TDDE62 Summary

Network Security

Starts with good network design:

- Segmentation
- Perimeter defence
- Containment

Vulnerabilities

- Networks are increasingly **complex**
- System are connected without a **security focus**

Network Security

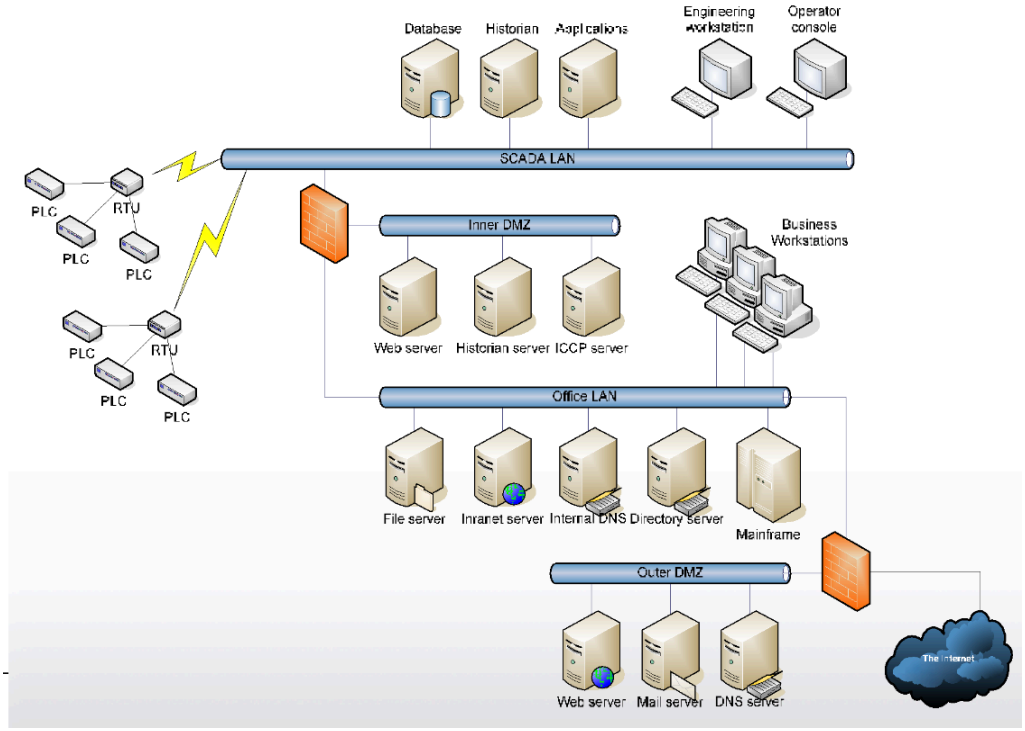
Is mostly focused on **reducing exposure** and increasing the **risk** for an attacker. Goes hand-in-hand with **system security.** Make sure auditing, monitoring, access control and other parts of the system are also working.

Designing for Security

If security is not part of the design, lots of time will be spent on patching the system.

- **Network Segmentation** - Divide the network into different parts with barriers between them. Have different zones for different functions. It contains threats to specific resources.
- **Perimeter Defence** - Protect the borders between the networks. Protect against outside attackers. Is usually a **firewall** or **intrusion detection system.**

- **Network Containment** - Limit network to a known extent, very hard with wireless networks.



Separation Mechanisms

Air-gaps

Physically disconnect network segments. Doesn't work practically. We often need to transport data to and from the network. If data transfer is **convenient**, it will be easier for an attacker. Even if tried, it does not always exist. Temporary connections can be made for updates, misconfiguration of switches etc. Laptops completely defeat the air gap. If a system sits on more than one network, access to one can be gained from the other.

Ex. A protected network uses the same DNS server as a network that is accessed from the internet.

Network equipment are systems of their own. A switch that manages two systems forms a connection between them.

Good network management defeats air-gaps. Virtual **LAN** is often used. They are logically disconnected but run on the same hardware. A **management LAN** is often used that can reach all devices, but usually from only a few places. Even if these are good practices, it completely destroys the air-gap.

Conclusion: Do not bother, assume air-gaps don't exist.

Firewalls

Computer that acts like a router with rules that can **filter traffic**. Normal routers are **not** for segmenting networks, they are only used to connect (home routers usually have both in one device). Enforces network policies and segmentation. They are **not an excuse** to not secure the inside.

Firewall rule - Consists of a **set of criteria** and an **action**.
Traffic criteria - Source/destination address, source/destination port, protocol, physical interface, etc.
Action - Allow, drop without notification, reject with notification.
Policy - What to do with traffic that doesn't match any criteria.

Weaknesses - They are only as good as their **configuration,** many are misconfigured. They give little protection against inside attacks. They place part of you overall security on a **single device.**

Network Address Translation

Rewrite addresses on packets going through the NAT box. Allows hosts with private addresses to access outside networks and prevents **direct connection** to NATed systems.

Sometimes **abused** as a **security mechanism!** It does not protect against stuff requested from the inside (malware).

Trust Relationships

Segmented networks still need to communicate. **Trust relationships** bridge these gaps. **Trusted systems** are given additional rights.

Ex. DNS servers, shared passwords (trust other systems not to share the secret), firewall rules.

Trust relationships are a necessary **risk**. Make sure to **map** existing trust relationships (for example using an adjacency matrix). Eliminate unneeded relationships or ones that cause too much exposure.

Backdoors

A hardware or software **entrance** into a computer that bypasses security controls. They are very common in **complex systems** (things are forgotten, misplaced, etc). They are **very dangerous** since they break assumptions that security is based upon.

Wireless Networks

Extremely dangerous and uncertain. Early versions are completely broken. It changes the extent of the network. They can be a challenge even for wired networks because of **rogue access points** or **laptops**.

Management Frames

Are used by stations to establish and maintain communications. Can be forged since they have no security.

- **Association frame -** Station sends frame to access point. Carries information about the station and SSID of the network it wants to connect to.
- **Disassociation frame -** Send when a station wants to disconnect from AP.
- **Reassociation frame -** Station moves away from current AP to one with a stronger signal. The new AP coordinates forwarding of data frames that it still has in its buffer.
- **Beacon -** Broadcast from an AP saying it exists, displays SSID.

If beacons are turned off, SSID can still be found by making **probe requests.** AP's will respond with their SSID. Even if these are turned off, if a station is connected to the AP a **dissociation frame** can be sent. A **reassociation frame** will be sent to the AP immediately revealing the SSID.

Wi-Fi Security Standards

WEP (Wired Equivalent Policy)

Broken by design, breaks in 5-10 minutes. Goal was to offer the same security as **wired networks.** Features authentication, confidentiality, and integrity.

WEP uses **RC4**, one of the most popular **stream ciphers**. It is "secure" under correct usage. In WEP, the input keys were **related.** The IV was sent in clear text and available for attackers. Keys were also too short, among other things.

WPA

Uses longer RC4 keys, has new checks for integrity and replay attacks. Can be implemented on WEP hardware. Problems can arise if passwords or SSID are easy to guess.

WPA2 (IEEE 802.11i)

Uses AES, has longer keys. Requires new hardware.

Wireless Attacks

If an attacker knows a station's **encryption key**, they can respond quicker than anything on a different network.

Ex. DNS query from the victim is responded to by an attacker.

Wireless networks are also very vulnerable to **DoS attacks.** Disassociation frames can be **forged** forcing victims to disconnect. Attackers can also send **ACK frames** every half second to reserve radio channels.

Evil Twins

Attacker dissociates the victim who then reassociates to the attacker. If the attacker has a better signal to the victim than the AP they will be connected to instead. The attacker can then relay traffic to the real AP, or use its own connection to stage a **woman-in-the-middle-attack.**

Rogue Access Points

Huge problem for both wired and wireless networks. Access points are put into place without the **network admins knowledge.** Doing this is easy, detecting them is difficult.

Jamming

Transmit garbage on the same frequency.

2G/3G/4G/5G Security

Based on symmetric keys stored on **SIM cards.**

ICMP

Is used to send **error** and **control messages** in a network (requested service not available, time to live, echo replies, traceroute, etc). Not designed for security and has a reputation of being **dangerous**.

- **Source quench -** Tells recipient to slow down sending, could be used for low-bandwidth **DoS attack.**
- **Redirect -** Tells a system to send traffic destined for a particular system to a specific router. Allows **DoS** or **woman-in-the-middle attacks.**

TCP

Client sends connection request to the server. The server responds with an acknowledgment. The client acknowledges and the connection is established.

Problem - When the server receives the second packet, it must confirm that it is part of a previously accepted connection. TCP therefore has a **connection queue** where all accepted connections are parked until the last **ACK** is received.

SYN Flood

The **SYN flood** exploits this. A large number of connections are requested without being acknowledged. In theory it will eat up the server resources but is not much of a problem anymore. However, it can be easy to miss when **developing new protocols.**

Prevention - Do not save any data on the server. Instead, a **cookie** is sent along with the ACK with information only the server knows about. This has been attempted in TCP but it didn’t quite work out.

DNS Security

DNS is distributed over a **huge number** of hosts in an hierarchical fashion. **Caching** is used to ensure efficiency but is also one of the main **security issues**.

Cache Poisoning

An attacker compromises a DNS database. The changes will propagate through the network. It is quite hard to accomplish nowadays but can still be done. DNS should not be trusted blindly.

DNSSEC

Secure DNS by cryptographically signing the server responses. It creates a **chain of trust** that starts at the root servers. Is used by org, com, net, and edu domains today, among others.

Scanning

Purpose is to gather information that can be used for an attack. Information can be firewall configurations, open ports, RPC services, known vulnerabilities, operation system details, etc.

Firewalking

The goal is to figure out the configuration of the firewall without connection to any systems beyond it. A datagram is sent to systems behind the firewall with a **Time-to-Live** set such that it will decrease to zero if it gets past it. If the firewall does not respond, the port is filtered. If an ICMP destination unreachable/time exceeded is received, the port is open.

TCP Scans

Knowing if a port is open is useful, knowing if there's a process listening on it even more so.

- **TCP SYN Scan -** Send SYN segments to the server. If a SYNACK is sent back, the port is used. Immediately send a RST back (tears down the connection and prevents logging).
- **FIN Scan -** Send segment with FIN flag to the server. Firewalls that filter only **connection attempts** will let it through. TCP requires that an RST is responded to if the destination is closed.
- **ACK Scan -** ACK segment is sent to the server. RST is returned regardless but information can be inferred in the TTL and window size of the response.

UDP Scans

Depends on the target responding when a UDP datagram is sent to a closed port. Response is unpredictable when sent to an open port. Makes it hard to distinguish between open and filtered ports.

SSL/TLS

Secure Socket Layer and Transport Layer Security. It is placed between the **application** and the **transport layer.** It supports multiple protocols such as HTTP, FTP, Telnet, etc. It has strong encryption, integrity protection and mutual authentication.

TLS

A TLS session is a long-lived **association** between two hosts. It may span many connections. The client sends which version and encryption algorithm it prefers. The server responds with a **certificate**, which must be certified by a known **certificate authority**. This certificate gives the client the server's **public key**. The client chooses a key for **symmetric encryption** and sends it, encrypted with the server's **public key**.

It is very secure and resilient to many attacks. Sequence numbers make sure **replay attacks** cannot happen. **MITM** are made ineffective since all communication is encrypted and integrity checks are made with **HMAC.**

The big problem is the **certificates**, and people accepting **invalid certificates**.

Certificate Validation

A **certificate** is an electronic document containing public information about an entity. If used for **secure communications**, a public key is made available. The certificate must be **verified.** Special certificate authorities (CA) sign them with their private key. We can verify the signature with the CA's public key.

IPSec

It involves security at the **network layer.** It provides confidentiality, integrity, and authentication. It is made up of a different set of protocols.

AH and ESP

Authentication Header Protocol and Encapsulating Security Payload Protocol add **headers** and **trailers** to IP packets to protect the head and/or the body.

IPSec can operate in two modes:

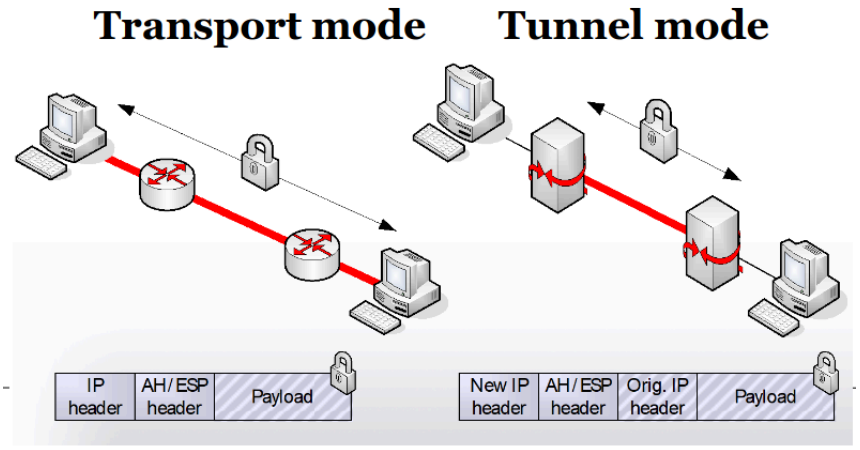
- **Transport Mode -** Two peers communicate, they add AH/ESP headers and the data remains protected until it reaches the peers.
- **Tunnel Mode -** Existing IP packets are encapsulated in new IP headers. Used to design VPNs.

- **ESP Transport Mode -** Adds encryption to the payload, can also add authentication.
- **ESP Tunnel Mode -** Protects the original IP header. The new IP header is not authenticated.

- **AH Transport Mode -** Authenticates the IP headers. Some headers change, like TTL, and cannot be authenticated. It is not compatible with NAT and does nothing to encrypt data.
- **AH Tunnel Mode -** The new IP header will also be authenticated.

There are two ways of getting both **authentication and encryption**:

- **Nest ESP in AH -** Apply ESP to the packet, encrypting it. Then add AH to
- authenticate.
- **Authenticated ESP -** Apply authenticated ESP to the payload. ESP header encrypts and ESP trailer authenticates. However, the IP header won't be authenticated.



Security Associations (SA)

A one way relationship between IPSec hosts, it allows the sending of protected packages. Determines how packets are processed, algorithms, parameters, keys, etc. Two SAs are needed for two-way communication.

Security Association Database (SAD)

SAs are stored in SAD. They are uniquely identified by their **Security Parameter Index** (SPI).

Security Policy Database (SPD)

At the **sender**, the processing is determined by the SPD. When a packet is received, a lookup is made to determine what should be done. It could be let through or discarded. If IPSec is applied, the SPD references an SA in the SAD for processing details. At the incoming side, SA is looked up in the SAD.

IPSec and IKE (Internet Key Exchange)

IPSec is dependent on SA, which is fine for static associations such as VPNs. For dynamic associations, a protocol is needed, IKE.

IKE sends a clear text message between hosts confirming that they can connect. **Diffie-Hellman** key exchange is then used and SAs are negotiated.

Diffie-Hellman

todo: is this really relevant?

Intrusion Detection

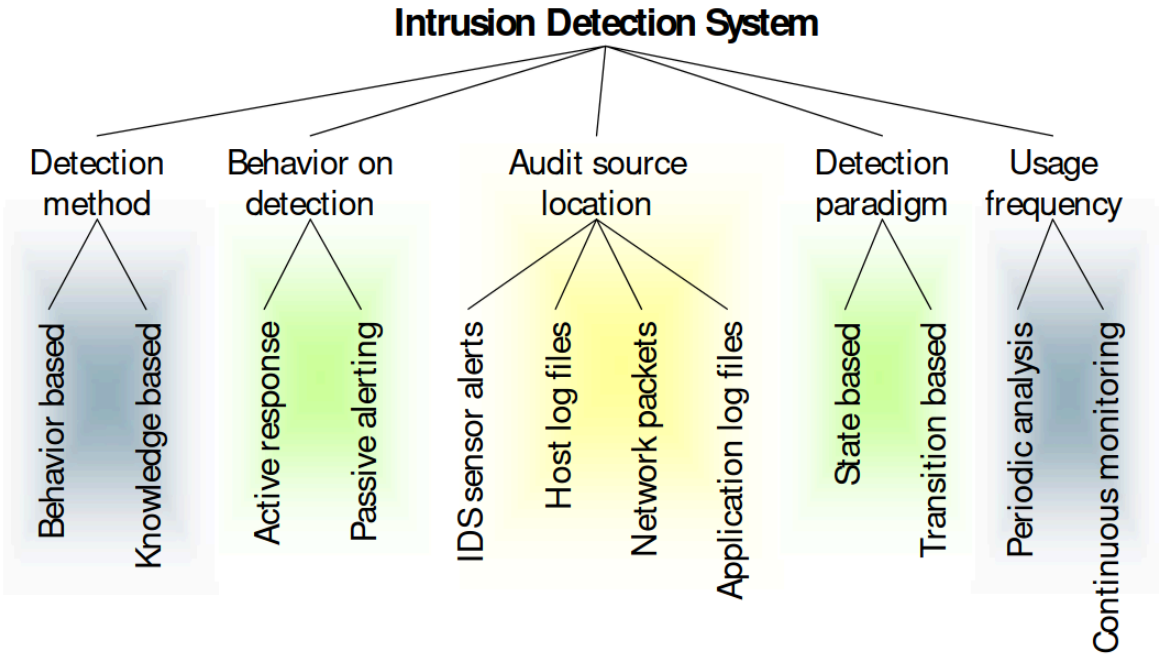
Does nothing to directly **prevent** intrusions, but may help in understanding or mitigating their effects. They are divided into **NIDS** (Network Intrusion Detection System) and **HIDS** (Host-based Intrusion Detection System).

Intrusion Detection System (IDS)

A device that collects information, analyzes and determines if it's a problem or not.

Classification

- **Behavior-based -** Use information about the system's normal behavior to detect intrusions.
- **Knowledge-based -** Use knowledge about intrusions, e.g. worm signature.
- **Behavior on detection -** Separates systems that only alerts from systems that try to prevent intrusions.
- **Detection paradigm -** Separate systems that evaluate if a system is secure or not (state based) from systems that detect when a system becomes insecure (transition based).

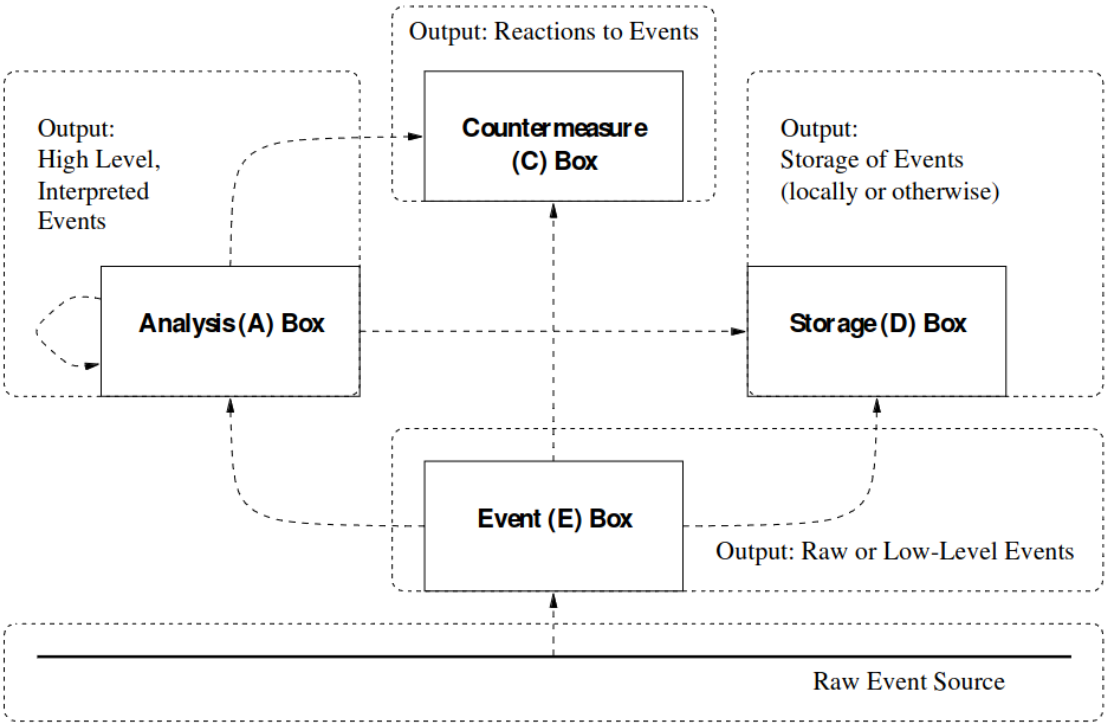


Snort

Open source NIDS, passive alerting, transition-based, continuous monitoring system. It uses signatures from known attacks to match against network packets in real time.

Components:

- **Event Generators (E-Boxes) -** Provides information about events to the rest of the system.
- **Analysis Engines (A-Boxes) -** Analyses information from E-Boxes.
- **Storage Mechanisms (D-Boxes) -** Defines how the security information should be stored.
- **Countermeasures (C-Boxes) -** Defines what to do if an attack is identified. Alarm, shut down, filter, etc.



An attack on a NIDS is almost always to hide **another attack.** Hitting the E-Boxes disables the NIDS, A-Boxes disables all analysis, D-Boxes can result in data not being stored, or even manipulated to look safe. Finally, the C-Boxes result in no action being taken if an attack is found.

NIDS Challenges

- **False positives -** Heavy traffic might raise alarms.
- **Performance -** NIDS that miss traffic may be unable to detect attacks.
- **Security features -** IPSec, VPNs, TSL/SSL might prevent NIDS from doing its job.
- **Desynchronization -** If the end system behaves differently than NIDS expect, no accurate detection can be done.

E-Box Attack

If the host has a known vulnerability and we know that the NIDS is sitting on one network segment before the host, we can craft an attack. The work HQACK is sent to the host, but the datagram containing Q has TTL set so that it will only get to the NIDS. The NIDS see HQACK, the host sees HACK.

IP Fragmentation

IP packets can be broken up for transit. In order for NIDS to see the whole packet it must collect all the fragments. A **DoS attack** can be made by infinitely sending fragments that are not marked as **final**.

Abusing Reactive ID Systems

If NIDS take action, it can be exploited to make the NIDS perform a **DoS attack** on its own network.

Ex. Attack the system with a SYN flood with spoofed IPs from Sweden. Suddenly real connections from Sweden are disallowed.

Social Engineering

Plays on ignorance, insecurities, and fear of people.

Usability

Usually underestimated. Security is extremely complex, asking users and developers to understand it is a big task. Security products are often **too complex.** End-users are not experts.

Privacy

Privacy Enhancing Technologies

End-user should have enough guarantee about the design that they don't have to trust a third party. PETs reduce/minimize the perimeter of trust.

- **Hard Privacy -** Technologies that avoid/reduce disclosure of personal data.
- **Soft Privacy -** Technologies for enforcing rights of users after their data is disclosed or processed. Focus is to provide accountability.

Communication Services

To not be eavesdropped during communication is a basic expectation. It is solved by cryptography but requires trusting a CA or a third party. They don't guarantee that identity (ip, role, time) cannot be inferred.
Ex. Inferring someone's health status by their browsing.

Technologies that protect against this are called **anonymous communications.**
Intermediate nodes are introduced in the network that reroute the messages in a hard to track way.

Anonymity

To enable anonymity, there must **always** be a set of similar subjects.

Anonymity Guarantees in Communication Services

- **3rd Party Anonymity -** When two known parties communicate with each other, 3rd party anonymity is provided it cannot be inferred that they are communicating with each other.
- **Sender Anonymity -** Recipient receives a message without the sender being identified.
- **Receiver Anonymity -** Other parties in the network should be able to contact me without knowing my identity.

Unlinkability

It should not be obvious if two or more items of interests are **related** or not.

Pseudonyms

- **Person Pseudonym -** Substitute for holder's name.
- **Role Pseudonym -** Limited to specific roles, e.g. customers.
- **Transaction Pseudonym -** Each transaction pseudonym should be unlinkable to every other, and is created for every transaction.

Undetectability

It should be indistinguishable whether the item of interest even exists.

Unobservability

- **Undetectability** against all subjects not involved in the communication.
- **Anonymity** for all subjects involved.

Relationships Between Properties

Unobservability (sender and receiver) reveals less than **anonymity**. It also reveals less than **undetectability.**

Mixnet

Anonymous email system that provides **3rd party anonymity** for both senders and receivers. Messages are bounced around in a **mix** making them hard to track. However, the anonymity set is small and there is about a 25% risk that the messages are able to be linked.

Privacy in Access Services

Users are linked to devices they have access to in order to execute privileged actions are retrieve confidential information.

Federated Identity Managers - Has a user, service provider, and identity provider. The user registers with the identity provider that then authenticates them for using a service. Example is Googles SSO.

Shibboleth Identity Management System - User decides if they want to reveal their id to a service provider.

Attribute Based Credentials

Attributes are in focus, e.g. name, age, hair color, etc. They are stored in a secure capsule called **credentials**. It is issued by a trusted **credentials issuer**.

Ex. Skatteverket.

Anonymous Credentials

Unlinkability is expected in the systems.

Single Credentials Authentication Systems

Credentials are only used once. **Blind signature** protocols ensure unlinkability between issuing and disclosing credentials. They are hidden from the issuer.

Ex. Microsoft U-Prove.

Multiple Credentials Authentication Systems

Enables the prover to use their credentials multiple times. Unlinkability is ensured by hiding the credential from both the verifier and the issuer. Crypto protocol called **zero-knowledge proof** is used.

Ex. IBM Idemix

Private Computation Technologies

- **Homomorphic Encryption -** Allows receivers of a cipher text to make operations on them (like adding fees) without decrypting.
- **Secure Multi-party Computations -** Allows several parties to perform common computations on their individual values without disclosing them to other.s

Whose Privacy?

- **Responded Privacy -** Protect the information of individuals records.

- **Owner Privacy -** Protect the information about entities computing the queries.
- **End-user Privacy -** Protect end-user's queries to interactive databases, e.g. search engines.

Statistical Databases

Enables users to receive knowledge. Exploited for disease control, market research, etc. It should provide **anonymity** in the form of **unlinkability.**

There are ways to try to find links:

- **Record linkage -** Re-identify the individual to a record by comparing to publicly free information.
- **Attribute linkage -** Infer confidential attributes with attributes present in the database.

Categories of Identifiers

- **Explicit Identifiers -** Name, social security number, IP address, etc.
- **Quasi Identifiers -** Gender, age, telephone number, etc.
- **Sensitive Attributes -** Disease, salary, etc.
- **Non-sensitive Attributes -** Other attributes.

Challenge

87% of the population can be identified by ZIP, DOB, and sex.

$X = s * t$ matrix. s respondents, t attributes.

- **Non-perturbative Approach -** Let X' be a modified version of X that is obtained by reduction of details. The new values in X' are now the true values.
- **Perturbative Approach -** Let X' be a modified version of X so that all statistical information is preserved.
- **Query Result Perturbation -** Queries are executed on the original database with added random noise.

K-Anonymity

Table T has k-anonymity if every combination of quasi-identifiers is shared by at least $k - 1$ records.

Database Reconstruction Attack (DRA)

k-anonymity does not guard against inference attacks.

Differential Privacy

Provides **strong** privacy guarantee with only a small loss in accuracy. It provides a way to **quantify** the plausibility peak (loss of privacy) and **bounds** (the maximum loss) for the individuals in a dataset.

Ex. Statistical Query. How many people have a cold? Answers will be almost the same no matter if David is in the underlying database or not.

- **Database Neighbours -** Databases that only differ at one record.

Laplace Mechanism to Differential Privacy

Add noise from the Laplace distribution.

Federated Machine Learning

Multiple entities collaborate in solving a machine learning problem, under a central server. Each clients data is stored locally and not exchanged.

The idea is to separate the data and training. It limits data exposure.

Secure Aggregation

Is used to make sure data cannot be reconstructed from the models. Before anything is sent, a zero sum mask scrambles the training results. When all parameters are added together the masks cancel out.

Security and Security Controls on Operating Systems

Protection and Security Controls

The operating system should **identify** and **authorize** users and prevent unauthorized access to **OS resources**.

Some Concepts and Principles

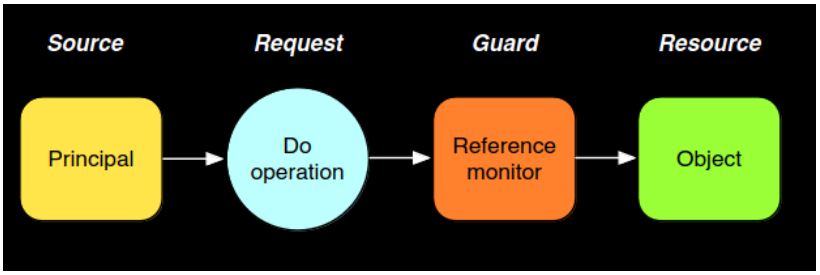
- **TCB -** Trusted Computing Base.
- **RBAC -** Role Based Access Control.
- **MAC -** Mandatory Access Control, principle of least privilege.
- **DAC -** Discretionary Access Control, principle of least surprise.

Capabilities and Requirements

- Protecting a system resource is needed to prohibit **malicious** access. Ex, system tables, I/O units.
- Authorization checks for system calls and resources are needed to maintain system integrity.
- Separation of resources is needed, physical, logical, temporal, or cryptographical separation.

Reference Monitor

Uses a Trusted Computing Base (TCB) to determine whether access should be allowed or not.



Principles for Secure Design

- **Economy of Mechanism -** Keep design as simple as possible.
- **Fail-safe Defaults -** Base access on permission, not exclusion.
- **Complete Mediation -** Every access to every object must be checked.
- **Open design -** Design should not be secret.
- **Separation of Privilege -** Program is divided according to needed privileges.
- **Least Privilege -** Every user and program should have as little privilege as possible.
- **Least Common Mechanism -** Minimize mechanisms common to more than one user that all depends on.
- **Psychological Acceptability -** Human interface must be easy to use.

Memory Handling

RAM must be shared between OS, components, and applications. **MMU** is a hardware supported memory protection.

File System

Central component in computer system. Besides content, **metadata** must also be protected (owner, creation data, etc). Manipulation of metadata can sometimes be more serious than the content itself.

Bugs and Vulnerabilities

Concept and Principles

- **Attack Vector -** Different paths to reach a vulnerability.
- **Reverse Engineering -** To recreate the original design by observing the final result. Recreate source code from binary.

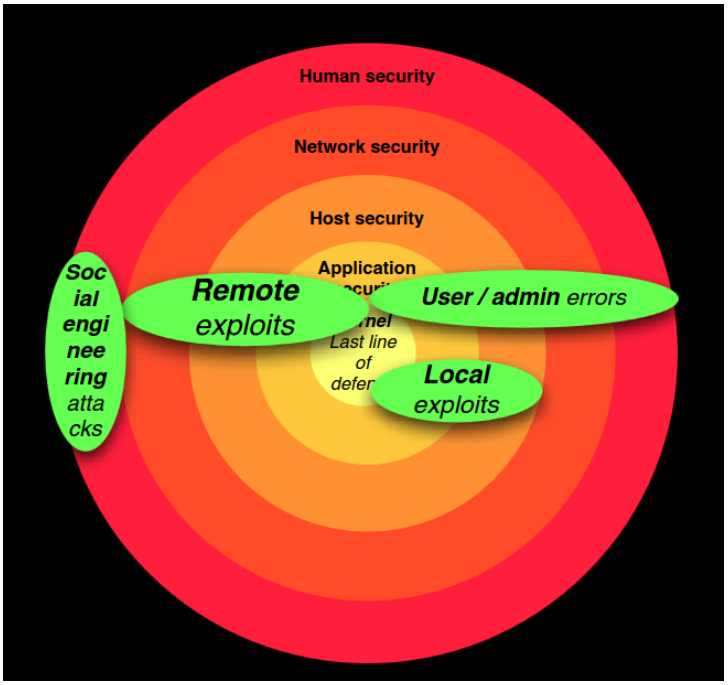
General Control Principles

- **Encryption** protects against eavesdropping or unauthorized access in network traffic, files, memory pages, etc. It is used by OpenSSL, IPSec, SSH, the kernel, etc.

- **Electronic signatures** protect against changes or unauthorized modification in network traffic, file content, or disk partitions. Also used by OpenSSL, IPSec, SSH; the kernel, etc.
- **Cryptographically strong hash values** protect against unauthorized changes and detect them in passwords, file content etc. Used for passwords, databases, file checksums, etc.
- **Random numbers** are used to make resources such as file names, process IDs, port numbers, session keys, etc, nondeterministic.
- **Compiler generated airbag (canary)** is used to make sure buffer overflows are detected.
- **ASLR** is used to randomize addresses used by applications. Make sure it's hard to write code that knows addresses.
- **KASLR** used to randomize addresses used by the kernel.
- **DEP, NX, W^X,** is used to make sure memory is **not executable.**
- **MTE** is a memory tagging extension used in ARM to protect against memory safety violations.
- **Secure boot / verified boot** makes a systems startup sequence secure by cryptographically signing each step. Ex, BIOS or UEFI.
- **Secure pairing** is used to connect peripherals, for example via bluetooth.
- **Scrubbing and zeroing** is used to clean old data areas before they are returned to the system.
- **Logs and audit trails** are used to trace errors and dumps from systems and applications.

Examples of Vulnerabilities and Attacks

Where Attacks Occur



Examples of Attacks

- **Buffer Overflows -** Attacker writes executable code outside the memory boundaries and executes it, for example by manipulating the return address.
- **Backward Compatibility -** Attacks that allow an attacker to use older versions of services or protocols.
- There are ways to bypass OS security controls, especially if the OS is not running.

A Classic Attack

Trojanized c compiler. The compiler's source code is modified to recognize if it is recompiling itself or the login program to insert a backdoor into the login.

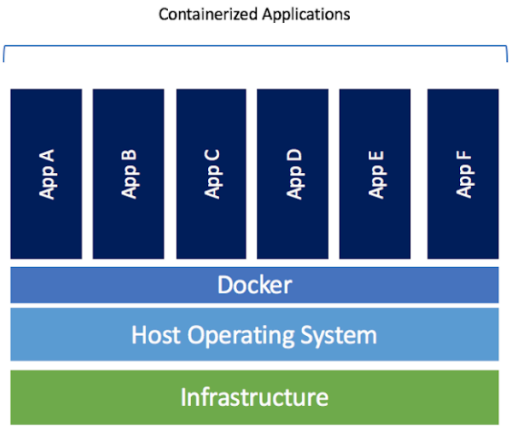
Virtualization and Isolation

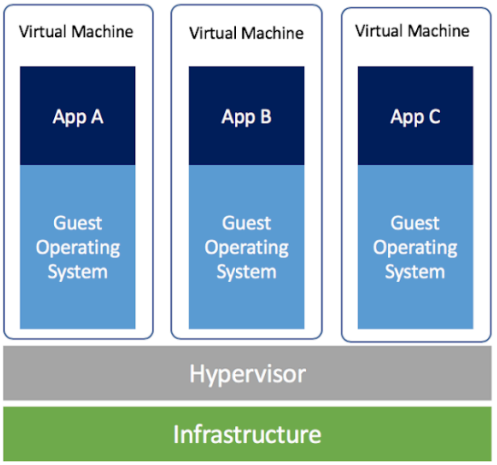
Sandboxing

Creates **temporary execution environments** for certain tasks. Used to perform tasks that are sensitive or prone to attacks. Provides isolation from other parts of the system. Some technologies are not complete virtualization or separation, e.g. share name space, kernel or drivers.

Isolation, Separation, and Virtualization

- **Chroot -** Change root for network service, classic Unix concept. Has no virtualization, only isolation.
- **Jails -** Userland, can run FreeBSD and Linux binaries.
- **Containers -** 3rd party tool on top of the OS. Needs a container engine. Ex, Docker.
- **Virtual Machines -** Hypervisor based and has stronger isolation than a container.
- **Hardware Partitioning -** Best isolation and separation.





Pros and Cons

Isolation is a standard way to minimize attack surfaces. Virtualization can help. However, it is easy to believe that it automatically makes things secure and that there is no way to escape, which is not true.

Advanced Attacks

- Attacks by attaching malicious hardware to buses and ports.
- Firewire and other DMA based methods to access memory.
- UEFI attack via thunderbolt.
- Removal or direct attachment of memory chips (cold boot).

Rowhammer

Flip bits without accessing them. Method of reading or writing memory cells so that adjacent ones become changed. It is based on an unintended side effect of DRAM that causes memory cells to leak their charges between each other. Can be used to attack virtual machines. Have been implemented in JavaScript and run in a browser.

Meltdown & Spectre

Low-level **cache** attacks, allows malicious reads. It breaks isolation between user and kernel mode. They will remain unfixed otherwise CPU designers will have to disable out of order execution.

System Security

Booting

Typical startup sequence of a PC:

1. **BIOS (Basic Input Output System) -** Program stored on read-only memory (ROM) on the motherboard. Loads the **boot loader** into RAM.
2. BIOS transfers execution to the boot leader, which loads the OS kernel into memory and transfers execution to it.
3. The OS kernel loads **device drivers** to handle communication with hardware.
4. Kernel loads programs.

OS Access Control

Processes

Basic entity to which access control is applied. They act as a **proxy** on behalf of a real life user. Whenever a process attempts to access a resource, the kernel checks their permissions.

Access Control Models

- **Discretionary Access Control (DAC) -** Users can assign permissions. Most Unix systems work like this.
- **Mandatory Access Control (MAC) -** Apart from DAC rules, access is also restricted according to system-wide policies. For example, users' home directories should **never** be writable by other users.

- **Role-Based Access Control (RBAC) -** Access is determined by role.

Enforcement

Each process should only access its down data. Functionality of a system should not depend on processes behaving. Processes' memory should be completely **isolated.** Furthermore, they should not be able to access hardware directly, but must go through the kernel.

Memory Protection

Virtual Memory (VM)

Every process has its own **virtual memory.** Through the processes' point of view, **all** memory is available. Kernel keeps track of these. The memory is divided into **pages** (usually 4kB). Virtual addresses are translated to physical addresses by the kernel using the **page table**. Hardware for doing the translation is called a **Memory Management Unit (MMU).**

Swapping

The virtual memory might be much larger than RAM. Only the pages currently active are kept in memory.

Kernel Interface

CPU Modes

The CPU can run in **kernel** and **user mode.**

- **Kernel mode -** CPU can execute all instructions and interact with hardware and physical memory.
- **User mode -** Only some instructions are available.

System Calls

The CPU has a special instruction that saves registers, switches to kernel mode, and moves execution to **predetermined** kernel code. This is used to implement **system calls.**

They are performed in the following way:

1. Process puts parameters in registers.
2. One parameter determined the system call to execute.
3. Process executes a **kernel-switch** instruction.
4. CPU mode is switched and the predetermined system-call code is run in the kernel.
5. Kernel checks parameters and the permissions of the process and executes if allowed.
6. The kernel restores registers and switches back to user-mode.

Each process has a copy of the necessary kernel code. The page table specifies that that part of memory should only be accessed in kernel mode.

Interrupts

Interrupts are an **asynchronous** communication mechanism used by the CPU. Many OS's use them to implement **system calls**.They are also used to handle **access violations**.

Process Switching

The OS rapidly switches execution between running processes.

1. Execution of A is paused, kernel code is executed.
2. Kernel saves values in all registers.
3. Old registers values for B are copied into the CPU
4. Kernel sets B's page table to active.
5. Kernel transfers execution to B's code.
6. Execution of B is resumed.

The OS does this by a built-in hardware timer that sends interrupts at a fixed interval.

Attacker with Physical Access

The OS can only apply access control when it's running. Attackers can physically remove the hard drive and read unencrypted files. Swap may also contain old RAM contents.

Cold Boot Attacks

The RAM memory is pulled out and rapidly cooled down. Their contents can then be preserved for hours and contents read.

DMA Attacks

Devices that require high throughput can implement standards that allow peripherals to bypass the kernel and send data directly to physical memory (Firewire, Thunderbolt, etc.) Attackers can plug in devices that write malicious code into the kernel mode segment, or read out data from RAM.

Reading Buses

Attackers can connect probes onto the buses and read the data flowing in them.

Attacks from Below

Higher-level components depend on the security of lower-level components.

Rootkits

Malware which embeds itself in system software to hide its presence. For example, loads as a device driver very early in boot. It runs in kernel mode and has complete control over the system. It can hide its presence on the file system by manipulating system call results and disable antivirus software.

BIOS Attacks

Computers usually have processes to update the BIOS code. If an attacker gets access **once**, they can establish a permanent backdoor. Even if the harddrive is wiped, the BIOS malware can reinstall e.g. a rootkit.

Firmware Attacks

Computers add-on hardware also have **firmware**. For example, hard drive firmware controls what is returned when the system asks for data. It can be silently manipulated to insert malicious behavior. Firmware is inaccessible to antivirus software and is almost impossible to detect.

User-mode Exploits

If a program has certain types of bugs, attackers can craft inputs that corrupt internal data structures. Allows attackers to inject code that run with the privilege of the process.

Kernel-mode Exploits

Device drives have to run in kernel mode to communicate with the hardware. They often have bugs.

Trusted Computing

It is a way to protect sensitive data and cryptographic keys inside a device.

Confidential Computing

Also includes data and code. It should be able to run in a secure environment.

Zero Trust Principle

Only engage with entities after **proper** verification.

Homomorphic Encryption

An alternative to trusted computing. Send encrypted data to a remove server and let the server process the data without decrypting it.

Trustworthiness

Trust - Mental view/perception.
Trustworthy - Meets defined requirements, can be measured.

Common Criteria (CC) is an ISO standard for measuring trustworthiness. Used for smart cards, crypto libraries, etc.

Ex. Hardware can be trusted because of a reputable vendor, recommendations, and different design assurances such as reviews and proofs.

Remote Attestation - To gain trust of a remote system.

Hardware Trustworthiness - Hard/costly to change, but has high performance.
Software Trustworthiness - Easy to change, but difficult to hold private keys.

Trusted Execution Environments (TEE)

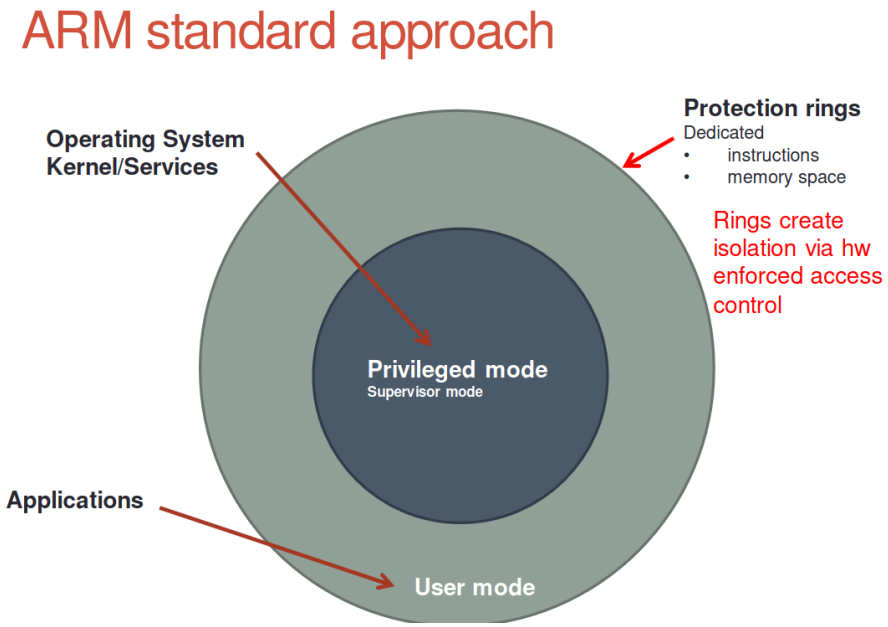
Combine the best of both hardware and software. Software is running with hardware protection mechanisms.

Course Focus - Attestation by hardware modules and later secure execution by the CPU.

Ex. ARM Trustzone, Intel SGX

ARM Trustzone

Can run both normal and secure OS at the same time in a single core.



The basic idea is as follows:

1. Introduce a **NS-bit flag**, used to tag secure data in the system, such as buses, cache, pages, etc.
2. A **monitor** manages this flag and the transition between the modes.
3. The isolation is **hardware enforced**, a secure OS can reach data outside of it but not the other way around.
4. **Secure interrupts** are used to switch to the secure OS.

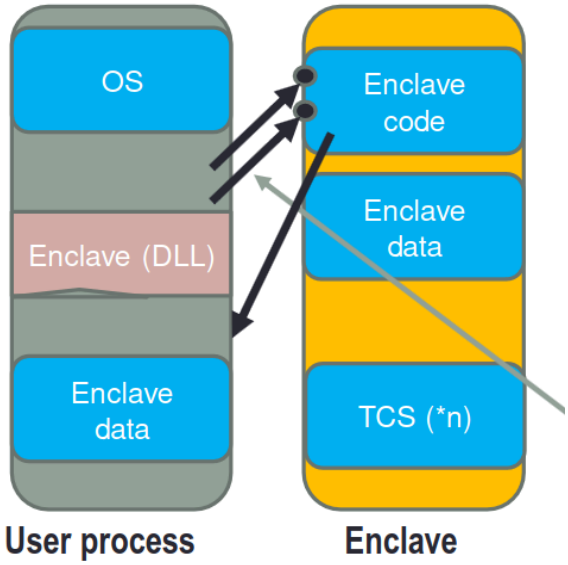
Some shortcomings of the system is that the Trustzone system is not an isolated part of the ASIC. It means it is impossible for it to be rated highly by the Common Criteria. Secure boot and setup is also not a part of it.

SGX (Software Guard eXtensions)

New technology in Intel chipsets. It can load an **enclave** into protected memory. An application or parts of it can be contained in an **enclave**.

An enclave is an isolated memory region. One part of RAM is reserved for them and is called **Enclave Page Cache** (EPC). It is encrypted and managed by the OS.

This defends against memory snooping on buses since it will be encrypted outside the CPU. It also has support for multiple threads and applications can be run in a trusted environment completely inside an enclave.



- **Launch Authority -** An enclave on an ASIC must be authorized by a **launch authority.**
- **MRENCLAVE -** Recorded, cryptographic hash with information about the enclaves code, data, and meta-data. It represents a specific enclave identity.
- **MRSIGNER -** A signature by the enclaves **sealing authority**.

Keys

Different cryptographic keys are needed for the system. The **Root Provisioning Key** (RPK) and **Root Sealing Key** (RSK) exist in hardware while symmetric keys for sealing and reporting are derived and unique for the instance.

Sealing

Process of encrypted enclave secretes for persistent storage. A **Private Sealing Key** is used and is unique to that particular platform and enclave.

Attestation

SGX can either attestate locally on the CPU or remotely.

Use Cases

Some use cases for the system is to protect **machine-learning models** or to implement **blockchain.**

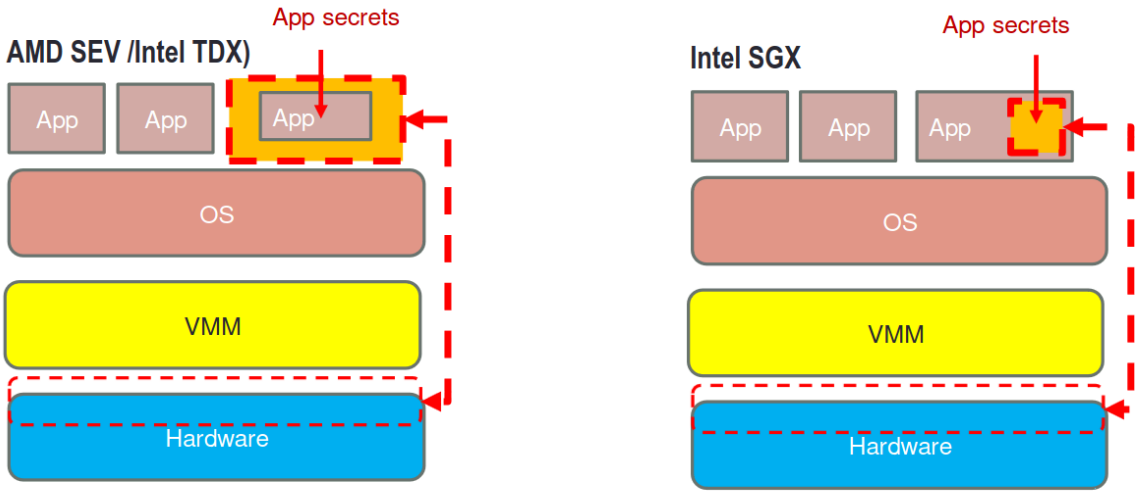
Intel TDX

It is a system that creates a trusted execution environment where a whole VM can be run, in contrast to SGX that primarily protects parts of applications.

AMD SEV-(SNP)

- **SEV -** Secure Encrypted Virtualization
- **SEV-SNP -** Secure Nested Paging

Controls whether a page is private or shared using an **enCrypted bit** (C-bit). Encrypted pages can only be used by the VM. It requires no changes of applications to be able to run and can be used by already virtualized systems.



General Problem of TEEs

Side channel attacks. Information can be deduced from power consumption, timing, electromagnetic radiation, etc. ASICs are very complex and are often not designed to deal with these threats.

Crypto Primitives

- **Cryptographic Hash -** Fixed bit **signature** or **digest** of a message.

- **Hash-based Message Authentication Code (HMAC) -** Keyed hash for providing integrity/authenticity. A cryptographic key is combined with the message before computing.
- **Symmetric Cryptography -** Shared secret keys.
- **Public key/asymmetric cryptography -** Public/private pair of keys.

Trusted Computing Technologies

- **Trusted Execution Environment -** Hardware features allowing strong isolation between trusted and non-trusted CPU zones. ARM TrustZone, Intel SGX, AMD SEV.
- **Secure Element -** Standalone tamper-resistant hardware running its own software. Smart cards, SIMs. Secrets **never** leave the chip.
- **Trusted Platform Module (TPM) -** A secure element in modern PCs.

Trusted Platform Module (TPM)

Features

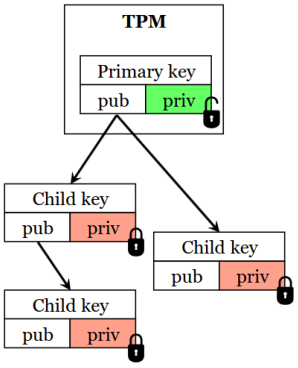
- Built-in crypto operations such as: secure random number generator, symmetric and asymmetric ciphers, hash functions, HMAC, digital signatures, etc.
- Volatile RAM that is lost in reboots.
- Nonvolatile storage.
- Platform Configuration Registers (PCRs), used to record **measurements** of machine state.

Keys

Safe storage of keys is one of the TPMs core functions. The **private part** of a key never leaves the TPM **unencrypted.** It can however be stored outside while encrypted.

Key Wrapping

The **primary keys** always reside in the TPM. These keys can wrap **child keys** that can wrap keys of their own. If a key wraps another key it is called a **storage key** and is done by encrypting the private part of the key. The **primary key** becomes the **root of trust**.



Key Hierarchies

Each hierarchy has a **seed** (embedded secret) used to derive the **primary keys**. The primary keys are HMAC:ed based on a key derived from the seed.

- **Endorsement Hierarchy -** Seed set during manufacture. Cannot be changed or reset.
- **Platform Hierarchy -** Seed set by manufacturer during initial setup. Locked for the user.
- **Owner Hierarchy -** Seed can be reset by the owner or user of the machine.
- **Null Hierarchy -** Seed reset on every reboot.

Key Creation

The Key Derivation Function is deterministic. Keys can therefore be re-generated (using the unique field) instead of directly stored on the TPM.

Key Parameters

- **Unrestricted Decryption Keys -** Can be used to encrypt/decrypt arbitrary data.
- **Restricted Decryption Keys -** Can only decrypt data blobs created by the TPM that will **reside inside** the TPM after decryption. Storage keys are always restricted.
- **Unrestricted Signing Keys -** Can be used to sign arbitrary data

Typical Setup

The **endorsement key** is a primary key on the endorsement hierarchy. Its authenticity is certified by a certificate stored in NVRAM. It is used to verify that the TPM is legit. Two important keys in the owner hierarchy are the **storage root and attestation identity key**, used to store and sign data.

Platform Configuration Registers (PCRs)

Initialized to zero on every reboot. Has two operations: **read** and **extend**.

Extend is used to record measurements (hash digest of something) to a hash chain. The old hash is used to calculate the new one. Different **banks** are used for different hash functions, with 24 PCRs per bank.

Measured Boot

PCRs can be used to **seal** data/keys to a certain **system state**, and for verifying system integrity.

During a **measured boot**, each element in the boot process measures its successors code and data before handing off execution. Different PCRs are used to measure different parts.

The **quoting** mechanism is used to securely send PCR values to a remote verifier. A set of PCRs and a nonce are signed with the **restricted signing key**. Since everything happens within the TPM, it is secure from **rootkits**.

Authorization

Used to prevent unintended access to keys and data.

- **Hierarchy Authorization -** Protects whole hierarchy using a password
- **Key Authorization -** Protects access to single keys.

Instead of passwords, **HMAC authorization (**shared secret + nonce**)** and **policy authorization (**combinations of biometrics, smartcards, with requirements on PCR values**)** can be used.

A **brute force protection** is also present. The TPM will enter locked-out mode on too many failed attempts.

Sealing

Lock a key to a **specific system state.** The key can only be unsealed if the system is in a, for example, trusted state. It uses the **policy authorization** functionality.

Malware

Malware Types

- **Spyware -** Steals sensitive information.
- **Adware -** Litters e.g. browser with ads.
- **Botnet Clients -** Turns victims device into a botnet client. Can be used for DDoS attacks.
- **Cryptojackers -** Use infected devices to mine crypto.
- **Ransomware -** Encrypts all files and requires payment to restore them.
- **Droppers -** Executables designed to drop other malware on the system.
- **Remote Access Tools (RATs) -** Provides remote back door access.
- **Advanced Persistent Threats (APTs) -** Advanced malware designed to evade detection for a long time.

Methods of Infection

Statistics show that 90% of malware is delivered via email. **Drive-by-downloads** is also quite common, a device is automatically infected when visiting a malicious website.

Viruses

Viruses are the earliest malware form. They need a **host** program to function. When run, they will insert their own code into other executables. They are too simple to support “useful” functionality and are therefore very uncommon today.

Worms

Standalone programs that can automatically spread.They usually exploit network protocol vulnerabilities. Modern systems have hardened their default configuration to avoid **automatically** exploitable flaws.

Trojans

Malware that poses as useful software to trick the user into running it. In practice, most software that isn't a virus or worm. Often performed using an **exploit kit.**

Exploit Kit

Web app specifically designed to infect visitors. They usually use old exploits and take advantage of users who don't update their browsers.

1. Attackers get ad distributors to show malicious ads or they hack a legitimate website.
2. Visitors of the affected web page are redirected to the EK landing page.
3. EK uses the **user agent** to find information about the OS and browser version.
4. Exploit ruins in victims browser.

Antivirus Software

Good for protecting against variants of **known** malicious programs. Protects mostly against **non-targeted** attacks. Newly created malware or malware designed for evasion usually slip through. They are useless for detecting directed APTs.

Malware Detection

Antivirus programs match files against **signatures** and **heuristics**. The exact inner workings are kept secret. Databases are usually updated several times a day. Malware others often **obfuscate** the code to increase the chance of misclassification.

- **False Negative Rate (FNR) -** Determines how many malwares are missed. Must be reasonably low.
- **False Positive Rate (FPR) -** Determines how often software is mistaken as malicious. Must be **extremely** low. File system files mistaken for malware can have disastrous effects.

The Cat-and-Mouse Game

Signatures use simple string matching at fixed offsets in a file. Hashes over the entire file are also used. Still the main line of defense in many AV products. **Polymorphism** is a simple **countermeasure,** just change a few bytes and throw off the offsets.

AV software has evolved to use more complex signatures and **heuristics.** Signatures can be format aware, e.g. headers are parsed individually. It increases the risk of attacks against the AV software itself.

Fuzzy hashning

A hash function with high collision rate, opposite of cryptographic hashes. H(x) and H(y) should be proportional to the difference between x and y.

Heuristic Matching

Used to check software for “general suspiciousness”. Often used as a pre-filter to determine if a file should be scanned further. Examples are malformed headers, use of obfuscation, etc.
A **countermeasure** is **packing**, wrapping a compressed or encrypted copy of the malicious executable inside another executable.

Polymorphism and Metamorphism

- **Polymorphism -** Transform without changing the code (Appending data, packing, encrypting).
- **Metamorphism -** Create **syntactic** changes while retaining **semantics** (change registers, control flow).

Static Unpacking and Emulation

For simple packers with known functionality, their payload can be **statically unpacked**. Most AV products use emulation to run suspicious binaries.
Malware countermeasures:

- **Emulator Fingerprinting -** Perfect emulation is not possible in practice, malware can detect it is simulated.
- **Malware Creation Kits** - Allows easy creation of new variants, used to overwhelm AV companies.

Cloud-based Detection

Send samples for analysis in the cloud. Threat signatures can be updated in real time and allows for more **expensive** analysis.

Evading Antivirus Software

Possible with moderate effort. Signature based detection is still most common, systematically modify different parts of the binary until it is no longer detected. **Heuristic detection** of entire families is more common due to ease of creating new malware. Evade through black-box testing like above or manual reverse-engineering of AV.

Mobile Malware Specific Challenges

- Personal info and privacy (banking, personal photos, etc).
- Widespread access to networks.
- Less computation power, limited capabilities for on-device detection.
- Almost exclusively trojans.
- Most trust is moved to app stores.
- Strong isolation between apps makes it hard for 3rd AV to detect.

IOS Security Model

Startup and updates are authorized. Files are encrypted with strong encryption keys. Applications run in sandboxes. Before an app is released on the store, it goes through manual testing and static analysis.

Android Security Model

Each app is assed a unique user ID at installation and is run as a unique user. App can only access its own files and world-accessible resources. Further access is defined in **androidmanifest.xml**.

Android Vetting Process

More lenient compared to iOS. Apps are dynamically tested with **Bouncer**. It attempts to run different code paths by interacting with the app in a simulator. However, it may be feasible to fingerprint Bouncer and detect while software is running in it.

Mobile Malware Detection

- **Signature-based Techniques -** Match specific strings or patterns in the byte code.
- **Permission-based Techniques -** Analyze the requested permissions.
- **David Bytecode-based Techniques -** Analyze the bytecode.
- **Dynamic Behaviour Analysis -** Sequence of system calls, accessed files.

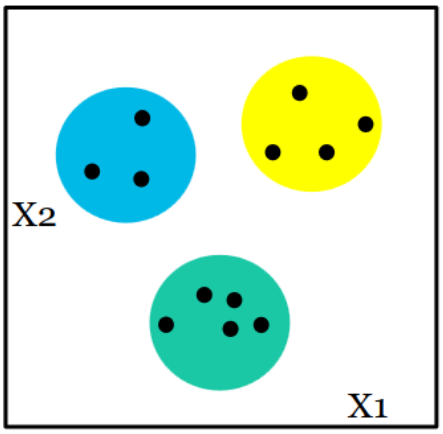
Obfuscation

- **Identifier Renaming -** Replace identifiers (variable or method names) with meaningless names.
- **String Encryption -** Replace constant string with encrypted forms and decrypt them on the fly.
- **Control-flow Obfuscation -** Inject dead code, re-order statements

Machine Learning for Malware Analysis

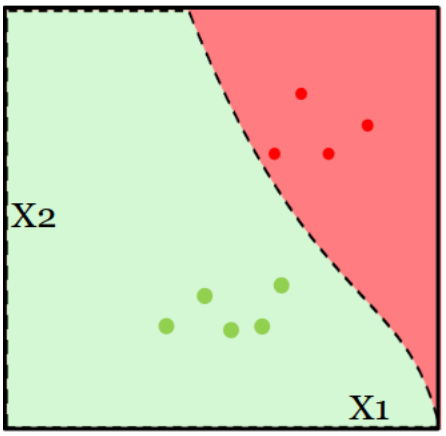
Unsupervised Learning

Given features **X**, discover the structure of the data. Split data into similar groups (group malware into families).



Supervised Learning

Having both **X** and y (colors in the model, benign and malware), try to find their relation.



Classification vs. Regression

- **Classification -** When **y** is a **categorical variable (**e.g. malware or benign).
- **Regression - y** is a **continuous** variable (e.g. probability of belonging to a specific family).

ML-based Malware Detection Procedure

1. Collect the training data.
2. Extract features from the training data.
3. Train the model.
4. Test the model.

Collect Training Data

Dataset should be representative of real-world malware.

Ex. Don't use benign apps where all have a bigger size than the malware. The model might classify all small apps as malicious.

Extracting Features

The extracted features should be relevant. Header values of executables for PC, permissions for mobile, obfuscation status for both, etc. Features that are similar for all types of apps can be removed.

Training

Parameters are optimized using the training data. Based on a particular metric, usually the classification or regression error.

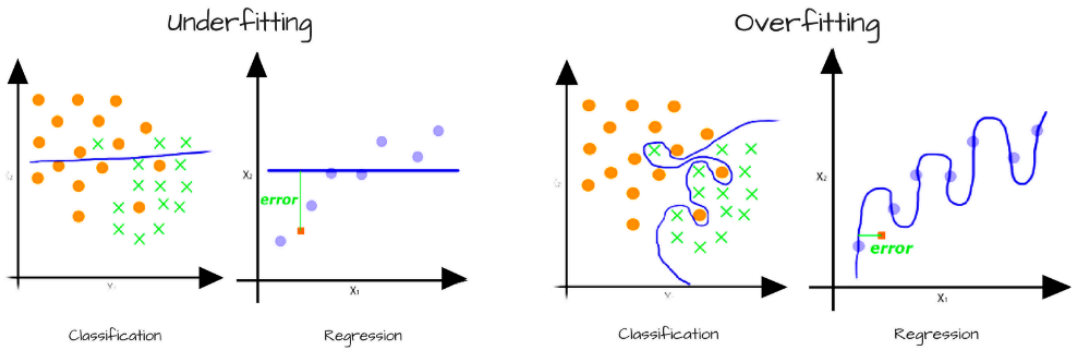
Testing

Discover whether it can generalize unseen data or if it has just memorized the training data. Training and testing errors are different and must not overlap, otherwise the testing will be biased.

Under- and Overfitting

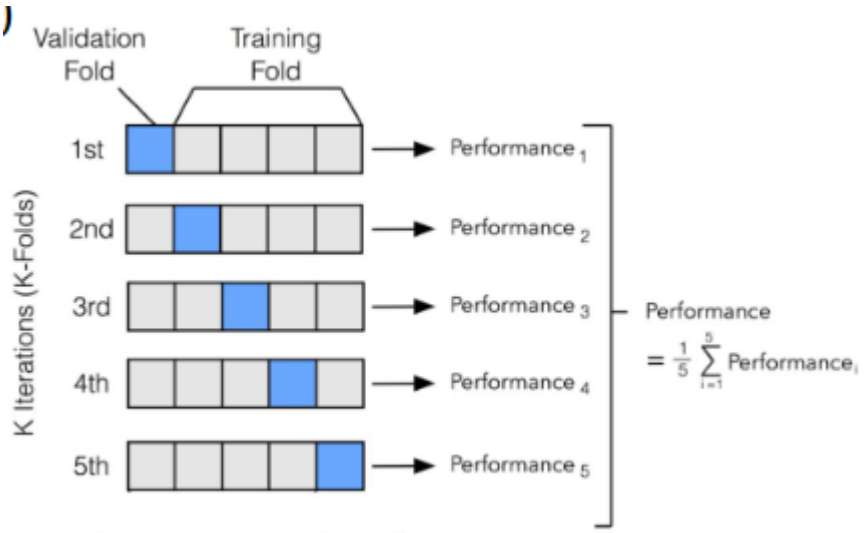
- **Underfitting -** Model is unable to obtain low error. Model might be too simple to accurately reflect the training data.
- **Overfitting -** Training error is small, but not testing error. Model might have **too many** parameters.

Solution: Adjust model complexity to minimize error. Must algorithms have tunable **hyperparameters** (number of hidden layers, maximum depth, etc). Test different combinations of these.



Cross-Validation

Each observation in the dataset can be tested.



Imbalance datasets

Malware datasets are often **imbalanced.**

Ex. 99% of samples are benign. Naive malware detection classifier classifies all samples as benign and reaches 99% accuracy. Accuracy is only meaningful if there is a 50/50 distribution.

Performance Measures

TDDE62 - Malware Defense II 47

Performance Measures

- Accuracy

$$\frac{TP+TN}{TP+FP+TN+FN}$$

- Recall (Sensitivity)

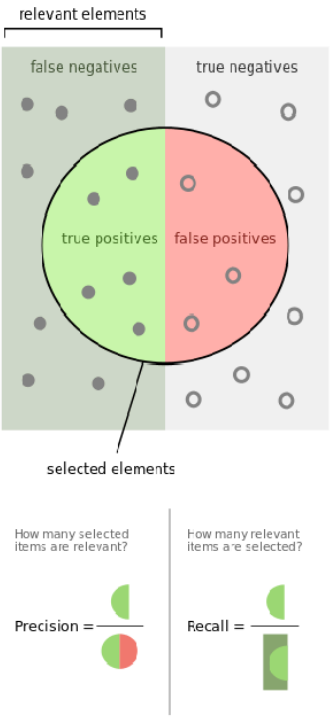
$$\frac{TP}{TP+FN}$$

- Precision

$$\frac{TP}{TP+FP}$$

- F-score : F-Score is the weighted average of Precision and Recall.

$$\frac{2 * precision * recall}{precision+recall}$$



Dataset Quality

Having representative data is crucial. Training can not be done on singular devices and is done in the cloud.