKI):*** The key identifier of the Issuing CA certificate that signed the TLS certificate.
- ***Subject Alternative Name (SAN):*** Various values associated with the certificate owner.
- ***Subject Key Identifier (SKI):*** Hash value of the certificate, used to identify a certificate.

11.3. TRUST SERVICE PROVIDERS (TSP) – CERTIFICATE AUTHORITIES

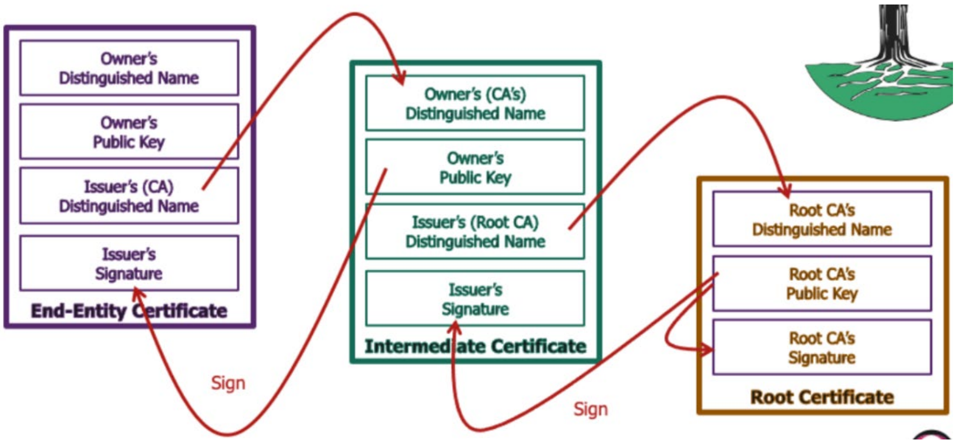
Trust Service Providers ***establish trust between communication parties***. In most cases, this is the Registration Authority (RA) and the Certificate Authority (CA).

- Providing trusted identity information
- Support secure authentication
- Support integrity-assured communication
- Support of encrypted communication

TSPs are verified by a forum of browser manufacturers. (It usually takes more than a year for a new root certificate to be issued).

11.3.1. CA hierarchy

- ***Root CA*** is the trust anchor of a PKI, installed in the certificate store of all devices
- ***Issuer CA*** ultimately used to issue X.509 certificates to users



11.3.2. Certificate Issuance

The subscriber prepares on the basis of the public key a ***certificate signing request*** (CSR). The CSR ***contains information*** on the subscribers ***object*** to be signed (Common name, etc), the ***public key*** to be signed and information on ***key type and length***.

The CSR is going to be sent to a CA in a Base64-PEM format. The CA will register the request, ***verify the data*** and in case of positive assessment ***sign the CSR*** and ***turn it into a X.509 certificate***.

11.3.3. Certificate Pinning (HPKP)

Certificate pinning ***explicitly trusts only one certificate***. ***All other root anchors are ignored***. There are 3 different pinning models: root pinning, intermediate CA pinning and end entity pinning. Pinning poses a ***big risk if the HPKP is hacked*** and because certificates are no longer fully validated. ***Legacy since 2017***.

12. E-MAIL SECURITY

12.1. SPF (SENDER POLICY FRAMEWORK)

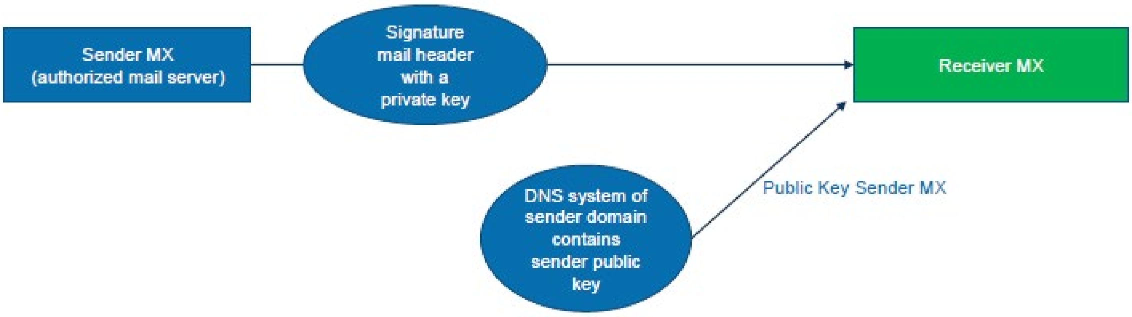
v=spf1 mx ip4:193.135.215.47/32 ip4:193.135.215.55/32 -all

Allows senders to define ***which IP addresses are allowed to send mail*** for a particular ***domain***.

12.2. DKIM (DOMAINKEYS IDENTIFIED MAIL)

Provides an encryption key and digital signature that ***verifies that an email message was not faked*** or altered.

Specifically, it uses an approach called «public key cryptography» to verify that an email message was sent from an authorized mail server, in order to detect forgery and to prevent delivery of harmful email like spam. It supplements SMTP, the basic protocol used to send email, because it does not itself include any authentication mechanisms.


Sender MX (authorized mail server)
Signature mail header with a private key
Receiver MX
Public Key Sender MX
DNS system of sender domain contains sender public key

It works by adding a ***digital signature to the headers*** of an email message. That ***signature can be validated against a public cryptographic key in the organization's DNS records***. In general terms, the process works like this:

- A domain owner ***publishes a cryptographic public key*** as a specially formatted TXT record in the domain's overall DNS records.
- When a mail message is ***sent*** by an outbound mail server, the server generates and attaches a unique DKIM signature header to the message. This header includes ***two cryptographic hashes***, one of specified ***headers***, and one of the message ***body*** (or part of it). The header contains information about how the signature was generated.
- When an inbound mail server ***receives*** an incoming email, it looks up the sender's public DKIM key in DNS. The inbound server uses ***this key to decrypt the signature*** and ***compare*** it against a freshly computed version. If the two values match, the message can be proved to be ***authentic*** and unaltered in transit.

12.3. DMARC (DOMAIN-BASED MESSAGE AUTHENTICATION, REPORTING AND CONFORMANCE)

Unifies the SPF and DKIM authentication mechanisms into a common framework and allows domain owners to declare how they would like email from that domain to be handled if it fails an authorization test. DMARC policies are published in the DNS as text (TXT) resource records (RR).

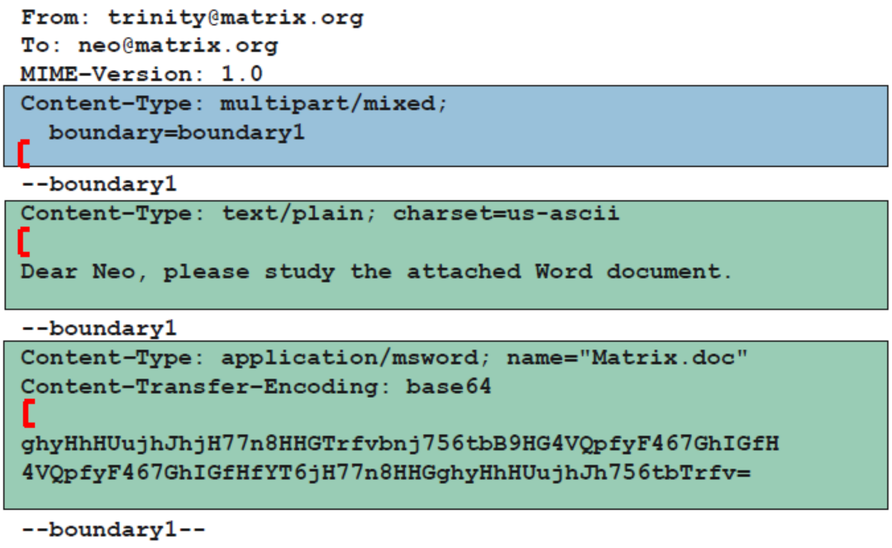
- ***Sender:*** Author Composes & Sends Email -> Sendin Mail Server inserts DKIM header -> Email sent to receiver
- ***Receiver:*** Standard validation Tests -> Retrieve verified DKIM domains -> Retrieve «Envelope from» via SPF -> Apply Appropriate DMARC Policy, if passed, do the standard processing (Anti-Spam Filters etc.)

12.4. S/MIME (SECURE / MULTIPURPOSE INTERNET MAIL EXTENSIONS)

12.4.1. Mime (Multipurpose Internet Mail Extension

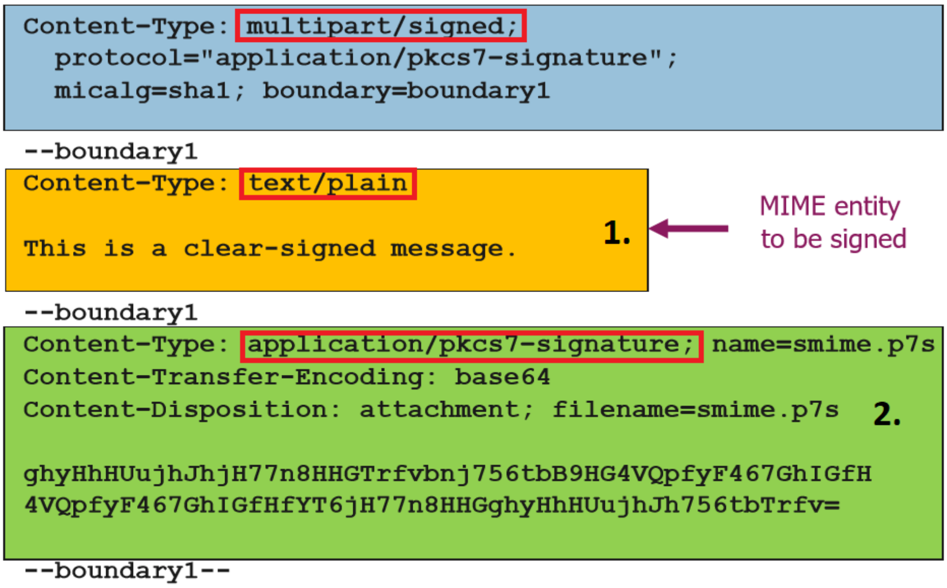
Expansion of a mail message with extensions for e.g. security. It is an Internet standard that extends the format of email messages to support text in character sets other than ASCII, as well as attachments of audio, video, images, and application programs.

S/MIME = Secure MIME



```
From: trinity@matrix.org
To: neo@matrix.org
MIME-Version: 1.0
Content-Type: multipart/mixed;
  boundary=boundary1

--boundary1
Content-Type: text/plain; charset=us-ascii

Dear Neo, please study the attached Word document.

--boundary1
Content-Type: application/msword; name="Matrix.doc"
Content-Transfer-Encoding: base64

ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfH
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbTrfv=

--boundary1--
```

12.4.2. S/MIME signed message format 1



```
Content-Type: multipart/signed;
  protocol="application/pkcs7-signature";
  micalg=sha1; boundary=boundary1

--boundary1
Content-Type: text/plain

This is a clear-signed message.

--boundary1
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s

ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfH
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbTrfv=

--boundary1--
```

1. The content to be signed
2. Digital signature

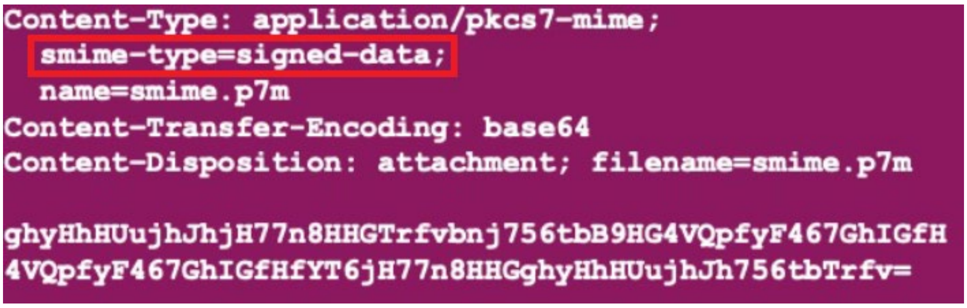
Instead of «text/plain», the signed data can also be multipart/mixed. This subdivision is then marked with «boundary2» and again consists of several parts, e.g. text/plain or application/msword.

- *Advantage:* Even if a mail client does not support S/MIME and thus cannot check the signature, it can still read the content (i.e., the content).
- *Disadvantage:* If the order of the MIME parts changes on the way, the signature becomes invalid. MIME parts are still 7-bit, because not all mail hops support 8-bit (ä,ö,ü).

If several people sign a mail, then the signatures are simply appended one after the other. The signatures are independent of each other.

12.4.3. S/MIME signed message format 2

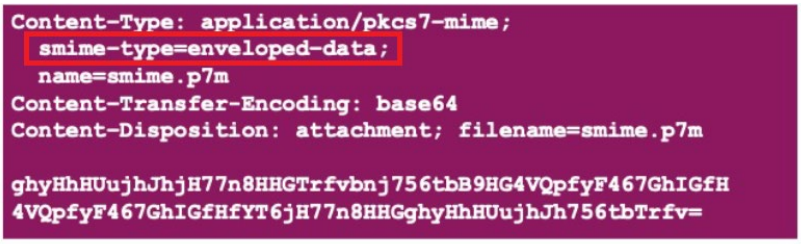
The MIME content is transmitted in the PKCS#7 signed object. This alternative signing format is optionally used by MS Outlook.



```
Content-Type: application/pkcs7-mime;
  smime-type=signed-data;
  name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m

ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfH
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbTrfv=
```

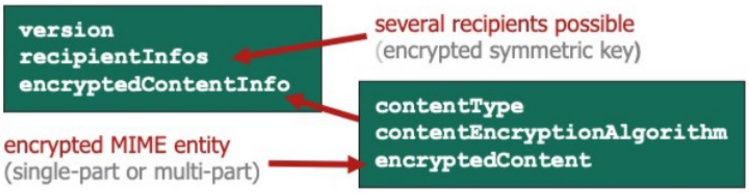
- *Advantage:* MIME content is not prone to changes of the transfer encoding enforced by intermediate mail transfer agents.
- *Disadvantage:* In order to read the embedded MIME message, the receivers mail client must support S/MIME

12.4.4. S/MIME Encryption



```
Content-Type: application/pkcs7-mime;
  smime-type=enveloped-data;
  name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m

ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfH
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbTrfv=
```

ASN.1 structure for the **EnvelopedData** content type



```
version
recipientInfos
encryptedContentInfo
```

several recipients possible
(encrypted symmetric key)

```
contentType
contentEncryptionAlgorithm
encryptedContent
```

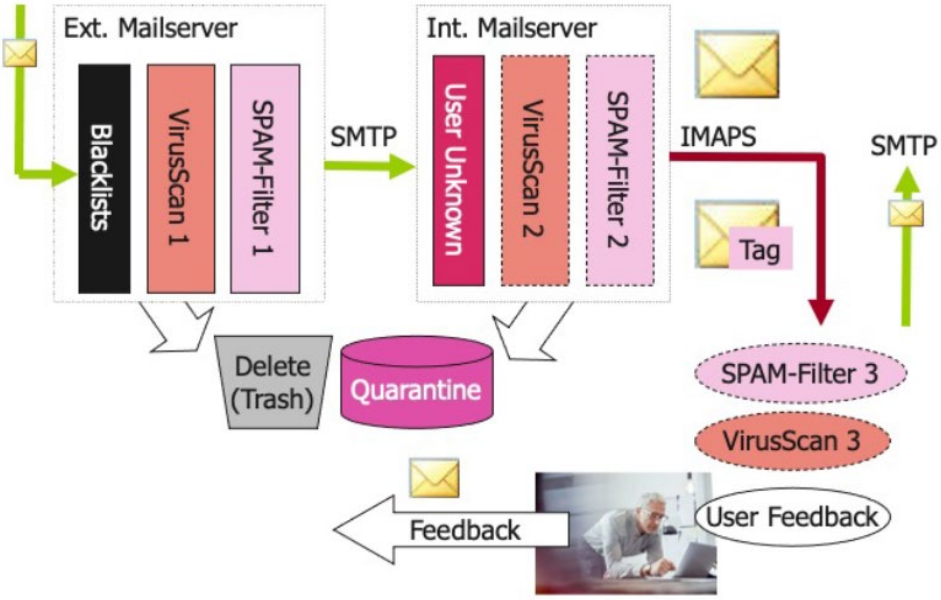
encrypted MIME entity
(single-part or multi-part)

- ***Sign, then encrypt (more commonly used): Signature*** remains private, anonymous.
- ***Encrypt, then sign:*** Trust can be established before decrypting.

12.4.5. Storage of Signature and Encryption Keys

- ***Signature Key:*** Equivalent to a person's digital identity. Thus there must not be more than a single copy of the private signature key. Best stored on a tamper-proof smart card. Mandated by digital signature legislation.
- ***Encryption Key:*** Backup copy protects against information loss. If the private encryption key gets lost or damaged, all encrypted communication received in the past cannot be accessed and read any more. In a corporate environment, a deputy should be able to access the encrypted email of an employee who is absent.

12.5. SPAM



3x spam filter for better protection.

Spam-Assassin: Is a framework that uses various spam detection techniques.

13. KERBEROS (KEY DISTRIBUTION SYSTEM)

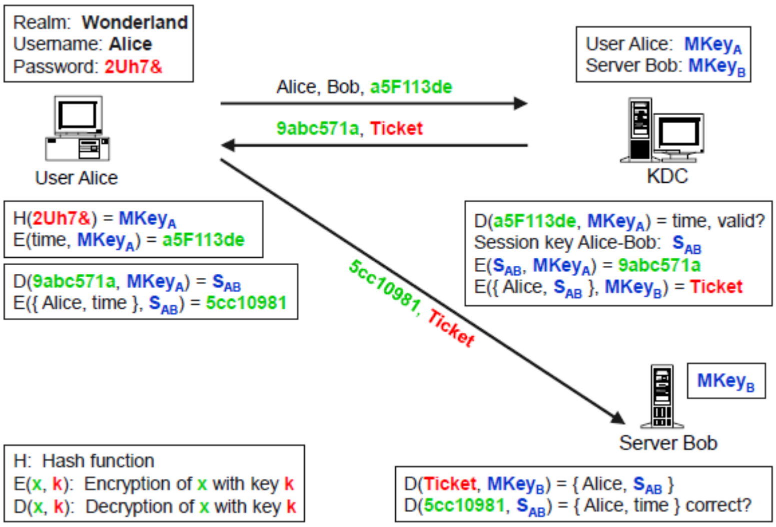
Is a Key Distribution Center. It is a network authentication protocol that works on the basis of tickets (security tokens) to allow nodes communicating over a non-secure network to prove their identity to one another in a secure manner. Each Kerberos participant – called a principal – shares a common secret with the Key distribution Center (KDC) – the principals master key.
Developed in 1983. Relies on the mediation services of a trusted referee or notary.

13.1. MEDIATED AUTHENTICATION BY MEANS OF A KEY DISTRIBUTION CENTER (KDC)

Every principal has a secret master key registered with the KDC. User master keys are usually derived from a login password. Machine master keys must be configured on the host. All master keys of the principals are stored in the KDC database, encrypted by using the KDC master key.

Because all master keys reside in the KDC, the server which is hosting the KDC service must be physically secured by locking it away in a burglar-proof room.

13.1.1. Simplified Kerberos Protocol



Realm: Wonderland
Username: Alice
Password: 2Uh7&
User Alice: $MKey_A$
Server Bob: $MKey_B$
Alice, Bob, a5F113de
9abc571a, Ticket
User Alice
KDC
$H(2Uh7\&) = MKey_A$
$E(time, MKey_A) = a5F113de$
$D(9abc571a, MKey_A) = S_{AB}$
$E(\{ Alice, time \}, S_{AB}) = 5cc10981$
5cc10981, Ticket
$D(a5F113de, MKey_A) = time$, valid?
Session key Alice-Bob: S_{AB}
$E(S_{AB}, MKey_A) = 9abc571a$
$E(\{ Alice, S_{AB} \}, MKey_B) = Ticket$
$MKey_B$
Server Bob
H: Hash function
E(x, k): Encryption of x with key k
D(x, k): Decryption of x with key k
$D(Ticket, MKey_B) = \{ Alice, S_{AB} \}$
$D(5cc10981, S_{AB}) = \{ Alice, time \}$ correct?

Disadvantages: The principals master key must be used every time a server must be accessed and a ticket must be issued by the KDC. This means that either the user password must be typed in for every time a ticket request must be sent to the KDC or the principals master key must be kept in temporary storage which is a security risk***. Solution:***

13.1.2. Long-lived Ticket-Granting Ticket

The KDC issues for each user an individual session key and a ticket granting ticket (TGT) with a typical lifetime of 8-24 hours, so that the entry of the user password is required less often.

Advantages: The Kerberos Authentication Service requiring the principals master keys and the Ticket Granting Service based on the principals TGT can be run on separate servers thus improving the performance.

Security risk: The initial Kerperos authentication message can be sniffed and an offline dictionary attack can be run to crack weak user login passwords. If successful the attacker will be in the possession of the principals master key.

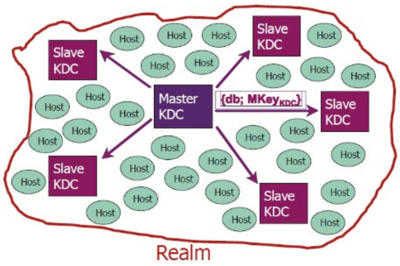
Improvements: Kerberos v5 supports the alternative PKINIT authentication method based on RSA public key pairs that can be securely stored on a smartcard or USB token.

13.2. SINGLE SIGN ON AUTHENTICATION WITH KERBEROS

- The client sends the principal name and a request for a TGT (Ticket granting ticket) to the KDC.
- The KDC generates a session key and a TGT and encrypts it.
- It sends the encrypted combination of the session key and the TGT to the client.
- The client extracts the session key and the encrypted TGT. When it wants to use a service, it encrypts a copy of the encrypted TGT with the IP address, time stamp and a service ticket request and sends it to the KDC.
- The KDC extracts everything and validates the client. It then generates a service session key and sends it encrypted to the client.
- The client extracts the encrypted service ticket and sends it to the service provider together with the service session key, the principal name and a timestamp.
- The service provider enables the service for the client.

13.2.1. KDC Replication

The entire Master KDC database is periodically downloaded to the Slave KDCs. The Slave KDC databases are read-only.



14. FEDERATION

Federated identity is a way to ***use an account from one website*** to create an account and ***log in to a different site***.

Protocols: SAML (Security Assertion Mark-up Language), OAuth 2.0, OIDC (OpenID Connect)

14.1. SAML

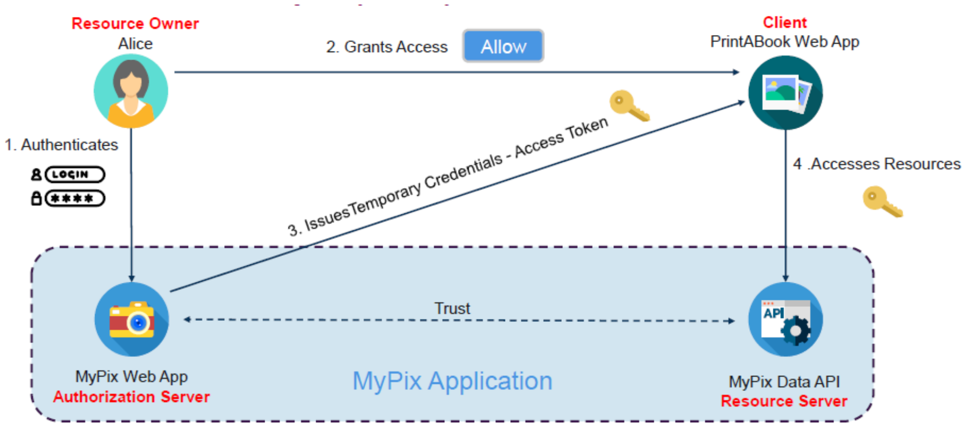
XML-Based standard for exchanging authentication and authorization data. Uses a XML document called Assertion that the Identity Provider sends to the service provider.

- The user goes to the Service Provider and clicks on SAML login
- The SP redirects the request to the Identity Provider
- The Identity Provider shows the login page to the user to enter credentials
- Upon entering the credentials, the SAML IdP verifies the credentials in its Active Directory or Database
- Once verified, a SAML response is sent with Assertions in XML format as seen above
- Then the user will log in to the application

14.2. OAUTH

Provides secure access delegation mechanism for websites. OAuth is not an authentication protocol. Defines a process for end-users to grant controlled access for third-party website to their private resources stored on a resource server.

OAuth 2.0 is not backwards compatible. Protocol messages are not protected by cryptographic means, TLS must be used for all messages.



14.2.1. Terminology

Name	*Description*
Resource Owner	The user that owns a particular resource
Client	Application accessing protected resources on behalf of the user
Resource Server (Protected Resource)	Server hosting protected resources of a user
Authorization Server	Server identifying the user and issuing access tokens

14.2.2. Client Types

OAuth defines two client types:

- ***Confidential Client:*** Applications capable of maintaining a secret (like Web-applications with a backend), can establish a back-end channel to the target APIs.
- ***Public Client:*** Clients incapable of maintaining a secret (like single page applications or mobile applications). Use front-end channel to fetch data from the target APIs.

14.2.3. Access Token

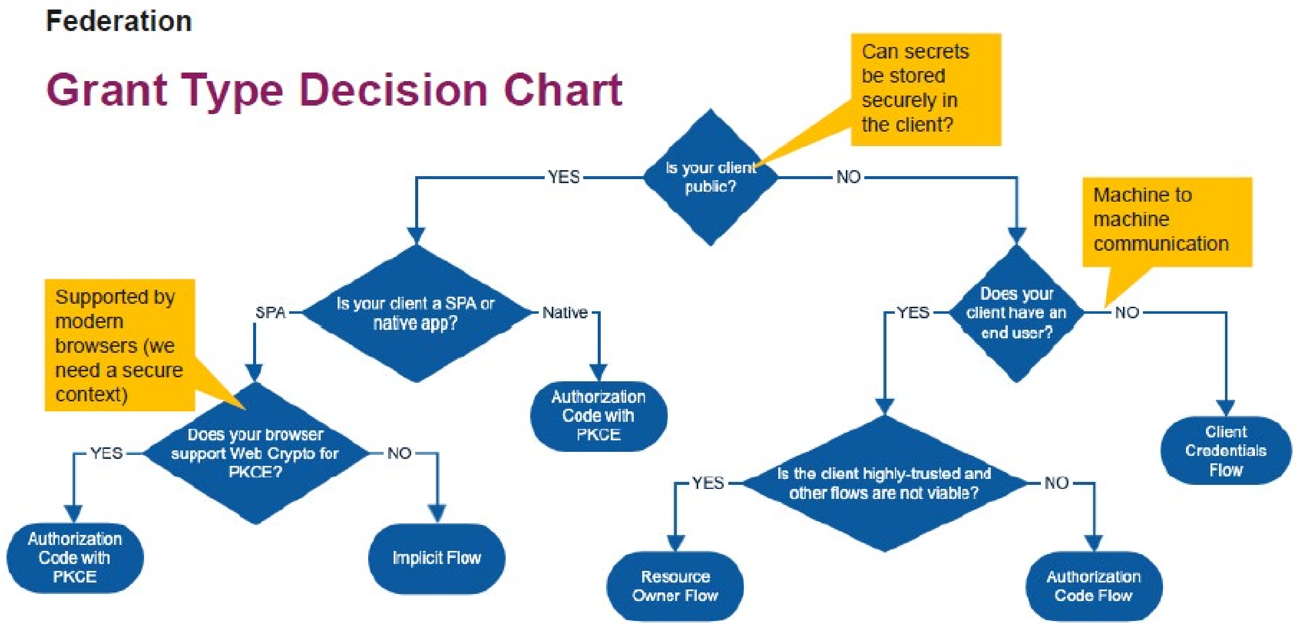
Credentials used to ***access protected resources***. Can be valid for a ***limited time***. Grant access to a specific protected resource defined by the scope parameter. Issued by the ***authorization server***. Access token format is opaque to the client, design is mostly open. ***Can be refreshed*** using a refresh token.

14.2.4. Grant Types (aka Flows)

The grant type defines the messages that are exchanged between the parties to issue an access token to the client. OAuth 2.0 defines several grant types:

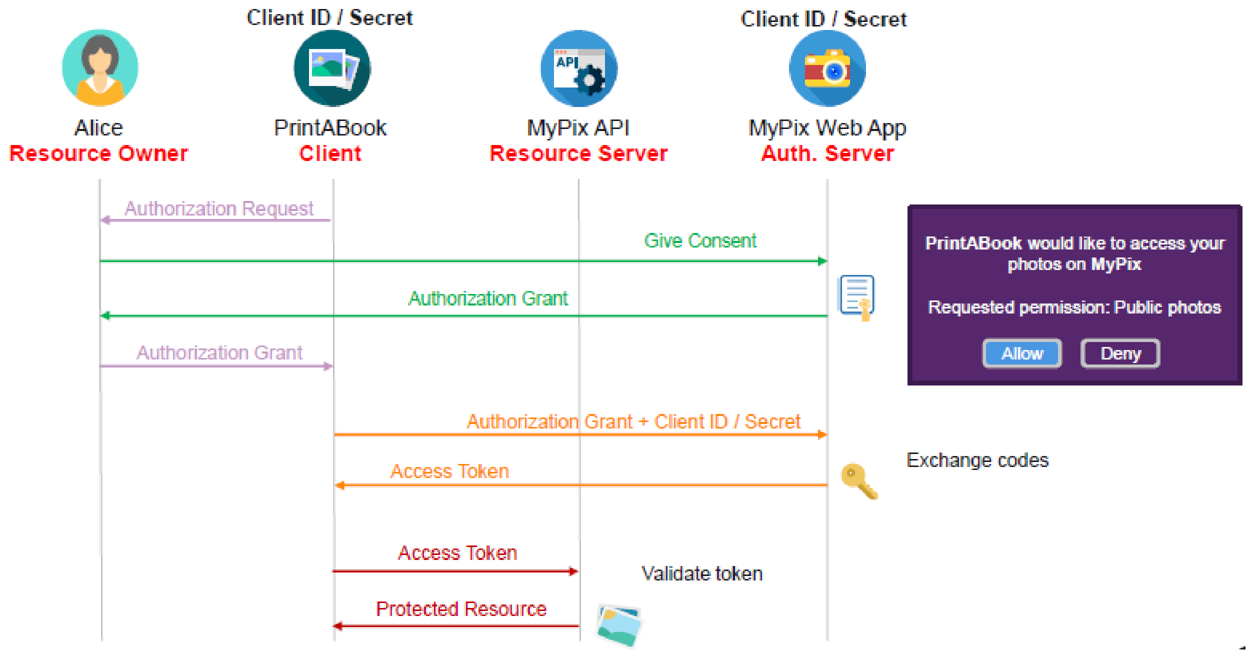
- Authorization Code *not recommended*
- Authorization Code + PKCE *recommended*
- Implicit Flow *deprecated in OAuth 2.1*
- Client Credentials Flow *used with Applications API access (machine-to-machine, no users involved)*
- Resource Owner Flow *deprecated in OAuth 2.1*

Federation


Grant Type Decision Chart
Can secrets be stored securely in the client?
Is your client public?
YES
NO
Is your client a SPA or native app?
SPA
Native
Supported by modern browsers (we need a secure context)
Does your browser support Web Crypto for PKCE?
YES
NO
Authorization Code with PKCE
Implicit Flow
Authorization Code with PKCE
Machine to machine communication
Does your client have an end user?
YES
NO
Client Credentials Flow
Is the client highly-trusted and other flows are not viable?
YES
NO
Resource Owner Flow
Authorization Code Flow

14.2.5. Authorization Code Flow

Recommended with PKCE



PrintABook would like to access your photos on MyPix

Requested permission: Public photos

Start OAuth Process

User accesses the application of the client, client provides an ***authorization code link*** pointing to the consent dialog:

`https://mypix.com/authorize?response_type=code&client_id=6779ef20e7581&redirect_uri=https://printabook.com/callback&scope=public&state=Ax3B0fqK`

- `https://mypix.com/authorize` The authorization API of the auth server
- `?response_type=code` Specifies that your application is requesting an authorization code grant
- `client_id=6779ef20e7581` The client ID
- `redirect_uri=https://printabook.com/callback` Where the service redirects the user-agent after an authorization code is granted

- `scope=public` Specifies the level of access/scope that the client is requesting
- `state=Ax3B0fqK` Randomly generated value to prevent/detect CSRF attacks

Issue Grant

When the user clicks the link, they must log in to the auth. server. Next, the auth. Server prompts the user, if the client is allowed to access the users resources. If the user authorizes the access, authorization server issues a redirect containing the authorization grant.

```
HTTP/1.1 302 Found
Location: https://printabook.com/callback?authorization_code=Ax3lfsx492Aldv
&state=Ax3B0fqK
```

- `https://printabook.com/callback` API of the client that accepts the auth code
- `authorization_code=Ax3lfsx492Aldv` Authorization code that can be used to request an access token
- `state=Ax3B0fqK` Reflected value received in the authorization request. Must match the previous value (CSRF prevention)

Redirect Validation

Authorization servers should always validate redirect URLs. Normally, redirection URLs are linked with a particular client during registration. Redirection should only occur if the particular URL is valid for the given client. If the redirect validation fails, the authorization server should not redirect the resource owner to the invalid URL and instead inform the resource owner of the failure.

State Validation

If the ***client issued*** a ***state parameter*** in the initial authorization request, ist must ***check ist presence*** in the ***response***, the value must be ***identical***.

```
https://mypix.com/authorize?response_type=code&client_id=6779ef20e7581&redirect_uri=htt
ps://printabook.com/callback&scope=public&state=Ax3B0fqK
```

```
https://printabook.com/callback?authorization_code=Ax3lfsx492Aldv &state=Ax3B0fqK
```

The state parameter «ties» an authorization request and ist response together. It allows the client to make sure it only continues authorization processes which it initiated itself.

Exchange Grant for Token

The client now possesses an authorization grant. This grant is used to request an access token from the auth. Server (backend to backend)

```
https://mypix.com/token?client_id=6779ef20e7581&client_secret=xB837sdfoBqq2842Bd&grant_
type=authorization_code&code=Ax3lkdfaB33jfsx492Aldv&redirect_uri=https://printabook.com
/callback
```

- `https://mypix.com/token` Access token API of the auth. Server
- `client_id=6779ef20e7581` The applications client id
- `client_secret=xB837sdfoBqq2842Bd` «password» of the application to authenticate itself to the auth. Server
- `grant_type=authorization_code` Authorization code flow that is used
- `code=Ax3lkdfaB33jfsx492Ald` Authorization grant issued to the client
- `redirect_uri=https://printabook.com/callback` Must be identical to the previous redirect URI

Grant Verification
The authorization server must verify the authorization code request.

- Are the client credentials correct?
- Was the code issued to the same client that is presenting it now?
- Has the code been used before? (Prevent Reply Attack)
- Has the code expired?

If all checks are passed, an access token is generated and returned to the client. Otherwise revoke all access tokens exchanged with this authorization code.

Issue Token
The access token is returned in a standardized format. The authorization server can specify additional info if neccessary.

```
{"access_token":"aSxl2bw958djkjcvqiowefasdf234dfDFWdf2", "token_type":"bearer",
"expires_in":2592000, "refresh_token":"2xk02dkjfoFDBjf29865uhSdhbodfasdfF",
"scope":"public","uid":923,"info":{"name":"Alice","email":"alice@gmail.com"}}
```

- The **`access_token`** eventually grants access to the users data
- The **`token_type`** specifies how the token is to be used
- The **`refresh_token`** (if issued) can be used to fetch a new access_token

Access Resources
The client now possesses an access token. This can be used to access resources on the resource server.

```
GET /pictures HTTP/1.1
Host: api.mypix.com
Accept: application/json, text/plain, */*
Authorization: Bearer aSxl2bw958djkjcvqiowefasdf234dfDFWdf2
```

- **`https://api.mypix.com/pictures`** Data API of the resource server
- **`Authorization:`** Previously issued access token

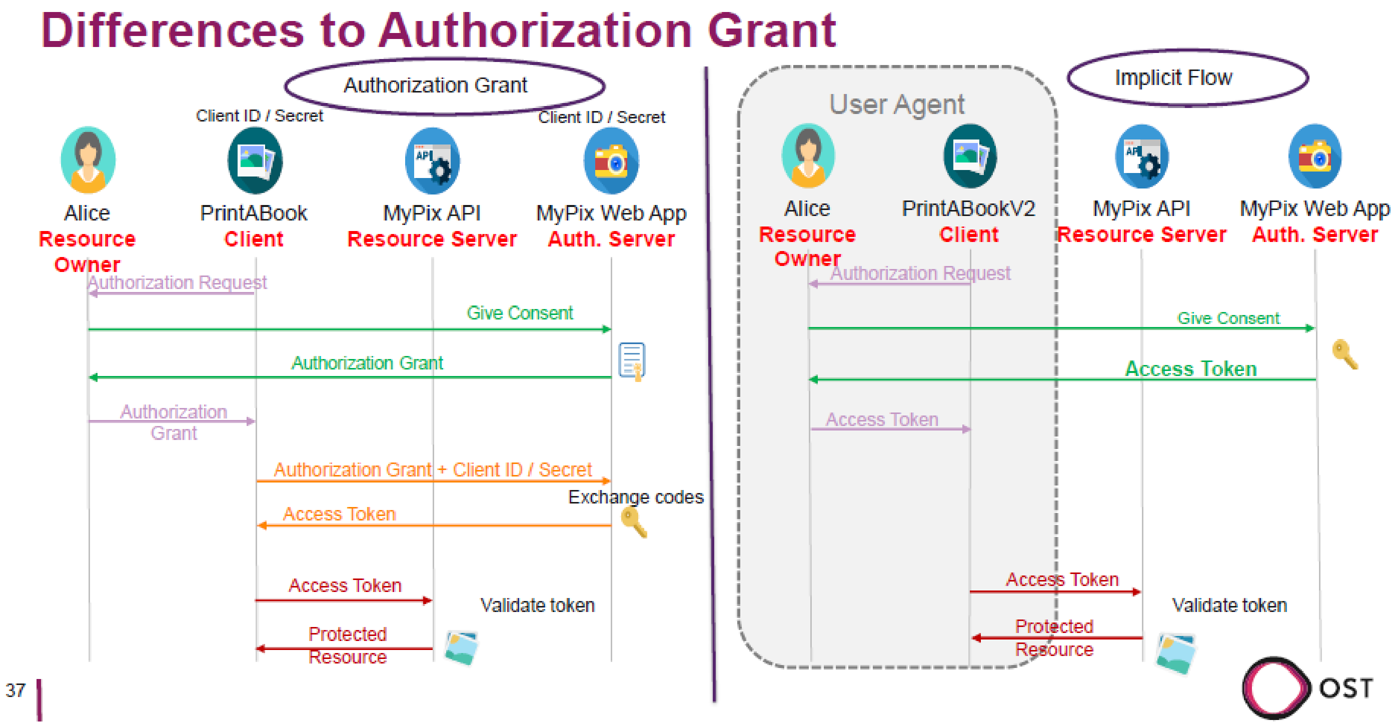
Verify Access Token
The resource server ***must perform validation on the received access token*** (Expiration / Validity, Scope). The access token is ***issued*** by ***another instance*** (auth.server), however depending on the scenario, this makes validation ***more or less complex***: Resource and authorization server have ***access to a common store***. Authorization server offers an API to verify access tokens (called Introspection). Access tokens are self-contained and can be verified by cryptographic means.

14.2.5.1. Refresh Tokens

Refresh tokens may or may not be issued by the auth server. They allow for relatively ***short-lived access tokens*** (valid for 5-10 minutes). If an ***access token expires***, the client can use ist ***refresh token*** to ***fetch a new access token***.

This mechanism provides ***improved security***. Access tokens, if compromised somehow, can be used to directly access resources. Therefore, limited validity reduces the time window for a successful attack. Refresh tokens must be used in combination with the clients credentials and can therefore not be abused as easily.

14.2.6. Implicit Flow



Differences to Authorization Grant

Authorization Grant

Client ID / Secret

Client ID / Secret

Alice
Resource Owner

PrintABook
Client

MyPix API
Resource Server

MyPix Web App
Auth. Server

Authorization Request

Give Consent

Authorization Grant

Authorization Grant

Authorization Grant + Client ID / Secret

Exchange codes

Access Token

Access Token

Validate token

Protected Resource

Implicit Flow

User Agent

Alice
Resource Owner

PrintABookV2
Client

MyPix API
Resource Server

MyPix Web App
Auth. Server

Authorization Request

Give Consent

Access Token

Access Token

Access Token

Validate token

Protected Resource

37

OST

14.3. AUTORIZATION CODE FLOW WITH PROOF KEY FOR CODE EXCHANGE (PKCE)

Originally designed for ***mobile / native clients*** as they ***cannot store a static client secret*** securely and usually use custom URL schemes to capture redirects, which may allow malicious app to steal codes/tokens.

Main Threat: Malicious app on the user device gets hold of the access token by stealing the authorization code and ***posing as the original app***. With PKCE, the client (mobile/native app) generates a ***dynamic secret***, which can later be ***verified*** by the auth. server. This guarantees that the auth. server is always talking to the actual client who initiated the OAuth flow.

It is recommended to use Authorization Code + PKCE for all interactive applications.