Cybersecurity Zussammenfassung

Security & Cryptography

↳ it's about communication in the presence of adversaries

Relay Attack

↳ intercept and relay communication between two parties to gain unauthorized access to a system or to information

ex relay car key signal from the house to the car to trick the car into opening, as it thinks the key is near by

Countermeasure

↳ Distance bounding protocols

↳ check timing between communication and calculate distance based on timings

How to Achieve Security

- Security Policy
  - ↳ confidentiality → who can access
  - ↳ Integrity → who can modify
  - ↳ Availability → ability to access data at any time

- Threat Model → Assumptions about the adversary
- Mechanism → Hardware & Software Implementations to achieve security/privacy.

⇒ policies have to be followed, up to date and you always need the right set of assumptions.
This is an iterative process and therefore prone to vulnerabilities

⇒ every system has a breaking point

⇒ the more secure a system is, the less likely it is to be attacked

Confidentiality

→ can be achieved using encryption

Encryption

↳ a secret key is used to encrypt a plaintext using some kind of an algorithm.
This can later be decrypted using another secret key.

Integrity

↳ Determin if contents have changed

Checksums

↳ use a function to map contents to a numerical value. This value can then be checked to see if the content has changed.

Types of Threats & Attacks

- Eavesdropping → listening in on communication
- Alteration / Tampering → change contents of communications
- Denial of Service → Block access to something
- Masquerading (Impersonating)
- Brute force → try everything until success

Protection Mechanisms

- Access Control → control who can do what
- Secret key cryptography → encrypt & decrypt using a key

Malware

→ can be classified into several categories

- Propagation
  - ↳ Virus → human-assisted propagation (ex: open email attachment)
  - ↳ Worm → automatic propagation without human assistance
- Concealment
  - ↳ Rootkit → modifies OS to hide its existence
  - ↳ Trojan → provides desirable functionality but hides malicious operation
- Various Types of Payloads
  - ↳ from annoyance to crime

Computer Virus

↳ software that can replicate itself by modifying other files/programs to insert code

⇒ needs some type of user assistence

Behaviours

- Slow down System
- Auto-run or close applications
- Display pop-ups
- Corrupt files and may cause system crashes

ex: Cryptolocker → encrypted files and demand ransome

Virus Phases

- Dormant Phase → sleep to avoid detection
- Propagation Phase → replicates itself to new files/systems
- Triggering Phase → logical condition that triggers the virus
- Action Phase → performs malicious actions

Virus Concealment

- Encrypted Virus → encrypts payload
  ↳ antivirus can use decryptor to detect
- Polymorphic Virus → random encryption variation
- Metamorphic Virus → variable internal structure
  ↳ rewrites itself

Computer Worms

↳ spreads copies of itself without human interaction

Behaviour

- carry malicious payload ex: deleting files, installing backdoors
- replicate rapidly, causing system to slow down
- can be controlled remotely

Development

- find vulnrability
- write code to exploit it
- generate target list
- Install & payload execution
- querying/reporting if a host is infected

Trojan Horse

↳ malware program that masquerade as legitimate software but steals sensitive information

Behaviour:

- does not replicate itself
- captures important data

Rootkits

↳ designed to infiltrate machines, give attackes remote control and remain hidden for extended periods

⇛ hard to detect by software that relies on the OS itself.

Adware

↳ ads delivered to a user via software payloads

Spyware

↳ Software that spies on the users activity

Locic Bombs

↳ malware that peforms malicious action as a result of a certain logic condition

ex. crash system on date xy

Backdoor

↳ hidden feature in a program that allows a user to perform actions he should normally not be allowed to

Malware Zombie

↳ can turn computer into a zombie → machine is controlled externally to peform attacks

Cryptohijacking

↳ malicians crypto mining → utilizes machine resources to mine crypto

Signatures

↳ malwae countermeasure

A virus has a bluprint, some sort of signature that can be calculated based on its code.

These signatures can be matched by an anti-virus softwae

Buffer Overflow

↳ program writes more data to a block of memory than it is supposed to hold.
This extra data then overtites adjacent memory.
This can lead to unexpected behaviour.

⇒ attacker could rewrite a return address and point the program to some malicious code

Protection

→ canary bytes

↳ random bits that come after the buffer.
A Program can then verify if these bits have been overwritten, as they are known

IP Address

Subnet

128.148.32.110

network host

IP header

includes:

- Source address
- destination address
- packet length
- Time to live (ttl)
- Ip protocol version
- Fragmentation Information
- Trasport layer protocol Information

Denial of Service (DOS)

↳ making a target system/network resource unavailable for its intended use

⇒ sends large number of packets to host providing service

↳ slows down, or crashes host

↳ often executed by botnet

Internet Control Message Protocol (ICMP)

↳ used for network testing & debugging

↳ simple messages encapsulated in single IP packets

Tools based on ICMP:

→ Pings

→ Traceroute

ICMP Attacks

Ping of Death

↳ sends a ping request that exeeds the maximum size of a ping request. The network splits this request into fragments. On reasembly, the target machine fails to reasemble correctly as the size exeeds the maximum. This leads to a buffer owoflow. → target host crashes

Smurf Attack

↳ large number of ICMP packets are sent withe a faked (spoofed) source address. Responses will all go to the same address, flooding the machine with traffic

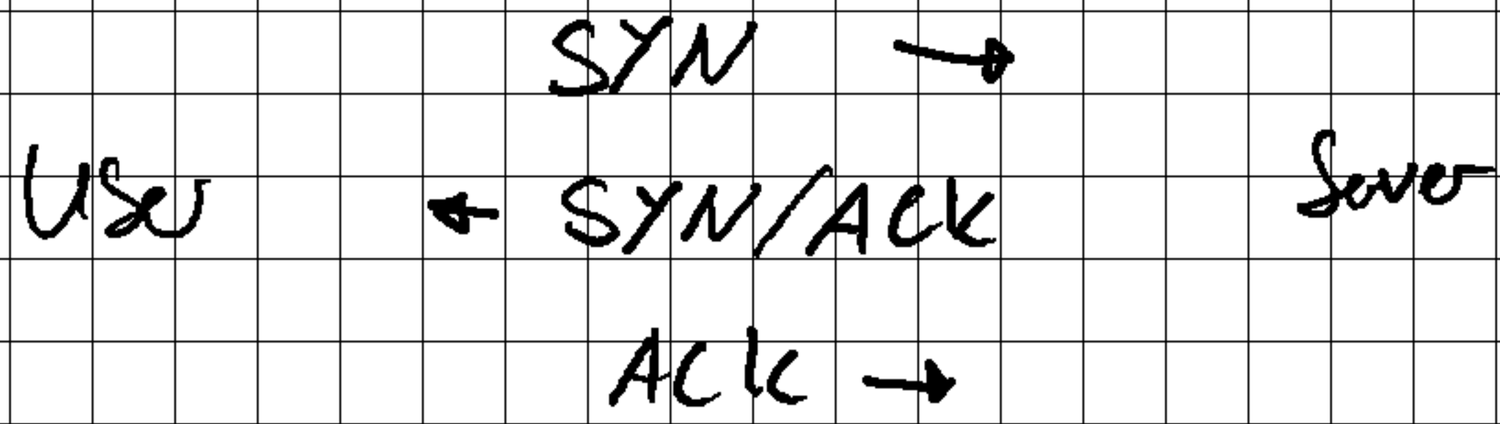
↳ slows target system down to unusability

IP Vulnrabilities

- Unencrypted transmission → ability to eavsdrop
- no source authentification → ability to spoof
- no integrity checking → forge content, redirect, etc..
- no bandwidth constraints → denial of service

TCP SYN Flood Attack

→ TCP connections are established in a 3-way handshake



Bot sends SYN request faster as server responds with SYN/ACK and never returns ACK, resulting in the overload and eventual stop of traffic

⇒ Denial of Service

Firewalls

↳ integrated collection of security measures designed to prevent unauthorized electronic access to a networked computer system.

⇒ predefined set of rules to filter incoming and outgoing traffic

policy actions

- Accepted → permitted through firewall
- Dropped → not allowed through, no indication of failure
- Rejected → not allowed through, inform source of rejection

Stateless Firewall

↳ processes each packet without regard to any packets it processed previously

⇒ Firewall keeps no state

Stateful Firewall

↳ can tell when packets are part of legitimate sessions

⇒ Maintain tables of active connections

Tunnels

↳ end to end encryption of traffic

Intrusion Detection System (IDS)

↳ actively monitors your computer/network for malicious activity & alerts you when it detects an attack

- Signature based detection
  - ↳ keeps list of attack signatures and compares incoming threats to this list
- annomaly based detection
  - ↳ monitors for system abnormalities throug established baseline

Intrusion Prevention System (IPS)

↳ blocks attempted attacks instead of just alerting

challenges of Intrusion detection

- Accuracy and Detection speed
- High detection rate
- low false alarm rate

Detection rate:

$$DR = \frac{TP}{TP + FN}$$

TP → True Positive

FN → False Negative

False Alarm rate:

$$FA = \frac{FP}{TN + FP}$$

FP → False Positive

TN → True Negative

Zero-Day Attack

↳ An attack that is unknown and therefore unaddressed by Security experts

⇒ no patches, fixes or signatures are known

Ceasar Cipher

↳ shift each letter x positions in the alphabet

x = secret key

⇒ to decrypt shift each letter back by x

⇒ only 25 possible keys → easy to crack

Substitution Cipher

↳ asign each letter of the alphabet another letter and substitute them

↳ bijective (one to one) function

↳ 26! possible keys

⇒ can be cracked using frequency analysis, as the frequenzy of individual letters are statistically fixed in each language, when there is a large enough text.

Exhaustive Search Attack

↳ Brute Force Attack

⇒ check every possible key

⇒ key space must be big enough to make a brute force Attack infeasible

Crypto System

↳ pair of algorithms (Encryption, Decryption) that take a key and convert plaintext to ciphertext and back

K → key space
M → Message Space
C → Ciphertext Space

} Cipher can be defined over (K, M, C)

E → Encryption Algorithm ⇒ $E: K \times M \to C$

D → Decryption Algorithm ⇒ $D: K \times C \to M$

Therefore $E(K, M) = C$

$D(K, C) = M$

Symmetric key Encryption

↳ using the same key for encryption & decryption

⇒ needs a secure way of sharing the key

One time pad (OTP)

↳ theoretically unbreakable encryption technique, when used correctly

→ generate a unique, completely random key that is at least as long as the plaintext.

→ convert plaintext into numerical form.

→ for each letter of the plaintext, add the coresponding letter from the key. If the result is greater 26, subtract 26.

→ result modulo 26 = encrypted letter

$\Rightarrow \quad m_i + k_i \ \% \ 26$

⇒ revese process to decrypt

⇒ Secure if every key is used only once, otherwise, the messages can by decrypted without knowing k, as

$$c_1 = m_1 \oplus k \quad , \quad c_2 = m_2 \oplus k$$

$$\Rightarrow \quad c_1 \oplus c_2 = (m_1 \oplus k) \oplus (m_2 \oplus k)$$

$$= (m_1 \oplus m_2) \oplus (k \oplus k)$$

$$= (m_1 \oplus m_2)$$

Stream Cipher

↳ uses a fixed-sized key to generate a pseudorandom key stream

⇒ use pseudorandom key stream to encrypt like OTP does.

Pseudo Random Generator

↳ takes a short random number (seed) and outputs a long string, that looks random

⇒ A good PRG must be unpredictable

Block Cipher

↳ plaintext is split into same-sized blocks. each block is then encyped into a same size block.

Padding is added if the sizes between blocks don't match.

Meet in the Middle Attack

↳ Attacker knows a plaintext, ciphertext pair in a double encryption system. He can now try to find a key that brings him from the plaintext to the ciphertext.

⇒ This brings down the complexity and therefore makes brute force a feasible attack.

Modes of Operations for Block Ciphers

- Electronic Code Block (ECB)
  ↳ encrypts each block of the plaintext independently
    ↳ simple & easy to implement
      ↳ two identical blocks result in the same ciphertext block
- Cipher Block Chaining (CBC)
  ↳ each plaintext block is XORed with the previous ciphertext block
    ↳ uses an initialization vector for the first block
      ↳ requires sequential processing
- Counter (CTR)
  ↳ converts block cipher into a stream cipher.

  Encrypts a counter & XORs it with the plaintext

  The counter is incremented each block

Public key Cryptography

↳ triple of efficient algorithms

- KeyGen → (pk, sk) ⇛ randomized key generation that outputs a private- & secret-key
- Enc(pk, m) → c ⇛ randomized encryption algorithm that outputs ciphertext c
- Dec(sk, c) → m ⇛ deterministic encryption algorithm that outputs the plaintext m that corresponds to the ciphertext c.

⇛ randomized encryption → never the same ciphertext

⇛ easy to verify, hard to crack

key exchange

↳ how to make sure a key really belongs to the person i am trying to talk to?

↳ give key in person

↳ Secure channel

↳ requires certification of each party

Certification Authority (CA)

↳ issues certificates to certify partys

One-Way-Functions

↳ easily computable in one direction, hard to compute the inverse

⇒ ex. factoring large numbers

↳ easy to multiply, hard to find factors

RSA (Rivest - Shamir - Adleman)

↳ private & public key encryption, where the shared secret is computed using the public key of the other party.

⇒ 1. choose two large prime numbers q and p

2. multiply them $n = p \cdot q$

3. choose a number e that is coprime to $\overbrace{(q-1)(p-1)}^{= \phi(n)}$

4. calculate the inverse $d = e^{-1} \mod ((q-1)(p-1))$

⇒ private key : d

public key : e

Encryption: $Enc(pk, m) \rightarrow c$

$c = m^e \pmod n$

Decryption: $Dec(sk, c) \rightarrow m$

$m = c^d \pmod n$

⇒ It is hard to compute $\phi(n)$ given only n.

Coprime → Two numbers are coprime, if their greatest common divisor (GCD) is 1.

Chosen Plaintext Attack (CPA)

↳ Attacker is able to compute ciphertext for chosen plaintexts.

AS RSA gives the same ciphertext for the same plaintext every time.

The goal is to galler enough information to deduce the decryption key, or decrypt other ciphertexts without access to the key.

Homomorphism

↳ property of encrytion schemes that allow for opvations to be poformed on ciphertexts, which, when decrypted yield the seme result as if the operation had been performed on the plaintexts

⇛ allows for operations over encrypted data

⇛ RSA has this poputy:

$$Enc(pk, m_1) \cdot Enc(pk, m_2) = Enc(pk, m_1 \cdot m_2)$$

Diffie - Hellman Protocol

↳ secure key exchange

1. choose public parameters
   ↳ choose a large prime number p
   ↳ choose a generator g with g beeing the primitive root modulo p ($g^a \bmod p$ can generate every integer from 1 to $p-1$ withe some a)

2. both parties choose a random integer as their private key: a, b

3. compute Public key:

$$A = g^a \bmod p$$
$$B = g^b \bmod p$$

4. exchange Public keys

5 compute shared secret:

$$s = B^a \bmod p$$
$$s = A^b \bmod p$$

⇒ relies on the discrete logarithm problem
↳ retriev a from A is computationally infeasible

El Gamal Encryption

↳ public key cryptosystem based on Deffe-Hellman

1. choose a prime number p
2. choose a generator g, with g beeing the primitive root of p ($g^a \bmod p$ can genrate each integer between 1 and $p-1$ with some a)
3. select random integer $x \in \{1, 2, \dots p-2\}$
4. compute $h = g^x \bmod p$

⇒ Public key: (p, g, h)

Private key: (x)

Encryption:

$$Enc(pk, m) \rightarrow c(c_1, c_2)$$

$$c_1 = g^r \quad (r \in \{1, 2 \dots, p-1\})$$

$$c_2 = m \cdot h^r$$

Decryption:

$$Dec(sk, c) \rightarrow m$$

$$m = c_2 \cdot c_1^{-x}$$

⇒ randomized encryption scheme → CPA secure

⇒ homomorphic

Discrete Logarithm Problem

⇛ given $y = g^x \bmod p$, find x

⇛ This can only be solved by trying every possibility, therefore making it a strong Foundation for cryptographic protocols. (ex. El Gamal, Diffi-Hellman)

Data Integrity

↳ assure data has not been modified in an unauthorized matter

→ Modification can be deletion, insertion, reordering ...

Man in the Middle Attack

↳ third pary intercepts the initial key exchange and establishes a channel to both parties.
He can now intercept messages and decide to relay them or change them.
The two original parties think they are communicating directly.

Cryptographic Hash Function

↳ maps long messages into fixed size bit strings

⇒ This is called a checksum or fingerprint

⇒ Hash Function should be one-way, making it unfeasible to generate a message with a specific Hash.

Collision

↳ two different inputs produce the same output

Collision Resistent

↳ A function is collision Resistent if it is hard to find collisions for it.

⇒ There is no explicit, efficient algorithm to generate collisions

⇒ Collisions will always exist, as the input space is much larger than the output space

Hash Function Requirements

1. Variable Input size
2. Fixed output size
3. Efficiency
4. Preimage resistant (one way)
5. Second preimage resistant
   ↳ (find a hash that matches the hash of a given input)
6. Collision resistant
   ↳ (find two inputs for the same hash)
7. Pseudorandomness
   ↳ (output seams random → small input change = big output change)

weak hash function → property 1-6

strong hash function → property 1-7

Birthday Attack

↳ Hash Function Attack based on the Birthday Paradox

Birthday Paradox

↳ In a group of 23 people, the probability of two people sharing a birthday is >50% as there are $\binom{23}{2} = 253$ possible pairs and each pair has the possibility of $\frac{1}{365}$ of sharing a birthday.

In Hash Functions:

→ A n-bit Hash function has 2^n possible values.

⇒ The Birthday paradox states that you now only need $2^{\frac{n}{2}}$ different Inputs to have a chance of >50% to find a collision.

⇒ n-bit Hash function has $\frac{n}{2}$ bit security against the Birthday Attack

Merkle - Damgård Construction

↳ method to build hash functions that produce a fixed length output

1. split input in a sequence of same-sized blocks
2. add padding to ensure all blocks are the same size
3. generate a hash using the first block and a initialisation vector
4. generate the second hash with the previous hash and the next block

⇒ Iterative hashing

⇒ Hash function needs a block & a hash input

⇒ if the compression function is collision resistant, the overall Hash function will also be collision resistant.

Password Hashing

↳ Password is hashed and the output is saved

↳ to check the input password hash input and compare hashes

⇒ one way & more secure way of storing passwords

Vulnrability

↳ people use weak passwords

↳ attacker can build a dictionary of common passwords and their hashes (rainbow table)

Improvement

→ before hashing, add a randomly generated value (salt) to the password and then hash it. Save the hash and the plaintext salt. ($salt $hash)

⇒ on verification, take the plaintext salt and add it to the input. Then compare resulting hashes

⇒ makes a rainbow table attack more costly

dictionary attack

Message Authentication Code (MAC)

↳ used in crypto system to ensure integrity

→ uses a message and the shared secret key to generate a short tag

→ MAC is sent alongside the message so that the receiver can verify the integrity by computing the same MAC and comparing his and the received one

⇒ $MAC(k, m) \rightarrow t$

Cipher Block Chaining (CBC)

↳ MAC is generated by computing a temp MAC for each Block and using it to generate the next MAC. The MAC from the last block is the final MAC.

⇒ given message M and MAC(M), an attacker could add data and generate the new MAC for it.

Encrypted Cipher Block Chaining (ECBC)

↳ uses a second key to encrypt the tag, that was generated from CBC.

Hash-based Message Authentication Code (HMAC)

↳ uses hashing algorithm to generate secure MACs

⇒ as long as the Hash Function H is secure, HMAC is secure.

$$HMAC = H\Big(\underbrace{k \oplus opad}_{\text{Secret key}} \;\|\; \underbrace{H(k \oplus ipad \;\|\; m)}_{\text{message}}\Big)$$

(concatenation / Verkettung; inner hash)

ipad: some fixed padding for H

RSA Signature Scheme

private key : d (secret key)

public key : e

$$Sign(sk, m): \sigma = m^d \bmod n$$ p.9

⇒ make more secure by hashing m first

$$\Rightarrow Sign(sk, m): \sigma = (H(m))^d \bmod n$$

↳ this is secure as long as H is secure & collision resistant.

Digital Signature Algorithm (DSA)

1. choose a large prime p
2. find sub-prime q (prime divisor of p-1)
3. find generator g
   4($g^a \bmod p$ generats all integrs from 1 to (p-1) with some a)
4. Specify a Hash Function H
5. Select a random integer x = private key / secret key
6. calculate public key : $X = g^x$

Signature : $Sign(sk, m) \rightarrow \sigma(R, s)$

$$R = (g^r \bmod p) \bmod q$$

r → random integer

$$S = \frac{(H(m) + x \cdot R)}{r}$$

Verification : $Verify(pk, m, \sigma)$

$$R \stackrel{?}{=} (g^{H(m) \cdot s^{-1}} \cdot X^{R \cdot s^{-1}} \bmod p) \bmod q$$

⇒ everyone can verify a message using the public key, no need for individual keys.

Secure Socket Layer (SSL) & Transport Layer Security (TLS)

↳ protocols designed to provide secure communication over a computer Network.

⇛ TLS is the successor to SSL

Goal:

- Cryptographic Security → establish secure connections
- Interoperability → ability to communicate no matter the systems
- Extensibility → ability to change encryption methods
- Relative Efficiency → ability to use session keys

Hand shake

3 parts:

- Hello messages → agree on parameters for connection
- key exchange → exchange encryption scheme specific parameters and establish shared secret
- finished message → confirm established connection & start sending data

Key exchange

↳ we can use protocols like Diffie-Hellman or any other private key encryption protocol

Pefect Foreward Secrecy (PFS)

↳ property of a secwe communication protocol that ensures tle confidentiality of past session keys, even if tle sewers long tum private key is compromised

⇛ private key is only used to authenticate tle server, not for generating a shared secret.

TLS 1.2 vs TLS 1.3

↳ 6 step Handshake vs 4 step Handshake

⇛ TLS 1.3 combines tle Hello messages and tle key exchange messages

↳ TLS 1.3 removed support for unsecwe algorithms

↳ TLS 1.3 added mandatory foreward secrecy

⇛ TLS 1.3 is more secure and offers a simplified and faster handshake and added mandatory foreward secrecy

SSL / TLS Attacks

- Drown Attack
  - ↳ uses vulnrability in the SSLv2 protocol to initiate a hand shalce with specifically crafted messages. This allows the Attacker to gain partial Information about the Servers private key. As the same private key is used for more secure TLS protocols on the same server, the attacker is able to use this partial information to decrypt session keys and therefore communication.
- Man in the Middle Attack
- Heartbleed
  - ↳ uses vulnrability in OpenSSL's implementation of the TLS Heartbeat extension, that allows attacker to read memory from the server, potentially exposing sensitive data like private keys.
- Poodle
  - ↳ uses vulnrability in SSL 3.0's padding