SWS2 Summary

19. Februar, 2026; rev. 4. August 2026

Linda Riesen (rieselin)

Inhaltsverzeichnis

1 Vorlesung 01

1.1 Grouping of Security Controls

Information Security Management System (ISMS) Overview / Checklist, no direct implementation

- **Information security properties**: Which characteristics of information does the control help to preserve [Confidentiality, Integrity, Availability]
- **Category**: What it concerns [people, physical objects, technology, other (organizational)]
- **Function**: The function within the ISMS [identify, protect, detect, respond, recover]

1.2 CIS Critical Security Controls

Best-practice guidelines, helps with prioritization

Abbildung 1: CIS overview

IG1 - Basic Cyber Hygiene

- Safeguards that can be implemented even with **limited cyber security expertise**.
- Designed to thwart general, non-targeted attacks
- Designed for use with Common Off-The-Shelf (COTS) hardware and software in small businesses or home offices.
- Used to protect IT resources and personnel.
- ***used when:*** limited IT and cyber security skills, primary interest: maintaining business operations, sensitivity of data is low (employee and financial info)

IG2 (includes IG1)

- Helps security teams cope with increasing operational complexity
- Often requires enterprise-grade technology and requires specific cybersecurity expertise
- ***Used when:*** multiple departments with different risk profiles, must adhere to compliance regulations, retention and processing of sensitive customer / company data, major concern: loss of public trust

IG3 (includes IG1 and IG2)

- Requires security experts from different security areas
- e.g., risk management, penetration testing, application security.
- Protective measures help against targeted attacks by sophisticated adversaries
- Protective measures help reduce the impact of zero-day attacks
- ***Used When***: Attacks can cause significant harm to public, availability of services, confidentiality and integrity of sensitive data = heart of business, retention and handling of sensitive information/functions subject to regulatory oversight and compliance

1.3 Security Measurement

Security:

- Audit - A systematic evaluation measuring how well (parts) of an information system conforms to a set of established criteria
- Perform a penetration test

- Using measures for the performance and effectiveness of an ISMS

Risk:

- Estimate the risk using something like: Risk = Likelihood x Impact
- Estimation of likelihood is very difficult

Measure comparatively only, with goal of reducing uncertainty about security, measure progress and maturity

2 Vorlesung 02

Threat Landscape: A collection of threats, threat actors (threat agents) and observed trends

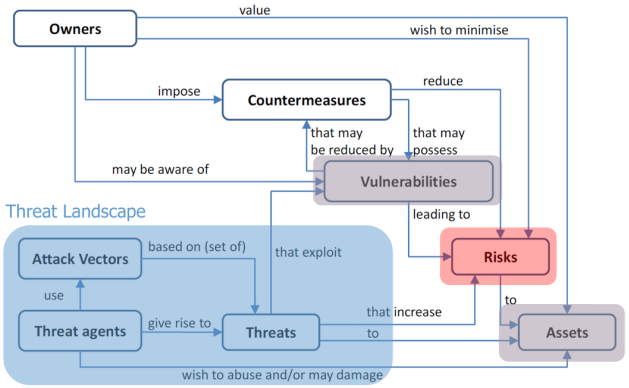


Abbildung 2: Elements of Risk

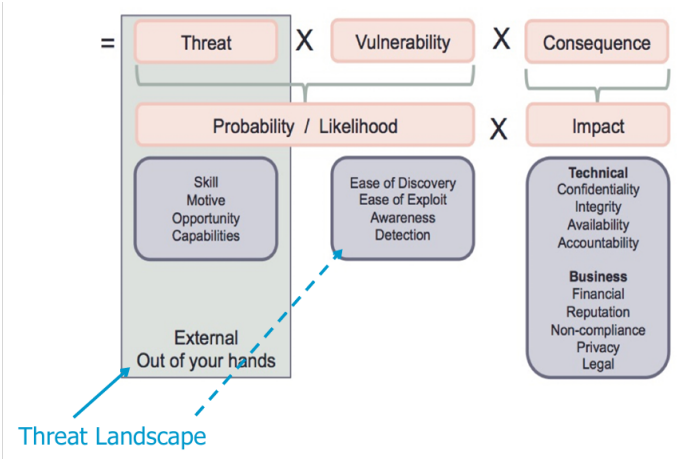


Abbildung 3: Threat Landscape for Risk

2.1 APT (Advanced Persistent Threat)

Advanced: Use of advanced technologies and techniques:

- Penetrate existing defenses (e.g., using vulnerabilities known to the attacker only
- Operators develop more advanced tools as required
- Multiple targeting methods, tools, and techniques to reach and compromise a target and maintain access to it

Persistent

- Maintain long-term access
- Keep trying until it gets in / achieves its goals (e.g. Lateral movement wait for suitable vulnerability)
- Hides from detection until it attains its objective

Threat

- Coordinated human actions, not mindless and automated pieces of code
- Operators have a specific objective, are skilled, motivated, organized and well-funded

2.2 Cyber Kill Chain

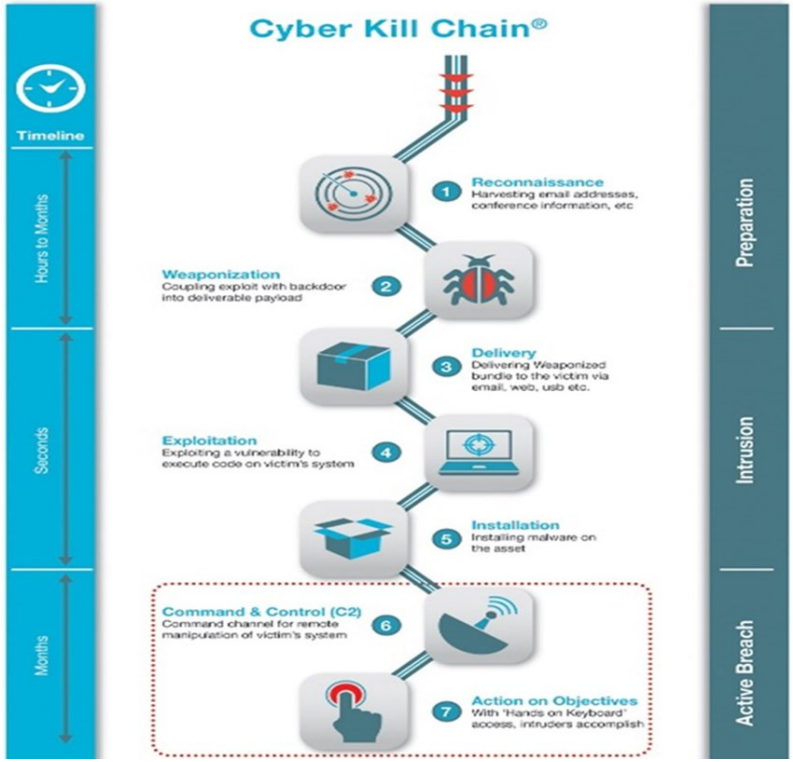


Abbildung 4: Cyber Kill Chain

3 Vorlesung 04

Challenge: Data overload => log management, making sense of security-relevant log data: use Security Information and Event Management System (SIEM)

3.1 SIEM

Gathering, aggregating, correlating, analysing, prioritizing and presenting information from disparate systems

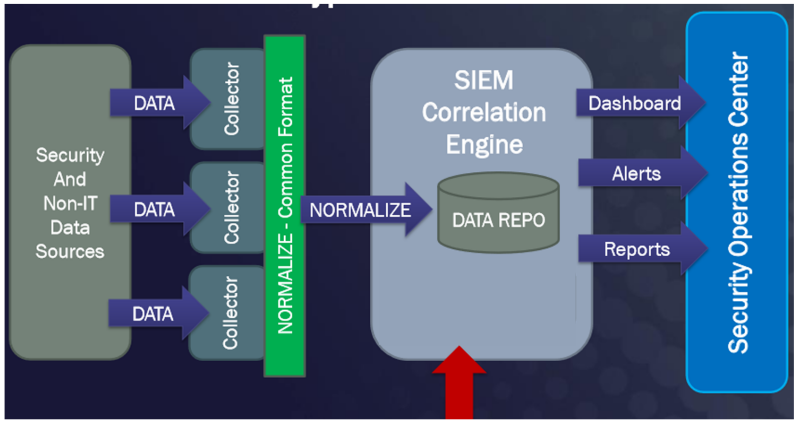


Abbildung 5: SIEM Architecture

3.1.1 SIEM Components

- Asset Directory: asset value for risk calc / priorization
- Vuln Scanner: Alerts for hosts + info about its vulns

Log Sources:

- Firewall / Auth / Service / System
- Host / Network / Serice avail Logs
- Antivirus System
- Honeypots
- Host & network based Intrusion detection system (HIDS / NIDS)

3.1.2 Host- / Network-Based: Intrusion Detection Systems / Intrusion Prevention Systems

Intrusion Detection Systems (IDS) monitor network and/or system activity to detect attacks, attack attempts and general abuse (e.g. file mods, sys calls, app logs, running processes, network data, network traffic) **Intrusion Prevention**

Systems (IPS) add a component to prevent/mitigate detected attacks, attack attempts and general abuse. (achtung: false positive, damage done when detected) (e.g. restore files from storage, suspend user, quarantine files/exes, updating firewall settings, killing process, shutdown systems Quality is measured by False positives and false negatives

3.1.3 SIEM Task: Event Correlation

Takes multiple isolated events, combining them into a single relevant security incident – "If this, followed by that, take some action"

3.1.4 SIEM Beispiel AlienVault

1. Receive Normalized Events (also from vuln scanner results)

2. Evaluate Event against policies (conditions = should be processed by policy?) + consequences = what will happen?)

3. Risk assessment: risk = asset * (reliability * priority / 25) [asset = asset value (0-5), priority = event type (0-5), reliability value = is attack? (0-10)] => alarm when risk >=1

4. Event correlation and cross-correlation (correlation rule = same data source? (e.g. same IP) cross-correlation = different data sources, correlation directive = several rules => decide connect events?)

Goal: Threat Use case a logical, actionable and reportable component

3.2 Indicators

Some indicators are much better than others - they cause more pain to the adversaries if they must tune an attack to evade such an indicator

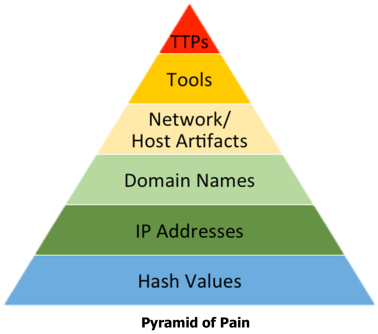


Abbildung 6: Pyramid of Pain

- **Hashes**: hashes that correspond to specific suspicious/malicious files => high accuracy, but very easy to flip single bit to avoid

- **IP Addresses:** (set of) ip address(es)/subnet => medium high accuracy (same ip used by legitimate people unlikely but possible), easy to avoid (anon proxy services, fast fluxing)

- **Domain Name**: (what its actually called and where / who is it registered) => medium easy to avoid (often anon registrations, lax standards)

- **Network and Host Artifacts**: Network: uri patterns, C2 info in netw protocols, distinctive http user agent, smtp mailer values, ... Host: registry keys/values known to be created by malware, files / directories in certain places with certain names, names / descriptions of malicious service => medium-high accuracy, medium to avoid => need to figure out why they are detected + how to fix

- **Tools**: multiple host / network artifacts of tool used by attacker hard to fix => attacker must create new tool

- **Tactics, Techniques and Procedures (TTPs)**: How the adversary goes about accomplishing their mission => operate on adversary behaviour (not against tool) => hard to fix : need to lern new behaviour to avoid

4 Vorlesung 05

4.1 Security Testing Methods

Purpose	Identify well-known vulnerabilities Compliance (e.g., GDPR, HIPAA, PCI DSS,…)
Attacker Goal	-
Assets in Scope	Source code, applications, systems, entire infrastructures
Result	Potential vulnerabilities and a generic risk rating
Method	Fully automated (software / appliance / SaaS) • e.g., OpenVAS (infrastructure), LGTM.com (source code)
Requirement	Mature vulnerability management process • resources/skills to verify, prioritize and fix them
Frequency	Continuous • Vulnerability signatures and assets are not static

Abbildung 7: Vulnerability Scanning

Purpose	Find as many vulnerabilities as possible and get fixing-advice Compliance[1] (e.g., GDPR, HIPAA, PCI DSS,…)
Attacker Goal	Efficiently find many vulnerabilities • Mainly «easy to find» vulnerabilities
Assets in scope	One / a few • e.g., a single application, system or service
Result	Verified vulnerabilities and their risk rating plus advice how to fix them
Method	(Semi-)automated tools/scanners and standardized testing procedures (for web-apps, e.g., OWASP testing guide)
Requirement	Mature vulnerability management processes, test environment[2]
Frequency	1 – 4 times a year Once per new version/release

Abbildung 8: Classical Pentesting

Purpose	Test the organization's detection and response cababilities
Attacker Goal	Find a way to achieve a stated goal and don't get caught trying • e.g., steal customer data, become domain admin, install spyware on a designated asset, …
Assets in scope	Many / All [physical, human, cyber]
Result	Goal achieved (yes/no) and how it was achieved
Method	Goal and scope dependent, social engineering is often part of it
Requirement	Mature security program – i.e., vulnerability management, security testing & monitoring, and incident handling in place
Frequency	Periodically (?)

Abbildung 9: Red Team Testing

Purpose	Improve security posture • focus on detective/preventive controls • identify malicious actors in environment
Attacker Goal	Help the blue[1] team to catch them / learn from them
Assets in scope	Predetermined systems/employees
Result	Improvements to security controls (e.g., new detection rules) and plan to resolve issues that could not be addressed during the exercise
Method	Simulate (relevant) attack patterns and scenarios
Requirement	Interfaces and collaboration between incident response, security tooling, network engineers and vulnerability management work well
Frequency	Periodically (e.g., once per quarter)

Abbildung 10: Purple Team Testing

Purpose	Improve security posture: • Focus on detective/preventive controls • Use a set of «prominent» attack vectors (e.g., MITRE ATT&CK) • Usually, multi-step attacks along the cyber kill chain
Attacker Goal	-
Assets in scope	Any* (determined by the selected attack types/scripts)
Result	Report on resilience against the scripted cyber attacks
Method	Automated testing platform (e.g., SaaS) implementing scripted multi-stage attacks (e.g., implementing the steps in the cyber-kill chain)
Requirement	Same as for purple-team testing, big-budget blue-team
Frequency	Continuous

Abbildung 11: Breach & Attack Simulation

Purpose	Find vulnerabilities and pay only if vulnerabilities are found • Public – anyone can participate • Private – selection process / by invitation only
Attacker Goal	Earn money (sometimes also «prestige»)
Assets in scope	Often a single application/service/system
Result	Vulnerability reports
Method	According to the program • Specifies what is allowed (or not allowed)
Requirement	Appropriate legal framework, willingness to «go public», accept the risks that you do not know/trust the testers,...
Frequency	Continuous

Abbildung 12: Bug Bounty Program

	Vulnerability Scanning	Penetration Testing	Red Teaming	Purple Teaming	Bug Bounty	Breach & Attack Simulation
Finding known vulnerabilities (in 3rd-pary libs/apps/systems before or during operation)	x	x			(x)	
Finding vulnerabilities ([self-developed] apps/systems before or during operation)	(x)[1]	x			x	
Improve product security	(x)[1]	(x)[2]			x	
Improve defensive measures				x		x
Test defenses, incident response			x	(x)		x
Educate/train blue team			(x)	x		(x)
Compliance	x	(x)				

[1] Specialized scanners. Can detect vulnerabilities that can be found with "simple" testing patterns (e.g., web-application scanners looking for SQL-injection or XSS vulnerabilities) under the condition that the vulnerable item/action in the application/system can be found/triggered (e.g., web-form supplying the data) by the scanner.

[2] Pen-testing can help here, but activities earlier in the product development cycle are far more effective and should be prioritized (e.g., threat modelling, security architecture review, code review and static code analysis)

Note: For security testing methods it might be relevant whether the system under test is **internal**, **public** or **hosted on third party infrastructure** (in the cloud). For example, public bug bounty programs are usually not an option for internal systems.

Abbildung 13: When to apply which Method

4.2 Penetration Testing Methodologies

OSSTMM - Open Source Security Testing Methodology Manual

- Methodology for testing security systems for everything (From guards and locked doors to mobile communication and satellites)
- Includes a mathematical model to determine the security level and make systems/tests comparable

OWASP Testing Guide (V 4.0)

- Security testing of web applications
- Includes hands-on advice and references for tools

The NIST SP 800-115 Technical Guide to Information Security Testing & Assessment

- Covers process typically applied in pentesting (planning, detailed analysis, dealing with validation of discovered concerns)
- Appendix lists some Linux-tools for typical pentesting tasks

Penetration Testing Execution Standard (PTES) Explains the basic principles and steps required to conduct a penetration test Effort of people from the industry Incomplete/unfinished and with respect to technical guidance outdated

A lot of other resources (checklists, guides, training material,...) are available on the Internet

- E.g., from the SANS Institute, offering a series of courses in this domain

4.2.1 Common Test Methods

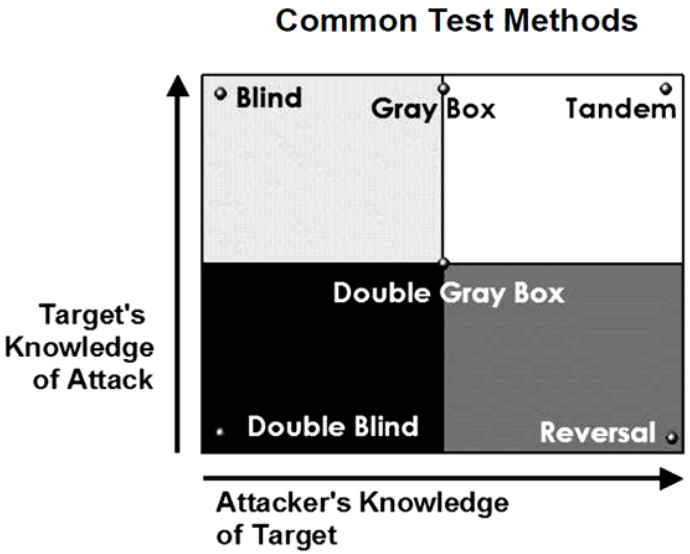


Abbildung 14: Common Test Methods (OSSTMM)

5 Intelligence Gathering

Intelligence gathering (or footprinting) is performing reconnaissance against a target to gather as much information as possible (when open source only => OSINT)

5.1 Intelligence Gathering Levels

- **Level 1** – Compliance driven“
  - Information can be collected almost entirely by automated tools
- **Level 2** – "Best practice“
  - Mix of automated tools and some manual analysis
  - A good understanding of the business, including information such as physical location, business relationships, organization chart, etc.
- **Level 3** – S̈tate Sponsored“
  - Automated tools and hours of in-depth and thorough manual analysis
  - Cultivating relationships on social networks, profiling of all key personalities of the company, in depth study of systems and technologies they use, . . .
  - Most advanced, full-scope (red team)

5.2 Intelligence Gathering Method

- **Passive** – Using data from third parties that is already there
  - Access to information cannot be detected/tracked by the target (E.g., use of Shodan instead of an active network scan)
  - Can be quite limited and information is likely to be not up-to-date
- **Semi-passive** – Using data gathered using legitimate behaviour
  - Query only sources that are there to be queried and stick to the protocol (Don’ts: In-depth reverse lookups, brute force DNS requests, searching for “unpublished” servers or directories, network level port scans, using crawlers or actively looking for)
  - Use anonymization networks or similar to hide the origin of the queries (Post mortem - Reconnaissance activities might be identified but the target shouldn’t be able to attribute the activity back to anyone)
- **Active** – Should be detected by the target
  - Most common form of information gathering (Vulnerability scanning, enumeration, profiling middle boxes (IDS, FW,. . .) with test traffic,)

5.3 Search Engine Hacking / Google Dorking

5.4 Finding Domains

Problem: There is no publicly accessible register of all domains (often you can get access to zone files)

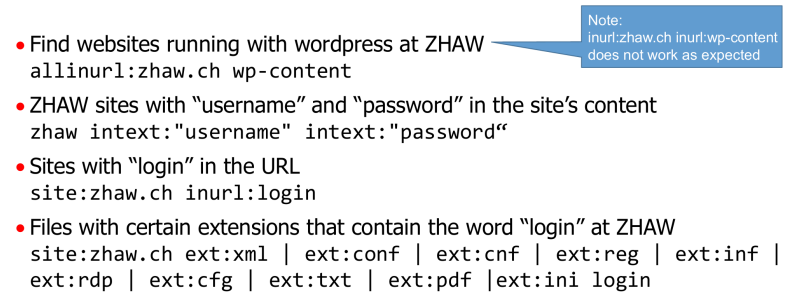


Abbildung 15: Google Hacking Google Hacking DB

- WHOIS is a protocol for querying databases storing the registered users or assignees of an Internet resource (mainly domain name, IP address block, autonomous system)
- WHOIS servers operated by regional Internet registries (RIR) can be queried to find the Internet service provider responsible for a resource -> query the provider's server
- Entries are cross-referenced: A query to ARIN for a record which belongs to RIPE returns a pointer to the RIPE WHOIS server
- Search Engines, DNS Tools, https://findsubdomains.com/, https://www.robtex.com/dns-lookup/init.zhaw.ch

5.5 Finding IP Addresses

Identify IP addresses used by the target company in case the company owns IP address blocks or their own autonomous system(s)

- DNS system: Get IPs/Hostnames of name- and mail servers (e.g. nslookup)
- WHOIS and related APIs (ICANN, IANA, RIPE,...)
- BGP Toolkit (based on BGP and other data) / Shodan / Censys (scanning based)

5.6 Scanning

Based on the previous results, the target company's networks and hosts are examined in more detail during scanning

- Determine the network structure
- Find hosts that are reachable / visible by the tester
- Analyse the hosts in detail (OS, Services on hosts)

Tools

- traceroute is the tool of choice to analyse the network structure
- nmap Host Discovery (Detection of services/versions): Nmap works by sending specially crafted network packets to a target and analyzing the responses to discover hosts, open ports, services, and operating systems; when run as root, it can use raw packets and advanced scan types (like SYN scans and OS fingerprinting) that are faster, stealthier, and more accurate than the limited TCP connect scans available to non-root users.

Switch	Technique	Description
-sS	TCP Syn	Default option, fast, relatively stealthy, reliable, also referred to as half-open scanning
-sT	TCP Connect	Uses «connect» system call of the underlying OS rather than writing raw packets. Slower than TCP syn scan, more easily detectable
-sA	TCP Ack	Used for testing firewall rulesets
-sU	UDP	UDP scan, much slower than TCP
-sN	TCP Null	TCP scan with different flags set (FIN, PSH, URG) to avoid stateless firewalls
-sF	TCP FIN	
-sX	TCP XMAS	
-sO	IP Protocol	Used to detect IP protocols supported by the target

Abbildung 16: NMAP Scan Techniques

5.7 Footprinting Defenses

Motivation: Additional attack vectors, know how to evade detection (stealth)

- Identify defensive systems: Firewalls, WAF, IDS, Anti-Virus
- Inspect banners, HTTP responses, device fingerprinting,...
- Social engineering (trick employees into providing information)
- Search engine hacking (partners, presentations, projects,...)
- Inspect banners, device fingerprints etc. (e.g., using Censys, Shodan etc.)

5.8 HUMINT (Human Intelligence

Online Accounts , Social Media Accounts, (Automated via does username exist : http://knowem.com/ http://checkusernames.com/)

6 Vorlesung 08 Pentesting 3

6.1 Threat Modelling Defender-Centric

=> used by attackers, where might they have good defenses

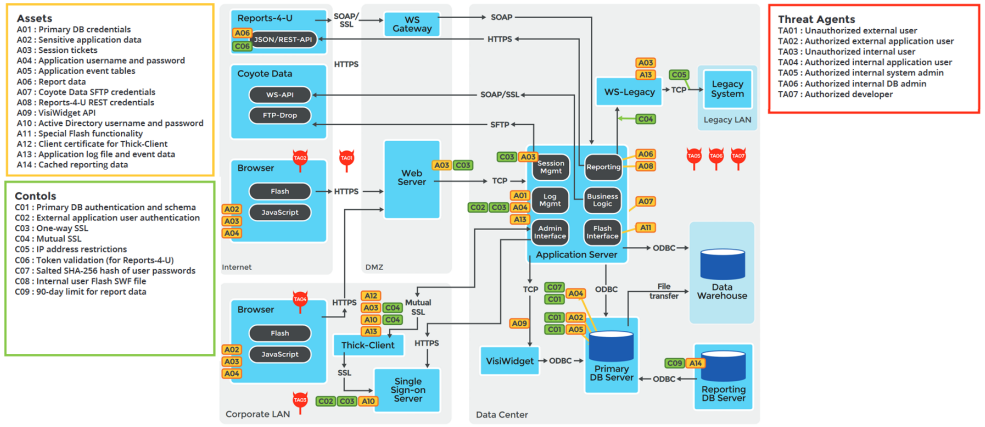


Abbildung 17: Process Flow Diagram

- **Primary asset:** Part of the system/functionality under test (in scope) (can imply compromise of secondary asset)
- **Secondary asset:** Not in scope but shared with or linked to the assets in scope (impact what threat agents to consider)

6.2 Attack Patterns

MITRE Common Attack Pattern Enumeration and Classification (CAPEC)

- Focus on application security
- Enumerates exploits against vulnerable systems
- Includes social engineering / supply chain
- Associated with Common Weakness Enumeration (CWE)
- describes attack patterns

MITRE Adversarial Tactics, Techniques & Common Knowledge (ATT&CK)

- Focus on network defense
- Based on threat intelligence and red team research
- Provides contextual understanding of malicious behavior
- Supports testing and analysis of defense options
- describes specific techniques in attacks

Attack Matrix MITRE

6.3 Vuln Analysis

discovering and confirming security issues in systems and applications which can be leveraged by an attacker (highly dependent on component that is being tested): goals: validate that exists & can't be exploited

6.3.1 Vuln Scanner

- Determine all assets reachable over the network (e.g. nmap, sublist3r)
- Determine the products/technologies they run (e.g. BuiltWith, Wappalizer)
- Match databases of known vulnerabilities with the list with the identified
- products/technologies and their versions
- Run product/technology specific recipes and/or scanners (e.g. Nessus, scanmeter.io)

Vuln Scanner problems

- New vulnerability data is added continuously: when to scan?
- Analysis based on product and version can be misleading
  - Products might be forked/copied and given new names, and version numbers => false negative
  - Products might have been patched without modifying their version => false positive
- Results might depend on time/location/user account etc.
  - E.g., scan from intranet vs. extranet with different filters/controls in-between
  - E.g., authenticated vs. unauthenticated

6.3.2 Vulnerability Analysis Active: Fuzzers

Make use of fuzzers to discover yet unknown vulnerabilities

6.3.3 Vulnerability Analysis Tracking

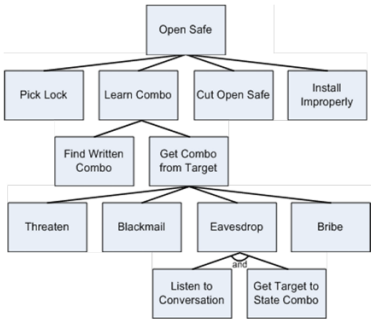


Abbildung 18: Attack Tree

6.4 Attack Vectors

Attack vector(s) can be assessed in terms of their probability to succeed and the probability of getting detected when executing the attack For a **precision strike**, choose the most promising attack vector (in terms of success probability and probability of getting detected)

7 Vorlesung 09

An exploit is a piece of **software**, a **chunk of data**, or a **sequence of commands** that takes advantage of a vulnerability Exploits cause unintended or unanticipated behavior on computer software, hardware, or something electronic

Exploit Classification (additionally to classification by action taken)

- A **local exploit** is targeting a vulnerability on the system on which it is executed, for example gain admin privileges
- A **remote exploit** is targeting a vulnerability on a remote system
- A **client-side exploit** targets a vulnerability of a client software

  - They often require the “help” of a user to work (open a malicious file, access a specific website etc.)
  - Drive-by attacks might use client-side exploits which do not require the help of a user (e.g., exploits triggered by malicious ads)

- A **server-side exploit** targets a server and does not require the “help” of a user to work

- **0-day/zero-day exploit:** An exploit for a vulnerability that was not publicly reported, announced or is unaddressed when the exploit becomes active

  - "Day Zero“: Day on which the vendor (or public) learns of the vulnerability

7.1 Repeat SWS1: Memory

- ASLR – Address Space Randomization

- Non-executable memory: Marks memory as non-executable at the page level (4 kB or greater) (NX (Linux), DEP (Windows), XN (ARM))

7.2 Return Oriented Programming (ROP)

ROP is a technique that allows to execute arbitrary code on the target without submitting/injecting any code, makes use of the code of the application itself, Pieces of code that are useful for ROP are called **gadgets**, ROP is a control-flow hijacking attack

8 Malware I

- maliciousness or harmfulness

- the ability to perform actions without the user’s consent or knowledge

8.1 Malware Classification

- By type: Based on the main features implemented by the malware (e.g., replication method): Terms frequently used in mass media, for example bot, virus or worm

- By behavior: Based on the exhibited behavior: e.g. “information stealer”

- By family/lineage: Focus on the authorship and the lineage / evolution: e.g. by analyzing the code-case for shared or similar code

8.2 Malware Types

- Trojan Horse (malware samples that employ some mechanism to conceal their true identity)

- Backdoor & Remote Access Tools (RAT): backdoor functionality provides some level of interactive 1-to-1 access to a compromised system, A Remote Access Tool is frequently used to describe a backdoor having a high level of interactive functionality and providing expansive access to the target’s host.

- Downloader: Downloader functionality provides a tool with a specific feature to download other tools (initial phase often delivers light-weight tool => breaking up attacks reduces detectability and facilitates surgical retooling)

- Dropper: In a multi-stage attack, one tool may need to write/drop another one to disk and execute it. (Code must first be extracted and stored in an OS-compatible, native container (e.g., EXE file).) (downloaders are usually also droppers but not vice versa)

- Bot / Botnets: A number of interconnected computers (=nodes) running a bot (The bot is coordinating their actions automatically or by command and control (C2/C&C) from the “operator”) (similar to backdoor but 1:m)

- Spyware (Monitor): record user activity or the environment employs monitor functionality, The recorded data is delivered back to the adversary

- Information Stealer (Spyware): automate the process of stealing information (searches data using a hard-coded or network-provided collection plan), similar to monitor but relevant data only)
- Scareware / Adware: Entices or frequently scares the user into taking specific actions (e.g. fake anti virus)
- Ransomware: social-engineer the target into paying a fee or some other activity (Typically, ransomware encrypts files on the local system and asks for money to decrypt them)
- Virus and Worm

	Virus	Worm
What does it do?	Inserts malicious code into a program or files that can contain «executable code» (e.g., macros, scripts,...)	Exploits a vulnerability in an application, service or operating system*
How does it spread to other computers?	Users transfer infected files to other computers. They are then executed/ used there	Uses a network to travel from one computer to another
Does it infect a file?	Yes	No
Does it spread with help of humans?	Yes	No
Example	Sality	WannaCry

Abbildung 19: Virus and Worm

- Mailer (or Spambot): Malware with functionality to send emails
- Cryptominer / Cryptojacking – Malware with functionality to mine crypto currencies. Impact is usage of your resources for free.
- Living Off the Land - Malware that utilizes already installed software to implement some/all of its functionality (e.g., PowerShell or a legitimate remote management software)
- Fileless Malware: No stand-alone malware is installed (Malware code is only in memory, not on disk/in files, In-memory only: Exploit vulnerability in a software to inject the malware into that process from remote without any file-system access, Software that is already installed is leveraged to load malware code into memory (subset of living off the land)
- Rootkits: utilizes operating system interfaces to conceal the presence of malware (=stealth)
  - User-mode rootkits

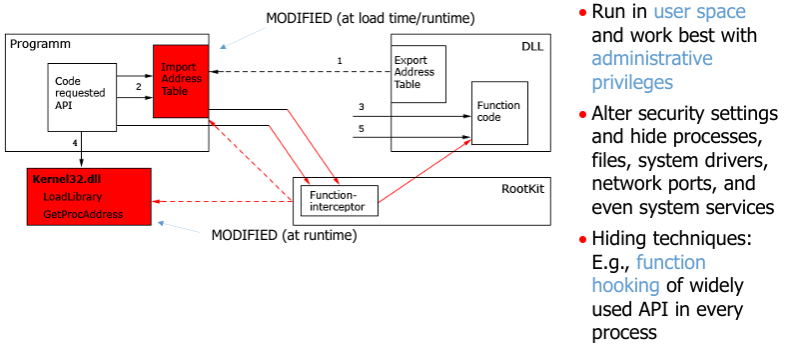


Abbildung 20: User Mode Rootkit

  - kernel-mode rootkits

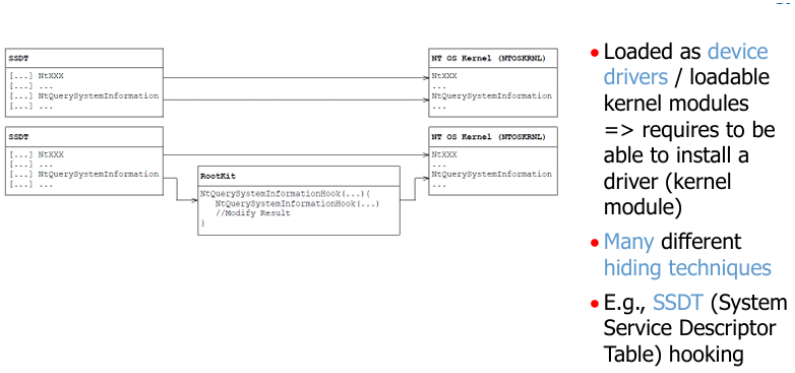


Abbildung 21: Kernel Mode Rootkit

- Bootkit: runs in real mode (and kernel mode after booting), Infects startup code (boot sector) to take control of the boot process, Can subvert the kernel and its security protections
- Hypervisor level rootkits: Exploits hardware virtualization features such as Intel VT or AMD-V, Runs in Ring -1 and hosts the target operating system as a VM, Does not have to make any modifications to the kernel of the target
- Firmware and hardware rootkits: Run on routers, network card, hard drive or system BIOS

8.3 Rootkits Guards

8.3.1 Rootkits Kernel-Mode Protections

- Windows Driver Signing, Starting with Windows 10, drivers must be submitted to, checked and signed by Microsoft
- PatchGuard (all x64 Win OS since XP) protects from SSDT-like rootkits by (trying to) preventing modifications of data structures like the SSDT
- Device Guard locks a device down so that it can only run trusted applications
  - Available on Windows 10 Enterprise or higher
  - Operating system only runs code that's signed by trusted signers
  - Who's a trusted signer is defined by your Code Integrity policy through specific hardware and security configurations

8.3.2 Detecting Rootkits

- Based on the «concept» used for hiding (E.g., look for alteration of the SSDT)
- Detection of sophisticated rootkits at ring 0 or lower is difficult
- Timing analysis as a generic detection method for rootkits
  - Hooked/modified operations show a different timing behavior
  - Baseline to compare to is the challenge
  - In case of hypervisor level rootkits, external time sources must be used

8.4 Malware Communication

- Communication consists of the communication architecture and the properties of the communication channel itself
- Important goals for malware communication:
  - Communication should be resilient vs. blocking/takedowns
  - Communication should be hard to detect and/or identify

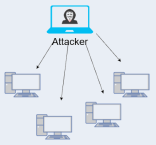
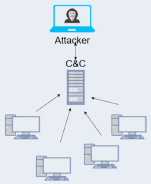
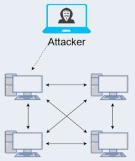
	Direct	Client-Server	P2P
Communication initiated by	attacker	malware	attacker/malware
Firewall compatibility	low	high	low-medium
Resilience / Ease of takeout	easy [hard]	easy [hard]	hard
Architecture Outline			

Abbildung 22: Communication Architectures

If the IP/domain/URL of the C2 server is known, we can take it down (e.g., arrest the attacker), block connections to it, or sinkhole it.

Common Evasion Techniques

- **Fast Flux:** Change the IP address(es) on a regular basis (e.g., Fluxer Botnet in 2015).
- **Domain Generation Algorithms (DGAs):** Change the domain name on a regular basis (e.g., Emotet malware).
- **Domain Fronting:** Malware connects to a seemingly legitimate domain.
- **Legitimate Services Abuse:** Communicate over platforms such as Dropbox or Evernote (e.g., BKDR_VERNOT.A in 2013).
- **IRC-based Botnets:** Use multiplexed communication channels (e.g., DorkBot botnet in 2015).

Countermeasures

- Partner with domain registrars where cybercriminals registered botnet-related domains (C2 servers).

8.4.1 Fast Flux – What It Is

- A single fully qualified domain name whose IP addresses are rapidly changed (TTL $<$ 300s) via DNS record updates.
- A DNS technique that hides a machine behind a constantly changing network of compromised hosts acting as proxies.
- Increases resilience of malware networks against IP-based blocking.
- One of the earliest uses was in the Storm Worm (2007).
- Often combined with peer-to-peer networking, distributed command and control, web-based load balancing, and proxy redirection.

Defenses Against Fast Flux

- Block the fully qualified domain name.
- Take down domains in collaboration with registrars or service providers.

8.4.2 Domain Generation Algorithms (DGA)

- Used to evade "bad domain" blacklists by continuously generating new domain names.
- Malware periodically generates many candidate domains that act as rendezvous points with C2 servers.
- The large number of possible domains makes it difficult for defenders to preemptively block or seize them.

Example (Simplified DGA)

```
def generate_domain(year, month, day):
    domain = ""
    for i in range(16):
        year = ((year ^ 8 * year) >> 11) ^ ((year & 0xFFFFFFF0) << 17)
        month = ((month ^ 4 * month) >> 25) ^ ((month & 0xFFFFFFF8) << 1
        day = ((day ^ (day << 13)) >> 19) ^ ((day & 0xFFFFFFFE) << 12)
        domain += chr(((year ^ month ^ day) & 25) + 97)
    return domain
```

8.4.3 Domain Fronting

- Malware communicates with its C2 server using a legitimate-looking domain.
- TLS connection (SNI) indicates a benign domain, while the actual HTTP request targets a malicious host.
- Helps bypass filtering systems that rely on visible domain names.

8.4.4 General Communication Characteristics

- Most common protocol: HTTPS (with or without additional content-level encryption).
- Advanced malware adapts communication to appear “normal”:
  - Uses common protocols (HTTP, HTTPS, SSH, etc.).
  - Uses legitimate applications/services (Dropbox, Google Docs, Citrix, etc.).
  - Mimics normal traffic patterns in timing and volume.

8.4.5 Covert and Side Channels

- **Side Channels:** Data exfiltration via indirect means even over encrypted tunnels (e.g., VPN).
  - Example: manipulating packet timing (inter-arrival patterns) to encode data.
- **DNS Covert Channel:** Data encoded within DNS queries/responses.
  - Useful when direct Internet access is restricted.
  - Example: encoding data in subdomains or IP-to-ASCII transformations.

8.4.6 Attributes of Communication Channels

- **Channel Protocol Scheduling:**
  - Static vs. dynamic
  - Periodic, random, or adaptive (“smart”)
- **Channel Protection:**
  - Plain vs. encrypted
  - Custom vs. inherited (standard protocols)
- **Channel Setup:**
  - Self-established or delegated
  - Piggybacking on existing communication channels
- **Data Transfer Type:**
  - Normal traffic vs. timing-based encoding
  - Protocol abuse (semantic or non-data fields)
  - Steganography

9 Vorlesung 11 Malware II

Can a tool detect 100% of all viruses that enter your system in form of a file?
Yes, but it has a high false positive rate (assume every file is malware)
Defense tools have faaaar more code than malware, hard to know how to detect before being able to analyse a sample of it, we don’t act until it is too late

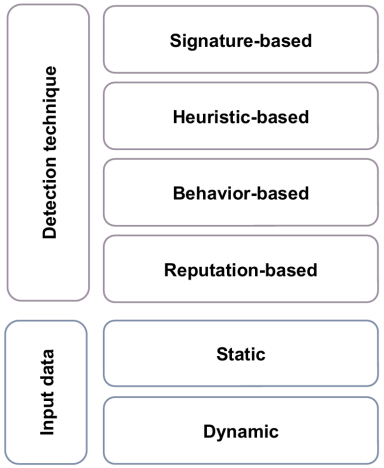


Abbildung 23: Malware Detection Engine Classification (often hybrid forms)

- Static analysis: data obtained without execution
- Dynamic analysis: Data obtained while the sample is being executed (sample can be run natively, in emulator, in VM)

- Data collection might be done at the same or at a different layer (e.g. run sample in vm, analyze from host => harder to detect for malware that its being analyzed)

9.1 Anti-Virus Technology (AV)

- Host-based and network-based
- Most of them have client- (local) and cloud-based (= instant updates, signatures hosted in cloud, usually cache exists locally with most relevant signatures) components
- Bandwidth limitation: At first, only meta data is submitted (unique identifier (hash), how the file came to system, how file behaved)
- Unknown files might be temporarily quarantined on the client machine and the file sent to the cloud for analysis

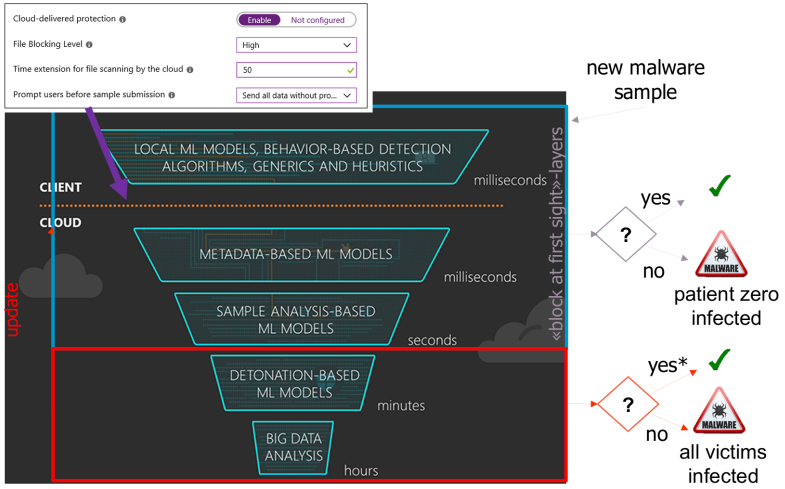


Abbildung 24: AV Architecture, Layered Approach

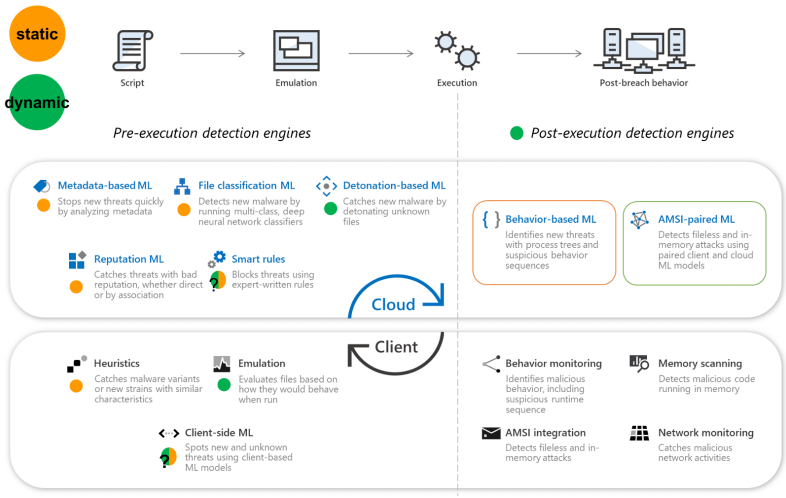


Abbildung 25: AV System Architecture – Client + Cloud

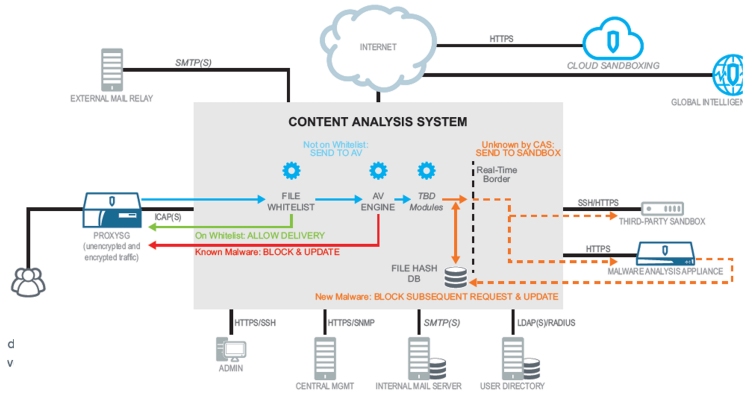


Abbildung 26: AV System Architecture – Network

9.1.1 Signatures

Traditionally, a signature captures the unique characteristics of a malware at the byte-level (hashes, fuzzy hashes, byte sequence(s)), usually supported by

whitelisting of known-good files (evtl only in context)

- Hashes: problem: changing one bit => completely different signature => today only in use to prevent analysis of known (benign/malicious)
- Fuzzy Hashes: (sped up file line-by-line comparison), Context-triggered piecewise hashes (CTPH) (e.g., ssdeep, MRS, mrsh-v2, bbHash), Other similarity digest approaches exist, for example sdhash and tlsh
  - Context Triggered Piecewise Hashes (CPTH): Segment file into pieces, hash each piece, Compile a hash from the "hashes" => context (content) determines bounds of segments (rolling hash => recovers if similar content is seen after difference) => calculate distance between hashes
  - Similarity digest hashing (sdhash): looks for statistically improbable sequences = features => finds samples where some parts are copied
  - Trendmicro locality sensitive hashing (TLSH): frequency distribution of n-grams (substrings of n bytes) => finds same building blocks

Will be improved / is being improved with machine learning

9.1.2 Heuristics

Heuristic-based engines make use of a set of rules and weighing methods written by domain experts (static or dynamic exist)

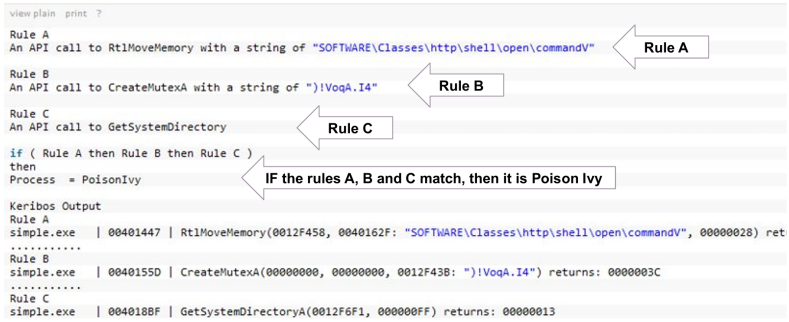


Abbildung 27: Heuristics Example

Heuristics is more general than behavioural based as it is not limited to behavioral features Heuristic engines are usually run prior to the runtime-behavioral analysis

9.1.3 Behavioral

Behavioral (sematic) detection is unaffected by changing a malware's form – it is based on what the malware does (on the host and on the network is sometimes considered separately)

Behavioral Detection Sandboxes Malware Sandbox - Captures the malicious program sample in a controlled testing environment (sandbox): Monitor all os calls, monitor network activity (e.g. Cuckoo Sandbox)

9.1.4 Reputation Based Analysis

Based on what we know about a resource (e.g., file, URL, IP): based on prevalence (file is used on many computers around the world), age (domain registration date), origin (e.g. internet + exectued=> probs bad)

9.2 Evasion

File Format

- renaming file extension => requires social engineering to rename it back and execute it, avoids blacklisting of file extensions, scanners must be able to read file format
- vulnerabilities in decoders can be exploited
- many different formats can be executed e.g. on windows

Compression

- compression is normal, hard to block
- time consuming to decompress for AV
- pw protected encrypted files often ignored by AV

Signature Evasion - Polymorphism

- Polymorphism - polymorphic code is code that uses a polymorphic engine to mutate while keeping the original function of the code (its semantics) the same
- Packers/cryptors contain a packed/encrypted part (payload) and a unpacking/decryption routine (stub) to extract+execute the payload (AVs attempt to generate stub's signatures, stub can be mutated with junk instructions / similar commands)

Signature (Heuristics/Behavior) Evasion - Metamorphism

- Metamorphism: metamorphic code is similar to polymorphic code but the whole binary including the metamorphic engine itself undergoes changes
- Translate their own binary code into an intermediate language, edit it and translate it back to machine code

Evasion Sandboxes / Emulation

- Detecting the behavioral monitoring or sandbox and behave differently (e.g. Fingerprint the environment / scan for registry entries (e.g. VM Tools))
- Outwait the behavioral monitoring by the AV solution (Difficult in case of endpoint security solution with always-on detection) e.g. Human interaction specific Implement some form of «CAPTCHA» / Time specific

10 Mobile Platform Security

Different Usage than Computers:

- carried around => often lost / stolen
- often thecentral device (wallet / 2nd auth / ...)
- loads of personal + sensitive data
- used in non-trusted Wi-Fi APs
- Apps store credentials
- easy app installation => users install all kinds
- even used by users that don't use computers => even lower sec awareness

Protection mechanisms (should be provided by phones beyond computer protections)

- difficult to load malicious apps
- apps shouldn't be able to access arbitrary data
- thiefs cannot use phone / access data on it
- users do not have admin rights

10.1 iOS Security

benefits from Apple controlling hardware and software

- Secure boot process: make sure that the original, untampered iOS kernel is run => secure boot chain
- File encryption and data protection
- Passcode and Face ID (or Touch ID)
- Keychain (used to store pw and key in FS)
- App code signing and app installation only from official App Store + nly apps signde by apple or using a certificate issued by apple can run
- Runtime security features such as sandboxing of apps with limited inter-app communication
- Permission model to grant access to general system functions

10.1.1 Secure Boot Chain

1. **BootROM** (= SecureROM): hardware contains code stored in BootROM (readonly), code is executed when device is turned on, code contained Apple Root CA, Code cannot be changed (even by Apple)

2. **iBoot**: loads, verifies (signature), runs iOS Kernel

3. **Kernel**: If everything is ok, we can trust the running iOS kernel to be the original one

10.1.2 File Encryption and Data Protection

- Every iOS device includes a hardware AES 256 crypto engine between system memory and flash storage => file system incl. all user data is encrypted

- Hardware includes RNG (random number generator) used to generate further crypto keys

- Hardware includes 2 IDs which are used as keys and cannot be read by software:

  - UID (unique id per device)
  - GID (device group id, same for specific processor class) => used for e.g. deliver system software installation / restore

- Each file has data protection class (complete (UID + passcode, passcode removed from memory after unlock/device lock => use cached class key)/ protection until first authentication (not removed from memory)/ none (only by UID))

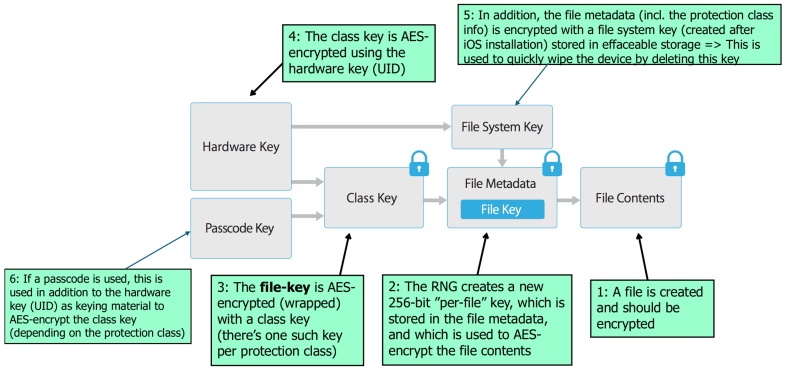


Abbildung 28: File Encryption

10.1.3 Secure Enclave - Data Protection

Secure Enclave stores keys in a secure environment to prevent their exposure to the CPU or main memory, runs on separate OS, also processes face and fingerprint

10.1.4 Passcodes and Face ID

Delays between attempts, bruteforce must be performed on device (check involves AES and UID)

Touch ID uses an additional coprocessor, the Secure Enclave

- Has its own secure boot chain and built-in AES engine and UID (key)

- Stores the fingerprint details of the user

- Verifies the "submitted" fingerprint by the user

- Does not replace the passcode, it's just a more user-friendly way to access it (File encryption and data protection still uses the passcode)

10.1.5 Sandboxing

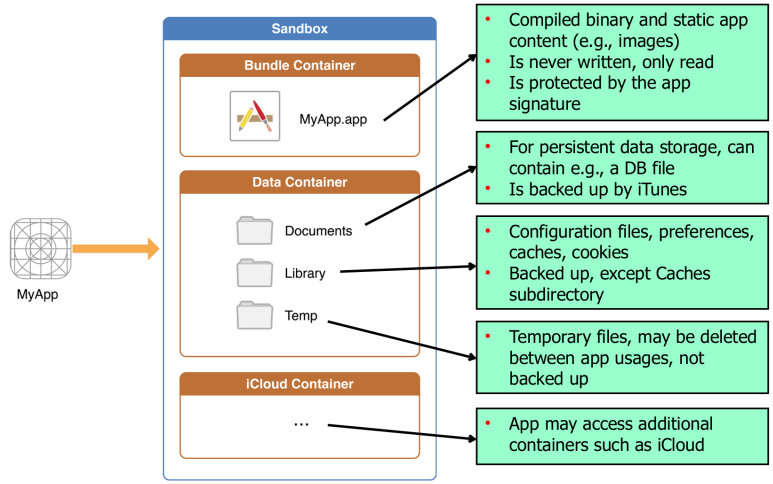


Abbildung 29: Runtime Security and Sandboxing

To enforce sandboxing, iOS uses the TrustedBSD Mandatory Access Control (MAC) Framework

10.1.6 Buffer overflow protection mechanisms

- Address Space layout Randomization (ASLR) to randomize all memory location upon launch
- Execute Never (XN) feature of ARM CPUs to mark memory pages such as the stack as non executable

10.1.7 Permissions

An app can access specific data or functions, but to do so it must ask the user for permission

10.1.8 Inter-App Communication

Via URL based scheme (e.g. http, tel, mailto)

10.1.9 Jailbreaking

removes the strict protection mechanisms
There are different vulnerability/exploit types (e.g. bootrom level/iboot level/ user space)

Types of Jailbreaks

- Untethered: Can boot normally - Boots jailbroken
- Semi-untethered: Can boot jailed - can jailbreak without PC
- Semi-tethered: Can boot jailed - can jailbreak with PC
- Tethered: Can not even boot without connecting to PC and using JB tools

10.2 Android Security

less strict and not enforced (compared to iOS) (runs on huge number of different hardware devices)

Runtime Security and Sandboxing When the app is installed on Android, two directories are created / used: apk file in common directory with owner system + data directory with new user = app is owner of directory

Permissions fine grained but user has to accept all => users rarely inspect list (=> overlay attacks)

Certificates for apps can be self-signed

- To update an app, the new version must be signed with the same certificate as the previous version
- If this is not the case, the app must first be uninstalled (which deletes the private /data/data/ directory) before the new version can be installed
- In addition, it is possible to establish trust relationships between apps that are signed with

- the same certificate => They can use the same user ID to access each others data

Inter-App Communication

- Android allows apps to communicate with each other
- The sender communicates with a specific component (an activity) of the recipient
- The object that is used for messaging is called intent
- Configured with intent-filter in AndroidManifest.xml

File Encryption Android support Full Disk Encryption (FDE) and File-Based Encryption (FBE) => someone steals your device, (most) data cannot be easily accessed but key is not required to be in hardware => bruteforce does not need to be performed on device

Rooting (= Jailbreaking) The approach of rooting typically means getting a su binary onto the device

11 Mobile Apps Security

Categories based on OWASP for mobile:

11.1 Data Storage and Data Leakage

storing sensitive data on the mobile device in a secure way and preventing that sensitive data is leaked (file system, buffer, copy/paste, screenshot, logs

Storage Options

- Store the data within the app-specific area in the file system, e.g., in folder Documents on iOS or shared_prefs on Android: requires root access, data is inculded in backups
- Store the data in a location that is intended for sensitive data, e.g., the iOS keychain or SecureEnclave or Android's keystore: not included in backups, root rights might still have access, secure enclave works with public key cryptography only
- store data in shared space => unprotected

Therefore would be best to never store sensitive data at all => not realistic => never store for e.g. bank data but okay to store medium sensitive data in shared_prefs

Data Leakage

- Copy paste has access in all apps,
- Screenshots in all apps with foto access + Screenshots by the OS (When an app enters the background, the OS takes a screenshot to show it in the list of running apps, In contrast to screenshots taken by the user, they are not accessible by other apps, but possible sensitive information is exposed in the file system nevertheless)
- Logs: written by all apps, read only on unlocked device => possible leakage of sensitive data in logs

11.2 Secure Communication

Exchange of sensitive data with backend servers use http + tls (= https) with secure versions of algorithms, secure server certificates from trusted stores (mobile OS has trust store), Don't allow to override failed certificate checks, use certificate pinning to avoid MITM (= store hashed version of CA or certificate in hardcode)

11.3 Authentication and Authorization

Often, mobile apps have less strict authentication requirements

Offline Authentication and Authorization access restricted content must be available somewhere in app, code access bought online. => can usually be circumvented

Alternatives to SMS as second factor (= mTAN)

- Shared secret between app and backend server (sent on login with pw and username) same security as SMS (=> attacker must have access to device)
- separate authentication app (e.g. in banking), must be configured with key, solves challenge and sends result to primary app => same security as mTAN (=> attacker must have access to device)

11.4 Inter-App Communication

both support URL-based communication, android also allows to explicitly call an Activity in another app

on Android Activity, Service or Content Provider must be registered in AndroidManifest.xml, To allow a component to be called by other apps, one does either:

- Mark them in den Android Manifest with android:exported=true"
- Add an intent-filter to the Activity or Service

11.5 Client-Side Injection

basically the same as on web, sql or file inclusion / file injection attacks possible

11.6 Reverse Engineering

Many techniques can only be done on jailbroken/rooted devices, => prevent that app can run on such a device but can usually be circumvented