Hochschule Bremerhaven

Lecture Notes

IT-Sicherheit

Wintersemester 2022/23

Version: 31. Januar 2023

Prof. Dr. Lars Fischer

lars.fischer@hs-bremerhaven.de

License

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

Inhaltsverzeichnis

1

Syllabus

> 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 IT-Sicherheit an der Hochschule Bremerhaven im Wintersemester 2022/23; Organisation, Lernform und Prüfungsform beschrieben.

Der Kurs IT-Sicherheit ist auf eine konstante Beteiligung an allen Teilen des Kurses ausgelegt. Wesentliche Prüfungsleistungen werden im laufenden Semester erbracht. Es ist deshalb weder möglich noch sinnvoll den Kurs als Blockkurs durchzuführen. Bestehen ernsthafte Hindernisse die einer konstanten Beteiligung entgegenstehen müssen diese frühzeitig angemeldet werden! Eine nachträgliche Anpassung der Lern- und Prüfungsbedingungen ist nicht möglich.

Für alle Probleme und Lernhindernisse gilt eine fundamentale Regel:

> Probleme müssen rechtzeitig **vor** dem Start- bzw. Abgabetermin einer einzelnen Prüfungsleistung mitgeteilt werden.

Mit dem Start (beziehungsweise der Abgabe) einer Prüfungsleistung ändert sich der Arbeitsmodus von “Lernen und Lehren” zu “Prüfen”. Im Modus “Lernen und Lehren” möchte ich Ihnen jede Unterstützung angedeihen lassen die möglich ist um ihren Lernerfolg zu fördern. Im Modus “Prüfen” ist es meine Aufgabe diesen Lernerfolg zu prüfen und zu bewerten und damit auch den Wert eines erfolgreichen Kursabschlusses sicherzustellen. Praktisch bedeutet dies, dass es nach dem Start einer Prüfungsleistung (beziehungsweise dem Abgabezeitpunkt von Übungsaufgaben) keine Entgegenkommen bezüglich Terminen, Prüfungsinhalten oder sonstige Anpassungen der Lehre und Prüfung geben kann. Natür-

lich stehen ihnen alle vorgesehenen formalen Wege, wie die Einreichung von ärztlichen Attesten oder die Anrufung des Prüfungsausschusses offen.

Im Anschluß an die Prüfungsphase haben Sie aber immer die Gelegenheit die Bewertung ihrer Prüfungsleistung einzusehen und die Korrektur von Bewertungsfehlern einzufordern.

Lern- und Prüfungsphasen

Abbildung 1.1: Lern-, Lehr- und Prüfungsphasen

Rahmenbedingungen:

Rahmenbedingungen

Präsenzzeit: 2 VL, 2 Ü (3h 20m)

Selbstlernzeit: (4h 40m)

Semesterzeiten: 14 Wochen

- Vorlesungsstart: 19.10.2020
- Vorlesungsende: 01.02.2021

Zielgruppe:

Studiengänge: INF und WINF

- Fachsemester: 5 (Pflichtveranstaltung)

Prüfungsform: Portfolio

1.0.1 Lernziele

Lernziele

Basic Security Knowledge:

- Security as a Process

Vorlesungsvideos:	1,5h/Woche
Übungen: 3h/Woche	

Tabelle 1.1: Arbeitsbelastung Selbstlernzeit

Plenum:	1:30h/Woche
Präsenzübung:	1:50h/Woche

Tabelle 1.2: Arbeitsbelastung Präsenzzeit

- Threats, Vulnerabilities and Controls
- Communication and Protocols
- Architectures
- Computer Systems
- Software Security

1.1 Lern- und Lehrkonzept

Dieser Kurs lehnt sich an das Konzept des Inverted Classroom

1.1.1 Zeitplanung

Arbeitsbelastung Selbstlernzeit:

Arbeitsbelastung

Arbeitsbelastung Präsenzzeit:

Arbeitsbelastung

Wochenplan

Mi
- Vorlesung/Plenum
  - Vertiefung der letzten VL
- Update Webseite
  - Vorlesungsvideos
  - Skript

  - ggF. Aufgaben/Fragen

Di Übung

  - Betreutes Arbeiten am Übungszettel
  - Vertiefende Experimente

Mo-So Matrix itsec-ws22

1.1.2 Vorlesung (Video)

Die Vorlesung wird in der Form von Kurzvideos als Selbstlernteil, nach eigener Zeitvorgabe, durchgeführt. Die Videos werden jeweils durch leichte Verständnisfragen abgeschlossen. Die Beantwortung dieser Fragen ist nicht Teil der Prüfungsleistung. Die Fragen müssen aber korrekt beantwortet werden, bevor das nächste Video angeschaut wird.

Die Vorlesung ist in thematische Unterabschnitte unterteilt. Die Themen bauen teilweise aufeinander auf. Inhalte der Unterabschnitte werden in den Übungsaufgaben vertieft und müssen dementsprechend zum Abgabezeitpunkt der Übungsaufgaben erfolgreich abgeschlossen sein.

1.1.3 Plenum

Im wöchentlichen Plenum ist Platz für Nachfragen zum Kurs und vertiefende Diskussion zu ausgewählten Inhalten statt. Zu jedem Plenum werden von, vorher ausgewählten Studierenden, ein Ergebnisprotokoll erstellt und von jeweils zwei Studierenden korrigiert und freigegeben. Die freigegebenen Protokolle sind Teil der Kursdokumentation. Die Erstellung und Korrektur sind verpflichtende Prüfungsteile.

Die Kommunikationskanäle für Nachfragen im Plenum werden jeweils durch Studierende moderiert.

1.1.4 Präsenzübung

In der Übung wird anhand von konkreten Aufgabenstellungen der Inhalt der vorangegangenen Vorlesung erarbeitet und erweitert. Die Aufgaben sind derart gestaltet, dass in der Übungsstunde die von den Studierenden vorbereitete, Lösungen diskutiert werden.

Musterlösungen werden in der Regel nicht vom Lehrstuhl ausgegeben, es besteht die Möglichkeit, dass Studierende ihre eigenen Musterlösungen in den Ellie-Foren, nach der Abgabefrist veröffentlichen und diskutieren.

Die Übungen finden jeweils in Präsenzgruppen (ca. 20 Personen) statt. In den Übungen arbeiten die Studierenden in betreuten Kleingruppen

an den Übungsaufgaben. In den Präsenzübungen finden weiterhin regelmäßige Konsultationen zur Semesteraufgabe statt.

1.2 Kursunterlagen

Alle Unterlagen des Kurses werden über Elli bereitgestellt. Die Unterlagen umfassen, unter anderem:

Videos

Übungsaufgaben

Verständnisfragen

Vertiefende Quellen

Skript: für ausgewählte Teile der Vorlesung wird zusätzlich ein begleitendes Skript bereitgestellt. Das Skript ist liegt größtenteils in englischer Sprache vor und enthält eine Obermenge der Inhalte des Kurses. Das Studium des Skriptes ist kein vollwertiger Ersatz der konstanten Teilnahme am Kurs.

1.2.1 Forum und Chat

Zur Vorlesung wird ein Kursforum bereitgestellt über welches Musterlösungen und inhaltliche Diskussionen geführt werden können.

Die Teilnahme an den Foren und dem Chatraum sind nicht Teil der Prüfungsleistung. Eine Teilnahme wird aber dringend empfohlen.

1.3 Prüfungsbedingungen

Die Prüfung erfolgt als Portfolio aus sechs Aufgabenzetteln aus Einzel- und Gruppenanteilen, sowie einer Hausaufgabe in den Semesterferien mit einer Bearbeitungszeit von einer Woche.

Prüfungselemente

Übungsaufgaben
- Gruppen-/Einzelleistung
- 2-wöchentliche Abgabe von Aufgabenzetteln

Hausaufgabe Individualleistung, Umfang 1 Woche, mehrere Aufgaben

Gruppengröße 2-3 Personen

Anmeldung mit erster Abgabe

1.3.1 Übungsaufgaben

Die Übungsaufgaben ergänzen und erweitern die Vorlesungsinhalte und sollen wesentlich während der Präsenzübung, mit minimaler Vorbereitung, bearbeitet werden können. Es wird erwartet, dass die Studierenden sich die nötigen Arbeitsmittel (Bücher, Anleitungen,...) vor der Präsenzübung einsatzbereit vorbereitet haben.

Bewertung der praktischen Übungsaufgaben erfolgt in einer vorher bereitgestellten Ausführungsumgebung.

1.3.2 Hausaufgabe

Die Hausaufgabe umfasst die Bearbeitung mehrerer individueller Aufgaben innerhalb eines vorgegebenen Zeitraums von einer Woche. Die Aufgaben umfassen die Reproduktion und Anwendung theoretischer und praktischer Themen aus der Vorlesung.

1.4 Bewertungsschlüssel

Die Übungsaufgaben gehen mit 60%, die Hausaufgabe mit 40% in die Endnote ein. Grundsätzlich verwenden wir die Notenskala[1] des International Office. Dies bedeutet, dass der Kurs ab 50% der erreichbaren Punkte als bestanden gilt. Werden jeweils 5% mehr der erreichbaren Punkte erzielt, verbessert sich die Note um eine Drittelnote.

1.5 Quellen

Die Vorlesung basiert grundlegend auf den Lehrbüchern von William Stallings: „Computer Security“ [**stallings2008computer**], „Cryptography and Network Security“[**Stallings11CryptographyandNetwork**] und „Network Security Essentials“ [**Stallings07NetworkSecurityEssentials**], sowie dem Buch „IT-Sicherheit: Konzepte - Verfahren - Protokolle“ von Prof. Dr. Claudia Eckert [**eckert2011sicherheit**]. Diese sind entsprechend mit Seitenzahlen referenziert und verweisen in der Regel auf weitere Quellen.

Es wird erwartet, dass Teilnehmer dieser Veranstaltung zuerst selbstständig diese Werke konsultieren.

Weitere Quellen

[1] https://www.hs-bremerhaven.de/organisation/dezernate-und-stabsstellen/international-office/internationale-und-austauschstudierende/vorbereitung/leistungspunkte-und-notenskala/

- Willam Stallings:
  - „Computer Security“
  - „Cryptography and Network Security“
  - „Network Security Essentials“
- Claudia Eckert: „IT-Sicherheit: Konzepte - Verfahren - Protokolle“

1

Introduction

1.1 IT-Security Objectives

The central question of security is one of authorisation. That is whether a specific action by a defined subject is allowed at the given circumstances to be executed on a given object. Work on the topic security is concerned expression and definition of allowed vs. forbidden, prevention and detection of misbehaviour, and the recovery from incidents.

Definition IT-Security

> **Computer Security:** The protection afforded to an automated information system in order to attain the applicable objectives of preserving the integrity, availability, and confidentiality of information system resources (includes hardware, software, firmware, information/data, and telecommunications)
> [NIST Computer Security Handbook, 1995]

1.1.1 CIA-Triad

The central concept of security is that of Authorization. With respect to authorization, IT-Security is understood to be defined by classes security-objectives. The most common understanding is given by the "CIA-Triad" in [**guttman1995introduction**]:

Confidentiality (Geheimhaltung)Only authorized users get the information.

Integrity (Integrität) Information is correct, complete, and current or this is detectably not the case.

Availability (Verfügbarkeit) Information and resources are accessible where and when the authorized user requires them.

e. g., [NIST Computer Security Handbook, 1995]

Confidentiality

Integrity

Availability

Although there are different opinions in the community whether this classisfication is too granular or completely covering all nuances of security, but the CIA-Triad provides an accepted baseline for the understanding of the scope of IT-Security. The terminology is applied in a very practical manner, with subcategories and limited extensions if the need arises.

And, while the dicipline focusses on information technology it must be stressed, that it can not be expected that attackers limit themselves only to digital attack vectors. Defenders can only be successful, if they also don't limit their methods to the digital domain but integrate security into the larger environment where IT is embedded in or executing control.

The common ground is the understanding, that security is the topic of intentional unauthorised actions by threat actors and protections and controls by defenders or asset-owners.

1.1.2 Further Security Related Objectives

The CIA-Triad 1.1.1 provides a categorisation requires some refinement into sub-objectives to allow work on solutions. There is no general consensus about refined sub-objectives, but a widely used of common terms that are to be introduced here.

In Germany the canonical reference on IT-Security by Claudia Eckert distinguishes six different categories of security objectives. Figure 1.1 provides a mapping of objectives.

Authenticity (Authentizität) denotes that the source of an information object can be proven. This implies that the integrity of that object is also protected. Therefore this is considered a part of the Integrity category in the CIA-Triad. (See 1.1) It can be argued that the term integrity is not sufficiently covering specific problems of authentication.

Integrity (Integrität) has the same meaning than the same term in the CIA-Triad without authentication.

Confidentiality (Vertraulichkeit) similar to the CIA-Triad, although in this definition it is not addressing specific topics from the domain of privacy protection.

Availability (Verfügbarkeit) denotes roughly the same topics as in the CIA-Triad.

Non-Repudiation (Unabstreitbarkeit) denotes the specific problems related to proving that defined actions have taken place, i. e., to proof that some other party has signed a contract. But also problems of secure logging of events where the other party is not actively providing proof of its actions. Techniques and problems are often similar to the topices of the Integrity-category in the CIA-Triad, but this category also often has requirements related to availability, e. g., availability of a current and correct time.

Ano-/Pseudonymity (Anonymität/Pseuyonymität) This term covers specific problems in privacy protection. The techniques and proceedures differ from the cryptographic techniques used for protection of confidentiality (in the Eckert-Model). One requirement, for example, could be unobservability of communication. It might be obvious that Anonymity-protection is in direct conflict with Authentication.

Refinement of Security Objectives

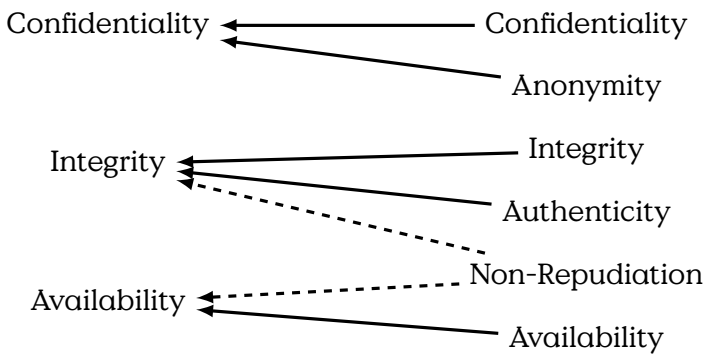


Figure 1.1: Mapping Eckert as sub-objectives onto the CIA-Triad.

1.1.3 Interedependencies of Security Objectives

Interdepentencies between S.O.

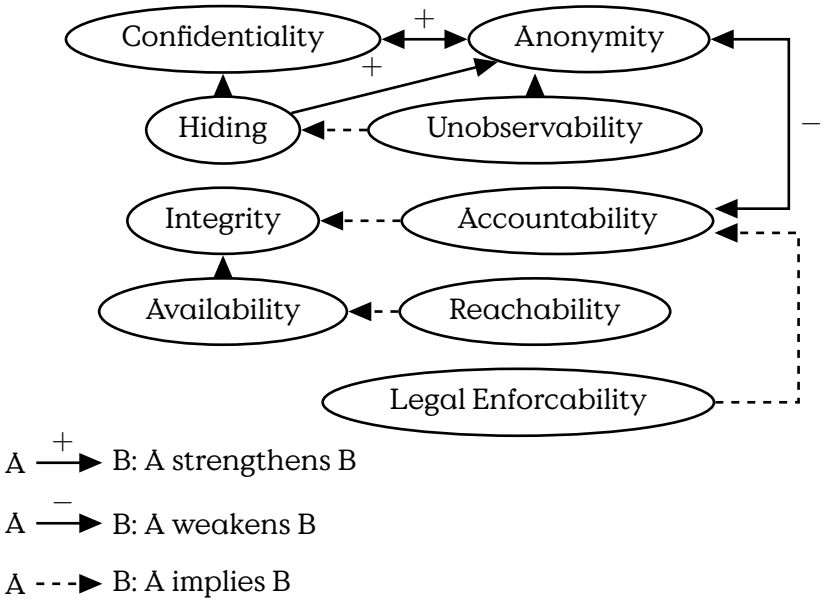


Figure 1.2: The security model of Wolf and Pfitzmann [**Wolf2000**]

> Security is a process, not a product.
>
> Products provide some protection, but the only way to effectively do business in an insecure world is to put processes in place that recognize the inherent insecurity in the products. The trick is to reduce your risk of exposure regardless of the products or patches.
>
> Bruce Schneier:
> Computer Security: Will We Ever Learn? [seen 2020-08-26]

1.2 Security vs. Safety

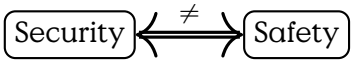
Dieser Abschnitt basiert auf einem Artikel von Fischer und Messerschmidt für die AG KRITIS. [**FM2020_safety_security**]

Wo der Brandschutz einen lebensrettenden Notausgang sieht (Safety), sieht der Sicherheitsberater eine Schwachstelle im Zugangsschutz (Security). Dieses simple Beispiel zeigt den Konflikt, der oft zwischen dem besteht, was im Englischen als Security und Safety unterschieden wird. Im Kern des Konflikts steht die Frage, welche Sicherheit für welche Schutzfunktionen benötigt werden und welche Schutzmaßnahmen notwendige Sicherheitsmaßnahmen untergraben. Die Beschäftigung mit dieser Frage ist notwendig, weil die beiden Seiten oftmals von unterschiedlichen Ex-

perten bearbeitet werden. Kommunikation ist notwendig, um die unterschiedlichen Ziele miteinander zu verbinden und Missverständnisse zu vermeiden.

Kommunikation setzt ein gemeinsames Verständnis der Begrifflichkeiten voraus, weshalb wir im Folgenden den Versuch einer Begriffsbestimmung der Sicherheit durch die zwei Begriffe Safety und Security unternehmen wollen. In der Literatur finden sich verschiedene Merkmale die zur Unterscheidung herangezogen werden.

German Terminology Fallacy



„Sicherheit" ≠ „Sicherheit"

Die Unterscheidung in Kurzform:

Safety vs. Security

1. Unterscheidung nach

   Schadensursache:

   Safety als Schutz eines Systems vor zufälligen, nicht-bewusst herbeigeführten Schadereignissen, z.B. Wetterphänomene oder Fehlbedienung.

   Security als Schutz eines Systems vor bewußt herbeigeführten, zielgerichteten Schadereignissen, z.B. Cyberangriffe.

2. Unterscheidung nach Art des Schadens und nach Schädigungsrichtung:

   Safety als Schutz vor Personenschäden, in der Regel durch das betrachtete System.

   Security als Schutz vor Schaden am betrachteten System.

Diese Situation zeigt dass auch grundlegende Begriffe komplexer sind als vielfach angenommen. Das ist natürlich unbefriedigend und erfordert, dass das Verständnis der Sicherheitsgrundbegriffe neu ausgehandelt werden muss.

Safety und Security nach dem Ursachekriterium

Von Safety sprechen wir, wenn es darum geht Anlagen, Prozesse oder Personen vor Beeinträchtigung durch ungesteuerte, zufällige oder natürliche Ereignisse zu schützen. In diese Klasse fallen Ereignisse die durch "ungewollte Fehlbedienung" ausgelöst werden, ebenso wie die Beschädigung durch Umweltereignisse wie Erdbeben oder Stürme.

Demgegenüber verstehen wir Security als Verhinderung oder Vermeidung von unerlaubter, absichtlicher Beeinflussung, mit dem Willen zur Schädigung von Werten (Assets). Im Gegensatz zur Safety sind der wesentliche Faktor im Bereich Security die Angreifer (Attacker oder auch Threat Agents), die sich insbesondere dadurch auszeichnen, dass sich ihre Fähigkeiten und ihr Verhalten schlecht vorhersagen lassen. Während es vergleichsweise einfach ist, die erwarteten Sturmereignisse in einer Region historisch aufzuzeichnen, statistisch zu beschreiben und damit zu prognostizieren, ist das Verhalten von Angreifern nicht durch empirische Beobachtung vorhersagbar. Schlimmer noch, es ist anzunehmen, dass, wenn Sicherheitsmaßnahmen auf bekannte Angriffsmuster optimiert werden, sich die Angreifer anpassen und insbesondere die verbleibenden Lücken oder genau die Sicherheitsmaßnahmen selbst angreifen.

Die Definition über die Schadensursache ist für die Planung von Schutzmaßnahmen einfach anzuwenden. Bei der Analyse und Reaktion auf Schadensereignisse ist jedoch gerade dieses einfache Kriterium der bewussten Schädigung schwer zu ermitteln. Denn es erfordert eine Attribution der Schadensursache bzw. der Verursacher. Eine eindeutige Attribution ist in einer digitalen Welt aber im Allgemeinen nicht möglich. Deshalb ist es sinnvoll, zusätzlich die Definition nach anderen Kritierien zu berücksichtigen.

Safety und Security nach dem Schadenskriterium

Das Kriterium der Schadensrichtung besagt, dass die Unterscheidung sich dadurch ergibt, ob ein möglicher oder tatsächlicher Schaden am betrachteten System oder ein Schaden durch das betrachtete System an einer anderen Sache vorliegt. Der Begriff Safety bezieht sich deshalb auf Schaden, der durch das System selbst entsteht, z.B. ein Zug dessen Bremssystem versagt. Der Begriff Security bezieht sich auf Schadenswirkung am System.

Das informelle Glossar der Internet Enginieering Task Force, welche wesentliche Internet Standards spezifiziert hat, empfiehlt die folgende Definition:

> **Safety**: "The property of a system being free from risk of causing harm (especially physical harm) to its system entities." [**rfc4949**]

Sicher, im Sinne von Safety, ist ein System dann, wenn kein Risiko besteht, dass von einem betrachteten System Schaden ausgeht.

Für den Begriff Security werden, je nach Kontext drei unterschiedliche Definitionen angeboten. Als Zustand wird Security als Ergebnis der Einführung und Aufrechterhaltung von Maßnahmen zum Schutz des Systems verstanden. Security kann weiterhin als Oberbegriff für genau diese Maßnahmen verstanden werden. Als Gegenstück zur Safety wird Security analog zur IT-Sicherheit als Zustand in dem ein System frei ist von möglichem Schaden von aussen, in Bezug auf die Unmöglichkeit von unauthorisiertem Zugang, Veränderung, oder Datenverlust, aber auch zufälligen Schadenseinwirkungen.

> **Security**: "A system condition in which system resources are free from unauthorized access and from unauthorized or accidental change, destruction, or loss." [**rfc4949**]

Safety und Security unterscheiden sich auch nach der Art des Schadens. Safety wird immer als ein Schutz vor Personenschäden verstanden, also Verletzung oder Tod von Menschen.

Demgegenüber bezieht sich Security sowohl auf den Schutz vor Personenschäden wie auch vor materiellen und ideellen Schäden. Deshalb ist es sinnvoll, bei Security zusätzlich nach IT Security und Physical Security zu unterscheiden, um besser nach Schadensart differenzieren zu können.

Der Begriff IT-Sicherheit bzw. IT Security oder Computer Security im Englischen kann als Spezialfall der oben definierten Security betrachtet werden. Informationstechnik (IT) ist ein wesentliches Element der aktuellen Veränderungen in Kritischen Infrastrukturen. IT-Sicherheit geht üblicherweise von der Definition des National Institutes for Standards and Technology (NIST) aus. Das NIST definiert Computer Security als die Sicherstellung von Integrität, Geheimhaltung und Verfügbarkeit von Systemen und Daten:

> **Computer Security:** The protection afforded to an automated information system in order to attain the applicable objectives of preserving the integrity, availability, and confidentiality of information system resources (includes hardware, software, firmware, information/data, and telecommunications). [**NISTSP800-12**]

Der Begriff Physical Security kommt aus dem Militär und bezeichnet im engeren Sinne alle physischen Schutzmaßnahmen gegen unerlaubten

Zugriff. Ein vergleichbarer deutscher Begriff ist Objektschutz. Im weiteren Sinne umfasst Physical Security alle Schutzmaßnahmen durch Sicherheitskräfte, auch im nicht-militärischen Bereich wie zum Beispiel durch die Polizei.

Während die IT Security sich überwiegend auf finanzielle Werte bzw. Schäden konzentriert, ist die Physical Security mit finanziellen (Raub), menschlichen (Mord, Verletzung, Entführung, Stalking), ideellen (Rufschädigung, Beleidigung), wirtschaftlichen (Sabotage) und gesellschaftlichen (Terrorismus) Schäden bzw. Werten befasst.

1.2.1 Security Priorities

Selection an prioritisation of security objectives in practice depend obviously on application- and system-specific requirements. But often you will find that security experts prioritise secrecy above integrity with availability being considered sometimes in the aftermath. This is only noteworthy because it is a common point of conflict with other stakeholder groups.

For most cyber-physical systems Safety provides the key objective that comes on top.

One example is conflicting priorities with security requirements in Industrial Control System (ICS), or more generally Cyber-Physical System (CPS), shown in Figure 1.3. Physical industrial processes often dependend on timely provision of precursors in the process. Safety-related parts of physical processes even must be functional at all times and under tight stress-conditions. These high requirements on availability in ICS transfer directly to the IT-System that controls these measures.

Priorisierung Sicherheitsziele

1.3 Basic Threat Model

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

Definition: Attack

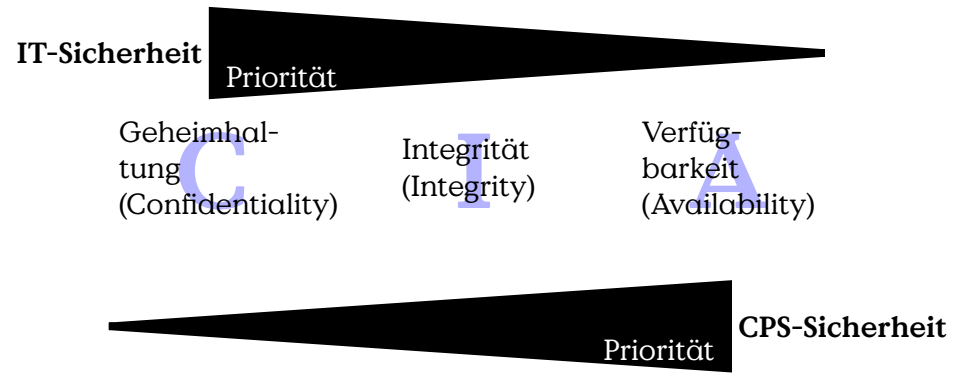


Figure 1.3: Prioritätsumkehr der Sicherheitsziele von information technology (IT)-Sicherheit und CPS-Sicherheit

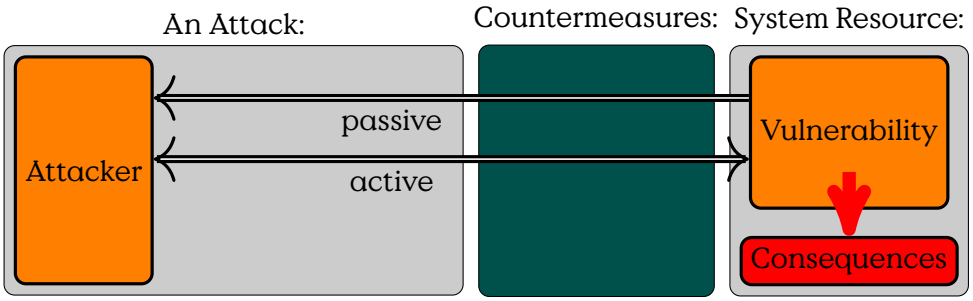


Figure 1.4: Basic model of an attack [**rfc4949**]

Abbildung 14.1 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 er-

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

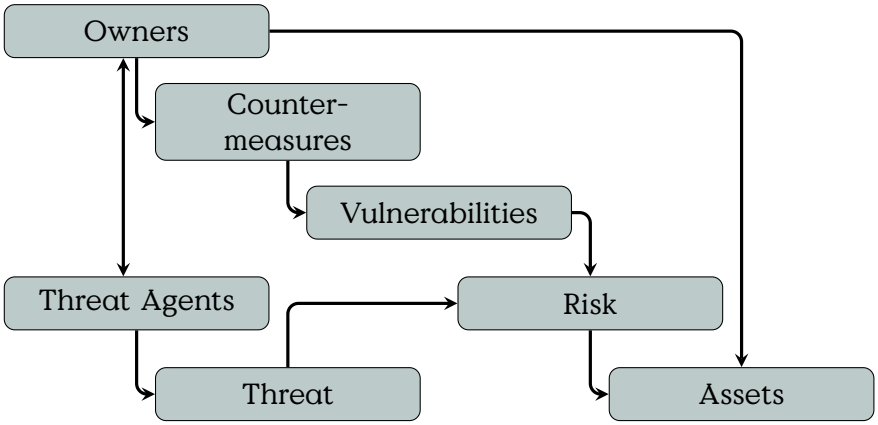


Figure 1.5: Security concepts and relationships [**stallings2008computer**][**CommonCriteria**]

greifen.

Attacker - Owner — Assets

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.

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.

Threat Terminology Summary

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

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

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

1.4 Cyber-Kill Chain

Cyber Kill Chain

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

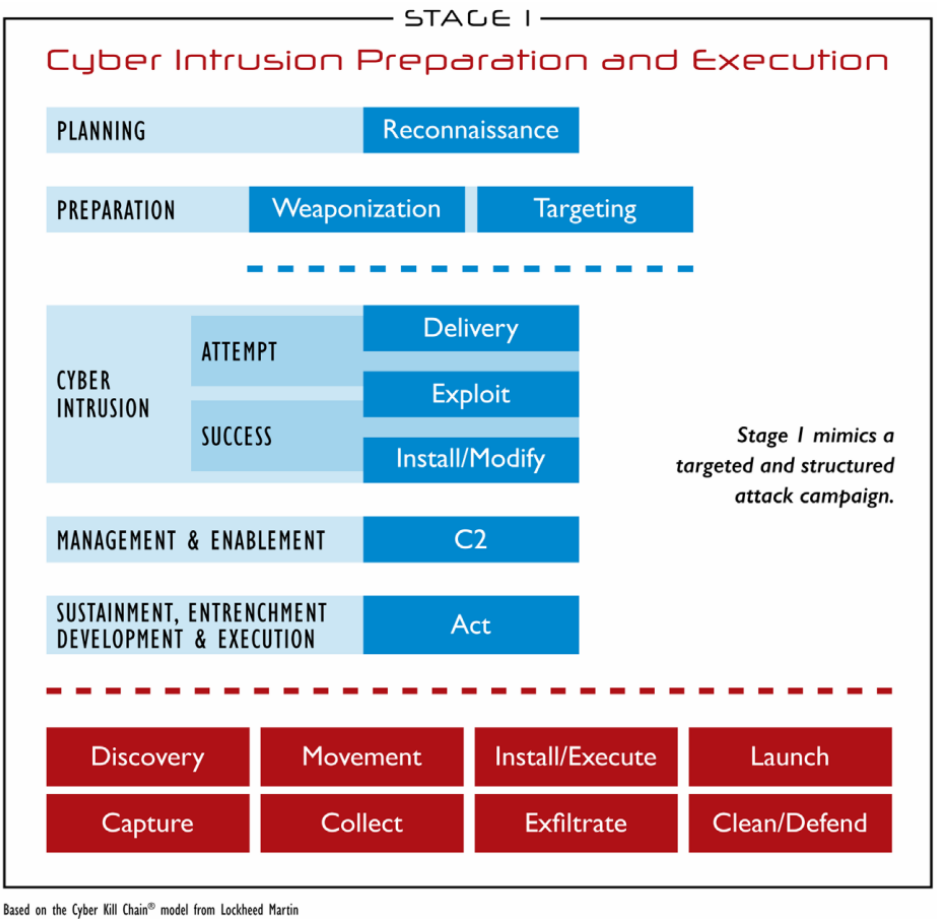


Figure 1.6: Industrial Control System Cyber Kill Chain, first stage representing the classical attack process in IT systems as defined in [**martin2014cyberkillchain**], taken from [**assante2015icsckc**].

wurden. Weitere Informationen zur politischen Einordnung in den Konflikt finden sich in [**baezner2018cyber**].

Ukraine Outages

2015:

- Sicherungsschalter in 57 Substations (Ortsnetzstationen) geöffnet
- Ca. 280.000 Kunden, 3h ohne Strom
- 2 Verteilnetzbetreiber betroffen
  - Prykarpattyaoblenergo:
    - 27 Substations,

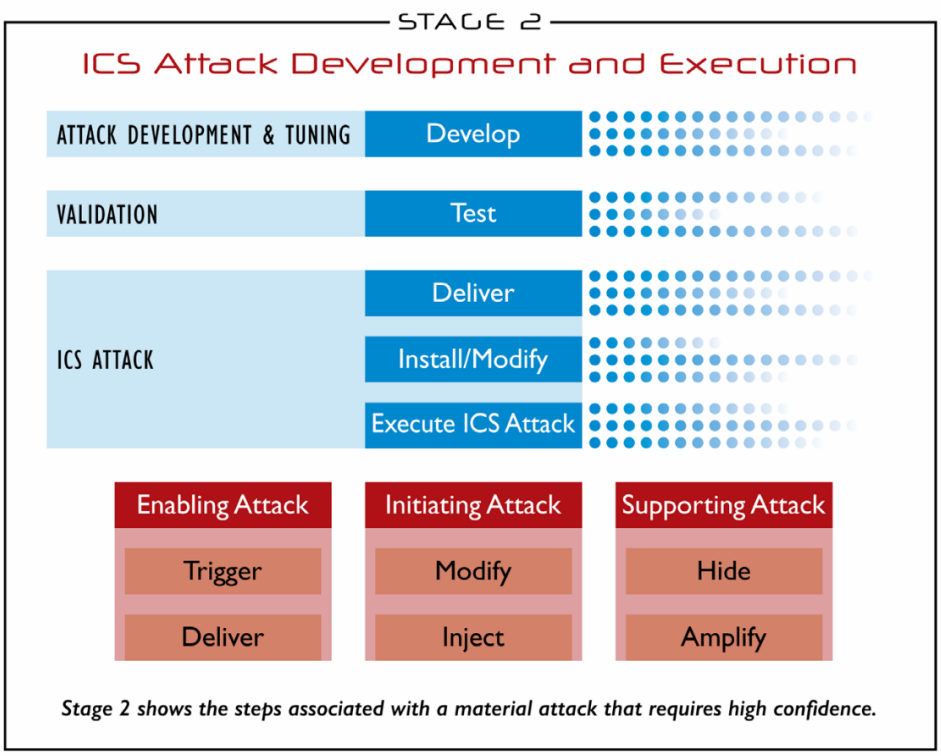
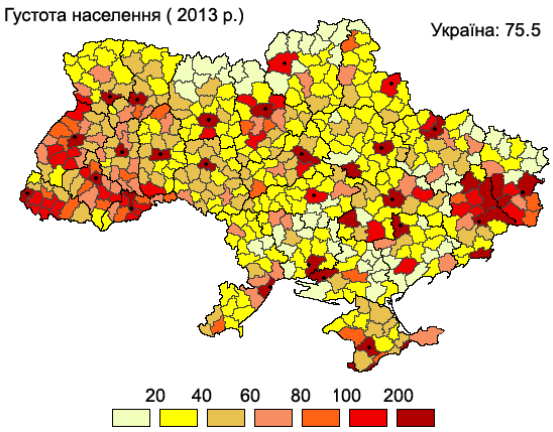


Figure 1.7: Industrial Control System Cyber Kill Chain, second stage describing the attack process into the ICS, taken from [**assante2015icsckc**].

    * 103 Städte ohne Strom,
    * 186 teilweise betroffen
  - Chernivtsioblenergo:
    * nicht veröffentlicht

2016:

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



Weitere Angriffe: Behörden, Radiostationen, Firmen

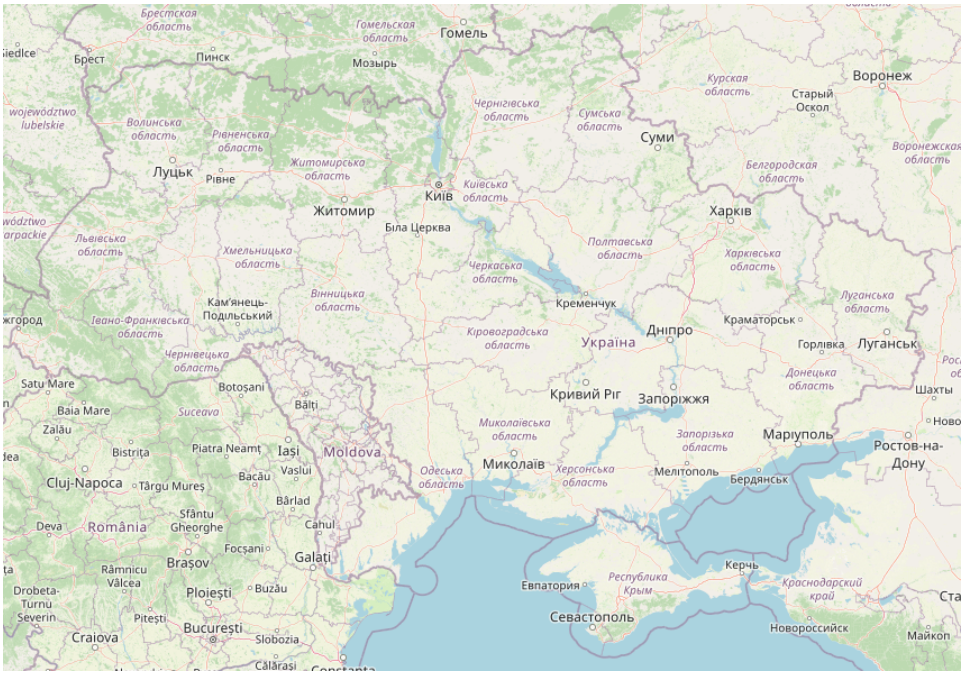
Ukraine'15

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

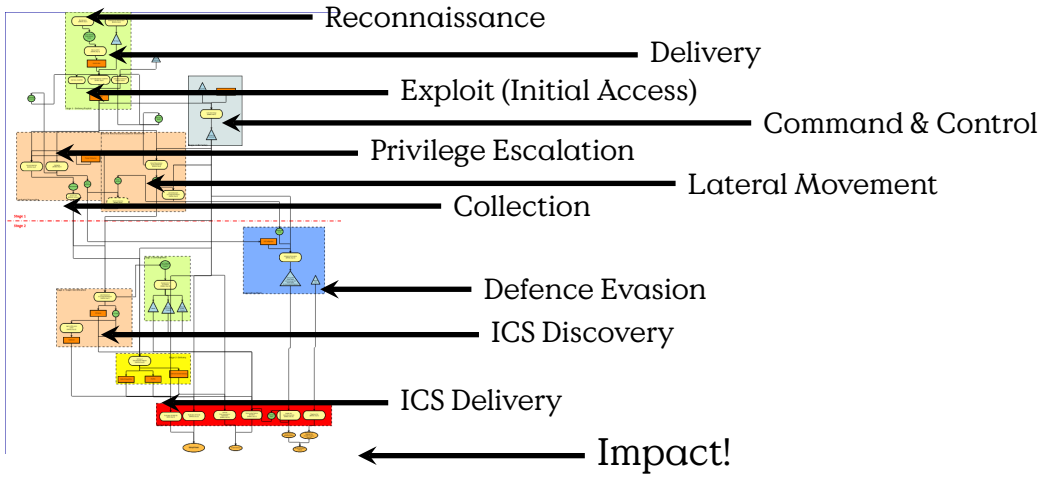


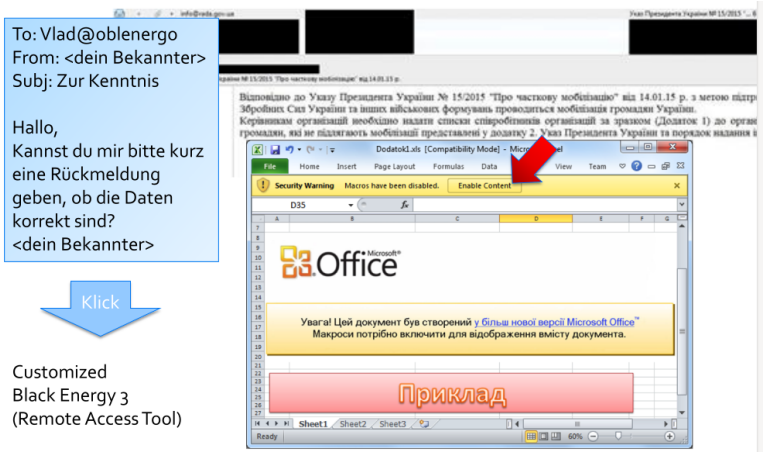
Figure 1.8: Complete Attack Flow of the Ukraine 2015 Incident

Ukraine 2015

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)

Ukraine'15



Ukraine 2015

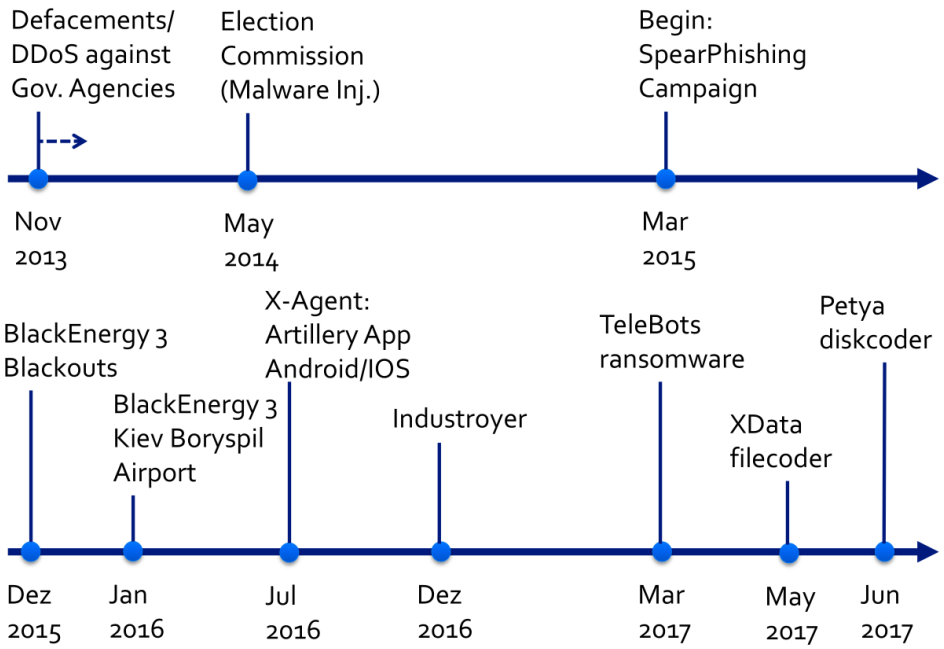
Malware Tools:

- 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

Ukraine



Sources:

- [baezner2018cyber]

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

1.4.1 Tactics, Techniques, and Proceedures (TTP)

Tactics, Techniques, and Proceedures (TTP) ist ein Begriff der die üblichen Werkzeuge und Vorgehensweisen einer Gruppe von Angreifern adressiert. Bei der Analyse eines Sicherheitsvorfalls sind die beobachteten TTP häufig ein erster, und auch vergleichsweise stichhaltiger Indikator auf die Herkunft eines Angriffs.

Auf der anderen Seite können Angreifer die TTP einer anderen Gruppe imitieren um die Herkunft eines Angriffs zu verschleiern. Werkzeuge werden genau deshalb oft “veröffentlicht”. Weniger leicht lassen sich allerdings festgefügte Prozesse der Angriffsteams ändern.

TTP

T actics

T echniques

P roceedures

1.5 Vulnerabilities

Schwachstellen

Vulnerability: eine ausnutzbare Schwachstelle

1.6 Malware

The shady world of malware terminology is full of references to biological and mystical entities. Sometimes playfully named categories of malware help to distinguish different behaviour patterns of malware. Knowing these basic terminology helps to communicate and establish fitting countermeasures. This section provides an overview over the basic terms encountered most often.

Malware

Definition 1.6.1. Software that is created with the intend to be deployed on a computer system without permission in order to attack system, communication or data.

Malware may be comprised of or denote:

Propagation Mechanism Malware or part thereof used to transfer it to its target.

Infection Mechanism Malware or part thereof that installs it within a target system. This could mean a simple file copy, installing a daemon or embedding code within existing software.

Payload The "functional" part of malware that is not distribution and replication but with the original effect of a malware. Effects range from purely visual nuisances to spying and manipulation of the system.

Malware Terminology

Infectious Worm A computer program that replicates and distributes itself to different computers. A worm usually utilises security vulnerabilities to gain access to remote computers.

Virus A program that resides within other programs where it hijacks the execution flow in order to replicate itself and distribute to other computers. Different to a worm, a virus is no standalone program but needs a host program. Fred Cohen

first defined the term "virus" in his 1984 paper "Computer Viruses - Theory and Experiments"[2]

Trojan Malware that is disguised as a benign program in order to manipulate the user to distribute and install the malware.

Rootkit Persistant installation of a service that grants continued, often privileged (mostly remote) access to a computer system. Interestingly rootkits have two faces, one side is the usage as malicious backdoor to a system, the other is a benign remote administration tool.

Backdoor:

Botnet: Collection of rootkits, installed on hosts that are controlled by a Bot Herder. Bot networks are often used to sell services of dubious legality: Spam Services, Denial of Service Attacks are among the most popular.

Ransomware

1.6.1 Worm

The term "Worm" denotes a self-sufficient program which can produce replicates of itself and transfer these copies to remote computers. Worms usually exploit vulnerabilities to infect remote computer systems. A worm may take up a completely different form in transit, for example hide away in an installation package or within a thumbnail file on a USB-Stick. On the target system a dropping mechanism creates a new copy of the worm as a independent process.

Worm

- Standalone program
- Self-replication
- Multiple Segments (common)
- Infection:
  - Vulnerability Exploitation
- Transfer:
  - Network (common)
  - Mobile Storage

[2] https://web.eecs.umich.edu/~aprakash/eecs588/handouts/cohen-viruses.html

  - Mobile Devices

The term “Worm” was coined in the science fiction novel “The Shockwave Rider”.

The first — and arguably most famous — historical worm is the “Morris Worm”

1988-11-02: The Morris Worm shut down 1/10th of all unix internet computers. With an estimated size of the Internat at that time of about 60.000 computers, this are only 6.000 machines in total. 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**].

Worm

Protection agains worm infection means security hygiene in the operating system, process and file access management.

Worm

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

1.6.2 Virus

The term “Virus” describes malware that is embedded within benign programs. Thus the main difference between virus and worm is that a virus in not a self-sufficient but consists of “pure” code. That means it requires

an existing program for execution. A virus propagates by modification of program files or process memory.

Virus

Computer Virus:

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

Virusses are also complex software. And as such can be separated into different functional parts. As virus technology is in a constant competition with defence mechanisms the following list only covers the most basic mechanisms.

Virus Signature () prevents that a virus infects an already infected program, which could either render it nonfunctional or increase the likelyhood of detection.

Infection () is the mechanism that embedds virus code into a target process or programm.

Payload () is the part of the code that implements the actual attack of the virus, e. g., exfiltrating data, installing backdoors, or encrypting the system.

Jump () is often a necessary part of the code because virus-code can often only be attached at the end of a program — otherwise it would completely scramble the original address space of the programm.

Virus

- 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
```

Figure 1.9: Abstract virus code

Virus Types

- 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

Countermeasures

- Principle-of-Minimum-Rights
- Encrypted Program
- Sandboxing (Data/Macro)
- Email-Isolation/Containment
- IDS/Antivirus
- Monitoring (Fingerprinting IDS)
  - (e. g., Tripwire)
- Liability (Haftung)

Anti-Virus Software

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

1.6.3 Trojan

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.

Trojan Horse

Functionality ≠ Proposed Functionality IST-Funktion ≠ 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[3] 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.

[3] See, for example the Farewell Dossier

Trojan Horse

- 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)

Trojan Horse

- 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

Attack Terminology

1.7 Security Controls

Wie soll man Sicherheit herstellen? Werden die Maßnahmen zu offen gestaltet, dann können sie umgangen werden und die Schutzziele werden nicht erfüllt. Sind sie zu strikt, dann verhindern sie legitime Nutzung.

[Dieser Abschnitt ist aus [**eckert2011sicherheit**] entnommen und wird, um Übersetzungsfehler zu vermeiden, in deutscher Sprache präsentiert.]

Die Sicherheitsgrundfunktionen bieten einen Baukasten zur Sicherstellung von grundlegenden Sicherheitseigenschaften und finden im Grundschutz insbesondere dann Anwendung, wenn eine niedrige Schadenshöhe zu erwarten ist. Eine vollständige Bedrohungs- und Risikoanalyse stellt in solchen Fällen eine unnötig hohen Aufwand dar. Grundsätzlich sollte jede Sicherheitsstrategie zumindest Konzepte zur Realisierung der Grundfunktionen enthalten.

Die Sicherheitsgrundfunktionen sind:

Sicherheitsgrundfunktionen

- Identifikation und Authentifikation
- Rechteverwaltung
- Rechteprüfung
- Beweissicherung
- Wiederaufbereitung
- Gewährleistung der Funktionalität

[eckert2011sicherheit]

1.7.1 Identifikation und Authentifikation

Identifikation und Authentifikation

- Abwehr von Maskierungsangriffen (Spoofing)
- Wann und Wer wird authentifiziert?
  - Welche Aktionen erfordern (welche Form der) Identität
  - z.B.: Systemzugang, DB-Zugriff,...
- Fehlerbehandlung

1.7.2 Rechteverwaltung

Rechteverwaltung

- Welche Subjekte auf
- Welche Objekte unter
- Welchen Bedingungen, mit

- Welchen Rechten?
- Rechtegranularität

1.7.3 Rechteprüfung

Rechteprüfung

- Policy Enforcement Points
- Bsp.: `open` file vs. `read` file
- Ausnahmebehandlung bei unauthorisierten Zugriffsversuchen

1.7.4 Beweissicherung

Beweissicherung

- Unabstreitbarkeit/Non-Repudiation herstellen
- Welche Ereignisse Wie protokollieren
- Zugriff auf Protokolle?
- Fälschbarkeit der Protokolle?

1.7.5 Wiederaufbereitung

Wiederaufbereitung

- z.B. Hauptspeicher, Festplatten, USB-Sticks
- Informationen sicher entfernen

1.7.6 Gewährleistung der Funktion

Unterthema Backup:

Gewährleistung der Funktion

> "Nobody is interested in Backups, what everybody wants is Restore." Michi Nagorsnik
>
> "Only wimps use tape backup: real men just upload their important stuff on ftp, and let the rest of the world mirror it ;)" Torvalds, Linus (1996-07-20)
>
> Kein Backup — Kein Mitleid!

Attackers
Hackers
Spies
Terrorists
Corporate Raiders
Professional Criminals
Vandals
Voyeurs

Tool
Physical Attack
Information Exchange
User Command
Script or Program
Autonomous Agent
Toolkit
Distributed Tool
Data Tap

Vulnerability
Design
Implementation
Configuration

Action
Probe
Scan
Flood
Authenticate
Bypass
Spoof
Read
Copy
Steal
Modify
Delete

Target
Account
Process
Data
Component
Computer
Network
Internetwork

Unauthorised Result
Increased Access
Disclosure of Information
Corruption of Information
Denial of Service
Theft of Resources

Objectives
Challenge, Status, Thrill
Political Gain
Financial Gain
Damage

Figure 1.10: Attack Terminology [**howard1998common**]

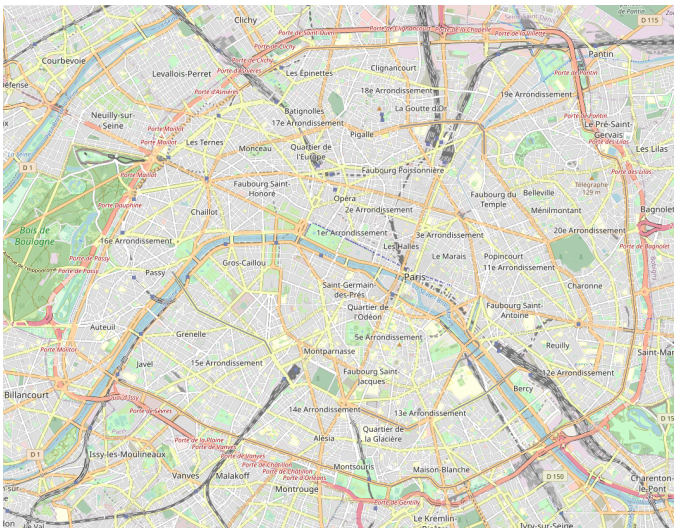
2

Network Architecture

2.1 Zones and Conduits

An old and proven principle of physical and digital security is separation. It can be found in the Saltzer and Schröder Security principles as well as in the architecture of castles, office buildings or law.

Zones and Conduits



2.1.1 Network Segmentation

In networking, separation

ICS Networks

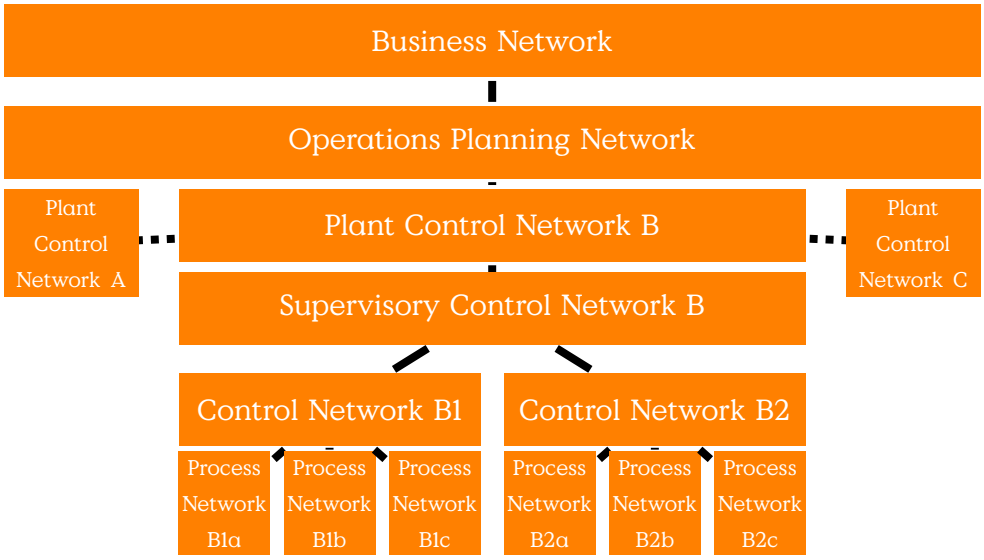


Figure 2.1: Conceptual representation of network segmentation. Based on [**knapp2015ics**].

System Intrusion

- Significant issue hostile/unwanted trespass
  - from benign to serious
- User trespass
  - unauthorized logon, privilege abuse
- Software trespass
  - virus, worm, or trojan horse
- Classes of intruders:
  - Masquerader: (Outside) user, who penetrates a system's access control to exploit user accounts
  - Misfeasor: Insider user, who accesses data, programs, or resources unauthorized, or who misuses authorized access
  - Clandestine user: Manipulates supervisory control of system, e. g., to evade auditing and access controls

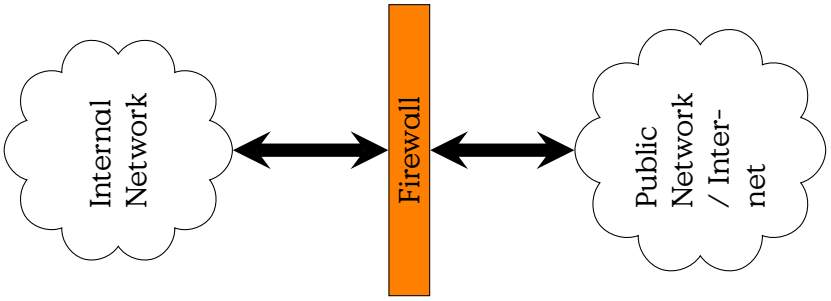


Figure 2.2: Firewall Principle

2.2 Firewalls

Overview

- Firewalls
  - prevention of intrusion
  - separation of segments
- Firewall Architecture
  - Layered Systems
  - Functional/Criticality
- Intrusion Detection
  - Eyes and Ears
  - Host and Network

A firewall is a set of one or more components, of hard- and software, through which all traffic between an inside network and and outside network is directed.

Firewall

Firewall Types

Firewall "An internetwork gateway that restricts data communication traffic to and from one of the connected networks (the one said to be "inside" the firewall) and thus protects that network's system resources against threats from the other network (the one that is said to be "outside" the firewall).

[...] A firewall is not always a single computer. For example, a firewall may consist of a pair of filtering routers and one or more proxy servers running on one or more bastion hosts, all connected to a small, dedicated LAN (see: buffer zone) between the two routers. The external router blocks attacks that use IP to break security (IP address spoofing, source routing, packet fragments), while proxy servers block attacks that would exploit a vulnerability in a higher-layer protocol or service. The internal router blocks traffic from leaving the protected network except through the proxy servers. The difficult part is defining criteria by which packets are denied passage through the firewall, because a firewall not only needs to keep unauthorized traffic (i.e., intruders) out, but usually also needs to let authorized traffic pass both in and out." [**Shirey07InternetSecurityGlossary**]

Also see "guard" in [**Shirey07InternetSecurityGlossary**]

Filtering Router "An internetwork router that selectively prevents the passage of data packets according to a security policy." [**Shirey07InternetSecurityGl**

Packet Filter A firewall working on ISO/OSI-layers 3 and 4 (and sometimes 2 as well).

Proxy Firewall A proxy firewall acts as intermediary at ISO/OSI-layers 3 and 4. The proxy is presenting itself as the server to the exterior world, usually accepts the communication and may provide additional security services, like authentication, access control, auditing.

Usually the requests to the service are then posted by the firewall to the service provider, this means that the service will only receive communication from the firewall. Communication may also be repacked, eg. into different network protocols.

Application Firewall A application firewall provides an application specific intermediary that shields the interior application by acting instead of this application to the exterior. An application firewall understands the application protocols and may re-encode or otherwise "clean" the requests. This allows the firewall to undertake application specific analysis.

Bastion Host "A strongly protected computer that is in a network protected by a firewall (or is part of a firewall) and is the only host (or one of only a few) in the network that can be directly accessed from networks on the other side of the firewall." [**Shirey07InternetSecurityGlossary**]

2.3 Firewall Architecture

A good explanation is found at: http://www.invir.com/int-sec-firearc.html

2.3.1 Subnetworking Fundamentals

We would hope that this is already well known to any computer related person.

IPv4 Recap

An Internet address, with a few exceptions, consists of a network and a host part. The host part is defined by a host mask where the binary 1es define which bits are included in the host part of an address.

Thus, different sub-networks are differentiated by different network parts of the address. Usually the network part consists of a sequence of bits at the beginning of an address and the host part is the remainder.

IPv4 Subnet Example

32 Bit: 141.99.96.123

Hostmask: $\underbrace{11111111.11111111.11111111}_{network}.\underbrace{00000000}_{host}$

Hostmask/24 255.255.255.0

Hostmask/16 255.255.0.0

Hostmask/22 255.255.253.0

IPv4 Address Registry[1] defines, in summary:

IPv4 Address Spaces

0.0.0.0 Network Address

10.0.0.0/8 Private Use

78.0.0.0/8 RIPE NCC

127.0.0.0/8 Loopback [RFC 1122]

192.168.0.0/16 Private-Use Networks [RFC 1918]

[1] http://www.iana.org/assignments/ipv4-address-space/ipv4-address-space.xml

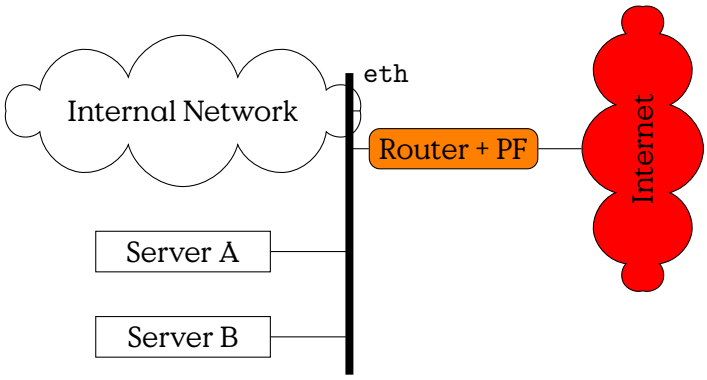


Figure 2.3: Screening Router

224.0.0.0/8 Multicast

255.255.255.255 Limited Broadcast

IPv6 Addresses

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.

2.3.2 Compartmentalisation

Please also see [**CZ95BuildingInternetFirewalls**] for more information.

Screening Router

Dual-Homed Host

Screened-Host Firewall

Screened-Subnet Firewall

Compartmentalisation means the separation

De-Militarised Zone (DMZ)

Screening Router

Screening Router

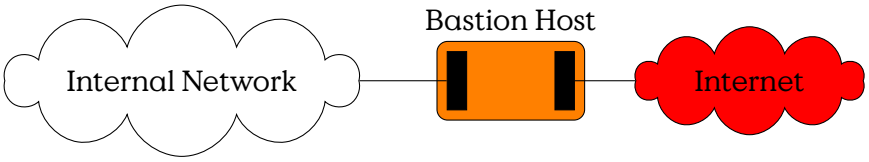


Figure 2.4: Screened-Host Firewall

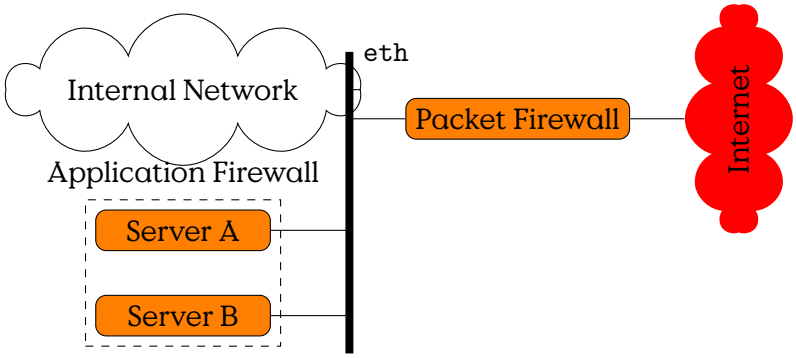


Figure 2.5: Screened-Host Firewall

Dual-Homed Bastion

Dual-Homed Bastion Host

Screened-Host Firewall

Screened-Host Firewall is a firewall architecture, where the routing firewall screens single hosts. Public servers, application firewall and internal network are situated on the same network segment (ethernet segment). This means, that the internal network is not separated from the internal network by a firewall.

Screened-Host Firewall

Screened-Subnet Firewall

A screened-subnet firewall goes one step further in that it has an additional firewall that separates internal from public networks.

Screened-Subnet Firewall

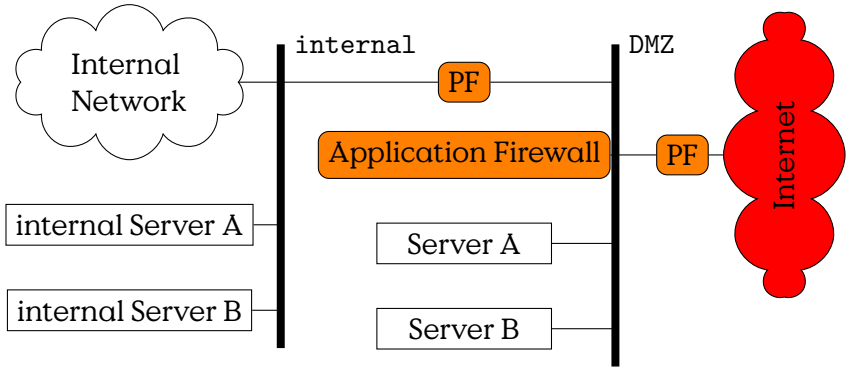


Figure 2.6: Screened-Subnet Firewall

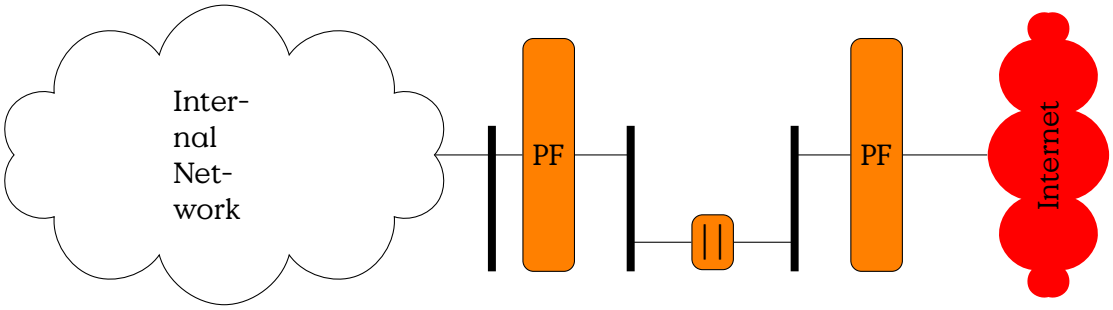


Figure 2.7: Split-Screen Firewall (Belt-and-Suspenders)

Split-Screen Subnet

A Screened Subnet architecture with an additional Dual-Homed Bastion Host that splits the Demilitarised Zone into two separate sub-networks.

Split-Screen Subnet

(also called Belt-and-Suspenders)

2.3.3 Packetfilter Firewalls

A Packetfilter Firewall is working on individual network packets (i. e., IP-packets including layer 4 headers).

Packet Filter Firewall

Stateless A stateless filter can make descisions based only on the contents of the currently inspected packet. It can not include information gathered from previous packets into its decision — that is,

because it has no state (i. e., memory).

Stateful A stateful firewall has a memory and can thus dynamically adapt its current decisions based on previous events. It could, for example, allow an incoming packet of a TCP-connection, if the establishment of that connection has previously been observed.

Stateless Example Filter Rules

```
iptables -P FORWARD DROP
iptables -A FORWARD -p tcp --dport 80,443 -j ACCEPT
iptables -A FORWARD -p tcp --sport 80,443 -j ACCEPT
```

Stateful Example Filter Rules

```
iptables -P FORWARD DROP
iptables -A FORWARD -p tcp --dport 80,443 \
              -s 192.168.1.0/24 -j ACCEPT
iptables -A FORWARD -p tcp -m state \
     --state ESTABLISHED,RELATED -j ACCEPT
```

Further rule-specifications

-i/-o Input/Output-interface

Rules for `iptables` are sorted in tables and chains. Tables are enabled by specific kernel-modules. Chains more specifically define which packets are addressed within a given table.

Tables and Chains

filter default for packets ...

FORWARD ... routet through the box

INPUT ... destined to local sockets

OUTPUT ... generated locally

nat packets for new connections are handled and modified here

PREROUTING as soon as packets enter the system

INPUT packets to local sockets

POSTROUTING packets before sending them out

Further tables: raw, security

2.4 Intrusion Detection

IDS Overview

- Network: e. g., Snort
- Host: e. g., Tripwire

Intrusion Detection

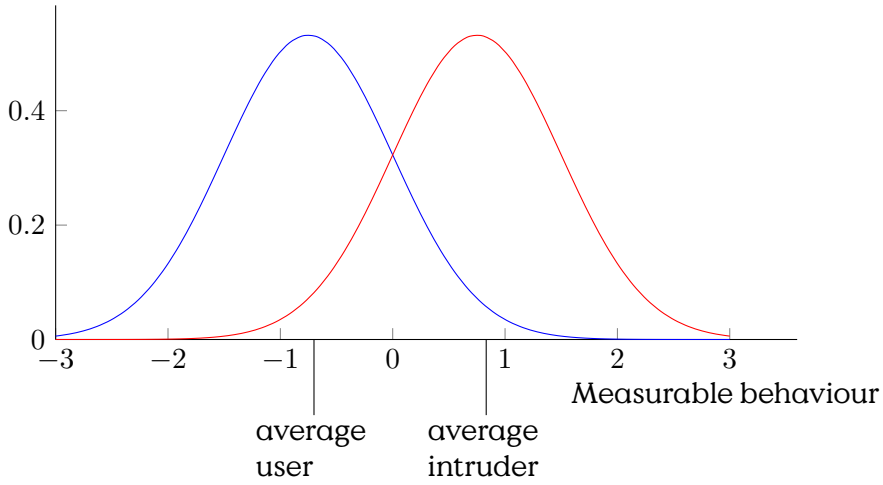
- Observations of non-attacked systems
  1. User, process actions conform to statistically predictable pattern
  2. User, process actions do not include sequences of actions that subvert the security policy
  3. Process actions correspond to a set of specifications describing what the processes are allowed to do
- As sumption underlying intrusion detection
  - Systems under attack do not meet at least one of the points

Intrusion Detection Systems

- Intrusion detection models:
  - Anomaly detection
  - Signature detection
- Intrusion detection structures:
  - Host-based IDS: monitor single host activity
  - Network-based IDS: monitor network traffic
- Logical components:
  - Sensors - collect data
  - Analyzers - determine if intrusion has occurred
  - User interface - manage / direct / view IDS
- Input of IDS: Audit records

2.4.1 Anomaly-based IDS

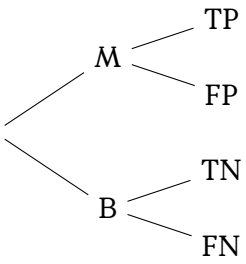
Intrusion Detection Principles



Base-Rate Fallacy

Extremely rare events — Many False Errors

Accuracy 99% (i. e., for every 100 benign events there will be 1 false positiv.



Assume: 0.95 of traffic is benign (i. e., 5% is malicious)

$$
\begin{aligned}
P(TP) &= 0.95 \cdot 0.99 = 0.94 \\
P(FP) &= 0.95 \cdot 0.01 = 0.009 \\
P(TN) &= 0.05 \cdot 0.99 = 0.04 \\
P(FN) &= 0.05 \cdot 0.01 = 0.0005
\end{aligned}
$$

400.000 events/day $\Rightarrow$ 3.600 False Alarms (200 undetected intrusions)

Statistical Moments This term (used in the slides) means the mathematical term of moments. First and Second Moment are probably better known as mean and variance by the audience.

2.4.2 Signature-based IDS

Example Snort Rules

```
# alert tcp $HOME_NET 2589 -> $EXTERNAL_NET any
  ( msg:"MALWARE-BACKDOOR - Dagger_1.4.0";
    flow:to_client,established;
    content:"2|00 00 00 06 00 00 00|Drives|24 00|",depth 16;
    metadata:ruleset community;
    classtype:misc-activity; sid:105; rev:14; )
```

- Intern to Extern
- TCP port 2589 to any port
- Contains "`2|00 00 00 06 00 00 00|Drives|24 00|`"

```
# alert tcp $EXTERNAL_NET any -> $HOME_NET 7597 ( msg:"MALWARE-
BACKDOOR QAZ Worm Client Login access"; flow:to_server,established; content:"qa
activity; sid:108; rev:11; )
# alert tcp $EXTERNAL_NET any -> $HOME_NET 12345:12346 ( msg:"MALWARE-
BACKDOOR netbus getinfo"; flow:to_server,established; content:"GetInfo|0D|"; me
activity; sid:110; rev:10; )
```

3

Access Control

This chapter is presenting selected mechanisms for authorisation and management of access rights.

The first computers where resources that where protected for themselves by keeping them in secured areas. Anyone who could access these areas (and had the knowledge to operate them) could use these computers. Access control was implemented in classical ways that restricted physical access. The most fundamental method probably is a door-lock. In this chapter we will concern ourselves with the access control methods that came with multi-user systems, networked computers and digital objects.

Access control allows or denies access to given resources by a given principal in a defined way. It requires authentication to establish the identity of the principal. It consists of an access control system, that makes decisions based on an access control database. In professional context, it further is completed by a separate auditing system.

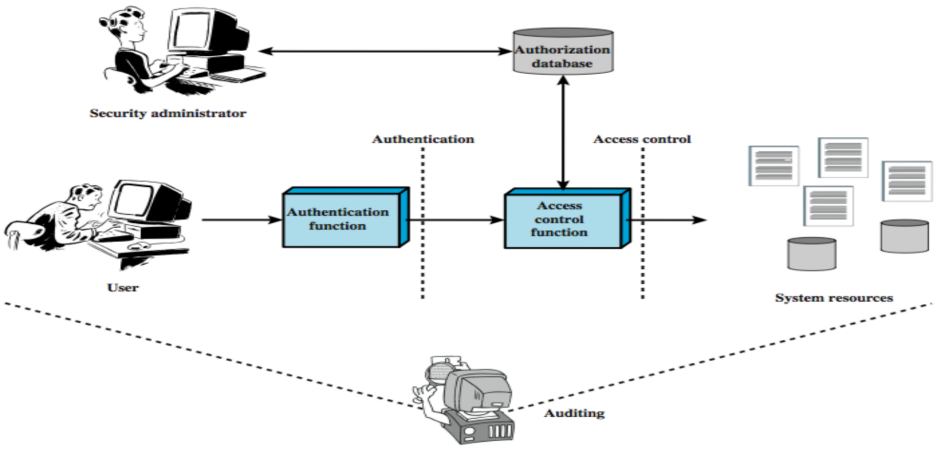
3.1 Access Control Components

Authentication function: Verification of claimed identity (See Chapter "Authentication")

Access control function: Determines if specific requested access of user is permitted based on authorisation database.

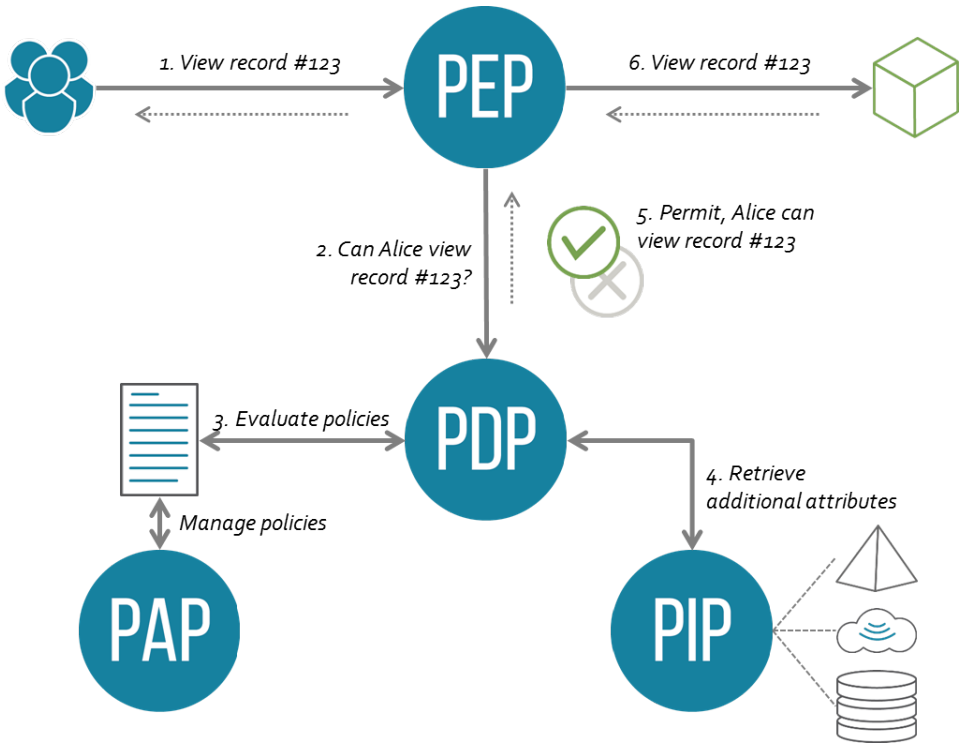
Audit: Monitors and keeps record of user accesses

Access Control Components



Let us take a closer look at the components of an access control function:

XACML Authorisation Framework



PAP (Policy Administration Point) Point which manages access authorization policies

PDP (Policy Decision Point) Point which evaluates access requests against authorization policies before issuing access decisions

PEP (Policy Enforcement Point) Point which intercepts user's access request to a resource, makes a decision request to the PDP to obtain the access decision (i.e. access to the resource is approved or rejected), and acts on the received decision

PIP (Policy Information Point) The system entity that acts as a source of attribute values (i. e., a resource, subject, environment)

PRP (Policy Retrieval Point) Point where the XACML access authorization policies are stored, typically a database or the filesystem.

Authorisation Policies

Discretionary Access Control (DAC) ("discretionary": (de) „Handlungsfreiheit", „Ermessens...")

Mandatory Access Control (MAC) ("mandatory": (de) „obligatiorisch", „zwingend")

Role-based Access Control (RBAC): Access Control based on roles instead of subjects. Is the most common type of access control in medium to large organisations.

3.2 Privilege-Separated Operating System

This section is a stub.

Privilege-Separated Operating System

- distinct identities (e. g., user,group,...)
- processes/applications related to identity
- e. g., Android (Linux), current Windows

We are talking about android[1].

3.3 Access Control Function

The Access Control Function embodies the decision of whether an action is allowed or not in a given execution context.

[1] https://developer.android.com/guide/topics/security/permissions.html, 2013-07-02

3.4 Process Space Protection

This topic is intentionally left out. Please refer to a lecture on operating systems.

3.5 Discretionary Access Control

3.5.1 Access Control Matrix

An Access Control Matrix (ACM) is a representation of access rights in discretionary access control. It consists of a list of all subjects and all objects represented in rows and columns and thus provides a complete model of all access rights. The fields of the matrix hold the access rights of the subject (in our examples found in the rows) to an object (in our examples found in the columns).

Access Control Matrix (ACM)

- Policy representation for discretionary based authorization
- Let
  - S be set of all subjects,
  - O be set of all objects and
  - R be set of all operation rights.
- Access Control Matrix (ACM) M defines for pairs $(s \in S, o \in O)$ set of possible operations $M_{s,o} \subseteq R$.

	o_1	o_2	o_3
s_1	rw	w	r
s_2		rw	rwx

Access Rights

It is not difficult to creatively define a whole heap of different rights. The question is, how much granularity you need. In the following we produced a summary list of the most common rights.

Access Rights

Owner is a "special" right that describes that a subject has all rights on an object (because it owns it).

Read/Write/Execute are the classic rights you probably know from using any standard Unix-like system. Reading and writing describe the rights to read data from an object and modify or add to its contents. By execution we describe the right to run the object as a program.

Append is a right rarely found, that describes, that one is not allowed to alter existing contents of an object, but may only add contents at the end of the object.

Create/Delete are slightly different from Write in that they allow to completely create or remove an object and not only its contents. This includes creation and removal in the ACM. Often Create and Delete are implemented as the right to Write of a directory-like object.

Search means the right to display contents of a directory. (This is again often represented by Read on the directory.)

The most common implementations have Read, Write and Execute rights and implement Create, Delete and Search as Read and Write on special directory-objects.

Operations on ACM

It is sensible to formally define operations on an Access Control Matrix to allow for formal analysis of state transitions. The operations either delete or create subjects (rows), objects (columns), or rights (in fields).

An operation is defined by pre-conditions that must be given before an operation may take place, and post-conditions that specify the effect of the operation. For the set of subjects S, of objects O, of rights R, and an ACL A. A system state is described by the triple (S, O, A). An operation c defines a state transition $(S, O, A) \underset{c}{\mapsto} (S', O', A')$:

ACM Operations

add subject s **Pre-Condition:** $s \notin S$

Post-Condition:
- $S' := S \cup \{s\}$ and $O' := O \cup \{s\}$ (create user)
- $\forall y \in O' : A'[s, y] := \varnothing$ and $\forall x \in S' : A'[x, s] := \varnothing$ (empty rights by default)
- $\forall x \in S, \forall y \in O : A'[x, y] := A[x, y]$ (copy existing)

delete subject s **Pre-Condition:** $s \in S$

Post-Condition:
- $S' := S \backslash \{s\}$ and $O' := O \backslash \{s\}$ (delete user)

- $\forall x \in S', \forall y \in O' : A'[x, y] := A[x, y]$ (copy remaining)

create object o **Pre-Condition:** $o \notin O$

Post-Condition:
- $O' := O \cup \{o\}$ and $S' := S$ (create object)
- $\forall y \in S' : A'[y, o] := \varnothing$ (empty rights by default)
- $\forall x \in S, \forall y \in O : A'[x, y] := A[x, y]$ (copy existing)

delete object o **Pre-Condition:** $o \in O$ and $o \notin S$

Post-Condition:
- $O' := O \backslash \{o\}$ and $S' := S$ (delete object)
- $\forall x \in S', \forall y \in O' : A'[x, y] := A[x, y]$ (copy remaining)

add right r **to** (s, o) **Pre-Condition:** $s \in S$, $o \in O$, and $r \notin A[s, o]$

Post-Condition:
- $S' := S, O' := O$ (copy subjects)
- $A'[s, o] := A[s, o] \cup \{r\}$ (redefine matrix cell)
- $\forall x \in (S \backslash \{s\}) : \forall y \in (O \backslash \{o\}) : A'[x, y] := A[x, y]$ (copy existing)

delete right r **from** (s, o) **Pre-Condition:** $s \in S$, $o \in O$, and $r \in A[s, o]$

Post-Condition:
- $S' := S, O' := O$ (copy subjects)
- $A'[s, o] := A[s, o] \backslash \{r\}$ (redefine matrix cell)
- $\forall x \in (S \backslash \{s\}), \forall y \in (O \backslash \{o\}) : A'[x, y] := A[x, y]$ (copy existing)

There are additional operations that could be defined on ACM, for example, transfer or grant of rights.

ACM Implementation

Issues of access control matrix:

Issues with ACM Implementation

Memory Intensive ACM are not suitable for implementation on large systems with many subjects or objects, i. e., they require $|S \times O|$ fields.

Complex Lookup ACM are not stored with either subjects or objects, i. e., they have to be separate from the objects concerned by them, which requires proceedures and storage additional to object-access for the lookup.

Furthermore, a large structure where individual entries are rarely accessed is not efficient to be kept in memory, thus lookup-strategies have to be implemented, which increase the complexity of the system.

Sparse Matrix Many entries in matrix will be empty, but memory is nevertheless allocated or complex algorithm (e. g., hash-tables) have to be deployed.

$\Longrightarrow$ Access control matrix is only abstract concept

Solution:

- Ignore empty matrix entries, i. e., no memory allocation for subject s without rights on object o.
- Store rights with subject or object:

  Capability List (CL) is a row-wise implementation of an ACL. Defines the rights of a subject to a list of objects. Can be compared to a credential that can be used to claim rights to objects, e. g., the paperwork that proves your ownership of a car.

  Access Control List (ACL) a list of access rights, related to given objects. Column-wise implementation of an ACL. Can be compared to a VIP-list at a concert, where the concert relates to an object and the list enumerates the subjects that have access right.

3.5.2 Capability Lists

You may compare capabilities to tickets that allow you entry to a concert. A capability is kept with the subject that enjoys the access rights granted by that capability.

- Each subject has a capability, which is an unforgeable token
- Capability corresponds to list of access control entries (row of Martrix)
- Each entry contains an object o and the rights of s on object o
- Subject can create object and grant other subjects capabilities on object.
- Subject s may be allowed to pass parts of capability to other subject s', e. g., in case that s invokes a process s'.

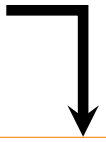
Issues:

- Who may read this file? It is difficult to find out which subject has access on which object.
- Revocation is difficult: Hard to revoke capability, because difficult to track transfer of access to other subjects.

Capability Lists

ACM

	o_1	o_2	o_3
s_1	rw	w	r
s_2		rw	rwx



Capabilities $s_1 \longrightarrow [(o_1, rw), (o_2, w), (o_3, r)]$

3.5.3 Access Control List (ACL)

Access Control List (ACL)

- Each object o is associated with list of access control entries (column of matrix).
- Each entry contains a subject and its rights on object o

Unix Access Control

Operation systems often implement a hybrid approach to access control utilising both Access Control List (ACL) and Capability Concepts.

In Unix-Systems, the primary access control model, represented in the file-system, is implemented as an ACL. But the call `open` provides processes with a file-descriptor, which is used to access the file without further verification of the ACL in the file system. File-descriptors thus resemble temporary capabilities, i. e., they are attached to the process acting on behalf of a subject.

The Unix-model provides a small set of rights (read, write, execute) but also an extension to the subject model. Each object has rights assigned for the "owner" of the file, which is an identity from the systems subject list. Additionally the Unix ACL also defines rights for a "group" of subjects and a different set for the rest of the "world". The group is a set of user identities managed in a special system group file.

Unix

3.5.4 DAC Safety

The main problem of ACM-based systems is to decide if there is a sequence of transitions that may lead to an undesired state, given the operations on an ACM.

The first problem is to define what states are desired and which are undesired.

Safety question: Is it possible to reach a state within which a right can be given or an operation executed that is not "intended"? For example: Can an attacker grant access rights to an object without consent of the owner of that file? A system is considered safe, if it can be proven for every safe state that there exists no sequence of commands[2] that lead to unsafe states.

mono-operational Commands that are only mapped onto single primitive operations. For example there is no command that both creates a file and changes a right. Which implies that all newly created objects have the same rights.

primitive operation operations that only produce single effects

[2] i. e., state changes

3.6 Mandatory Access Control

Mandatory Access Control provides a concept for systemic control of resources. Different from DAC, some baseline access rules are set by the access control system an cannot be changed by users (or administrators). Often the objective is to provide control over information flow.

3.6.1 Bell-La Padula

The Bell-LaPadula model aims at flow control, by preventing information flow from higher security levels down to lower security levels.

Bell-LaPadula

Security Levels $L : S \cup O \rightarrow$

Top Secret

Secret

Confidential

Unclassified

Objects have security classification $L(o)$

Subjects have clearance levels $L(s)$

BLP: Accepted Information Flow

No information shall flow downwards!

1. Simple Security Property/no-read-up rule— no subject of a lower security clearance may read from an object of a higher security classification
2. *-Property/no-write-down rule— no subject of a higher security clearance may write to an object with lower security classification.
3. Discretionary Security Property— an additional ACM implementation is used to further restrict access.

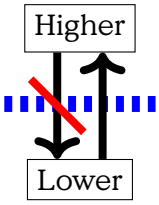
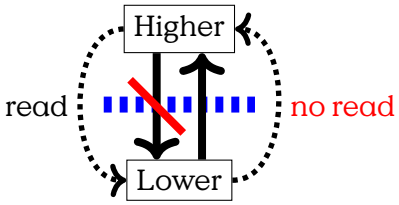


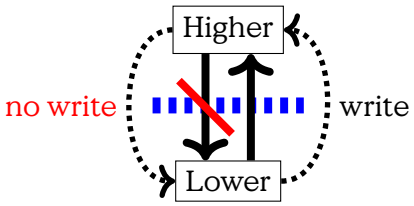
Figure 3.1: Bell-LaPadula Information Flow Directive

BLP: Simple Security Property



Subject s can read from object o if $L(s) \geqslant L(o)$ AND Discretionary Access is granted

BLP: *-Property



Subject s can write to object o if $L(s) \leqslant L(o)$ AND Discretionary Access is granted

Summary and Discussion

BLP: Issues

- Blind writing: A user may not read the effects of the modifications he just has made to a file, if he has lower clearance. This is problematic with respect to the next point.
- Low integrity protection: Subjects are able to blindly write upwards. As subjects with lower clearance are able to write to objects with higher classification, the multi-layer security model do not increase protection against manipulation.

- Covert Channels (see following slide)

3.6.2 Covert Channels

It is actually very hard, if not impossible to completely control all information flows.

2-Phase Locking Exploit

2-Phase locking is a method for concurrency control[3].

2-Phase-Locking Exploit

Use file-locking for communication:

- Given s_h, s_l, o_l: Subjects with high and low clearance and object with low classification.
- s_h may read o_l, s_l may write
- Read operation triggers lock
- Essential binary signal from high to low.

3.7 Role-based Access Control

The intention of Role-based access control is to simplify the task of managing access rights by providing functional roles. In professional context, the obvious question before granting a right is: For what is it necessary that an individual is able to do this? And the obvious generalised answer almost always will be: to fulfil its function. Thus the better way to manage access rights is to model the access rights of functions and then simply assign functions to individuals. This way of dealing with rights seems obvious and is probably used for ages outside the digital world.

Imagine system with 1000 subjects, 100000 objects and 10 access rights, there are $1000 \times 100000 \times 10 = 10^9$ authorization triples

- Reducing complexity to maintain access control lists like in CL and ACL by introducing intermediate layer between subject and object called roles
- Access rights are assigned to roles instead of to single subject, thus subject accessing an object must have corresponding role

[3] https://en.wikipedia.org/wiki/Two_phase_locking

- Subject can have several roles
- There are less roles than users, thus reducing access control overhead

Role-Based Access Control (RBAC)

User an individual (with access to the system)

Role a named job function. A role definition should describe the responsibility and authority of any user assuming this role.

Permissions the access rights of a role.

Session is a temporally valid mapping from user to a set of activated roles.

Constraints restrict the accepted types of sessions, i. e., which roles are allowed to be active at the same time (dynamic constraint). Can also define roles that may not be assigable to the same user (static constraint), i. e., the function behind the roles should not be combined in the same user.

[**Stallings11CryptographyandNetwork**]

User — Role — Object

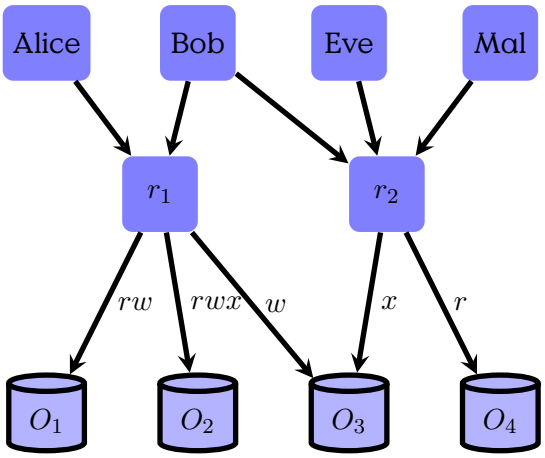


Figure 3.2: Multiple users can be assigned roles. Access rights are granted to roles only.

Figure 3.2 shows the mapping of users (subjects) to roles, which are then mapped onto access rights on objects. Roles take up the position of sub-

jects in the ACM model, which is beneficial if the number of roles is smaller than the number of users.

Furthermore, Roles allow to model access rights based on function of, i. e., the actual requirements of a position (job) with associated tasks. In practise this allows for more generic access right assignments, makes the idea behind the assignments much more transparent and provides long-term solutions.

Users/Persons can now be assigned to roles, depending on their job description.

The next step now is to define constraints on the roles that can be assigned to a user. That is, as roles define positions of subjects, we can now identify pairs of roles that are mutually exclusive. Perhaps because the function linked with these rules generally pose a conflict of interest, or, because those roles should not be assumed at the same time (to prevent information spill between different work areas).

Role Constraints

dynamic : not concurrent in same session by same subject

- e. g., Administrator and User
- Cashier and Guard
- CustomerAccount A and CustomerAccount B
- max. number of roles

static : not available to same subject

- e. g., President and Judge
- Manager and Employee

3.8 Location-Based Authorisation

This section introduces the concept of authorisation based on the current position of an entity. We utilise an example of a locking mechanism for doors that often has obvious spatial requirements before it opens.

Location-based Authorisation

- Based on Authentication
- Access depends on position

- Relation between person and "key"
- Example: Door Lock

A door usually is a device that provides access to a given area or passage. Often this function is coupled with the function of denying access to unauthorised persons.

Let us briefly summarise common requirements for authorisation of entry.

Requirements

Authenticity Usually entry is allowed only a defined set of entities, which have to prove that they belong to that set. This is a special problem of authenticity.

Freshness It often must be assured, that a request for entry is current and not a replayed request. This is less obvious in the "real" world, as it is such a fundamental attribute of physical interaction that we rarely think about it. In electronic communication it is, nonetheless, an attribute that is not necessarily given.

Spatial Closeness You usually want that the person opening a door is the one standing close to the door. Even if you do not think about replay and masquerading attacks, you also want to prevent, that some authorised entity opens a door without overseeing who is entering that door.

Temporal Interval Some doors will only open at certain times, or at certain times for some and other times for others. Bank vaults are a common example.

Revocation Withdrawing the right of passage is a quite common operation connected with doors. If the reason why a right of passage has been granted becomes invalid, it must be possible to revoke that right.

Unlinkability It should not be possible to know, whether two different opening requests have been initiated by the same person. It has to be decided whether this requirement should only consider a passive global observer or the authorisation system itself as "attacker". You may request various levels of privacy. Usage of pseudonyms is rather common in digital context. Unlinkability provides a stricter requirement, that is more similar to the use of keys.

3.8.1 Scenario

Scenario

- person
- mobile device (smartphone)
- door lock
- authorisation/authentication server
- communication medium (local WLAN, global Internet)

3.8.2 Solutions

Solutions for Presence

- Cryptographic challenge
  - at door (e.g. QR-Code Display)
  - local number radio
  - ...WLAN/IP Multicast
  - Polling server locally (WLAN)
- Spatial Closeness? (WLAN, changing challenge at the door?)
- Zero-Knowledge Proofs for Unlinkability/Anonymity

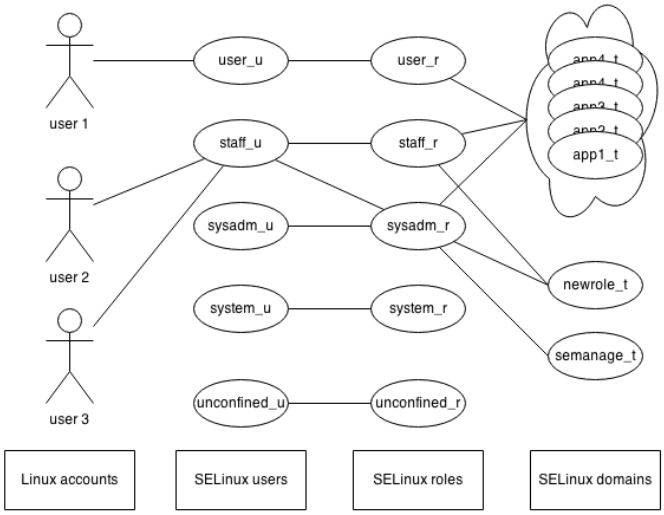
3.9 SELinux

SELinux provides an enhancement to Linux that provides Role-Based Access Control (RBAC) and MAC.

SELinux

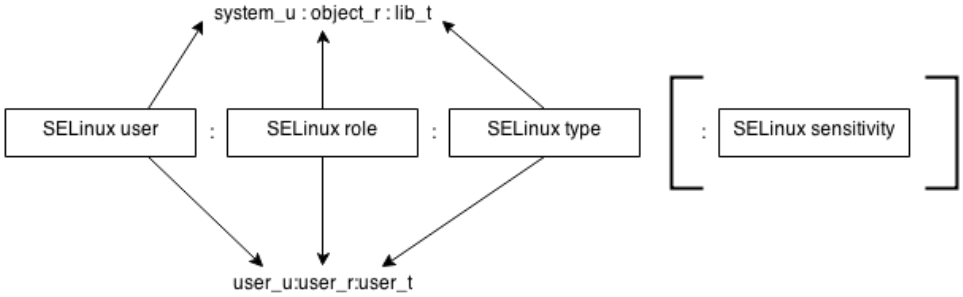
- Enhancement to Linux Kernel
- Provides
  - Message Authentication Code (MAC)
  - RBAC

SELinux Users



[source: Gentoo Wiki]

Security Enhanced Linux (SELinux) Context



[source: Gentoo Wiki] The context of a subject is given by its domain, defined as user, role and type. The important field for RBAC authorization is the type-field, which is checked against SELinux rules.

Rules

deny/allow —— subject label —— object label —— object class —— access right

```
allow user_t bin_t:file read;
```

- Users act only through processes (as usual)
- A Linux user account is mapped onto exactly one SELinux user
- A SELinux user is allowed one or more SELinux roles.

- A role is allowed a defined set of domains.
- Every process, i. e., subject, is, at any time, active in exactly one context.
  - Running shell in new context: `newrole -r ROLE -t TYPE`

4

Grundlagen Kryptographie

Lernziele

- Verständnis von Zufälligkeit?
- Ideal Block Cipher
- Symmetrische Verschlüsselung
- Einwegfunktionen
- Asymmetrischen Verschlüsselung
- Digitale Signaturen/Credentials
- Cryptographische Hashes
  - Eigenschaften (unvorhersehbar, schwach/stark kollisionsresistent)
  - Birthday Paradox

4.1 Symmetric Cryptography

Symmetric cryptography is the concept of the category of algorithms that use the same,i. e., a single key, for encryption and decryption of messages. The concept has two major fundamental problems: This means that you can never be sure, that your communication partner has the same security standards as you have. If the key is the foundation of a long-term secure communication relation, this is problematic, as with the passing of time, the possibility, that the key is somehow compromised is increased.

The second problem is, that each key is usable only for one communication channel. The result is, that you need a single (secret) key for every

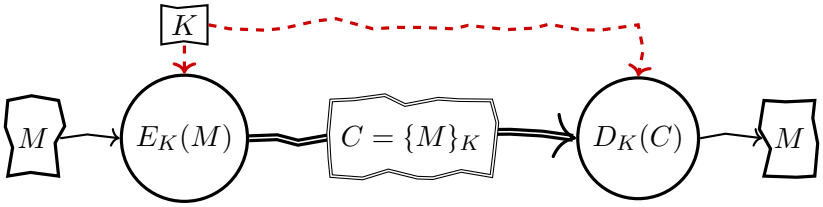


Figure 4.1: Symmetric Cipher Model

communication partner, that you can never share with a third person. Thus the number of all keys necessary to allow secure communication between n communication partners is n^2, i. e., quadratic growing to the number of communication partners. Also, every communication channel requires the perfectly secure exchange of the secret key before any communication can take place, or every time a key is potentially compromised. In practise this turns out to be very difficult and makes the establishment of communications too complex for every area but classical high security areas.

Symmetric cryptography, also called private- or single-key cryptography, has been the only way to do cryptography (if you don't count steganography) until the 1970's. Thus, until then cryptography has been a tool only for very limited personal communication or military use.

Symmetric Cryptography

- until 1970, the only encryption method
- Sender and Recipient share one key
- also:private- or single-key cryptography

Objective: hide information in transit

Symmetric Cipher

Plaintext M**:** original message

Ciphertext: scrambled message

Key K**:** control exact substitutions/transformations used in cipher

Encryption E**:** performs substitutions/transformations on plaintext

Decryption D**:** inverse of encryption algorithm

> Verschlüsselung ist ein Vorgang bei dem ein Datenblock derart verändert wird, dass er nur mithilfe eines Schlüssels wieder

in seinen Ursprungszustand versetzt werden kann. Praktisch bedeutet dies, dass Verschlüsselung den Datenblock für alle uninterpretierbar macht, die den Schlüssel nicht kennen.

4.1.1 Requirements for Secure Symmetric Encryption

Requirements for secure use of symmetric encryption

- A strong encryption algorithm
- A secret key known only to sender / receiver

Mathematical description:

- Encryption: $Y = E_k(X)$, where k is key
- Decryption: $X = D_k(Y)$

Assumption:

- Encryption algorithm is known to everyone
- Hard to decrypt message given ciphertext and encryption alg.
- Therefore sufficient to keep key secret
- There is a secure channel to distribute key

4.1.2 Historic Examples

Monoalphabetic ciphers are now only found in the history section of reading books. It is simply much too easy to break such things as the classical Cesar Cipher using simple frequency counts. Nowadays all ciphers are at least polyalphabetic substitution algorithms. But much more likely they are product ciphers that utilise complex combinations of substitution and transposition.

Nonetheless, historic ciphers are a very good starting point to learn about the basic mechanisms of cryptographic algorithms. Albeit, it should be obvious, why they are called "historic".

Monoalphabetic: Cesar Cipher

The most often used starting point and the most fundamental substitution cipher. It uses a single integer k as a key and works on every ordered alphabet (which practically is every alphabet ever used). It works by replacing each letter in a text by the k's letter in the alphabet counting

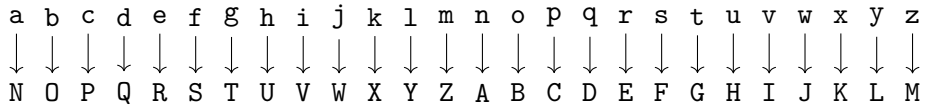


Figure 4.2: Cesar Monoalphabetic Substitution Cipher

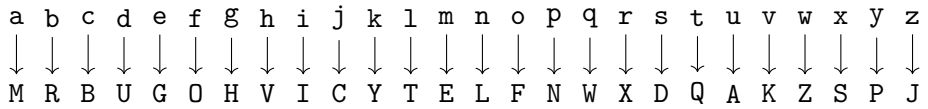


Figure 4.3: Random Monoalphabetic Substitution Cipher

from the former one. At the end of the alphabet, the count is continued from the first letter in the alphabet, which resembles the modulo-operation, that we will meet much more often in the near future.

Cesar Cipher

Monoalphabetic: General Single-Letter Substitution

A slight extension to the Cesar Cipher is the generalisation uses a permutation of the alphabet as key.

General Monoalphabetic Substitution

I used the following simple snippet to generate a random permutation of a list of letters

```
import Data.List
import System.Random

select [] = undefined
select xs = randomRIO (0,length xs - 1) >>= (\r -> return$(\
   xs !! r, take r xs ++ drop (r+1) xs))

shuffle (out, []) = return (out,[])
shuffle (out, xs) = select xs >>= (\(r,rem) -> shuffle (r:\
   out, rem))
```

Monoalphabetic Digraph: Playfair

The Playfair Cipher increases the complexity of the classical monoalphabetic ciphers by encrypting digraphs instead of single letters. This

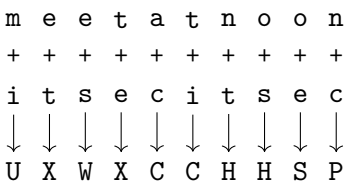


Figure 4.4: Vigenère Encryption of "meetatnoon" using key "itsec"

increases the complexity of frequency analysis as the number of digraphs is approaching the number of letters squared.

The idea is to randomly arrange the complete alphabet in a box, e. g., 25 letters of the english alphabet assuming "I" and "J" are equal. Then the plaintext is changed into a sequence of pairs of letters. Now, for every pair find their locations in the alphabet-box and substitute following these three rules:

- if both letters are in the same row, encode each letter with its right neighbour (rolling over to the left side if necessary),
- if both letters are in the same column, encode each letter with its neighbour below (rolling over to the top if necessary),
- if both letters form the edges of a box, encode each letter with the letter in the same row, in the opposite corner of that box.

Polyalphabetic: Vigenère

The Vigenère Cipher extends the Cesar Cipher (Section 4.1.2) by using a sequence of keys, represented as code word, onto subsequent letters of the plaintext.

Vigenère Cipher

- Codeword determines encryption

Transposition Ciphers

Transposition ciphers work by defining permutations to text blocks.

Transposition Ciphers

plaintext: hide the parcel behind the car

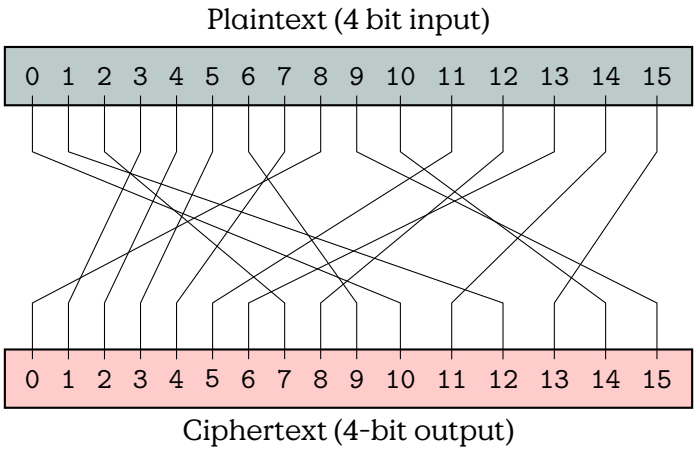


Figure 4.5: Block Cipher

Columnar transposition:

3	2	4	1	5
H	I	D	E	T
H	E	P	A	R
C	E	L	B	E
H	I	N	D	T
H	E	C	A	R

Key: 32415

ciphertext: HHCHHIEEIEDPLNCEABDATR

S-DES

S-DES is a smaller, more easy to learn variant of DES. It is, by no means to be considered as secure.

4.1.3 Ideal Block Cipher

Ideal Block Cipher

A block cipher is a cryptographic algorithm that encrypts/decrypts blocks of a given length. Formally it is a bijective map from plaintext to ciphertext.

Block Cipher (Substitution)

An Ideal Block Cipher is a permutation that maps n-bit blocks onto n-bit blocks. The key determines the permutation used for a specific cipher.

	plain	cipher
	0000	1010
	0001	1100
	0010	0111
	0011	0001
	0100	0010
	0101	0011
	0110	1001
Codebook:	0111	0100
	1000	0000
	1001	1111
	1010	1110
	1011	0101
	1100	1000
	1101	0110
	1110	1011
	1111	1101

Table 4.1: Block Cipher Map

The number of permutations is $2^n!$ which leaves us with a key length of $\log_2(2^n!)$ which is approximately $n \cdot 2^n$.

Ideal Block Cipher

Issues:

- Insecure for small n, but
- impractically long keys
  - $n = 64$ (length of block in bits)
  - key length: $64 = 64 \times 2^{64} \approx 10^{21}$
  - grows exponentially

4.2 Block Cipher Modes

Encryption algorithms usually encode fixed-size blocks of data. But the size of the payload to be encrypted rarely fits into a single block. It is thus necessary to cut the data into block-sized chunks for encryption. At the destination, after the chunks have been decrypted, they can be concatenated again to reveal the original data.

Sounds simple, but, as commonly with cryptography, the devil is in the details. In the following, we will discuss different ways to handle block-encryption to encrypt streams of data, and the shortcomings of different algorithms.

4.2.1 Electronic Code Book

With the size of blocks, the complexity (i. e., size) of keys increases exponentially. It is thus not feasible to use a block size n that is large enough to include all possible messages or files. To encrypt data p that is larger than the cipher block size, we have to break it down into chunks p_i that are not larger than the block size. You can now apply the symmetric key k to each chunk to derive encrypted chunks c_i. This mode of operation is the Electronic Code Book (ECB) in Figure and it has the disadvantage that it fails at hiding structures within the document. (See the example in Figure 4.7)

Both operations; encryption and decryption, can be executed in parallel for each chunk. This can increase the speed in specialised hardware.

It turns out, that even something seemingly simple as applying ideal block ciphers, that have been complicated to derive, onto data can be challenging.

4.2.2 Cipher Block Chaining

Cipher Block Chaining (CBC) solves the problem of the ECB by creating a dependency on the previous block. In CBC mode you are able to run decryption in parallel, but encryption must happen in sequence of the chunks.

Encryption:

$$c_i = E_k(m_i \oplus c_{i-1}), i \in \{1, \ldots, r\},$$

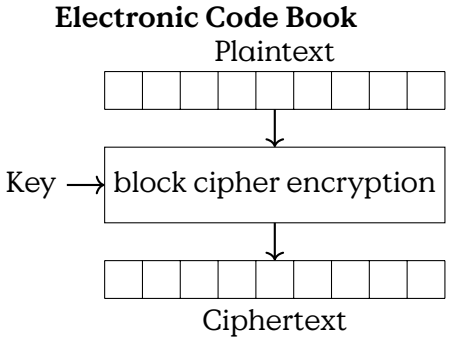
where $c_0 = IV$ is an initialisation vector, that must be agreed upon before communication and r is the number of chunks.

Decryption:

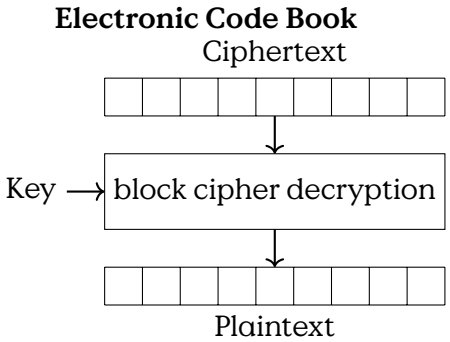
$$m_i = D_k(c_i) \oplus c_{i-1}, i \in \{1, \ldots, r\},$$

where $c_0 = IV$ is an initialisation vector, that must be agreed upon before communication and r is the number of chunks.

The problem of CBC is the strict dependency on previous operations which results in propagation of errors. If a single bit is flipped during transmission not only the concerned block, but also the following block will be corrupted.

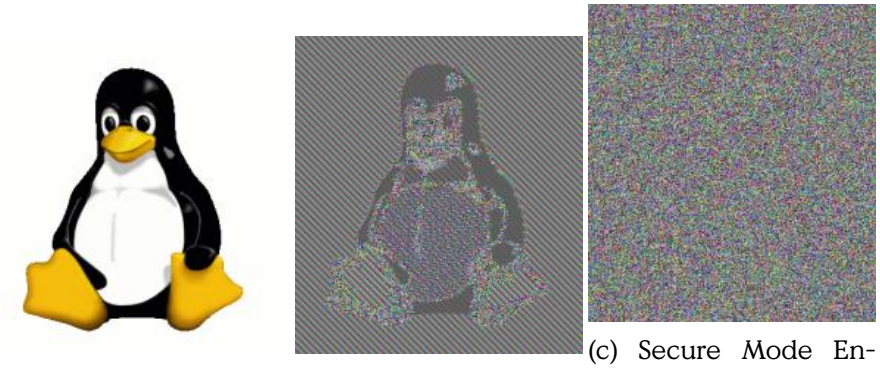


(a) Encryption



(b) Decryption

Figure 4.6: Electronic Code Book (ECB)



(a) Original (b) ECB Encrypted (c) Secure Mode Encrypted

Figure 4.7: Example for Electronic Code Book and Secure Encryption Mode

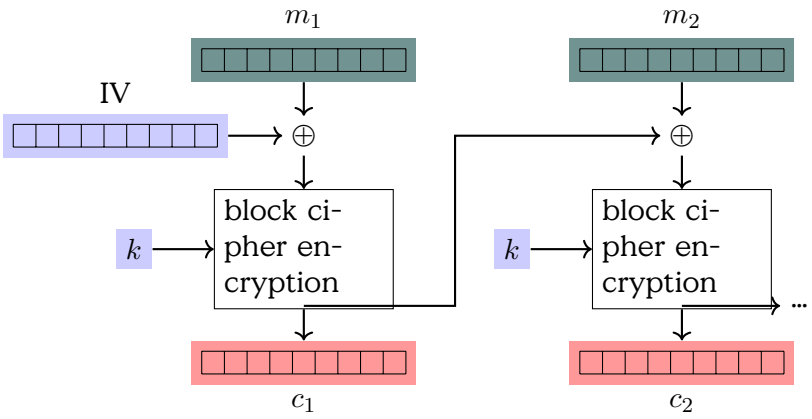


Figure 4.8: Cipher Block Chaining (CBC)

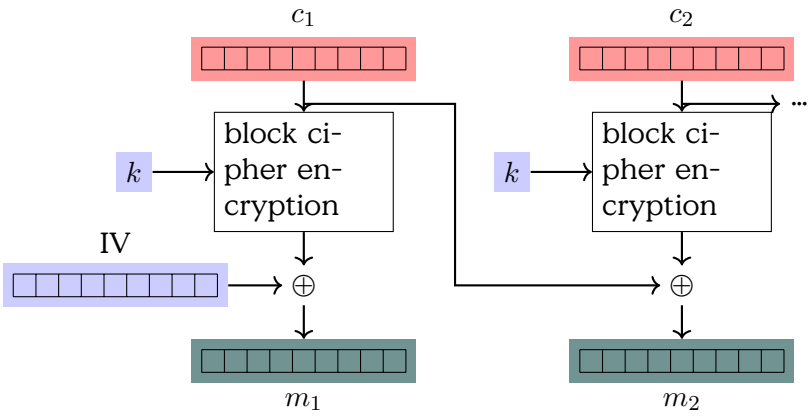


Figure 4.9: Cipher Block Chaining (CBC): Decryption

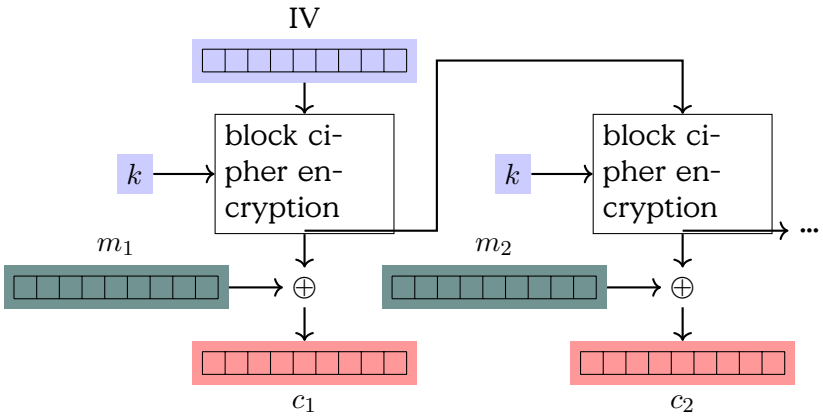


Figure 4.10: Output Feedback Mode (OFB): Encryption

4.2.3 Output Feedback and Cipher Feedback

OFB and CFB are modes that can make stream ciphers from block ciphers. Both cipher modes

Output Feedback Mode

The Output Feedback Mode (OFB)

$$\begin{aligned} c_j &= m_j \oplus o_j \\ m_j &= c_j \oplus o_j \\ o_j &= E_K(i_j) \\ i_j &= o_{j-1} \\ i_0 &= \text{IV} \end{aligned}$$

Cipher Feedback Mode

$$\begin{aligned} c_i &= E_K(c_{i-1}) \oplus m_i \\ m_i &= E_K(c_{i-1}) \oplus c_i \\ c_0 &= \text{IV} \end{aligned}$$

4.2.4 Counter Mode

The Counter Mode (CTR) allows for parallel computation of encryption and decryption of chunks. It uses a counter to increment an individual initialisation vector for every chunk. (Figure

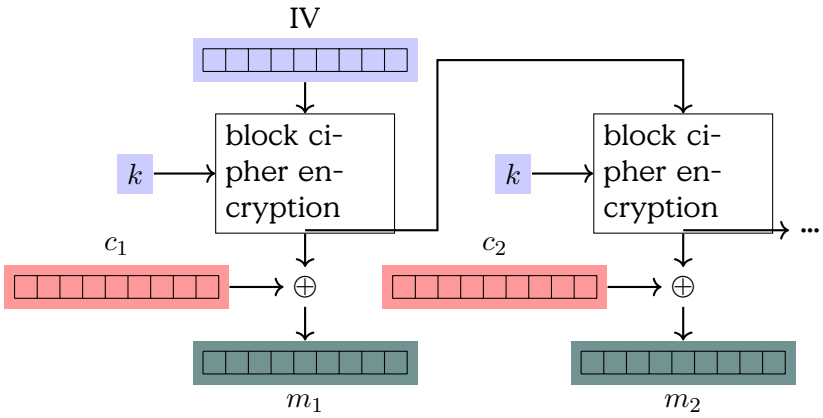


Figure 4.11: Output Feedback Mode (OFB): Decryption

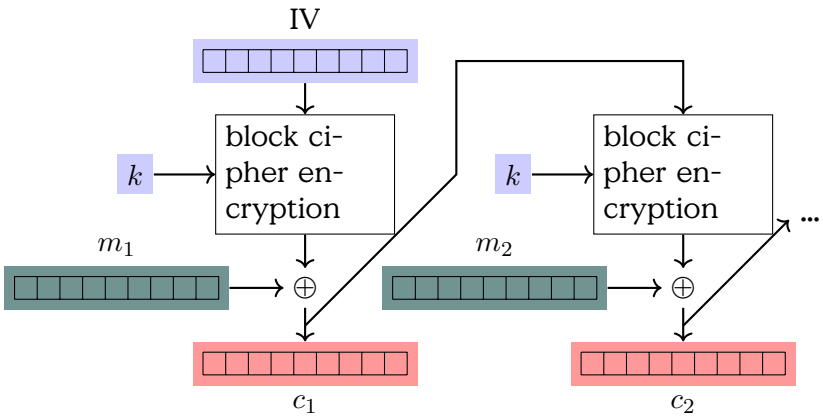


Figure 4.12: Cipher Feedback Mode (CFB): Encryption

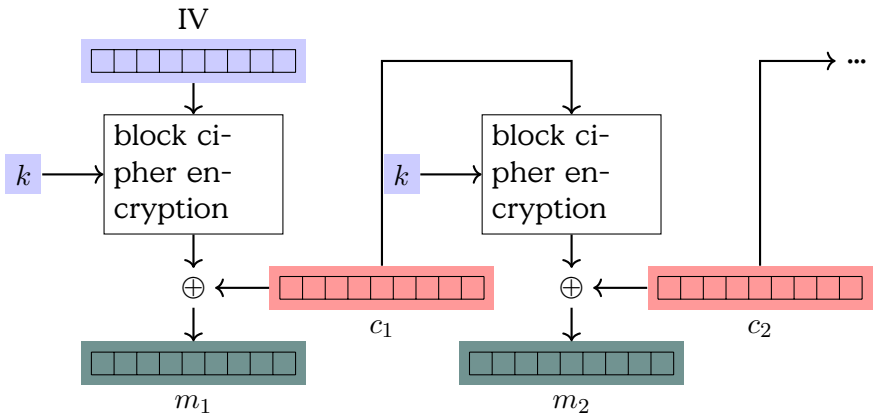
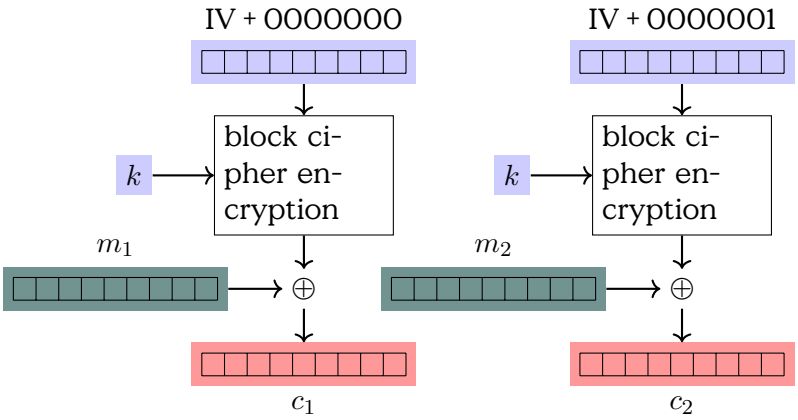
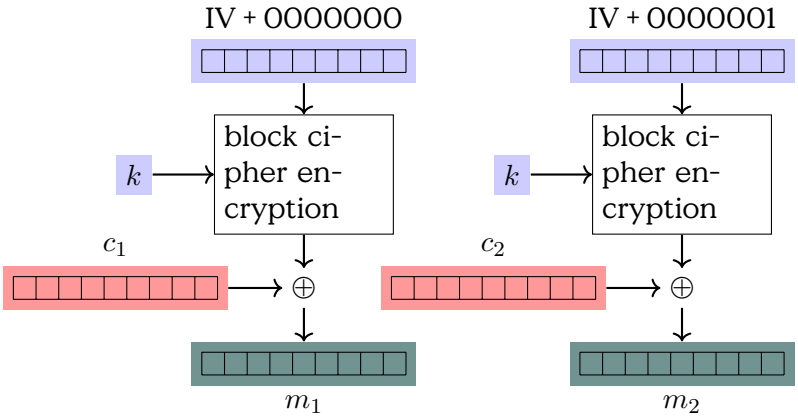


Figure 4.13: Output Feedback Mode (OFB): Decryption



(a) Encryption



(b) Decryption

Figure 4.14: Counter Mode (CTR)

4.2.5 Galois/Counter Mode (GCM)

https://en.wikipedia.org/wiki/Galois/Counter_Mode

4.3 Security of cryptographic algorithms

Security of computer systems is, to a large extend, based on security of cryptographic algorithms. For an algorithm to be attacked means that an unauthorized attacker is able to derive the secret key used for encryption. Generally we denote this as breaking the algorithm.

Thus, the big question is: how secure are the arcane algorithms that we use?

Cryptographic Security

Unconditional Security

- Cipher cannot be broken, because plaintext and cipher text do not match uniquely
- No matter how much computing power or time is available

Computational Security

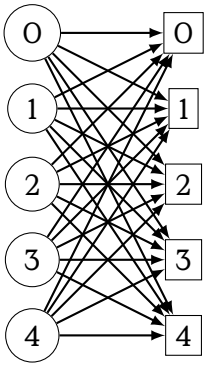
- Given limited computing resources (e.g. time, space) the cipher cannot be broken
- Computational cost is higher than the value of the encrypted information

Unconditional Security

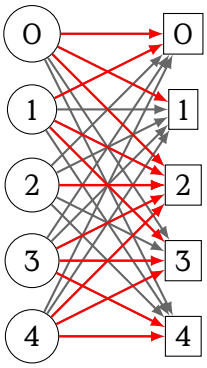
Unconditional security is an attribute of a cipher, that describes that the ciphertext reveals no information whatsoever about the corresponding plaintext. A cipher that is not unconditionally secure is a cipher where an attacker can infer that certain symbols or sequences of symbols are more likely corresponding plaintext symbols than others. In Figure 4.15 this is described by the potential mappings of plaintext onto ciphertext symbols.

Unconditional Security

Definition 4.3.1 (Unconditional Security)**.** The probability $P(p)$ of guessing the plain text p is equal (or at least not less) than the probability $P(p|c)$ of guessing the plain text under the condition, that the cipher text



(a) Unconditional Secure Cipher



(b) Insecure Cipher

Figure 4.15: Unconditional and Insecure Cipher

key:	101011
plaintext:	011001
ciphertext:	110010

Table 4.2: Example One-Time-Pad

is known. A cryptographic algorithm that provides this attribute for all plain texts and cipher texts is called unconditional secure, formalised as:

$$\forall p \forall c : P(p|c) = P(p)$$

A cipher text that is produced by an unconditional secure encryption algorithm reveals no information whatsoever about the plain text.

One-Time-Pad

The One-Time-Pad (OTP) is a symmetric algorithm that is the only known example of an unconditional secure algorithm.

One-Time-Pad

- Let $P, C, K = 0, 1^n$ be domain of p, c, k, where $|P| = |C| = |K|$
- $p, c, k0, 1^n$, key chosen randomly and uniformly distributed
- Encryption: $c = p \oplus k$, where $\oplus$ is exclusive or
- Decryption: $p = c \oplus k$

Example:

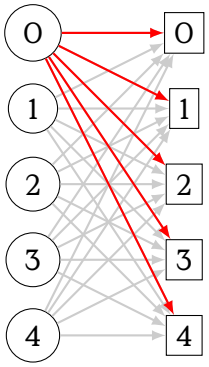


Figure 4.16: Complete Mapping of a One-Time-Pad

The main drawback of the OTP is that the key must have the same length as the plain- and ciphertext. The first problem is, that the key must be exchanged secure between the two communication partners before communication can take place. This is produces both overhead and restricts the usability of the scheme. Furthermore, it is very expensive to generate such large amount of random numbers. As a consequence, OTP is used mostly only by organisations that can afford such an overhead for high security applications.

Bayes' Theorem

Definition 4.3.2 (Conditional Probability)**.** The probability of an event A given event B, or A conditioned on B, is defined by Andrey Kolmogorov as the probability of the event A if the event space is limited to B:

$$P(A|B) := \frac{P(A \cap B)}{P(P)}$$

Theorem 4.3.3 (Bayes' Theorem)**.** The relation between the conditional probability $P(A|B)$ can be derived from the probability of the reverse condition $P(B|A)$ as

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}.$$

To proof that a cipher is unconditionally secure it must be shown that

OTP Security Proof

$$
\begin{aligned}
P(p|c) =& \frac{P(p \cap c)}{P(c)} \\
=& \frac{P(c|p)P(p)}{P(c)} \\
=& \frac{P(c|p)P(p)}{\sum_{p_i \in P} P(p_i)P(c|p_i)} \\
\overset{(OTP)}{=}& \frac{P(p)2^{-n}}{\sum_{p_i \in P} P(p_i)2^{-n}} \\
=& \frac{P(p)}{\sum_{p_i \in P} P(p_i)} \\
=& P(p)
\end{aligned}
$$

4.4 Asymmetric Cryptography

4.4.1 Basic Number Theory

This section provides some background helpful to understand the math behinde the algorithms. I mixed bits of Arithmetic, Algebra and Number Theory in order to cover the necessary basics to understand cryptographic algorithms.

Basic Algebra and Number Theory

Associativity $a \cdot (b \cdot c) = (a \cdot b) \cdot c$ for operation $\cdot$.

Commutativity for a set A and an operation $\cdot$ for every

$a, b \in A$: $a \cdot b = b \cdot a$.

Identity Element for a set A and an operation $\cdot$ there is an element $e \in A$, the identity element, iff

$e \cdot a = a = a \cdot e$, for all $a \in A$.

Inverse Element for a set A and an operation $\cdot$ with identity element $e \in A$, elements $a, b \in A$ are inverse elements of each other iff

$a \cdot b = e = b \cdot a$.

Divisibility "m divides n iff $m > 0$ and $n/m \in \mathbb{Z}$: Notation for "m divides n"

$m \backslash n$. (We adapt the notation from Graham, Knuth and Patashnik in [**graham1994concrete**], more commonly used is the notation $m|n$, but "|" is very ambiguously in different contexts and the backslash notation has the advantage of hinting towards m as denominator.)

Euclid's Algorithm for gcd

$$\begin{aligned} gcd(0,n) &= n \\ gcd(m,n) &= gcd(n \mod m, m), \text{ wlog.: } m \leqslant n. \end{aligned}$$

because $n \mod m = n - \lfloor \frac{n}{m} \rfloor m$

Abelian Group a set R with operation $+$ that satisfies closure, associativity, has an identity element, inverse elements and is commutative.

Finite Field (Endlicher Körper) or Galois Field[1],

Galois Field

or $\mathbb{Z}_n$. $\mathcal{GF}$, has order p^n where p is prime and n a positive integer.

Unit an element a in $\mathbb{Z}_n$ is called unit if it has an inverse element $b \in \mathbb{Z}_n$, which is equivalent to $a \cdot b \equiv 1 \mod n$.

Ring

an abelian group $(R, +)$ with an additional operation, usually $*$, that satisfies

$$\begin{aligned} a * (b * c) &= (a * b) * c \text{ [Associtivity]} \\ a * (b + c) &= (a * b) + (a * c) \text{ [Left Distributive]} \\ (a + b) * c &= (a * c) + (b * c) \text{ [Right Distributive].} \end{aligned}$$

If there exists an multaplicative unit e, i. e.,

$e * a = a = a * e$ then the ring is called a unitary ring or ring with unity.

Ring

[1] Évariste Galois died at the age of 20 in a duel. If ever you want to feel humbled, take a look at what this man achieved in only such a short time.

Residue Class

The residue class of a modulo n:

$$[a] = \{x \in \mathbb{Z} | x \equiv a \mod n\} \tag{4.1}$$

Residue Class is the set of all numbers congruent to a.

Residue Class Ring modulo n

$\mathbb{Z}_n$ is the set of residue classes from $\{[a]|a\mathbb{Z}\}$, with

$\mathbb{Z}_n = \{[0], [1], \ldots, [n-1]\}$,

where $0, \ldots n-1$ are called natural representatives.

It is a

commutative ring with addition and multiplication $+, \cdot$, where

$[a] + [b] := [a + b]$ and $[a] \cdot [b] := [a \cdot b]$.

Prime Residue Class Group

, is a multiplicative subgroup of $\mathbb{Z}_n$ consisting only of units of $\mathbb{Z}_n$, written $\mathbb{Z}_n^*$ of $\mathbb{Z}_n$. Example: $\mathbb{Z}_9^* = [1], [2], [4], [5], [7], [8]$: $1 \cdot 1 \equiv 1(mod 9)$, $2 \cdot 5 = 10 \equiv 1(mod 9)$,$4 \cdot 7 = 28 \equiv 1(mod 9)$, $8 \cdot 8 = 64 \equiv 1(mod 9)$

For prime residue class groups $\mathbb{Z}_p^*$ each representative $a \in \{1, \ldots, p-1\}$, excluding 0, corresponds to a unit.

Residue Class Ring modulo n (Restklassenring) **Prime residue class group modulo n (Einheitsgruppe)** an element $a \in M = \{0, \ldots, q-1\}$ is called generator if there is, for all $m \in M$ a $p \in M$ with $m \equiv a^p \mod q$.

Primitive root A generator $g \in \mathbb{Z}_n^*$ is called primitive root.

Cyclic Group

Let G be a finite group. G is cyclic, if there is a $g \in G$ which generates G, i. e., $G = \{g^1, g^2, \ldots, g^{ord(g)-1}, g^{ord(g)} = e\}$, where e is the identity element of G

Cyclic Group

- g is called generator of G.
- Example: Let $G = \mathbb{Z}_5^*$, $[2] \in \mathbb{Z}_5^*$, then $ord([2]) = 4 \in \mathbb{Z}_5^*$. $[2]$ is generator of $\mathbb{Z}_5^*$, because $\mathbb{Z}_5^* = \{[2], [2]^2, [2]^3, [2]^4\} = \{[2], [4], [3], [1]\}$

If p is prime, then $\mathbb{Z}_p^*$ is cyclic and the number of generators is $\phi(p-1)$.

Euler Totient Function

$\phi(n)$ is the number of elements in $\mathbb{Z}_n$ that are relative prime to n, i.e. $gcd(x,n) = 1$. Especially it is multiplicative, meaning that if m, n are relative prime to each other, then

$$\phi(mn) = \phi(m)\phi(n).$$

Let's have a prime decomposition $n = p_1^{e_1} \cdot \cdots \cdot p_r^{e_r}$, then

$$\begin{aligned} \phi(n) &= \phi(p_1^{e_1}) \cdot \cdots \cdot \phi(p_r^{e_r}) \\ &= p_1^{e_1}(1 - \frac{1}{p_1}) \cdot \cdots \cdot p_r^{e_r}(1 - \frac{1}{p_r}) \\ &= p_1 \cdot \cdots \cdot p_r(1 - \frac{1}{p_1}) \cdot \cdots \cdot (1 - \frac{1}{p_r}) \end{aligned} \tag{4.2}$$

Euler and Fermat

Euler's Totient Function **Euler's Theorem** if $gcd(a,n) = 1$ then $a^{\phi(n)} \equiv 1 \mod n$

Fermat's Little Theorem p is prime, then for all $a \in M = \{1, \ldots, p-1\}$: $a^{p-1} \equiv 1 \mod p$.

Cyclic Group $\mathbb{Z}_{11}$

Efficient Modular Arithmetic

Modular Arithmetic:

Calculations in $\mathcal{GF}(23)$:

$$\begin{aligned} 5^2 &\equiv 5 \times 5 \equiv 2 \\ 5^4 &\equiv 5^2 \times 5^2 \equiv 4 \\ 5^8 &\equiv 5^4 \times 5^4 \equiv 16 \\ 5^{16} &\equiv 5^8 \times 5^8 \equiv 3 \\ 5^{23} &\equiv 5^{16} \times 5^7 \equiv 5 \mod 23 \end{aligned}$$

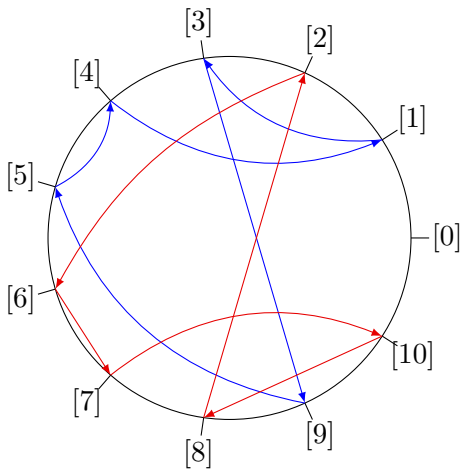


Figure 4.17: Cyclic Groups $\mathbb{Z}_{11}$, arrows show generating funktion "multiplication by three"r

In 1976 two cryptographers; Whitfield Diffie and Martin Hellman, rocked the world with the idea to revolutionise cryptography. Their requirements where quite simple: get rid of the excruciating need for secure key distribution, which almost ridiculed the idea of secure message exchange and provide a cryptographic system that would allow for signatures.

Asymmetric Cryptography

- Diffie, W. & Hellman, M. New directions in cryptography, 1976:
  - "minimize the need for secure key distribution"
  - "supply equivalent of a written signature"

Public Key Cryptosystem

"A public key cryptosystem is a pair of families $\{E_K\}_{K\in\{K\}}$ and $\{D_K\}_{K\in\{K\}}$ of algorithms representing invertible transformations,

$$E_K : \{M\} \longrightarrow \{M\} \tag{4.3}$$
$$D_K : \{M\} \longrightarrow \{M\} \tag{4.4}$$

on a finite message space M, such that

1) for every $K \in \{K\} : E_K$ is the inverse of D_K,

2) for every $K \in \{K\}$ and $\{M \in \{M\}$, the algorithms E_K and D_K are easy to compute,

3) for almost every $K \in \{K\}$, each easily computed algorithm equivalent to D_K is computationally infeasible to derive from E_K,

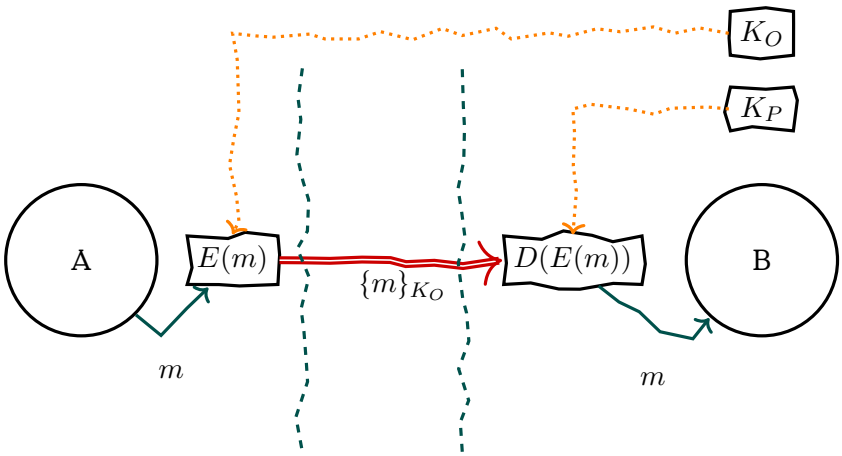
4) for every $K \in \{K\}$, it is feasible to compute inverse pairs E_K and D_K from K."[**DH76Newdirectionsin**]

They further introduce an algorithm for exchange of keys that can be done on a public channel. The Diffie-Hellman Key Exchange is using calculations over finite fields, which is, why we have to start with some basic number theory first.

Asymmetric cryptography makes encryption into a one-way-road: If you apply the public key, only the holder of the private key can reverse, i. e., decrypt, the message again. The one encrypting the message itself is not able to reverse the operation. But generally, this is not a hindrance, as asymmetric encryption is rarely applied directly to a message. (See)

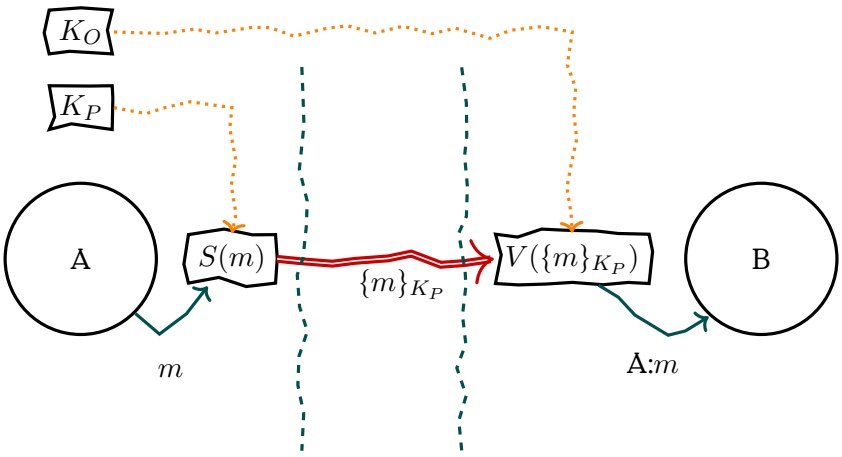
To use an asymmetric encryption algorithm, Bob has to prepare by creating a pair of related keys; one public and one private keys. The public keys he has to distribute to all principals he wants to receive encrypted messages from. If Alice wants to send an encrypted message to Bob, she has to know the correct, i. e., authentic, public key of Bob. She then can apply Bob's public key to a message, using the trapdoor function that is the asymmetric encryption algorithm, and send the encrypted message to Bob. Bob then can apply his matching private key and retrieve the original message.

Assymetric Encryption



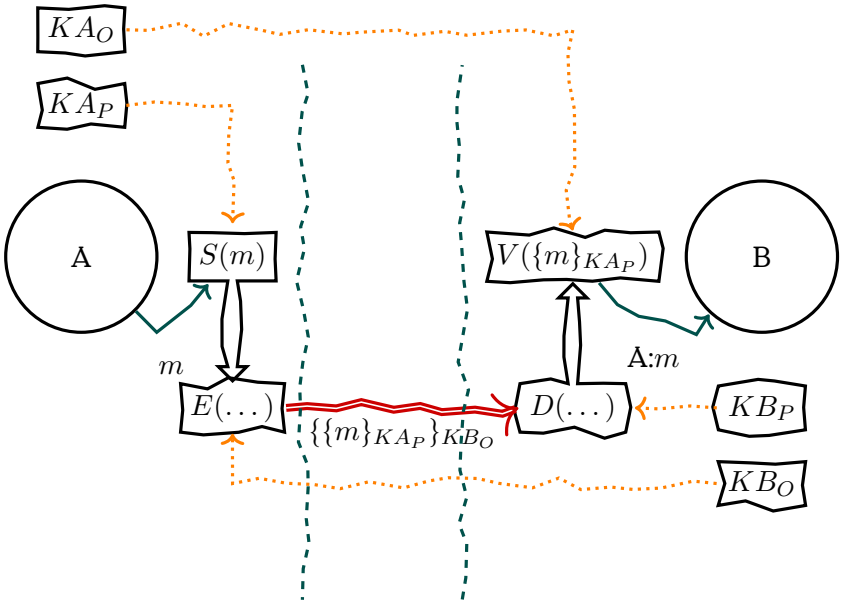
Asymmetric cryptography provides a completely new application for cryptographic algorithms as compared to symmetric cryptography. Using a private key is an operation only the holder of that key can facilitate. This allows, for example, to interpret the result of that operation as a proof that the holder of that private key has seen the message and approves it. The most common application of this is to provide authentication of the source of a message, or as it is commonly refered to, a signature.

Signature



On the other hand, asymmetric cryptography can be used for the original aim of cryptography, to keep the contents of messages a secret to all but the holder of a secret key. In general, it seems useful to combine both functions, encryption and signature.

Verschlüsselte, Signierte Nachrichten



The general, yet not mandatory, practise is to first sign a message and then encrypt both the message and the signature. In that way no information on the sender of an encypted message is leaked. The reason to do it in that order is, that oftentimes a signature is interpreted as agreement to the signed statement. This assumption can be made as long as the plaintext message is signed and not the encrypted message. This is denoted by the fifth principle in the "Prudent Engineering Practice for Cryptographic Protocols" by Needham and Abadi.

Prudent Engineering Practise

> Principle 5: When a principal signs material that has already been encrypted, it should not be inferred that the principal knows the content of the message. On the other hand, it is proper to infer that the principal that signs a message and then encrypts it for privacy knows the content of the message. [**abadi96prudent**]

4.4.2 Diffie-Hellman Key Exchange

Exchanging secrets has always been a difficult problem. For sure, you could meet "in private" and have a conversation. But this is very costly, as it requires to meet physically at the same space and time, and further ensure that no other is eavesdropping. In practise secret messages have to be transported over channels that do not fullfill the requirements of a private meeting. The message always has to be entrusted to a trans-

port that is not the intended recipient. And, by "entrusted" we explicitly mean, that sender or recipient have no way to ensure that the transport is not peeking.

Symmetric encryption has been a huge step forward, as it allows to apply an algorithm, via a secret key, to the message which makes it intelligible unless you are able to reverse that algorithm, via the same secret key. And, while this generally enables you to exchange secrets over untrusted channels, you now have a secret key that you need to exchange in secret before you can communicate secretly. An obvious chicken and egg situation, that is only relieved because you can do that exchange when you have an occassion to confer in private, as long as it happens beforehand.

We obviously gained a lot by encryption algorithms. And until Diffie and Hellman posed their requirements for asymmetric encryption algorithms very few have objected against the seemingly fundamental constraints of encryption. It seems that it helps to think about the seemingly impossible. Because shortly after Diffie and Hellman posed their white-paper on asymmetric encryption and requirements therefor, the provided a first solution.

The Diffie-Hellman Key Exchange is a distributed algorithm (or a communication protocol) that allows two (or more) parties to cooperatively generate a shared value over a public channel without disclosing this value. Thus, after the exchange, both parties know the same secret while no observer can know which value that is, although the observer has seen all exchanged messages.

Sounds like magic to you? You are not the first person to feel that way, but in the end it is all some fine but explainable mathematics.

Diffie-Hellman Key Exchange

Diffie-Hellman Key Exchange is secure as long as it is hard computationally hard to reverse the power-function in residue class rings. Later on this assumption has become known as the Strong RSA Assumption.

Strong RSA Assumption Given an RSA modulus n and an element $u \in \mathbb{Z}_n^*$ it is hard to find $e > 1$ and v such that $v^e \equiv u \mod n$.

The Diffie-Hellman Key Exchange is secure under one assumption. That is, the communication between the two parties is not intercepted by a Person-in-the-Middle (formerly known as Man-in-the-Middle). If an attacker is able to intercept the communication and pose (spoof) as Alice or Bob to the respectiv other side, it is easy to have an individual DH

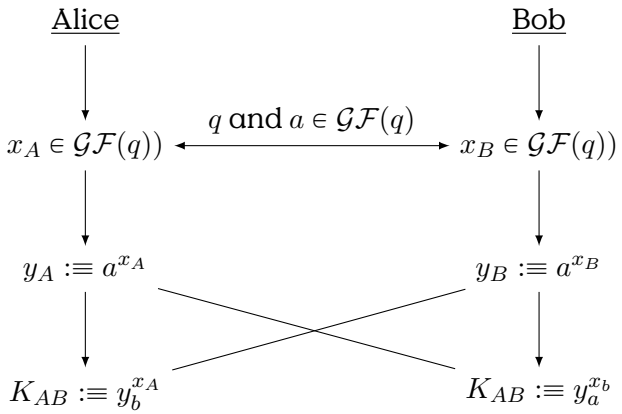


Figure 4.18: Diffie-Hellman Key Exchange (for two parties), all $\equiv$ are mod q

Key-Exchange with both sides. In that way we end with the Person-in-the-middle sharing a secret with both sides, leaving Alice and Bob with the wrong assumption of their respective secret being shared only by them.

DH- Person-in-the-Middle (PitM)

But, the attacker still is not able to partake in the shared secret of Alice and Bob. If Alice and Bob use their secrets to encrypt their further communication, the attacker will be able to read all messages, but Alice and Bob are not able to decrypt the messages send by the other because they are using different keys. That means the Person-in-the-middle will have to intercept and re-encrypt every and all messages exchanged between Alice and Bob using the secrets they falsely assume to be shared with each other. Alice or Bob may grow suspicious if they start receiving messages from each other that they cannot decrypt with the assumed shared secret.

4.5 RSA

RSA, named after his inventors Ron Rivest, Adi Shamir and Leonard Adleman, is one of the first and most commonly known and used algorithms for asymmetric cryptography. Much more important, the principles used are easy to understand and used or comparable to the underlying concepts of more recent algorithms.

You should take a glimpse at the section on number theory and algebra

to polish your math.

Rivest — Shamir — Adleman

- Popular asymmetric algorithm
- Based on discrete logarithm
- Vulnerable to Davida's Attack, if key pairs for signing and encryption are the same.

4.5.1 Key Generation

Key Generation

1) Choose two prime numbers p and q at random (stochastically independent) with $|p| \approx |q| = l, p \neq q$
2) Calculate $n := p \cdot q$
3) Choose c with $3 \leqslant c < (p-1)(q-1)$ and $gcd(c,(p-1)(q-1)) = 1$, remember $\phi(n) = (p-1)(q-1)$
4) Calculate d using p, q, c as multiplicative inverse of c in residue class group $\mathbb{Z}/n\mathbb{Z}$ so that: $c \cdot d \equiv 1 \mod \phi(n)$
5) Publish c and n as public keys.

4.5.2 Encryption

En-/decryption exponentiation with c, d in $\mathbb{Z}/n\mathbb{Z}$

RSA Encryption

Given:

Message $min\mathbb{Z}/n\mathbb{Z}$

Public/Private Key c, d of recipient

Modul $n = p \cdot q$

Encryption: $m^c \mod n$

Decryption: $(m^c)^d \mod n$

$$\begin{aligned}(m^c)^d &\equiv m^{c\cdot d} \mod n \\ &\equiv (m^d)^c \mod n \\ &\equiv m \mod n\end{aligned}$$

4.5.3 Signature

Signature

Exponentiation with d respectively c in $\mathbb{Z}/n\mathbb{Z}$

Proposition: $m \in \mathbb{Z}/n\mathbb{Z} : (m^d)^c \equiv m^{d\cdot c} \equiv (m^c)^d \equiv m \mod n$

4.5.4 Correctness Proof

Correctness

Proposition: $m \in \mathbb{Z}/n\mathbb{Z} : (m^c)^d \equiv m^{c\cdot d} \equiv (m^d)^c \equiv m(mod\, n)$

Remember:

$$c\cdot d \equiv 1(mod\, \phi(n)) \Leftrightarrow \exists k \in \mathbb{Z} : c\cdot d = k\cdot\phi(n) + 1$$

Therefore

$$m^{c\cdot d} \equiv m^{k\cdot\phi(n)+1} \equiv m(mod\, n),$$

by Euler's Theorem: $m^{\phi(n)} \equiv m(mod\, n)$.

It follows for all m coprime to p, i. e., $gcd(m, p) = 1$ (Fermat's Theorem):

$$m^{p-1} \equiv 1(mod\, p)$$

Because $(p-1)|\phi(n)$: $m^{k\cdot\phi(n)+1} \equiv m^{k\cdot(p-1)(q-1)+1} \equiv m \cdot \underbrace{\underbrace{(m^{p-1})}_{=1}^{k\cdot(q-1)}}_{=1} \equiv m.$

Because $\phi(n)$ is, by construction of the finite field of the exponent, equal to $(p-1)(q-1)$, with p, qprime, that means $m^{k\cdot\phi(n)+1}$ can be expressed as $m^{k\cdot(p-1)(q-1)+1}$. Now, we can reformulate this into $m \cdot (m^{p-1})^{k\cdot(q-1)}$ and because of Fermat's Little Theorem conclude that m^{p-1} equals 1 (in $\mathbb{Z}/n\mathbb{Z}$ and also $(m^{p-1})^{k\cdot(q-1)}$. This renders $m^{c\cdot d} \equiv m(mod\, n)$.

Security of RSA

Correctness

Proposition: $m \in \mathbb{Z}/n\mathbb{Z} : (m^c)^d \equiv m^{c \cdot d} \equiv (m^d)^c \equiv m (mod\, n)$

Remember:

$$c \cdot d \equiv 1 (mod\, \phi(n)) \Leftrightarrow \exists k \in \mathbb{Z} : c \cdot d = k \cdot \phi(n) + 1$$

Therefore

$$m^{c \cdot d} \equiv m^{k \cdot \phi(n)+1} \equiv m (mod\, n),$$

by Euler's Theorem: $m^{\phi(n)} \equiv m (mod\, n)$.

It follows for all m coprime to p, i. e., $gcd(m, p) = 1$ (Fermat's Theorem):

$$m^{p-1} \equiv 1 (mod\, p)$$

Because $(p-1)|\phi(n)$: $m^{k \cdot \phi(n)+1} \equiv m^{k \cdot (p-1)(q-1)+1} \equiv m \cdot \underbrace{\underbrace{(m^{p-1})}_{=1}^{k \cdot (q-1)}}_{=1} \equiv m.$

Because $\phi(n)$ is, by construction of the finite field of the exponent, equal to $(p-1)(q-1)$, with p, qprime, that means $m^{k \cdot \phi(n)+1}$ can be expressed as $m^{k \cdot (p-1)(q-1)+1}$. Now, we can reformulate this into $m \cdot (m^{p-1})^{k \cdot (q-1)}$ and because of Fermat's Little Theorem conclude that m^{p-1} equals 1 (in $\mathbb{Z}/n\mathbb{Z}$ and also $(m^{p-1})^{k \cdot (q-1)}$. This renders $m^{c \cdot d} \equiv m (mod\, n)$.

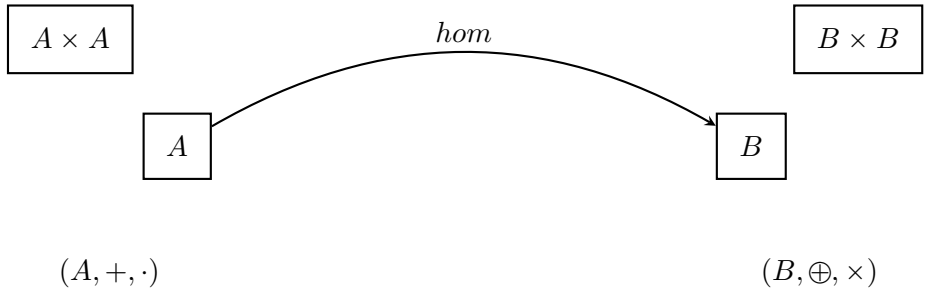
RSA is a partially homomorphic encryption algorithm. This remarkable feature provides that certain operations can be executed on encrypted data, without decrypting the data beforehand. If the algorithm is homomorphic, than this operation is equivalent to an operation on the decrypted data. This, homomorphic encryption algorithms allow mathematical operations on data, without being able to decipher the data[2]

A homomorphism is a mapping between the elements of two algebraic structures of the same typ. Meaning both structures provide a similar set of operations (with same arity) on their respective sets.

Sets A, B and operations $+, \cdot, \oplus, \times$.

[2] pretty magical, ain't it?

Homomorphism



e. g., $\forall x, y \in A : hom(a) \times hom(b) = hom(a \cdot b)$

Homomorphisms can be understood as maps that retain the essential structure of the preimage within the target domain. If an inverse homomorphism $hom^{-}1$ exists, both structures are called isomorph and the homomorphism is called an isomorphism.

The homomorphism in RSA allows for homomorphic multiplication. Thus, you could multiply on encrypted data. The only requirement is, that the data be encrypted with the same RSA-key pair.

RSA Homomorphism

$$E(x) \cdot E(y) = E(x \cdot y) \tag{4.5}$$

Now, homomorphic encryption can be used to delegate computations on sensible data, without disclosing that data.

In [**davida1982chosen**] utilises the RSA homomorphism to manipulate a principal into signing an encrypted message. If the same public/private pair of keys is used for both encryption and signature, then the application of the signature reverses the encryption. In order to hide this from the signer, the encrypted message is "blinded" beforehand.

Davida's attack not only teaches us a conceptual vulnerability of asymmetric cryptography but also the very useful technique of blinding, which can also be used for blind signatures in electronic money schemes and anonymous credentials.

Davida's Attack

In the first step the attacker intercepts a message m encrypted with the public key e of the victim. The attacker then multiplies an arbitrary value y encrypted for the victim.

$$\begin{aligned}
\text{given: } m^e & & \\
\text{select: } y & \longrightarrow & x = y^e \\
\text{blind: } m^e & \longrightarrow & x \cdot m^e \\
\text{victim signs: } x \cdot m^e & \longrightarrow & (x \cdot m^e)^d \\
(c^{\cdot e})^d & \overset{\text{RSA Hom.}}{\Longrightarrow} & x^d \cdot m^{e \cdot d} \\
& & = y^{e \cdot d} \cdot m^{e \cdot d} \\
& & = y \cdot m \\
\text{unblind: } y \cdot m & \overset{/y}{\Longrightarrow} & m
\end{aligned}$$

The combined value of encrypted message and value is the forwarded and the victim is foiled into signing the value, e. g., by using it as part of another statement. By the RSA homomorphism this is the same as applying the private key individually to the encrypted message and the encrypted, attached value.

The result is the product of attached value and message. As the value has been created by the attacker, he is able to extract the unencryptey message by dividing through y.

This provides us with two general rules on the handling of asymmetric keys:

1. Never use the same key-pair for encryption and signature, and
2. Be clear about what the meaning of a signature of a given document is.
3. Do not sign messages that you cannot read/understand.

4.6 Hash Functions

Please refer to [**https://nostarch.com/seriouscrypto**] for more details on hash functions and descriptions of existing hash functions.

The objective of a hash function is to map arbitrary data onto a shorter, unpredictable value in a way that cannot be reversed. The target domain is of finite size, i. e., the length of a hash-value is limited.

Hash functions are non-injective[3] and surjective[4], because they map a large domain (that of arbitrary-length messages) onto a small domain (a fixed-width bit-field).

Hash Function

Figure 4.19: Hash function with input message M and output hash value H. (See [**Aumasson18seriouscrypto**])

Hash functions have to be

Cryptographic Hash

- unpredictable
- preimage resilient (Einwegfunktion)
- second-preimage resilient (Schwache Kollisionsresistenz)
- collision resistant (Starke Kollisionsresistenz)

Unpredictability means, that it is not possible, except for calculating the hash function, do get to know the resulting hash value.

Unpredictability

```
$ echo "Hochschule Bremerhaven - Fachbereich 1" | sha512sum
680734a583a49ca0de59493ab25cf739f082f23ab21c44cdedd6b68ad0222f0ac
4b23d37ce54ac7c8a4f65bf96ec98ecd5d18845762aaad62ca26caeada1ed77  -

$ echo "Hochschule Bremerhaven - Fachbereich 2" | sha512sum
7cc3a49a63369e2bafc6698e31e3548543d3f627c9d5398e7d93ccee33e009a137
c9eb6d1a2474ba40db8098e6c5d57fbff3a689130b11d80c7eb58977e36551  -
```

(First) Preimage Resistance

Given: $h \in H$, $hash : M \rightarrow H$

Preimage: $m \in M$ with $hash(m) = h$

[3] injective is the principle also called "one-to-one", or $\forall a, b \in A : f(a) = f(b) \Rightarrow a = b$

[4] the principle of "onto", or $\forall y \in Y, \exists x \in X : f(x) = y, f : X \rightarrow Y$

Required: finding m is practically impossible

Second-Preimage Resistance

Given: $m_1 \in M, hash : M \to H$

Second-Preimage: $m_2 \in M$ with $hash(m_1) = hash(m_2)$

Requirement:

- Preimage Resistence, and
- finding m_2 is practically impossible

Collision Resistance

Collision Resistance

Given: $hash : M \to H$

Collision: $m_1, m_2 \in M$ with $hash(m_1) = hash(m_2)$

Requirement: practically impossible to find m_1 and m_2

4.6.1 Attacks on Hashes

4.6.2 Birthday Attack

How likely is a hash collision in general? Well the odds of finding a collision are surprisingly good, if only you can generate enough messages. The effect is known as the birthday paradox. The cause for its unintuitive behaviour lies in the number of pairs that can produce a collision grows quadratically.

Precisely, a collision among n messages can occur in any of $(n^2 - n)/2$ unordered pairs of messages.

Birthday Paradox

The Birthday Paradox describes the unexpected likelyhood of finding at least two persons within a room who where born on the same day of the year.

Birthday Paradox

Given:

- number of people $|N|$,

- number of days/year k,
- birthdays b

$P(b(n_i) \neq b(n_j)) = \frac{k-1}{k}$ (The argument goes as follows: you take the birthday of either of the persons as given, no matter what day it actually is. Then the probability of the other person been born on the same day is $1/365$. Likewise the probability that the second person is not born on the same day is the probability that she is born on any other of the 364 day, that is $364/365$).

If you follow above argument than, the likelyhood of three persons not being born on the same day in the year is the likelyhood of two persons times the likelyhood that the third person is not born on any of the two previous persons. For this to happen there only is a chance of $k-2/k$, or 363 out of 365 days.

Given that $B(n)$ is the event that any of n persons has the same birthday, we can calculate the likelyhood of none of the persons having the same birthday $\neg B(n)$ as Eq 4.6

$$P(\neg B(n)) = \underbrace{\underbrace{\frac{364}{365}}_{n=2} \times \frac{363}{365}}_{n=3} \times \frac{362}{365} \times \cdots \times \frac{366-n}{365} \tag{4.6}$$

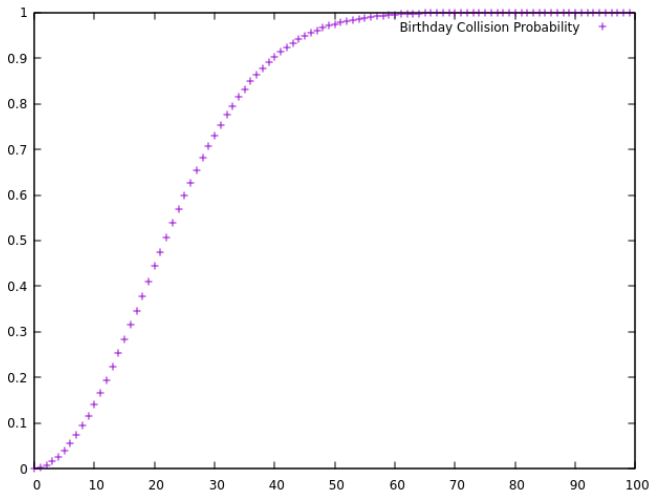


Figure 4.20: Probability of a Birthday Collision relative to the number of persons present

Let's do a quick visualisation and calculate the likelyhood of a collision occuring for n people $P(B(n))$:

```
p=1.0; for n in {1..50}; do
  echo "personen: $n; kollision: $((1-$p))"
  p=$(($p*(365-$n)/365))
done | while read a b c p;
   do echo "$p"
done | gnuplot -p -e "plot '<cat -' title \"Birthday Collision Probability\"
```

Birthday Attack Algorithm

Breaking an n-bit hash is surprisingly easy. The likelyhood of having found a collision grows over 0.5 (i. e., 50%) after only $2^{k/2}$ messages.

Birthday Attack Algorithm

$P(collision) > 0.5$ requires $2^{k/2}$ messages

Algorithm:

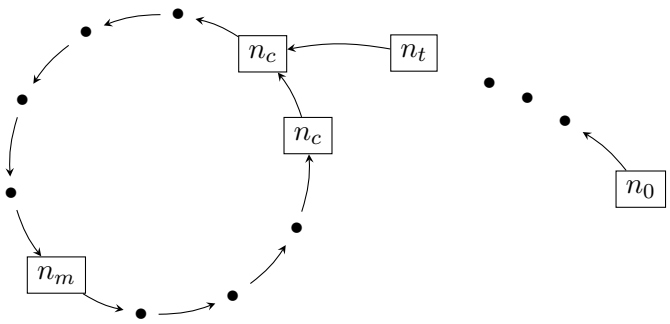
1. Generate $2^{n/2}$ random messages
2. Calculate hashes for all messages
3. Sort to have identical hash-values close
4. Find consecutive identical hashes
5. If that fails, add a new random message

While the above aproach requires a quadratic amount of memory, the Rho Method [**Aumasson18seriouscrypto**] requires only a linear amount of memory. The concept is based on the idea that hash-values are essentially pseudo-random. In fact, many random number generators employ hash functions to generate random numbers, using the unpredictability of hash functions.

In brief, the Rho Methods calculates and stores a hash chain until two hash values are equal. It is based on Floyd's two fineger circle detection method.

With Floyd's two finger circle detection method you can test if a directed graph is circular. It can be used to find a hash collision in a memory-efficient manner.

Floyd's two finger circle detection method



There are two pointers (fingers) called hare h and tortoise t.

1. Set both pointers to node n_0
2. Every round
   - t advances one step
   - h advances two steps
3. Until $t = h$ (both point to the same node)

   , call this node "mooring point" n_m
4. Set one pointer back to n_0
5. Advance both pointers one step, until they point to the same node. Call it n_c. This node is the entry point into the circle.

If you consider a directed graph of hashes, by starting with a random hash h_0 as first node, then the nodes preceeding the entry node n_c to the circle represent a hash-collision.

Pollard-Rho-Method

- Use Floyd's two finger circle detection
  1. Pick random hash value, set $n_0 = randhash$
  2. Use n_0 as starting node, setting $t_0 = h_0 = n_0$
  3. Advance t by calculating $t_{i+1} := \#t_i$
  4. Advance h by calculating $h_{i+1} := \#\#h_i$
  5. Advance until $h_i = t_i =: n_m$ (mooring point)
  6. Set $h'_0 = n_0$, $t'_0 = n_m$
  7. Advance with $h'_{i+1} := \#h'_i$ and $t'_{i+1} := \#t'_i$
  8. Until $h'_i = t'_i =: n_c$

9. Return h'_{i-1} and t'_{i-1} as hash collision.

4.6.3 Kerckhoffs's Principle

Auguste Kerckhoffs, a Dutch linguist and cryptographer, wrote two articles in 1883 in which he proposed a list of principles on cryptography.

Kerckhoffs's principle

1. The system must be practically, if not mathematically, indecipherable;
2. It must not be required to be secret, and it must be able to fall into the hands of the enemy without inconvenience;
3. Its key must be communicable and retainable without the help of written notes,

   and changeable or modifiable at the will of the correspondents;
4. It must be applicable to telegraphic correspondence;
5. It must be portable, and its usage and function must not require the concourse of several people;
6. Finally, it is necessary, given the circumstances that command its application, that

   the system be easy to use, requiring neither mental strain nor the knowledge of a long series of rules to observe.

4.7 Message Authentication Code (MAC)

A Message Authentication Code (MAC) is data added to a message to prove the authenticity of that message. MAC are produced using a cryptographic operation with a secret key with a corresponding operation to verify that the MAC is derived from the concerned message and the secret key.

Secure Hash

Message Authentication

Message authentication is concerned with:

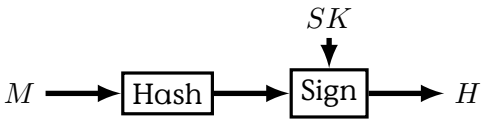


Figure 4.21: Secure (signed) Hash function with input message M and output signed hash value S, authenticated by application (sign) of a secret key SK (See [**Aumasson18seriouscrypto**])

- protecting the integrity of a digital message, i. e., unintended alteration of message is detectable
- validating identity of originator, which is, often, a necessary prerequisite to integrity protection. Or, the other way round, integrity of originator statements is a form of authenticity
- Sometimes: Timely and correct sequence of message, which can

The fundamental technique for MAC is to mix a given message with a secret and then perform a hashing operation on the result. The two most simple forms generate MAC from (unkeyed) hash functions

Prefix or Suffix Construction of MAC Secret-Prefix

$$\mathcal{H}(K|M)$$

- Length-Extension Attack works for some hash functions, e. g., SHA-2 family. It allows to compute

  $$\mathcal{H}(K|M_1|M_2)$$

  for unknown key K and message M_1.
- Variable Key-Length Attack if the length of keys is variable, then one could have equal hash-values for different messages, if the start of a message is equal to the remainder of the shorter key. Thus you can have two different messages with the same hash. That never is a good thing to have.

Secret-Suffix

$$\mathcal{H}(M|K)$$

- Exploit Hash-Collision of Messages

subsectionMerkle-Damgard Construction

Hash-based Message Authentication Code (HMAC)

Hash-based Message Authentication Code (HMAC) has superior security against message extension attacks and thus is probably the most

widely used concept for MAC. IPsec and TLS (Section **??**) are just two examples for usage. It is described in RFC 2104 and RFC 6151. You find the description of the algorithm below and depicted in Figure 4.23.

Hash-based Message Authentication Code (HMAC)

Define two fixed and different strings $ipad$ and $opad$ (the 'i' and 'o' are mnemonics for inner and outer):

$$\begin{aligned} ipad &= \text{the byte } \texttt{0x36} \text{ repeated to blocksize} \\ opad &= \text{the byte } \texttt{0x5C} \text{ repeated to blocksize.} \end{aligned}$$

To compute HMAC for the data 'text':

$$H(K \oplus opad, H(K \oplus ipad, \text{ text})).$$

HMAC

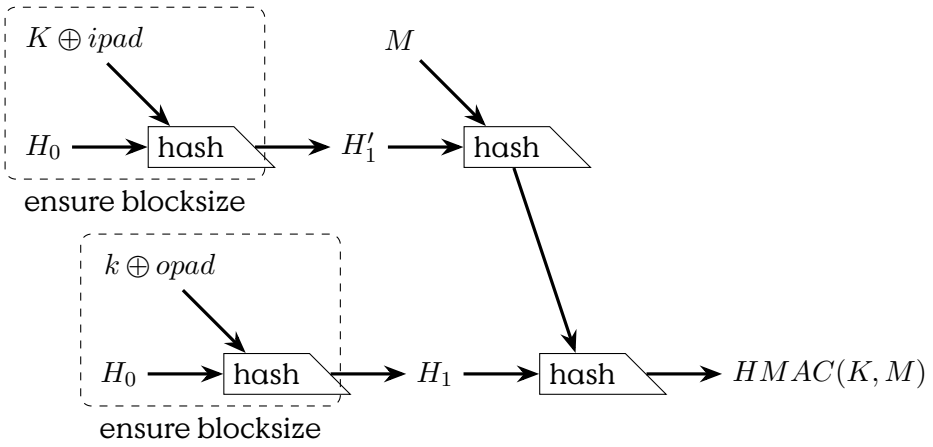


Figure 4.22: Hash-based Message Authentication Code (HMAC) (see [**Aumasson18seriouscrypto**])

Namely,

(1) append zeros to the end of K to create a B byte string (e.g., if K is of length 20 bytes and $B = 64$, then K will be appended with 44 zero bytes 0x00)

(2) XOR (bitwise exclusive-OR) the B byte string computed in step (1) with $ipad$

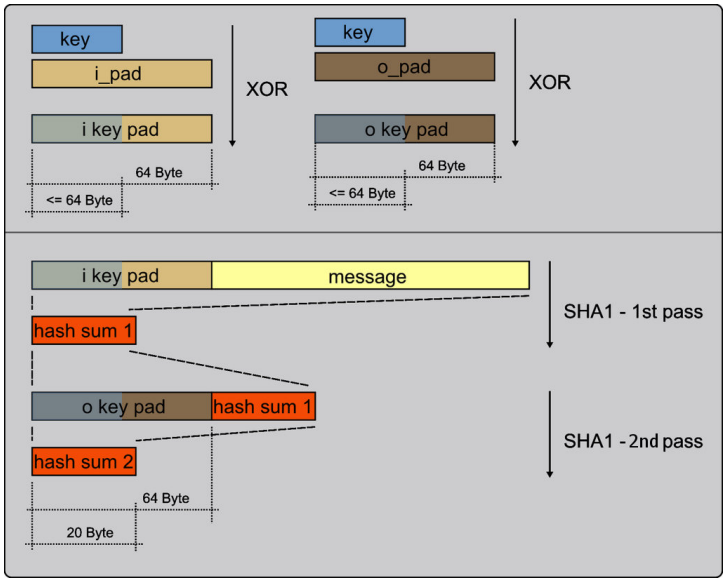


Figure 4.23: HMAC

(3) append the stream of data 'text' to the B byte string resulting from step (2)

(4) apply H to the stream generated in step (3)

(5) XOR (bitwise exclusive-OR) the B byte string computed in step (1) with $opad$

(6) append the H result from step (4) to the B byte string resulting from step (5)

(7) apply H to the stream generated in step (6) and output the result

[RFC 2104]

SHA-1 HMAC

5

Authentication

Authentication in general is the "act of confirming the truth of an attribute of a datum or entity."[1]

In this chapter we introduce techniques and structures to create and verify that a given entity is using an authentic identity. In general, three distinct classes of methods are distinguished:

Authentication Classes

Knowledge something one knows

Possession something one holds

Measurement something one is (i. e., Biometry)and which can be measured in any way.

It is possible (and more secure mostly) to combine multiple classes together. For example authentication of a human being by identity card uses possession (the paper/plastic document) and measurement (the picture on the card and probably the signature). Another form of combination can be used for authentication of machines towards a human, for example the form of casing, the "knowledge" of certain data, the location. (Find some illustrations in the slides.)

Authentication can happen between different types of entities, humans against humans, humans against machine, machines against machines and machines against humans.

Authentication Peers

[1] https://en.wikipedia.org/wiki/Authentication, 2013-12-06

Figure 5.1: Machine-to-Human (M2H) Authentication

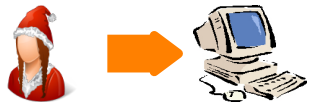


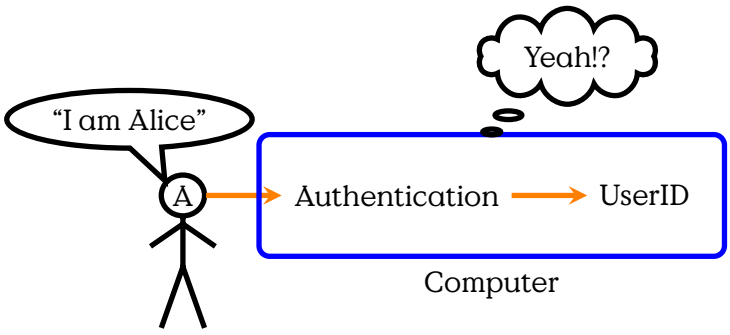
Figure 5.2: Human-to-Machine (H2M) Authentication

5.1 Password Authentication

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.

Identification



Password Login

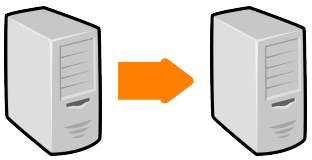
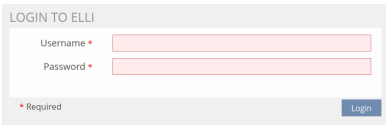


Figure 5.3: Machine-to-Machine (M2M) Authentication



Username:

- Identifies user within system
- (communication identifier)

Password:

- Prove identity statement correct
- Not guessable!

5.1.1 Password Vulnerabilities

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 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[2] will have a pipeline deployed that checks every uname/password-combination found with popular services.
- Evil Maid Attack:
  - Workstation hijacking
  - Electronic monitoring (key-logger, van-Eck)
- Scrape it from the memory (mimikatz)
- Offline
  - Wordlist enumeration
  - Modification Rules
  - Hash Calculation (optimized)
  - Rainbowtables

5.1.2 Password Countermeasures

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

Passwort Protection

Interfaces:

- user-"memory"
- authentication process
- password database

5.1.3 Login Process

Countermeasures

Login:

- Encrypted network links
- Exponential Waiting-Time
- Random-Session-ID

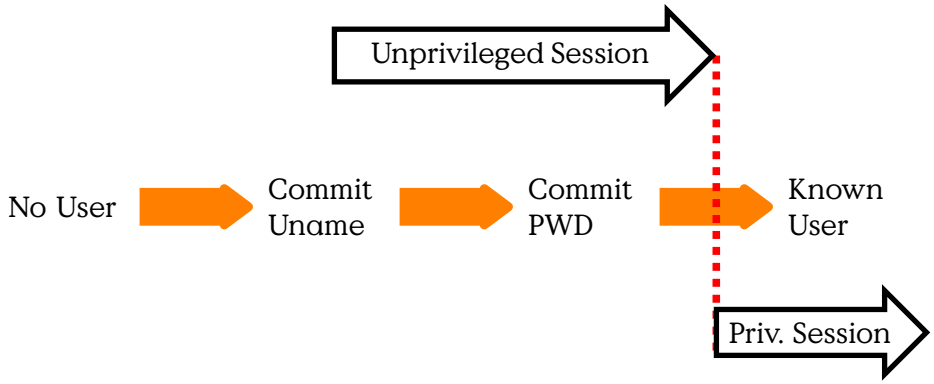
[2] pun intended

- Pinning-Prevention: separate the "public" unauthenticated session from the authenticated session.

Session:

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

Login Process



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

5.1.4 Password Storage

Countermeasures

- Never(!) store plaintext

- Stop unauthorized access to password file
- Intrusion detection measures

5.1.5 User-side Password Handling

Choosing Secure Passphrases

- Hard to guess passwords
- Unique Passwords (1/Account)

A crucial part of the security of passwords is handling of the passwords by the user. Users have to choose secure passwords and store them in a way that the clear text is accessible to them. There are essentially two ways for a user to handle passwords:

Password Handling

- Random Passwords and Secure Storage
- Manual Algorithms that are easy to remember

Password Data Usage

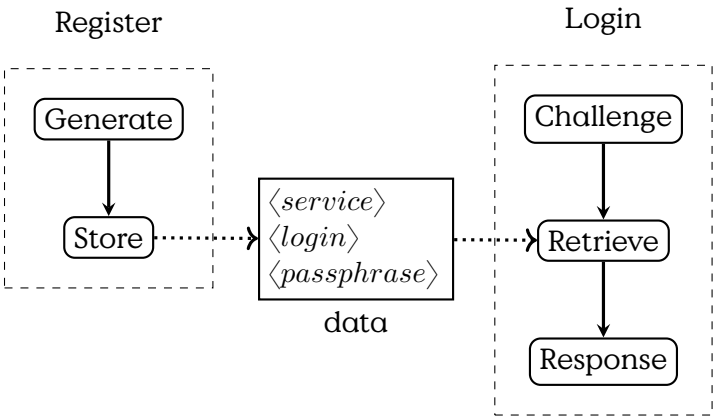


Figure 5.4: Password Data Usage

Passphrases are only one part of the data required in order to login to a service. In order to retrieve the correct passphrase during a login process, a user usually will have to know the service to which he/she is currently authenticating. The service usually requests a a service-local identifier for the account before a passphrase is requested. Thus, to identify

the right login and passphrase a reverse mapping from service name is required.

Password Data

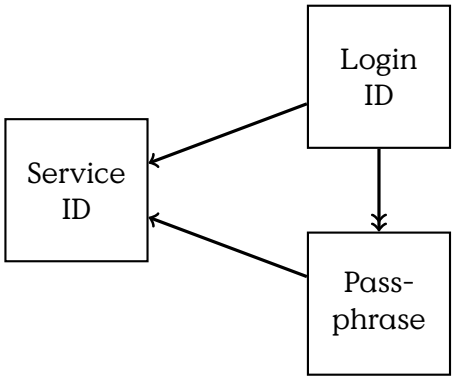


Figure 5.5: Simple OLOG of password credential data.

In Figure 5.5 the different maps between service, login and passphrase are shown. A user may have different login/passphrase pairs for a single service, but for every login/passphrase there is exactly one service for which they are intended. (Although syntactically the login name used in different services could be equal.) It should further be possible to map from each login to the related passphrase. This map is drawn as a surjective map, although there is a one-on-one relation between login and passphrase, to emphasize that it should not be possible to derive a login from a passphrase.

From this data-structure we can derive a few requirements for storing of passphrases. We require a reverse map that allows to retrieve a set of login/passphrase pairs for a known service. (As it is possible to have multiple accounts.) There further has to be a way to retrieve the passphrase for each login.

Necessarily, only the service ID has to be stored in a searchable (e. g., associative) way. It is not necessary to store the login identifiers searchable, i. e., readable, they could be encrypted. Logins identifiers should be handled as a shared secret between service and user. Passphrases are private secrets of users and should never be stored or communicated in the clear.

Passphrase Generation

The main attributes of a password are that it is hard to be guessed by an attacker but easy to remember by an authorized user.

Random password generation is actually not to complicated. You need a source of randomness and translate it into passwords by sequentially selecting letters from an alphabet.

Random Password Generation

- Manual Methods and Problems
  - Hit the keyboard → typing behaviour
  - Think of letters → Unconcious patterns
  - Flick through a book → language bias
- Generate 23-letter-password:
  - Random password `pwgen -sy 23 1`
  - Password manager `pass generate <pass name> 23`

Figure 5.6: 30-sided dice with german umlauts and double letter 'E'

Exercise 1

Generate "random" words by hitting at your keyboard. Generate 100 words of at least 8 letters length.

1. Can you spot any patterns?
2. Make a frequency analysis of letters and digraphs and plot the results in a frequency graph, sorted by frequency. Which stochastic model describes the graph best?
3. Calculate the entropy estimate of your randomness source based on the digraph frequency. What would be the best possible entropy you could gain?

Of course, random letters might be difficult to guess, but they are also hard to remember. Our brains simply are not normally wired that way, although it is possible to train them[3].

Diceware Passphrases

Another way for random passphrases, that is a little more suited to our brains, is to use words in a dictionary as "letters" and create longer passphrases by randomly choosing a sequence of words as passphrase. As with normal passwords, the resistance against guessing-attacks is based on the number of possible combinations. It is crucial to note, that secrecy of the word-list is not part of the security scheme.

Diceware Passphrases use a number of throws of a 6-sided dice to generate randomness and select words from a fixed wordlist. The original wordlist by Arnold G. Reinhold contain $6^5 = 15625$ words. A main security feature is, that the whole process is manual, i. e., no — potentially compromised — computer system is involved[4]. The idea might seem funny, but it produces secure and somewhat memorable passphrases and thus has become part of the security toolkit[5]

In order to generate a passphrase, you need to throw the dice five times per word, find the words inside the wordlist and note them down. A recommendation is to use passphrases of at least six words.

A human can memorize these words easier than random letters, because

[3] Few of you might remember that people where able to remember a large list of telephone numbers, before telephones learned to include telephone-books. Make it a habit to dial at least the two or three most important numbers for you by hand, it comes in handy if you drain the battery of your phone.

[4] But obviously many people, including me, could not keep themselves from implementing dice-word generators. Although, using a computer, it would make sense to get rid of the restrictions imposed by the dice and use arbitrarily long wordlist.

[5] Much funnier is, that Mira Modi, at that time eleven years old, became — arguably — more famous than the inventor for selling diceware passwords. She promises to generate all passwords by hand, using her "Top Secret Business Passwords"-list put them on a linnen paper in a linen envelope — and to forget them afterwards. US. Postal Service only, sorry.

the human brain understands the meaning of the words and can much easier store a connecting story for them. This actually is otherwise a technique of people who aim to memorize long sequences or numbers in memoization contests. Diceware Passwords are a good choice if you plan memorize the generated passphrases, want to store them on paper and type them manually. If you want to store and handle them digitally it has no advantages over sequences of random letters.

Manual Password Generation Algorithm

If it is hard to remember a password for every account you have somewhere, especially for services you might be using rarely. If you don't want to write your passwords down, maybe you could memorize a single algorithm with which to construct — seemingly random — passwords from the context of the service.

Password Algorithm Requirements

- Memorizable
- Manually computable (ideally in the head)
- Individual (use context)
- Hard to guess (include master secret)
- Hard to guess knowing pwds from other services

There are some problems with this approach. Firstly, you must be able to calculate your password easily in your head. Thus the algorithm has to be somewhat simple. Usually large parts of the algorithm will have to be secret, so you will have to design your own algorithm[6]. Therefore your passwords for different services necessarily have some common structure, even if you yourself are not able to see it. After all, everybody can construct a crypto-scheme that he himself is not able to break. As a result, your other passwords become less secure if a password from any of your accounts is disclosed.

Secondly, accounts often require more than a password, for example a login name, that you cannot always choose freely. For example the login name identifies an individual account and thus has to be unique within every given service. Some services require secret answers alternative

[6]Using secret algorithm is directly against the second of Kerckhoff's principles for cryptographic algorithms. That rarely is a good thing.

"security questions"[7]. Thus you most likely need to store some secret login-related data somewhere.

But, if you require some support in memorizing certain passwords, especially passwords that you need to enter physically into a keyboard, then password generation algorithms might be a good way to memorize them.

The general approach is to apply a secret sequence of syntactic operations to mix up secret phrases with the an identifier of the service. The result should contain letters from all allowed classes of letters, e. g., alphanumerical plus punctuation characters. Your algorithm should not produce results that are contained in common wordlists. The identifier of the service must not be recognisable in the result, otherwise an adversary could easily guess your password for other services.

The method itself is not fully recommended, but a few hints on how to build an algorithm:

Password Generation Algorithm Hints

- Take first/last/middle letters of domain name
- Transpose these letters (e. g., ROT-1)
- Append Secret Word (master-password)
- Append special character
- Permute some letters (to scramble secret)
  - e. g., n = length of domain name
  - count to nth letter
  - count n letters further (maybe wrap word)
  - exchange both letters
  - repeat
- swap some letters for numbers
- capitalize letters at some positions

(Motivated by Password Algorthms by Sumit Khanna, 2017-10-24)

[7] I hereby must note, that these alternative security questions are nothing more than alternative passcodes to your account and do not increase security, but often pose themselves grave security risk.

Password Managers

With the requirement to choose random passphrases and never reuse a single passphrase, remembering becomes a problem. Luckily humanity has invented writing a long time ago and thus one of the most basic ways to remember passphrases is to "just write them down". The simplest way is to use pen and paper and notch everything down in a notebook[8].

Obviously the notebok becomes a critical asset whose contents now need to be handled with an increased secrecy, accessible only to yourself. But one big advantage of paper-storage is that any access requires physical co-presence, i. e., it is (generally) save from remote attacks. A disadvantage is, that is lacking the comforts of digital storages that we have become used to — and that actually are the main reason you actually need a passphrase store.

Thus, to no surprise, the idea is to use digital notepads to notch down all your passwords. And, while at it, why not try to utilize the whole set of available functionality that makes digital data handling — as compared to plain old paper — comfortable? Namely searchability, single-sign-on, browser/tool-integration and synchronisation between multiple devices.

Usage of PW Manager Software

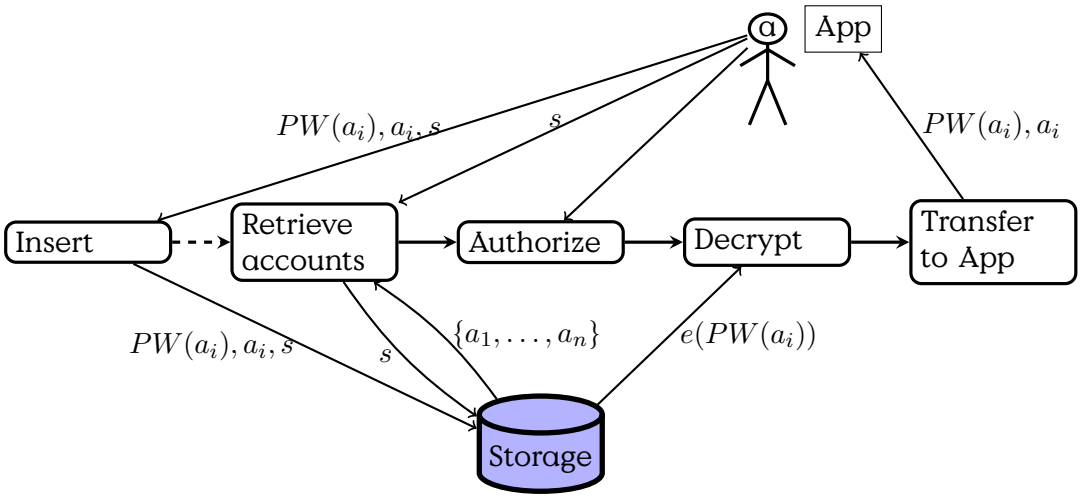


Figure 5.7: Using a password manager

But, security first! Using digital password managers means your most

[8] keeping a journal-like scetchbook close to you never has been a bad idea

precious passphrases are now available on one or more computers. We are not going to dig into "building a secure password manager", but you should be aware of some crucial distinctions between the ways your passwords are stored and handled.

Password Manager Functionality

- Encrypted storage
- Local vs. remote storage (i. e., cloud)
- Transfer from storage to login form
  - Temporary storage (e. g., clipboard) whiped after use?
  - Direct access through applications (e. g., browser)?
- Added security services
  - Random passphrase generator
  - Passphrase renewal reminder
- Synchronization between devices

With your pen-and-paper notebook the same question arises as with your digital password manager: where to store the data securely. Given that your storage uses secure encryption, the first line of defence is that the storage is not accessible from the outside so that an adversary cannot start brute-forcing your master password right away.

There are a few different password managers available already.

Some Password Management Software

HowTo Use pass

As an example for the usage of a password manager, we will take a look at pass, the standard unix password manager. The program makes extensive use of existing tools, uses GnuPG for encryption and, optionally, git for version control and synchronisation. It is itself a comparatively short program of less than 800 lines of bash-Skript. Around it a whole community of additional user-interfaces, graphical and command-line, has been developed. It can be integrated directly within many applications or used on a wide range of window managers and operating systems.

Name	OS	Pl	Enc	Sync
pass	Yes	most known plattforms via (G)UI clients	GnuPG	git
KeePass	Yes	via contributed/inofficial ports	Yes	local file
passbolt	(Yes) Community Version	Linux/CentOS	central cloud store	cloud
vaultwarden		Web		server

Table 5.1: Selection of Password Managers (OS=OpenSource, Pl=Plattform, Sync=Synchronization)

In the following I will give a very short introduction into the basic process of setting up a password-storage, inserting and retrieving passwords. For everything else please consult the excellent documentation on the man-page.

For a start you need to install GnuPG as pass relies on it for encrytion. (If you use a Linux Distribution this, most likely, will happen automatically during installation.) Git is optional, but if you want to remember old passwords after changing them, you'll need it. Also, if you maybe want to synchronize your store, a remote git-repository is a simple and proven way to do so.

Passstore Initialization

1. Generate a new PGP-key with gnupg
2. Create a directory for the store (e. g., `/home/me/.passwordstore`)
3. Init the store: `pass init <storedir> <GPG-key-id>`
4. Init git (optional): `pass git init`

After initialisation you can start to add or search and retrieve passwords or multi-line text. The password store essentially is a bunch of encrypted files within a directory tree, that you address as path starting at the passwordstore-directory. The pathname itself is readily searchable by everyone with read-access to the directory, the contents of the files — obviously — are not.

It is wise to take a few minutes to think about how you intend to access

your passwords. Basically there are two ways: using service- and login-name as part of the search path, or only using the service-name in the file path and encrypting login-name and passphrase in the file. Take a look at the website for an example.

If you store not only one single password-line in an item, than you probably want to create some syntax to distinguish passphrase and login-name. One way could be to have the login-name in the first line and the passphrase in the second line of a multiline file in the store.

Inserting a passphrase

- Blind entry
  1. `pass insert`
  2. Enter passphrase
  3. Confirm by entering again
- Visible entry
  1. `pass insert -e`
  2. Enter passphrase
- Multiline: use parameter `-m`

To use the contents of your password you have to know where in your password store the needed password is to be found. The best thing is if the hierarchy of your store is so well organised that you can directly enter the path of the needed file. Otherwise, use the search function.

The password is retrieved by the show-command, which is the default if you give no command.

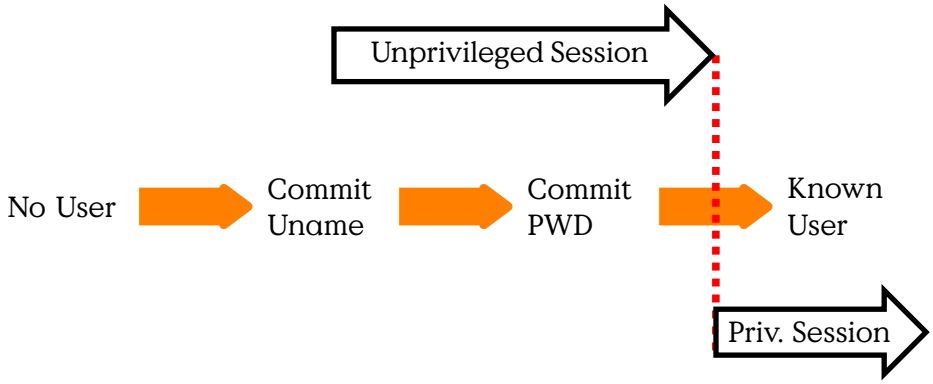
Searching and Retrieving

- Searching in path-name item `pass search <something>`
- Retrieving `pass <path>`
- Retrieving into clipboard:
  1. `pass -c <path>`
  2. Passphrase of the GnuPG-Key asked interactively
  3. Clipboard will be wiped after 45 seconds.

5.1.6 Server-side Password Handling

Passwords are never directly stored in cleartext in a database (or anywhere else). In order to verify passphrases, cryptographic hashes are used. Only the hash of a passphrase is stored with the verifier. To verify that a passphrase entered by an entity the hash of the entered phrase is tested against the stored hash.

Password Login



That way the possibility that a threat agent (including the verifier) can make use of a password database is limited.

Attacks on Passwords

- Brute-force: Test every possible password/message, until you find one that fits. On average you will have to try half of the possible passwords. How many are this for a 512-bit Hash value?
- Dictionary Attacks: Trying lists of common-passwords or using common password-generating methods to "brute-force" the most likely passwords first. Can you find a list of common passwords?
- Rainbow Tables: Pre-compute all possible passwords/messages, i. e., create a big table of (at least one) message for each hash value. Storage is cheap and in that way finding a matching password/message to a given hash is a fast and simple lookup in a huge table. How much storage is required for a rainbow-table?

Salting Passwords

In order to prevent Rainbow-Table attacks, an additional secret can be added to a password before the hash-function is applied. This secret is

called a "salt" and must be known only to the authentication function, i. e., the host that verifies passwords.

In that way, the password known by users, and entered into the login process, is not the message used to generate the hash value. Thus, an attacker gaining knowledge of a list of password hashes cannot simply lookup the corresponding password in a rainbow table, but has to know the used salt.

The salting process, as is schematically shown in Figure 5.8, can add the salt in different ways to the password. Commonly the salt is simply appended to the passphrase-string. Binary XOR would be another, more secure variant, as it basically implements a One-Time-Pad. Another version would be the use of HMAC.

Password Salts

Spicing it up!

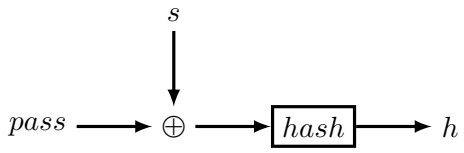


Figure 5.8: Salting a password before it is hashed.

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

Stupid password policies

- Strange Password Rule Enforcement
  - 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[9]

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

There are some guidelines, found, for example, in the "Grundschutz Handbuch" of the "Bundesamt für Sicherheit in der Informationstechnik" Section 2.11[10].

Administrators should

Administrator Guidelines

- 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.

5.2 Biometric Authentication

Biometric Authentication

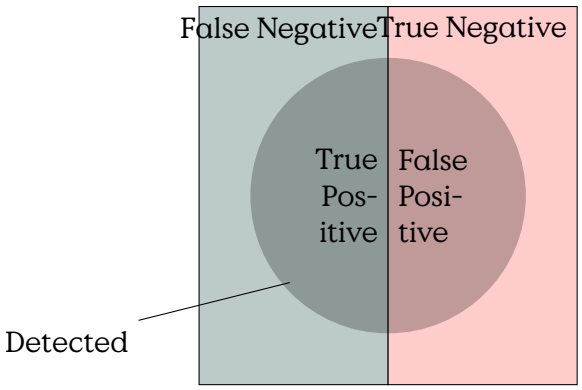
- Measuring human features
  - Physical
    - Fingerprint
    - Iris Scan
    - Brainwaves...
  - Behavioral
    - Typing Rhythm/Speed
    - Signature/Handwriting
- Impact of Biometry

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

Generally biometric authentication is a statistical binary classification test, i. e., a biometric scan leads to one of two possible results: rejection or acceptance. But biometry is inherently error-prone.

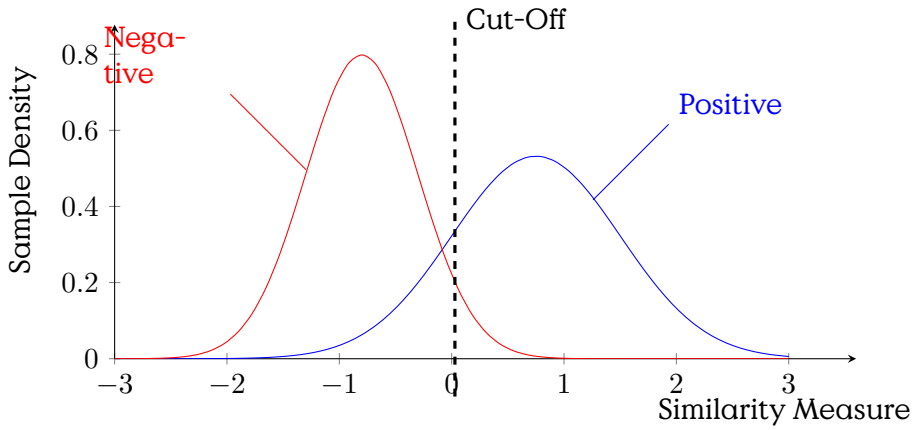
A biometric test can essentially fail in two ways: falsely rejecting a valid authentication and falsely accepting an invalid authentication. The termini for these errors are false negative and false positive, or Type II and Type I error. Usually a test method always has a probability for both types errors to a certain degree which can be adjusted by a threshold value. There are different ways to quantify the errors made. The most generic are Sensitivity and Specifity.

Binary Classification Errors



Any result of a binary classification consists of two distinct sets of samples which a classification mechanism should distinguish. A classification algorithms detects a set of samples, i. e., classifies them as "positive". The dual is the set of undetected samples, which is thus classified as "negative". Within the set of detected samples we denote the correctly detected samples as "true positive" (TP) and the set of wrongly detected samples as "false positive" (FP). Similarily the set of undetected samples is comprised of "true negative" (TN) and "false negative" (FN).

Binary Classification



Sensitivity (true positive rate) $TPR = \frac{TP}{P}$

Specifity (true negative rate) $TNR = \frac{TN}{N}$

Attributes of Biometry

- Misclassification is unavoidable
- Persistent personal attributes
  - Cannot be changed (easily)
- Cloned/Faked Attributes, e. g., 2008: Schäubles Fingerabdruck
- Dangerous Data-Leaks
- Unauthorized use: e. g., 2020: Taliban get hold of biometric database of local forces

Biometrical attributes differ in main aspects from knowledge and possesion as authentication methods. Namely they cannot easily be denied or changed by the individual authenticated by them. Changing biometrical features always means substantial change to body or behaviour which usually requires surgery to alter attributes of the body.

A consequence is, that biometrical attributes are unique to the individual and are similar to using a single password for everything. Which can be problematic as security research has constantly found ways to cheat sensors or algorithms for biometric attributes.

Furthermore the biometrical information stored in databases can be used for other purposes, e. g., yield information about health and genealogy, or can be used for identification without consent. Recent history sadly provides us with examples with potentially catastrophic consequences for the individual.

During the retreat of western forces from Afghanistan, Taliban Have Seized U.S. Military Biometrics Devices (The Intercept, 2021-08-18). The stored information on these devices possibly can be extracted and used to identify people that have worked with the western forces, which, if history repeats, will put them and their relatives and acquaintances in great mortal danger of retributions. Thus a tool for biometric authentication has been turned into a tool for identification. Different from other authentication methods, the chances of individuals avoiding identification from iris-scan or finger-print-scans involve dire mutilation of their bodies.

5.3 Token Authentication

Authentication Tokens are physical objects that can be carried and proof authentication by ownership. If implemented as active devices, e. g., SmartCards, they can provide two-factor authentication by utilising knowledge of a secret, e. g., a PIN.

Token Authentication

Token = Object/Possession to autheticate

- embossed card
- magnetic stripe card
- memory card
- smartcard
  - RSA-ID Token
  - nPA
  - Chip-TAN

Smart Cards

define size and protocol for smart cards.

One common type of smart cards are the EMV-Cards (abbreviation for Europay, MasterCard and Visa) widely used for payment.

5.3.1 HMAC-based One-Time Password (HOTP)

The concept of HMAC-based One-Time Password (HOTP) describes a group of algorithms that generate sequences of temporarily valid one-time passwords based on a shared key and an index value. A specific

HOTP Algorithm is defined in [**rfc4226**]. In this section a slightly simplistic version is introduced.

Thus, the objective is to provide a sequence of unpredictable numbers. Very commonly hash-functions are used to generate unpredictable, so-called pseudo-random, numbers. This can be used to generate a sequence of unpredictable HOTP.

HOTP

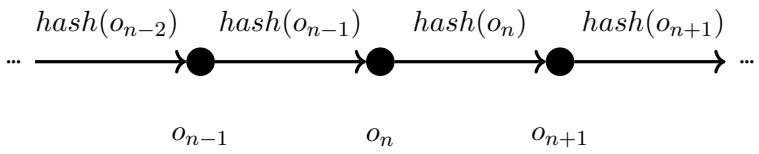


Figure 5.9: Hash Chain provides a dangerously simple one-time password chain.

Both sides have to agree on an initial value from which to start the hash-chain and would then be able to provide and verify a HOTP based on the index in the chain. But this is an overtly simplistic way to build HOTP. Even if the used hash-function properly provides unpredictability, meaning, it is not feasible to know a hash-value from its predecessor. The hash-chain will eventually produce a value that has been produced earlier in the chain, as all hash-chains have to be cyclic, given the finite size of the range of hash values.

Essentially everyone using the same hash algorithm would share the same hash-chain with the hash values within the cyclic structure even sharing the same order. An attacker would not have to break the cryptographic hash function but has to find the map between index and known hash values.

What is required is to use an individual hash function for every HOTP instance in order to generate an individual sequence of hash values. This can be achieved by using keyed hashes.

HOTP

5.3.2 Time-based One-Time Password (TOTP)

Time-based One-Time Password (TOTP) provides One-Time Password (OTP) with restricted validity for a given time-interval. TOTP essentially

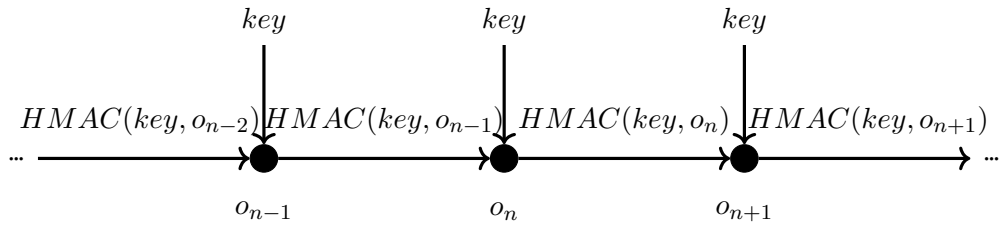


Figure 5.10: Keyed Hash Chain

C_T Time-interval index
T Current Unix-Time
T_0 Agreed starting time
T_X Length of validity of every OTP

Table 5.2: TOTP notation

is a HOTP (see Section 5.3.1) with an additional function that maps time-intervals onto index numbers of a hash chain.

Time-based One-Time Password (TOTP)

TOTP uses a HOTP function with shared secret K and a specific time-index C_T.

$$TOTP = HOTP(K, C_T)$$

An time-index is calculated as enumeration of time-intervals of length T_X from a start time T_0 for a current time $T > T_0$.

$$C_T = \left\lfloor \frac{T - T_0}{T_X} \right\rfloor$$

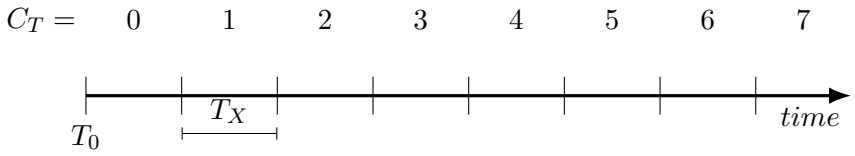


Figure 5.11: Time indicees for TOTP.

TOTP is fundamental to the Initiative for Open Authentication (OATH) and often used in 2-Factor-Authentication. It is implemented in hard-

ware authentication tokens, e. g., Yubi-Key or similarily the longer existing RSA-Identity Token.

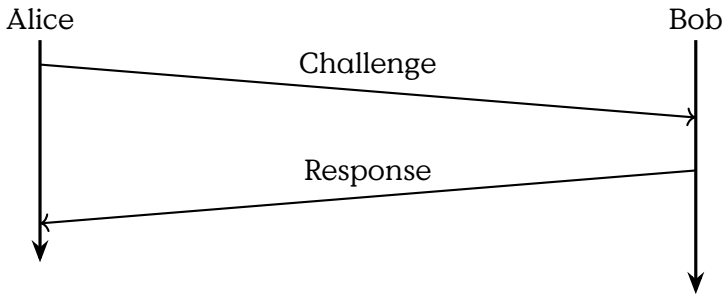
5.4 Remote Authentication

5.4.1 Challenge-Response

The term Challenge-Response describes a method used for authentication where the verifier sends a challenge to the entity that wants to authenticate itself. The entity then responds with an answer that proves that it has knowledge about some secret. The secret can be symmetric or asymmetric.

Challenge-response techniques can be found in symmetric and asymmetric flavours. The concept is very simple. The verifier generates a (random) challenge for the prover. The prover then calculates something based on the challenge that shows his knowledge of a secret. This, compared to static passwords, has the additional advantage of providing some proof of freshness.

Challenge-Response



- Challenge: Not Predictable, Fresh Nonce
  - e. g., random + datetime
- Response: Application of Secret
  - e. g., signature, HMAC, encryption

Mutual Challenge-Response

Mutual Challenge-Response

1. Server sends unique challenge sc to client
2. Client generates unique challenge value cc

3. Client computes $cr = hash(cc + sc + secret)$
4. Client sends cr and cc to the server
5. Server calculates the expected value of cr and compares
6. Server computes $sr = hash(sc + cc + secret)$
7. Server sends sr
8. Client calculates expected value of sr and compares

5.5 Public Key Infrastructure (PKI)

5.6 Web-of-Trust

6

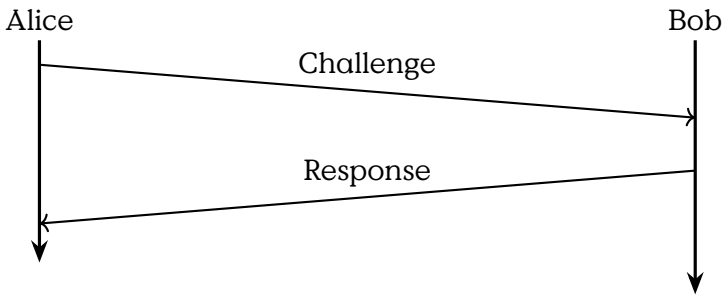
Remote Authentication

6.0.1 Challenge-Response

The term Challenge-Response describes a method used for authentication where the verifier sends a challenge to the entity that wants to authenticate itself. The entity then responds with an answer that proves that it has knowledge about some secret. The secret can be symmetric or asymmetric.

Challenge-response techniques can be found in symmetric and asymmetric flavours. The concept is very simple. The verifier generates a (random) challenge for the prover. The prover then calculates something based on the challenge that shows his knowledge of a secret. This, compared to static passwords, has the additional advantage of providing some proof of freshness.

Challenge-Response



- Challenge: Not Predictable, Fresh Nonce
  - e. g., random + datetime
- Response: Application of Secret

- e. g., signature, HMAC, encryption

Mutual Challenge-Response

Mutual Challenge-Response

1. Server sends unique challenge sc to client
2. Client generates unique challenge value cc
3. Client computes $cr = hash(cc + sc + secret)$
4. Client sends cr and cc to the server
5. Server calculates the expected value of cr and compares
6. Server computes $sr = hash(sc + cc + secret)$
7. Server sends sr
8. Client calculates expected value of sr and compares

6.1 Certificates

The most prominent application is certificates used for authentication, i. e., certification of the relation between an identification and a public-key pair. Thus, a certificate can be used for verification that a webserver is correctly serving under the identity of `www.example.org`, or that the sender is indeed jon.doe@example.com. The two remaining questions are: who is the issuer of the certificate and why can the issuer be trusted to know that the identified holder of the certificate is authentic.

Digital certificates are omnipresent. They are needed for secure communication or to prove access rights and qualifications. But, what exactly is a "digital certificate" in IT-Security?

What are certificates?

A certificate

- relates list of **attributes**
- authenticity of **provable authority**
- allows proof of **ownership**

Example: Passport

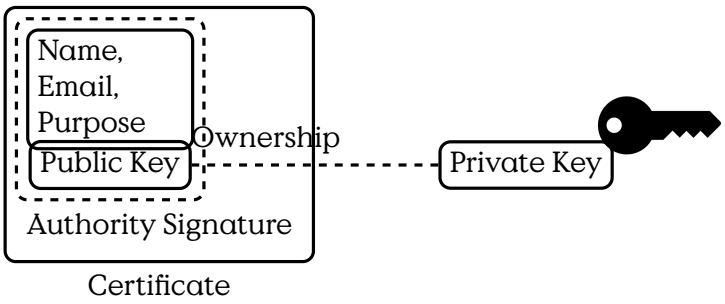
Attributes · Name,

- Birthday,
- Place of Birth,
- Unique ID,...

Provable Authority
- Security Features,
- official seal,
- well-known design

Ownership
- Biometric data,
- physical ownership

Digital Certificate



Certificates are secured by digital signature of an issuer. The first question regards the authenticity of the issuer of the certificate. Of course an issuer can be authenticated by providing a certificate, but this is essentially the same question for which the original certificate should have solved. Essentially this is usually solved by another a chain of certificates that leads to a trusted source.

The second question is obviously not only solved in the digital domain, as trust is foremost a human concept. [**Josang2003**]

6.2 Public Key Infrastructure (PKI)

A Public Key Infrastructure (PKI) denotes a hierarchical structure to issue, distribute and verify certificates.

PKI/PMI Hierarchy

Cross Certification means the mutual authorisation of certification authorities that expresses trust in the other CA and enables usage of sig-

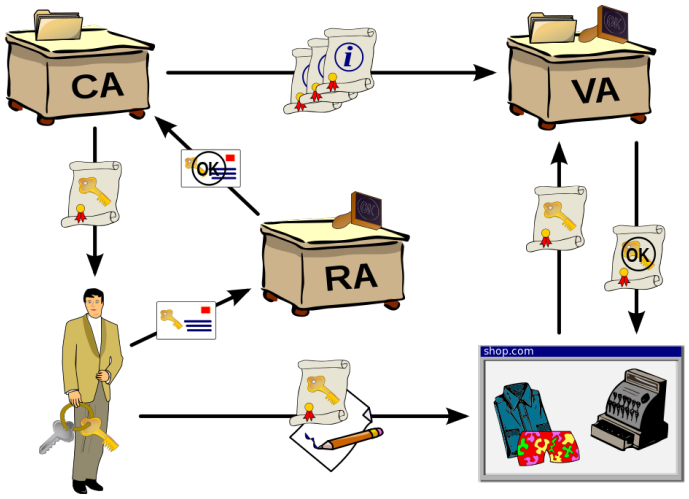


Figure 6.1: PKI Participants

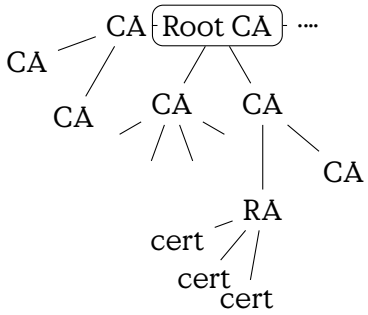
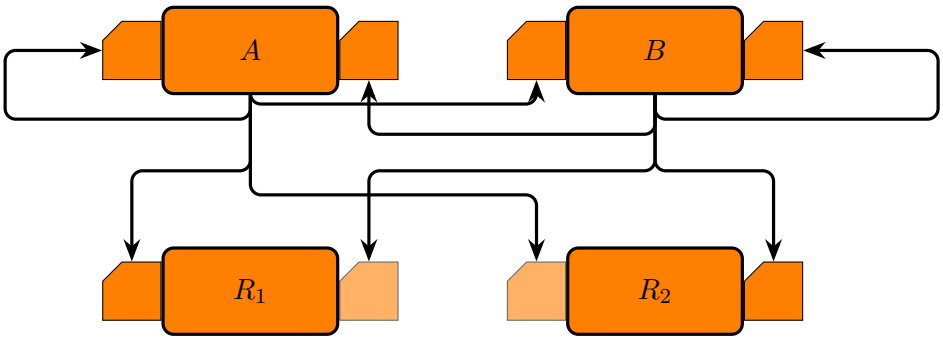


Figure 6.2: Tree of Certificates

natures from one PKI tree within another. In practise this is implemented by providing a second certificate to a self-signed root-certificate.

Take for example two CA A and B who have established mutual trust. That means that CA A authorizes all certificates signed by CA B and vice versa. This would establish an alternative trust-paths for the certificates signed by any of the two CA.

Cross Certification



X.509 Certificates

In this section we introduce the structure of X.509 certificates version 3. X.509 is an ITU-T[1] standard that defines a Public Key Infrastructure (PKI) and a Privilege Management Infrastructure (PMI). The PKI is expressed by certificates that bind identities to public keys by way of signatures of certification authorities (CA).

Certificates

Bind Name to public key

X509 http://tools.ietf.org/html/rfc5280

Formats: PEM, PKCS#7, PKCS#12

X.509 Certificate — Structure

- Certificate
  - Version

[1]International Telecommunication Union — Telecommunication Standardization Sector

  - Serial Number
  - Algorithm ID
  - Issuer
  - Validity
    * Not Before
    * Not After
  - Subject
  - Subject Public Key Info
    * Public Key Algorithm
    * Subject Public Key
  - Issuer Unique Identifier (opt)
  - Subject Unique Identifier (opt)
  - Extensions (opt)
- Certificate Signature Algorithm
- Certificate Signature

How does a browser decide which CA to trust? Where is the first certificate from a trust-chain defined?

Trusted CA

```
for i in /etc/ca-certificates/extracted/cadir/*.pem
do
  cat $i |
  openssl x509 -text |
  sed -n '/Subject:/p;/CA:TRUE/p;/Key Usage/,+1p'
done
```

```
         Subject: C = US, OU = www.xrampsecurity.com, O = XRamp Security Servic
            X509v3 Key Usage:
                Digital Signature, Certificate Sign, CRL Sign
                CA:TRUE
```

Certificate Revocation List

Certificate Revocation Lists (CRL) provide revocation of certificates in the X.509 PKI scheme. A revocation list is published by a CA and contains identifiers of certificates that became invalid before their validity time expired. CRL are standardised in RFC 5280.

There are various circumstances under which a certificate has to be revoked. Among others there are:

CRL — Revocation

Revocation vs. Hold

- Key Compromise The private key of the holder of a certificate has been disclosed to an attacker. Even the suspicion of a compromised key might be sufficient reason to order a re-certification.
- CA Compromise the issuing certification authority has been compromised and the currently signed certificates may have been faked. This obviously is a very big issue and there might be a lot of revocation certificates be circulating at the same time. Probably all certificates signed by this authority are affected.
- Privilege Withdrawn means that a privilege that has been granted by a certificate, for example the privilege to sign certificates, has been withdrawn from the holder — possibly a CA.

Critique of PKI

Problems of PKI

- Impersonal Trust Relation
- Difficult to Control Trustworthiness of CA
- Certificate Revocation is Slow and Blacklist-based
- Complicated Protocol for CRL/Cert. Distribution
- X.509 is complex
- X.500 Identifiers complex and not widely used

6.3 Web-of-Trust

The Web of Trust (WoT) is an alternative concept to PKI that is not based on institutionalised trust relations, but only on direct trust.

- Phil Zimmerman
- Traditional Format RFC 1991 https://www.ietf.org/rfc/rfc1991.txt
- MIME/PGP RFC 2047 RFC 3156

Web-of-Trust

- Authentication of Credentials
- Without direct (secure) contact
- Everybody may sign a public key and thus create a certificate
- Trust only trusted persons' certificates

Trust

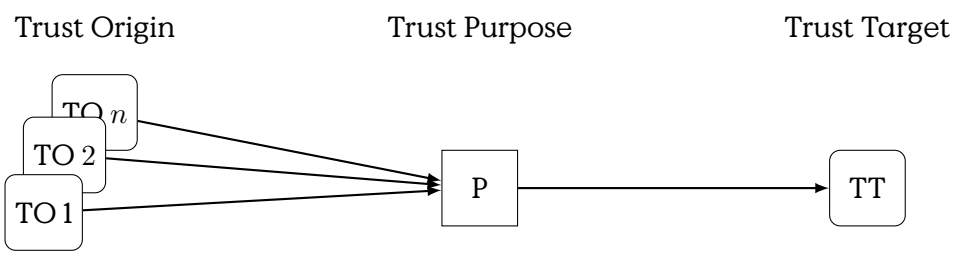
Trust

> Trust is the extent to which one party is willing to depend on something or somebody in a given situation with a feeling of relative security, even though negative consequences are possible. [**mcknight1996:meaningsoftrust**]

Remark: Negativ consequences are possible, because the outcome is not under the party's control.

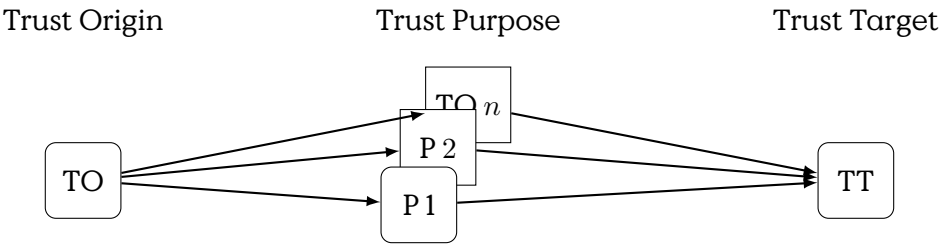
Trust Topologies (A. Jøsang)

Basic Trust Diversity



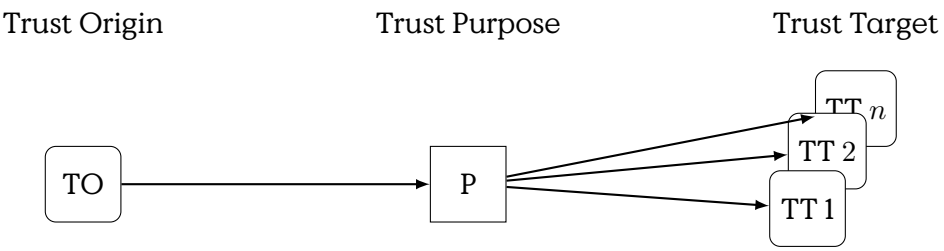
[Josang2003]

Basic Trust Diversity



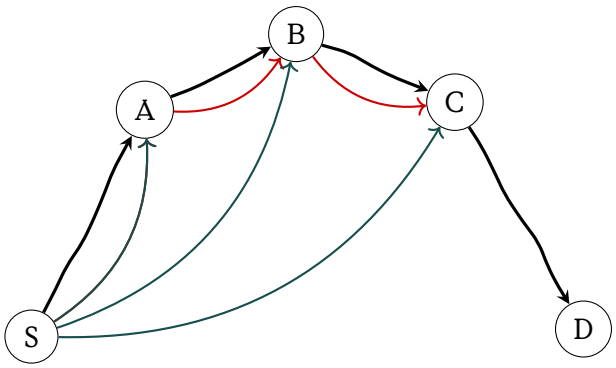
[Josang2003]

Basic Trust Diversity



[Josang2003]

Trust Path



Transitive Trust

WoT Decision Rules

A key is valid if:

1. Sufficient number of trusted signatures
2. Chain of certificates

untrusted Signatures by this key are ignored

marginal At least n signatures of marginally trusted keys are needed to make a key valid. A common value for n is two.

complete One or more signatures of completely trusted keys are sufficient to trust a key. This value also can be configured locally

ultimate To the owner of a secret key, the connected public keys are ultimately trusted.

PGP Signatures

A list of signatures that signs the key `65D0FD58` in Figure 6.3 contains various attributes of signatures. In general, a signature expresses some degree of certainty that some person, identified by email and name is holding the matching private key to the certified public key.

numbers 1-3 for certificate check level

0 means you make no particular claim as to how carefully you verified the key.

1 means you believe the key is owned by the person who claims to own it but you could not, or did not verify the key at all. This is useful for a "persona" verification, where you sign the key of a pseudonymous user.

2 means you did casual verification of the key. For example, this could mean that you verified the key fingerprint and checked the user ID on the key against a photo ID.

3 means you did extensive verification of the key. For example, this could mean that you verified the key fingerprint with the owner of the key in person, and that you checked, by means of a hard to forge document with a photo ID (such as a passport) that the name of the key owner matches the name in the user ID on the key, and finally that you verified (by exchange of email) that the email address on the key belongs to the key owner.

"L" for a local or non-exportable signature,

"R" for a nonRevocable signature,

"P" for a signature that contains a policy URL,

```
pub   rsa2048 2020-06-05 [SC] [expires: 2022-06-05]
      583263EAF239F5802E3786F1D48973AA9B0F4D73
uid   [ unknown] Lars Fischer <lars.fischer@hs-bremerhaven.de>
sig 3 D48973AA9B0F4D73 2020-07-17  Lars Fischer <lars.fischer@hs-bremerhaven.de>
sig   ABD2FB41822CF4D1 2020-07-27  Lars Fischer <lars.fischer@hs-bremerhaven.de>
uid   [ unknown] Lars Fischer <lafischer@hs-bremerhaven.de>
sub   rsa2048 2020-06-05 [E] [expires: 2022-06-05]
sig     D48973AA9B0F4D73 2020-06-05
Lars Fischer <lars.fischer@hs-bremerhaven.de>
```

Figure 6.3: Example PGP Signature List

"N" for a signature that contains a notation,

"X" for an eXpired signature,

numbers 1-9 or "T" for 10 and above to indicate trust signature levels that combine the notions of certification and trust in a signature. This is probably only useful in closed groups.

Example PGP-Key

6.3.1 Simple Public Key Infrastructure

Due to restrictions in space-time we will not introduce the Simple Public-Key Infrastucture (SPKI) here. But you are very free to research it for yourself. You will find sources with the SPKI working group[2] of the IETF. Or you directly take a glimpse into the standards they developed: RFC 2692[3] and RFC 2693[4]. We apologise for this inconvenience.

[2] http://www.ietf.org/html.charters/spki-charter.html
[3] https://tools.ietf.org/html/rfc2692
[4] https://tools.ietf.org/html/rfc2693

7

Secure Communication

Lernziele:

- Konzept: Hybride Protokolle
- Perfect Forward Secrecy
- Attacks
  - MitM
  - Replay
  - Session Hijacking/Pinning
- Prudent Engineering Practice
- TLS
- Key Distribution (PKI)

Cryptography alone is no guarantee for security, it must be deployed in a sensible way to achieve the intended security objectives. In this chapter we present example communication protocols that are selected for their importance in existing networks or for their teaching value. Without yet concerning ourselves with implementation errors, the communication protocols that utilise cryptography produced series of vulnerabilities as well.

Prudent Engineering Practise

> Principle 1: Every message should say what it means: the interpretation of the message should depend only on its content.

□

Participants A, B, C, I
Messaging $M \quad A \longrightarrow B : M$
Keys: Symmetric, Public, Private $K_{a,b}, K_a, K_a^{-1}$
Encrypted Message $\{M\}_K$
Signed Message $\{M\}_{K^{-1}}$

Figure 7.1: Notation for Cryptographic Protocols

> It should be possible to write down a straightforward English sentence describing the content though if there is a suitable formalism available that is good too.
>
> [**abadi96prudent**]

Notation for Cryptographic Protocols

7.1 Hybrid Encryption Protocols

Asymmetric encryption tends to require costly computation, as compared to symmetric encryption algorithms.

Hybrid Encryption

Problem: Asymmetric Algorithms are costly Solution:

- Use symmetric encription for data
- Session Key

The solution is to use a session key, i. e., a key valid and used only for a single conversation session, or even only a part of that conversation. Session keys are re-established often. Establishment can be done by a single principal or via shared key-establishment protocols, e. g., Diffie-Hellman (see Section **??**).

Hybrid Encryption

7.2 Security Attributes

7.2.1 Perfect Forward Secrecy (PFS)

Session Keys

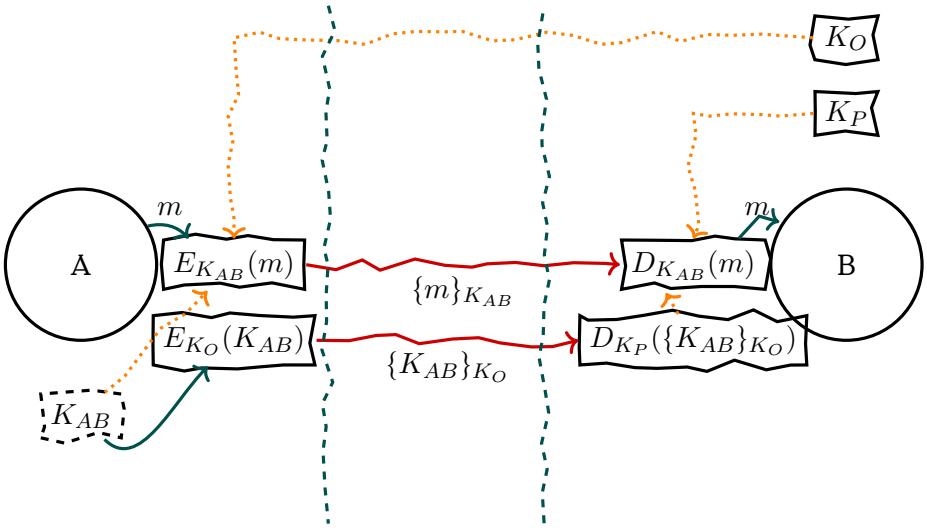


Figure 7.2: Hybrid encryption scheme, using symmetric session key K_{AB} for encryption of data m. Key is encrypted by asymmetric algorithm.

Problem: Compromise of master key reveals all communication Solution:

- Temporary valid keys
- Exchange/Generate at start of session
- Regular re-establishment, commonly the Diffie-Hellman protocol is used for this.
- (Authenticated!)

Perfect Forward Secrecy (PFS) means compromise of a master key will only compromise future (and at most current) data exchanges. Compromise of session keys will only compromise current exchange.

Perfect Forward Secrecy

To achieve PFS it is required that every session is protected by an independent, random secret session key which is not reused. This secret session key has to be exchanged between all authorized principals. It is crucial that all principals verify the authenticity of the session keys received to prevent Person-in-the-Middle (PitM) attacks.

7.3 Transport Layer Security (TLS)

At this point we finally went through all the necessary prerequisites to take a look onto real world examples for secure communication. This

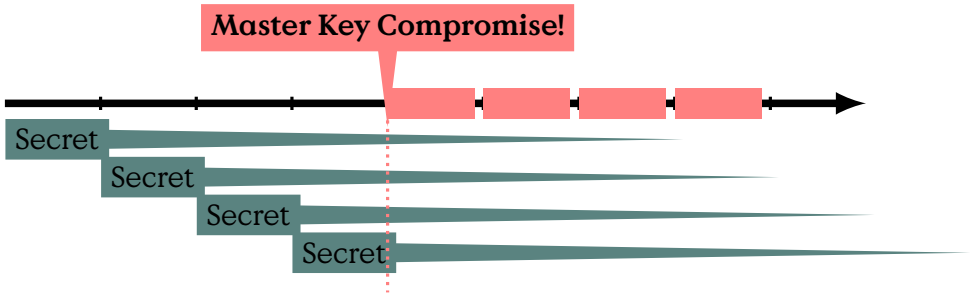


Figure 7.3: Compromised communication without Perfect Forward Secrecy (PFS)

section is based on Chapter ?? and the previous sections of this chapter here.

Transport Layer Security (TLS) is a protocol that provide secure communication, i. e., secret and authenticated, directly on top of some reliable transport protocol, e. g., implemented by TCP. In the layer model it is situated between TCP/IP (i. e., the 3rd and 4th layer) and the application layer. (See Figure 7.4)

The TLS protocol is currently standardised in version 1.2 by RFC 5246. The protocol has a history of three major revisions as Secure Socket Layer (SSL) that has been introduced by the almost forgotten now browser company Netscape.

TLS is comprised of two sub-protocols, the TLS Handshake and the TLS Record Protocol. Basically the former handles the establishment of a secure communication session and the latter handles the actual data transfer. Both sub-protocols are comprised of multiple separate messages.

TLS Layer Model

TLS already is defined for a number of different cryptographic algorithms, which can be replaced by newer algorithms in the future.

Supported Algorithms

- Anonymous/Uni-/Bi-directional Authentication
- HashAlgorithms: none(0), md5(1), sha1(2), sha224(3), sha256(4), sha384(5), sha512(6), (255)

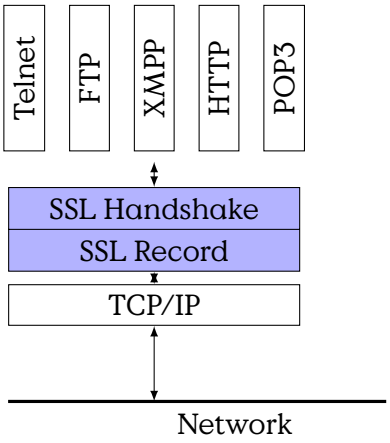


Figure 7.4: TLS Layer Model

- Signature Algorithms: anonymous(0), rsa(1), dsa(2), ecdsa(3), (255)

7.3.1 TLS Handshake Protocols

Handshake Steps

- Negotiate Algorithms, Random Values, Session Resumption
- Exchange Cryptographic Parameters
- Exchange Certificates
- Generate Master Secret
- Verify Integrity of Handshake Protocol Run

ClientHello First Message send to initiate a connection. Contains: Random structure: GMT and 28 random bytes, Session ID (Empty or old ID), list of supported cipher suites, compression methods, and extensions.

ServerHello Response to ClientHello. Contains essentially the same fields.

HelloRequest

ServerHelloDone Indicates the end of the server-side of the handshake, and that the server will now wait for the client's response.

Server-/ClientCertificate Contains a chain of certificates for authentication. These messages could already contain Diffie-Hellman parameters for the generation of the master secret.

CertificateRequest Message from the server to request TLS authentication from the client, if the server is non-anonymous. Contains lists of acceptable certificate types, signature algorithms and certificate authorities, the latter could be empty to indicate "any".

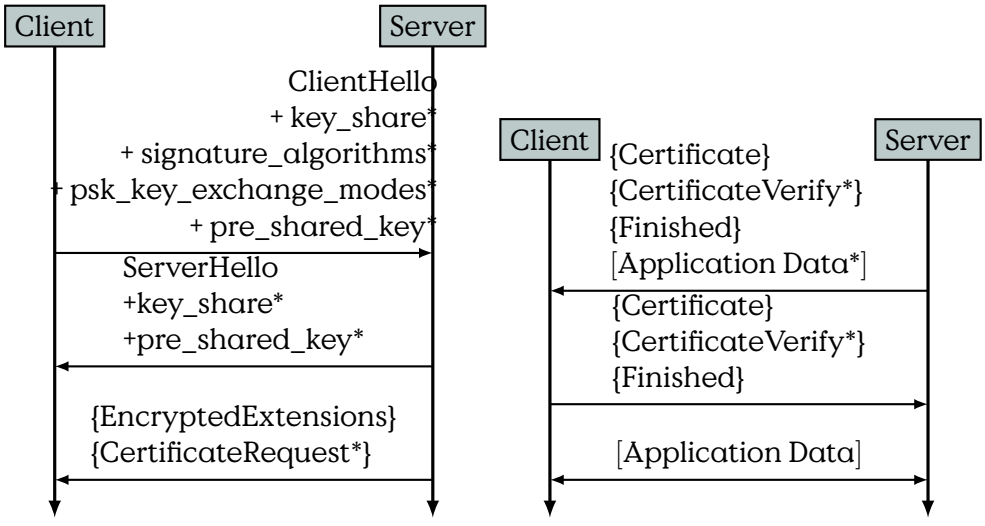
CertificateVerify Signature by the client over all handshake messages exchanged so far.

Server-/ClientKeyExchange Exchange of Diffie-Hellman Parameters (if not already included in Certificates).

ChangeCipherSpec Last message before switching to the selected encryption.

Finish These are the first messages exchanged in the negotiated secure connection. This message is essential for the whole negotiation as it includes a hash value over all handover messages and the master key.

The most noteworthy change in version 1.3 of the TLS Handshake Protocol is — arguably — that the handshake messages after the ServerHello are encrypted.



More informally the whole process can be described in six steps:

1. Exchange hello messages to agree on algorithms, exchange random values, and check for session resumption.

2. Exchange the necessary cryptographic parameters to allow the client and server to agree on a premaster secret.
3. Exchange certificates and cryptographic information to allow the client and server to authenticate themselves.
4. Generate a master secret from the premaster secret and exchanged random values.
5. Provide security parameters to the record layer.
6. Allow the client and server to verify that their peer has calculated the same security parameters and that the handshake occurred without tampering by an attacker.

[**rfc5246**]

TLS Certificate Request

TLS Certificate Request

- CertificateRequest
- immediately after ServerKeyExchange
- WebID: empty `certificate_authorities`

```
struct {
    ClientCertificateType certificate_types<1..2^8-1>;
    SignatureAndHashAlgorithm
      supported_signature_algorithms<2^16-1>;
    DistinguishedName certificate_authorities<0..2^16-1>;
} CertificateRequest;
```

TLS Client Verify

Client verification is used to prove that the client is in the possession of a private key associated with an identifier. Normally this identifier is bound to the individual person assumed to sit in front of and controlling the client. Thus the prerequisite to user authentication using this is, that users do not share browsers (or at least not their key rings).

Client Verification is prominently used in WebID described in Section **??**.

TLS Client Verify

- immediately after Client Certificate
- must have signing capabilities

```
struct {
  digitally-signed struct {
     opaque handshake_messages[handshake_messages_length];
  }
} CertificateVerify;
```

TLS Key Generation

A Pseudo-Random Function (PRF) as utilised in TLS must be able to generate a stream of pseudo-random numbers. The PRF used for all standards up to TLS 1.2 is defined as follows.

Pseudo Random Function

The PRF is based on a recursively defined application of a SHA-256 based HMAC:

$$\begin{aligned} P_{hash}(secret, seed) := & HMAC_{hash}(secret, A_1 + seed) + \\ & HMAC_{hash}(secret, A_2 + seed) + \\ & HMAC_{hash}(secret, A_3 + seed) + \ldots, \end{aligned} \tag{7.1}$$

where $+$ denotes bit string concatenation, and

$$\begin{aligned} A_0 &:= seed \\ A_i &:= HMAC_{hash}(secret, A_{i-1}). \end{aligned}$$

So that the pseudo random function can be defined as

$$\text{PRF}(secret, label, seed) = P_{hash}(secret, label + seed), \tag{7.2}$$

where $label$ is an ASCII-encoded string, $seed$ a nonce[1], and $secret$ some (shared) secret.

All key-material used for later communication is derived from a master secret that is negotiated during the handshake protocol from exchanged pre-master-secrets, commonly Diffie-Hellman parameters. Pre-master secrets are either included in Certificate messages or explicitly exchanged using `ServerKeyExchange` and `ClientKeyExchange` messages.

[1] number used once, hopefully random

Master Secret

48 bit source for keys

$$master_{secret} = PRF(pre - master_{secret}, \text{master secret}, ClientHello.random + ServerHello.random) \tag{7.3}$$

TLS Finish Message

The `Finish` message is a very crucial part for the protocol. It has to be exchanged after the exchange of `ChangeCipherSpec` messages. The messages contain the result of an application of the agreed upon pseudo-random function onto important parts of the protocol, including the master secret. This ensures, that a man-in-the-middle attacker is not able to exchange any of the messages of the handshake protocol in authenticated sessions, i. e., an attacker is not able to drop the communication onto the least-secure cipher suite.

Finish Message

$$PRF(master_{secret}, finishedlabel, Hash(handshakemessages))$$

7.3.2 SSL 2.0

The second version of the Secure Socket Layer protocol had security issues that make it advisable to cease using it. Currently the protocol is disabled by default in most browsers.

SSL 2.0

- Identical cryptographic keys are used for message authentication and encryption.
- weak MAC construction that uses the MD5 hash function with a secret prefix, making it vulnerable to length extension attacks.
- no protection for the handshake, meaning a man-in-the-middle downgrade attack can go undetected.
- TCP connection close to indicate end of data. This means that truncation attacks are possible: the attacker simply forges a TCP FIN, leaving the recipient unaware of an illegitimate end of data message (SSL 3.0 fixes this problem by having an explicit closure alert).

7.3.3 StartTLS

Classically TLS is used on dedicated ports only. For example, HTTP is usually handled on port 80 and HTTP encapsulated in TLS is usually handled on port 443. We could argue, that this is a little impractical because now every service would require two distinct default port addresses, instead of one. Alas this is a historical artefact and will remain for the forseeable future, yet there is a different way of handling this.

StartTLS allows to "upgrade" any connection, e. g., a TCP connection, into an TLS-protected connection. StartTLS is commonly used for email protocols, notably SMTP and IMAP. The general concept is that the client, at the beginning of a communication session, issues a STARTTLS-command which then initiates the TLS handshake. (I am not sure whether also the server could issue a STARTTLS, in theory this is perfectly possible.)

- RFC 2817: HTTP Upgrade to TLS
- RFC 2818: HTTP Over TLS

7.4 Key Exchange

8

Key Exchange Protocols

In this lecture, we will concern ourselves with the problem of distributing authentication and keys, especially session keys, between communication partners. This lecture is based on Chapter 9 from the textbook "IT-Sicherheit" by Claudia Eckert. We use this lecture to introduce some principles based on "Prudent Engineering Practices for Cryptographic Protocols" by Martin Abadi and Roger Needham. [**abadi96prudent**].

The introduced protocol is fundamental for authentication and communication security over un-secured networks in the Kerberos protocol. Kerberos itself is the default protocol for remote authentication in the Windows operating system, used to authenticate members of a service group. It is is also used in Unix-like operation systems, for example in the Andrew File System (AFS).

The underlying idea is, that a (central) authentication server establishes the authenticity of clients and provides session keys that authenticate the client against application servers while also securing the communication between clients and servers. The advantage, as compared to every server handling its own authentication, is, that the security checks are kept at one place, preventing that every server has to implement them itself.

The topic of this section must be distinguished from the common key-generation protocols, e. g., Diffie-Hellman in TLS, that are used to generate session keys. The protocols herein are more focussed on authentication by a single (often local) authority.

8.1 Key Exchange

Motivation for Key Exchange/Authentication:

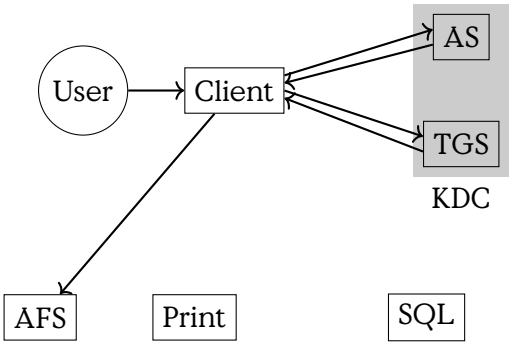


Figure 8.1: Kerberos Architecture

Motvation/Objectives

- Ephemeral Session Keys
  - Considered broken/disclosed after some usage
- Provide Perfect Forward Secrecy (PFS)
- Key Distribution
- Authentication of Participants

The canonic architecture of a Kerberos installation consists of the Key Distribution Centre (KDC) which itself consists of an Authentication Service (AS) and a Ticket Granting Service (TGS). A user wishing to use a service, e. g., the Andrews File System (AFS) first authenticates against his client, as usual by a password. The client then acts in the name of the user and requests an authentication ticket, i. e., a key to communicate with a service. To achieve this he has to authenticate its user against the AS and is then receiving a ticket from the TGS on request. This ticket then can be used to access the service. (See Figure 8.1.)

Kerberos Architecture

8.1.1 Simplistic Protocol

The most simplistic protocol (without actual authentication) is that all servers trust all clients A and simply accept key and user id from them.

Simplistic Protocol

The Simplistic Protocol (Figure 8.2) fails for various reasons.

1. A : generates $K_{A,B}$
2. $A \longrightarrow B$: $\{K_{A,B}\}_{K_b}$
3. B : decrypts to $K_{A,B}$

Figure 8.2: Simplistic Protocol

1. $A \longrightarrow S$: A, B
2. $S \longrightarrow A$: $\{K_b, B\}_{K_s^{-1}}$
3. $A \longrightarrow B$: $\{N_a, A\}_{K_b}$
4. $B \longrightarrow S$: B, A
5. $S \longrightarrow B$: $\{K_a, A\}_{K_s^{-1}}$
6. $B \longrightarrow A$: $\{N_a, N_b\}_{K_a}$
7. $A \longrightarrow B$: $\{N_b\}_{K_b}$

Figure 8.3: Needham-Schroeder Protocol with Asymmetric Keys

- No authentication of Alice. Anybody could have generated the key. An Attacker could easily distribute its own key to Alice and Bob and then listen on the conversation. (This could be countered by explicitly stating who initiates the conversation.)
- Man-in-the-Middle by intercepting the message and replacing it.
- Replay of key is possible. The session key could be cracked, given sufficient time, and then replayed back. (This would work even, if

8.2 Needham-Schroeder Asymmetric

The Needham and Schroeder are namesakes for two protocols for authentication and distribution of session keys.

Seven steps, described in [**Needham:1978:UEA:359657.359659**]

Needham-Schroeder Protocol Asymmetric

Problems with Needham-Schroeder:

- Missing Timelyness
- Replay of Authentication Messages (Nonces, see below)

8.3 Attack on Needham-Schroeder

The following attack is introduced by Gavin Lowe in [**Lowe95attackNeedhamSchroeder**]. It allows an attacker I to pose himself as A towards B.

$$
\begin{array}{rrl}
1.3 & A \longrightarrow I: & \{N_a, A\}_{K_i} \\
2.3 & I(A) \longrightarrow B: & \{N_a, A\}_{K_b} \\
2.6 & B \longrightarrow I: & \{N_a, N_b\}_{K_i} \\
1.6 & I \longrightarrow A: & \{N_a, N_b\}_{K_a} \\
1.7 & A \longrightarrow I: & \{N_b\}_{K_i} \\
2.7 & I(A) \longrightarrow B: & \{N_b\}_{K_b},
\end{array}
$$

Figure 8.4: Attack on Needham-Schroeder Protocol with Asymmetric Keys

> Principle 3: If the identity of a principal is essential to the meaning of a message, it is prudent to mention the principal's name explicitly in the message. **[abadi96prudent]**

Figure 8.5: Prudent Engineering Practice, Principle 3

Consider only the messages that facilitate authentication: 3, 6, and 7. An attacker I could now start an authentication with A and using this process to authenticate himself as A in a second conversation he starts with B:

Attack on Needham-Schroeder

where I is the attacker, denoted by $I(A)$ if posing as A, and the index number $i.j$ denotes protocol step j in the i-th conversation. (The first is undertaken with A and the second is undertaken with B.) The problem here is, that the messages do not make clear, for which communication the nonces are intended and thus do allow to swap them freely between different communications.

This flaw possibly motivates the statement of the third principle (Figure 8.8).

Prudent Engineering Principle

8.3.1 Revised Protocol

Thus it is important to explicitly state the identity of partners, especially in the authenticating parts of the protocol.

A way to prevent the previously described attack is proposed in [**Lowe95attackNeedhamSc**
Include responders identity in message 6:

6'. $B \longrightarrow A$: $\{B, N_a, N_b\}_{K_a}$ In that way the attacker is not able to just relay this message to A, who would expect a message from I, not

1. $A \longrightarrow S$: A, B
2. $S \longrightarrow A$: $\{K_b, B\}_{K_s^{-1}}$
3. $A \longrightarrow B$: $\{N_a, A\}_{K_b}$
4. $B \longrightarrow S$: B, A
5. $S \longrightarrow B$: $\{K_a, A\}_{K_s^{-1}}$

6'. $B \longrightarrow A$: $\{B, N_a, N_b\}_{K_a}$

7. $A \longrightarrow B$: $\{N_b\}_{K_b}$

Figure 8.6: Revised Needham-Schroeder Protocol with Asymmetric Keys

from B. With the full revised protocol shown in Figure 8.6

Revised Asymm. N-S

8.3.2 Session Key

One Problem with the asymmetric scheme is, that there still is no session key established. And asymmetric encryption schemes are usually not known for their efficiency. There is, thus an eighth message to be send — at least — to establish $K_{a,b}$. Or, as has been explained in previous lectures, negotiated, for example using the Diffie-Hellman protocol[1].

Asymmetric Key Exchange

Simplistic: 8. $A \longrightarrow B$: $\{K_{a,b}, A\}_{K_b}$

With freshness: 8'. $A \longrightarrow B$: $\{\{K_{a,b}, A, B, T\}_{K_a^{-1}}, A\}_{K_b}$

Clever: 8". $A \longrightarrow B$: $g^a mod n$
8". $B \longrightarrow A$: $g^b mod n$ (Diffie-Hellman)

8.4 Needham-Schroeder Symmetric

This protocol utilises a trusted server S to authenticate two participants A, B and securely generate and distribute a session key. We will successively learn, that this first version of the protocol thus had its flaws and had to be revised, before it could be incorporated into real-world protocols.

The protocol (Figure 8.10) runs as follows. A contacts the server, expressing the wish to communicate secure and authentic with B. The server S

[1]But keep in mind, that Diffie-Hellman was only two years published at that time and thus quite uncommon, as was public-key cryptography.

1. $A \longrightarrow S$: A, B, N_a
2. $S \longrightarrow A$: $\{N_a, B, K_{a,b}, \{K_{a,b}, A\}_{K_{b,s}}\}_{K_{a,s}}$
3. $A \longrightarrow B$: $\{K_{a,b}, A\}_{K_{b,s}}$
4. $B \longrightarrow A$: $\{N_b\}_{K_{a,b}}$
5. $A \longrightarrow B$: $\{f(N_b)\}_{K_{a,b}}$

Figure 8.7: Attack on Needham-Schroeder Protocol with Symmetric Keys

then answers with an message encrypted with the master-key between A and S. The included nonce N_a provides freshness and thus protection from replay attacks. $K_{a,b}$ is the session key for communication with B, which is included twice: directly, readable for Alice and within an enclosed data packet, which is encrypted with the master secret $K_{b,s}$.

The enclosed packet additionally contains the identification of Alice to state, that the enclosed key is usable only in communication between Alice and Bob. This enclosed packet is then relayed by Alice to Bob.

Symmetric Protocol

- Symmetric is faster
- Shorter Key Length
- Session Key are symmetric too

Needham-Schroeder Protocol Symmetric

At that point Alice can expect that only Bob is able to use $K_{a,b}$. But Bob has yet no proof that Alice is authentic. The next two messages are used to authenticate Alice. Bob crafts a nonce, and sends it encrypted to Alice. Alice, if authentic, is able to decrypt the nonce, apply a simple transformation that is known by both parties to it and sends it back using the session key again.

In the symmetric protocol encryption is used both for secrecy of the session key and mutual authentication of clients. If Alice is able to use $K_{a,b}$, then Bob assumes her authenticity and vice versa. Alice on the other hand has to trust S, that it encrypted the key in a way that only Bob is able to use it.

The following principle is not wronged by the use in the symmetric protocol, as long as we are clear about the meaning of encryption.

> Principle 3: If the identity of a principal is essential to the meaning of a message, it is prudent to mention the principal's name explicitly in the message. [**abadi96prudent**]

Figure 8.8: Prudent Engineering Practice, Principle 3

Prudent Engineering Principle

8.4.1 Timeliness of Communication

The problem of the protocol is, that it assumes, that no session key is ever disclosed to an attacker, or cracked.

But having gained knowledge on any session key, an attacker could simply replay the third message and would be able to correctly answer the challenge within the fourth message.

Spoofing Alice

3' $I \longrightarrow B$: $\{K_{a,b}, A\}_{K_b}$
4' $B \longrightarrow I$: $\{N'_b\}_{K_{a,b}}$
5' $I \longrightarrow B$: $\{f(N'_b)\}_{K_{a,b}}$

Thus, the attacker would be able to pose as Alice, because an old session key has been compromised. This is exactly not what Perfect Forward Secrecy is about. The reason is, that we have attached too much meaning to the nonce N_b, which proves knowledge of the key, but can not guarantee, that this is a recent key. Which motivates the next principle:

> Principle 6: Be clear what properties you are assuming about nonces. What may do for ensuring temporal succession may not do for ensuring association and perhaps association is best established by other means. [**abadi96prudent**]

Nonces itself are not generally sufficient to guarantee that messages are fresh and not simply replays. The attribute that the nonces in the above protocol shall prove is knowledge of a private key (via the ability to decrypt a nonce) but this does not prove freshness.

8.4.2 Revised Protocol (2)

Thus, a second way to improve the original Needham-Schroeder Protocol is to not only be explicit on the intended partners of the communication, but to provide freshness as well.

1.	$A \longrightarrow S$:	A, B, N_a
2'.	$S \longrightarrow A$:	$\{N_a, B, K_{a,b}, T, \{K_{a,b}, A, T\}_{K_{b,s}}\}_{K_{a,s}}$
3'.	$A \longrightarrow B$:	$\{K_{a,b}, A.T\}_{K_{b,s}}$
4.	$B \longrightarrow A$:	$\{N_b\}_{K_{a,b}}$
5.	$A \longrightarrow B$:	$\{f(N_b)\}_{K_{a,b}}$

Figure 8.9: Attack on Needham-Schroeder Protocol with Symmetric Keys

0.1	$A \longrightarrow B$:	A
0.2	$B \longrightarrow A$:	$\{A, N_0\}_{K_{b,s}}$
1.	$A \longrightarrow S$:	A, B, N_a
2'.	$S \longrightarrow A$:	$\{N_a, B, K_{a,b}, \{K_{a,b}, A\}_{K_{b,s}}, N_0\}_{K_{a,s}}$
3.	$A \longrightarrow B$:	$\{K_{a,b}, A, N_0\}_{K_{b,s}}$
4.	$B \longrightarrow A$:	$\{N_b\}_{K_{a,b}}$
5.	$A \longrightarrow B$:	$\{f(N_b)\}_{K_{a,b}}$

Figure 8.10: Attack on Needham-Schroeder Protocol with Symmetric Keys and prior nonce

The easiest way is to introduce time stamps, which thus would require synchronised time between communication partners.

N-S Protocol Symmetric Timestamped

Another way is to exchange nonce directly before communicating to the server. This could be seen as a commitment on a given time.

Needham-Schroeder Protocol Nonced

8.5 Certificate-based Key Exchange

What is the difference between Key Exchange Protocols and PKI infrastructures, i.e. certificates? Why not simply all use x509?

Comparision Certificates vs. Key Exchange

Key Exchange

- Interaction based
- Current communication
- Establish Session Key

Certificates

- Asynchronous
- Revocation difficult Either defined validity time, or certificate revocation lists, or both.

9

Federated Authentication

In this lecture I want to introduce WebID a, in my eyes, very intriguing technologically beautiful — yet pragmatic — solution to the problem of identities, authentication and control over one's identity. The fundamental idea is, that everything should be identified by an URI and that, by that URI, further information, credentials and more, can be found. The idea takes up wide-spread technology that every computer user nowadays knows how to use — the World Wide Web, and enhances it in a way that is backward compatible and still provides new functionality.

WebID is an authentication scheme based on URI, SSL, x509 certificates, and the semantic web. It has been proposed by Dan Brickley and Tim Berners-Lee in 2000 and is supporting distribution of services as well as individual control and well defined representation by agents. It further is notably well suited to be used in `rdf:foaf`.

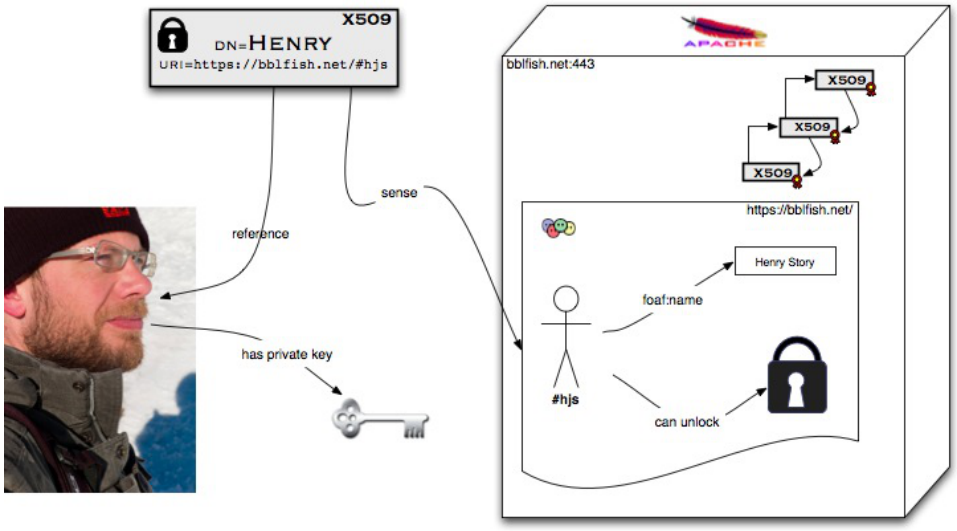
9.0.1 WebID Credentials

Henry has stored a public key in his FoaF-profile on his web server. He also has the matching private key stored locally accessible for the tools with which he accesses the web (which is mostly his browser). He further has the information stored on his server, that he has access to his profile.

9.0.2 WebID Authentication

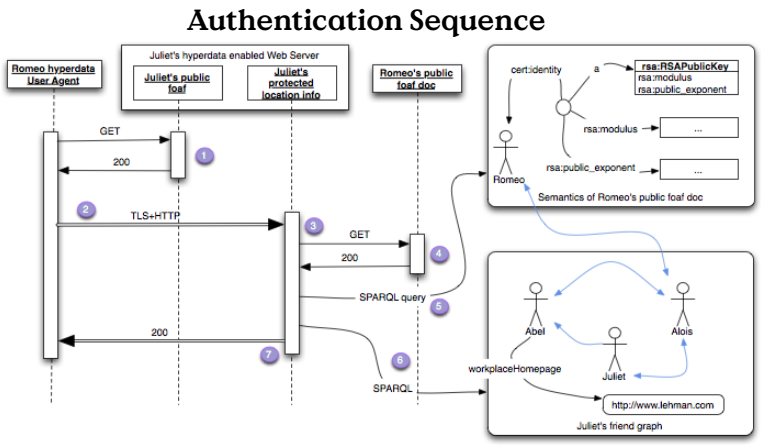
WebID makes use of TLS described in Section**??** to authenticate both users (clients) and servers.

The authentication sequence in Figure 9.2 describes the process of Romeo accessing the homepage of Juliet. Juliet's server has a database that



[http://www.w3.org/wiki/WebID]

Figure 9.1: WebID



[http://www.w3.org/wiki/Foaf%2Bssl]

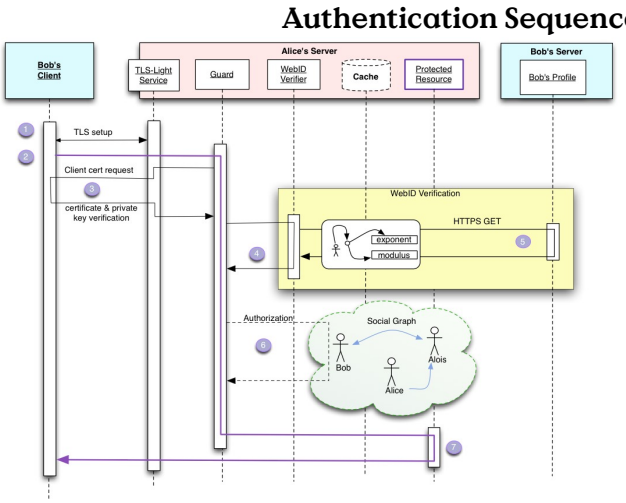
Figure 9.2: WebID Authentication

contains access control information in the form of a linked data graph. The steps are explained as follows:

1. In the first connection the user (Romeo) arrives on the home page of Juliet, after having received a note from her at a party. This page contains a login button or link.
2. Romeo clicks the login link, an https URL. Perhaps `https://juliet.example/login`
3. Juliet's server on receiving the HTTPS connection asks the client for his certificate. As a result a popup appears in Romeo's browser asking him to choose his WebId. (see the pictures of how different browsers present this). Romeo selects one of them, and this corresponding X.509 certificate is sent back to the server, which verifies that Romeo's browser has the private key associated with the public key published in it, using the standard https protocol.
4. The certificate also contains a Web ID, which is a URL such as `http://romeo.example/#me` that was placed in the Subject Alternative Name field of the cert. Juliet's server finds this and does an HTTP GET on it.
5. If the graph of the returned document contains the relations that `http://romeo.example/#me` has the public key specified in the certificate, the server knows that the person at the other end of the connection is `http://romeo.example/#me` .
6. Juliet' server can then check in its database if `http://romeo.example/#me` is known by any of Juliet's friends, as specified in their foaf files published on their servers.
7. Juliet's server authorized Romeo to see her file.

We can, more specifically, take a look at the precise sequence of messages exchanged for authentication. The process is essentially a client certificate request that is happening within an SSL-secured session that should be commonly opened with every server now.

1. Bob's Client must open a TLS [RFC5246] connection with the server which authenticates itself using well known TLS mechanisms. This may be done as the first part of an HTTPS connection [HTTP-TLS].
2. Once the Transport Layer Security [TLS] has been set up, the application protocol exchange can start. If the protocol is HTTP then the client can request an HTTP GET, PUT, POST, DELETE, ... action on a resource as detailed by [HTTP11]. The Guard can then intercept that request and by checking some access control rules deter-



[http://www.w3.org/wiki/Foaf%2Bssl]

Figure 9.3: WebID Authentication Sequence

mine if the client needs authentication. We will consider the case here where the client does need to be authenticated.

3. The Guard must requests the client to authenticate itself using public key cryptography by signing a token with its private key and have the Client send its Certificate. This has been carefully defined in the TLS protocol and can be summarised by the following steps:

   (a) The guard requests of the TLS agent that it make a Certificate Request to the client. The TLS layer does this. Because the WebID protocol does not rely on Certificate Authorities to verify the contents of the Certificate, the TLS Agent can ask for any Certificate from the Client. More details in Requesting the Client Certificate

   (b) The Client asks Bob to choose a certificate if the choice has not been automated. We will assume that Bob does choose a WebID Certificate and sends it to the client.

   (c) The TLS Agent must verify that the client is indeed in possession of the private key. What is important here is that the TLS Agent need not know the Issuer of the Certificate, or need not have any trust relation with the Issuer. Indeed if the TLS Layer could verify the signature of the Issuer and trusted the statements it signed, then step 4 and 5 would not be needed - other than perhaps as a way to verify that the key was still valid.

   (d) The WebID Certificate is then passed on to the Guard with the proviso that the WebIDs still needs to be verified.

4. The Guard then must ask the Verification Agent to verify that the WebIDs do identify the agent who knows the given public key.

5. The WebID is verified by looking up the definition of the URL at its canonical location. This can be done by dereferencing it. The Verification Agent must extract the public key and all the URI entries contained in the Subject Alternative Name extension of the WebID Certificate. A WebID Certificate may contain multiple URI entries which are considered claimed WebIDs at this point, since they have not been verified. The Verification Agent may verify as many or as few WebIDs it has time for. It may do it in parallel and asynchronously. However that is done, a claimed WebIDs can only be considered verified if the following steps have been accomplished

successfully:

(a) If the WebID Verifier does not have an up to date version of the WebID profile in the cache, then it must dereference the WebID using the canonical method for dereferencing a URL of that scheme. For an https://... WebID this would be done using the [HTTP-TLS] protocol.

(b) The returned representation is then transformed into an RDF graph as specified in Processing the WebID Profile

(c) That graph is then queried as explained in Querying the Graph. If the query succeeds, then that WebID is verified.

6. With the set of verified WebIDs the Guard can then check its access control rules using information from the web and other information available to it, to verify if the referent of the WebID is indeed allowed access to the protected resource. The exact nature of those Access Control Rules is left for another specification. Suffice it to say that it can be something as simple as a lookup in a table.

7. If access is granted, then the guard can pass on the request to the protected resource, which can then interact unimpeded with the client.

WebID other Authentication

If the Client does not send a certificate, because either it does not have one or it does not wish to send one, other authentication procedures can be pursued at the application layer with protocols such as OpenID, OAuth, BrowserID, etc...

9.0.3 Certificates in FoaF

Certificate in FoaF

```
<cert:key>
    <cert:RSAPublicKey>
    <cert:label>Lars Fischer</cert:label>
    <cert:modulus
       rdf:datatype="http://www.w3.org/2001/XMLSchema#hexBinary">
          BAAFB2E38A4E4FD49F9F0285D5929CA45EB1833607425E60CBB28AD
          31C61F146067989FBF3A47DB5B9D7E52C3AD337FC978093033A8E7B
          226AA6ACDC8FB6BBA7
    </cert:modulus>
    <cert:exponent
       rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">
         65537
    </cert:exponent>
    </cert:RSAPublicKey>
</cert:key>
```

9.0.4 WebID Conclusion

WebID Summary

- SSL based authentication
  - Browser has private key
  - any user action authenticated
- identifier: URI
- Webservices to write

OAuth is an approach to authorise third parties to access defined rights to resources of an owner on a server. A very often used example is that of a photo printing service (the client), at which a user (the owner) using a browser orders a few prints of pictures (the resources) stored on a remote database (the server). Before OAuth, it was common, that the user had to provide any service to access his resources with his personal credentials to let the services client act on his behalf towards the server. As most often passwords are used for authentication, the service gained full access to the owner's resources for the whole validity period of the credential.

The main objective of OAuth is to allow for authorisation that is both restricted in its validity time, fine granular resource and rights selection. Disclosure of the password obviously is not sufficient.

In order to facilitate this, the owner, who is assumed to browse the clients web-page, has to authenticate himself using his server-password. The client then is provided with a token to authenticate itself for access to the resources towards the server. The protocol uses HTTP Redirection to guide the owner from the clients web-page to the authentication pages of his (trusted) server and back to the client. The client explicitly has to state the requested access rights towards the server and the owner explicitly grants or denies these rights. In other words: the owner communicates directly with his server and not through the client.

OAuth Overview

resource owner (User) resource owner, the user that owns a web-resource and wishes to grant (temporary) access to that resource to the

client (Consumer) client, who might be a secondary webservice, a mobile app or similar, that should do something with a users resource hosted at a

server (Service Provider) server, that hosts and controls the resource owner's data.

(Terms in parentheses are used in the original spec, other terminology is from the RFC.)

client credentials

temporary credentials

token credentials

OAuth

OAuth 1.0 RFC 5849 [**rfc5849**]Session Fixation Attack

OAuth 2.0 RFC 6749: OAuth 2.0, RFC 6750 OAuth Bearer Tokens

- als OpenID Connect auch zur Authentifizierung
- Facebook (seit 2012), Microsoft, Google,...experimental

OAuth provides a protocol to grant an application limited access for a limited time to a web-resource.

OAuth in the Wild

Sources State 2011:

OAuth 1.0 Session Fixation `http://oauth.net/advisories/2009-1/`

OAuth 2.0 **Facebook** `http://developers.facebook.com/docs/authentication/?_fb_noscript=1`

Google `http://googlecode.blogspot.com/2011/03/making-auth-easier-oauth-20-for-goog html`

Microsoft `http://windowsteamblog.com/windows_live/b/developer/archive/2011/05/04/announcing-support-for-oauth-2-0.aspx`

9.1 OAuth Phases

OAuth Phases

Preliminaries:

0. (Client Credentials) Client Credentials are not part of the protocol, and are obtained beforehand to authenticate the client-software/instance.

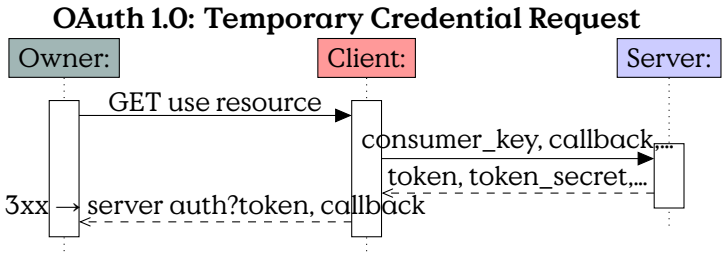


Figure 9.4: OAuth v1 Temporary Credential Request

Server Communication Endpoints:

1. Temporary Credential Request The first step in the protocol is for the client to request a temporary credential. These are used for Resource Owner Authorisation and afterwards by the client to claim a token from the server.

2. Resource Owner Authorisation The resource owner authorises the client by authenticating itself in a request to the server, including the clients temporary credential.

3. Token Request Using an previously authorised temporary credential, the client requests a token from the server for authentication of further requests.

9.1.1 OAuth Temporary Credential Request

As the owner requests some action from the client, that requires access to resources on the server, the clients sends a request to the server, requesting temporary credentials (in form of a token). The client then replies to the owner with a redirection to the authorisation endpoint of the server containing the token (not the token_secret). (Figure 9.4)

9.1.2 OAuth v1: Resource Owner Authorisation

The owner follows the clients redirection to the servers authorisation endpoint, using the token. There the owner authenticates himself to the server and explicitely authorises the client to access the given resource. The server then redirects the user to the client-callback previously set, providing a token_verifier. The owner calls (gets redirected) to the clients callback and thus issues the token_verifier to the client. (Figure 9.5)

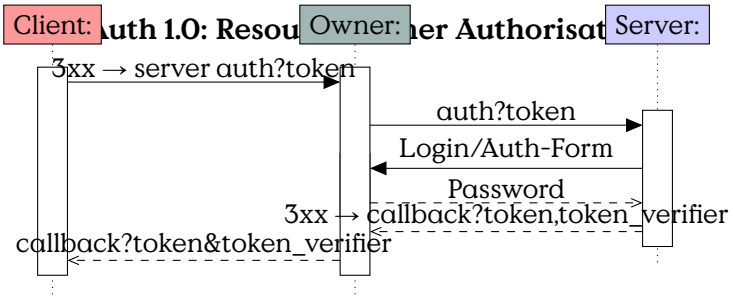


Figure 9.5: OAuth v1 Resource Owner Authorisation

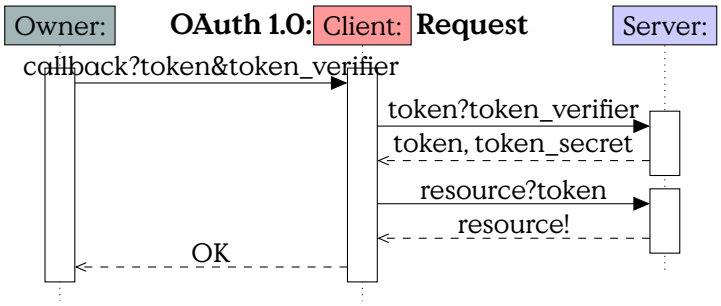


Figure 9.6: OAuth v1: Token Request

9.1.3 OAuth v1 Token Request

After the client received a token_verifier from the owner it requests an access token from the server, proving authorisation with the token_verifier. The server grants a new access token and the client can then request/use the resource. At the end it is nice to inform the owner of the result. (Figure 9.6)

OAuth 1.0: Session Fixation

- Attacker uses (honest) client to get temp. credential
- Attacker does not follow authorisation redirect
- Attacker tricks resource owner to click redirect
- Owner authorises honest client at server
- Attacker uses saved temp. credential to request token
- Attacker uses token to access resource

(source: http://oauth.net/advisories/2009-1/)

OAuth 2.0

Why update?

- OAuth 1.0 too complex
- Scalability issues
- Incompatible to existing Auth. Schemes

(source http://hueniverse.com/2010/05/introducing-oauth-2-0/)

State:

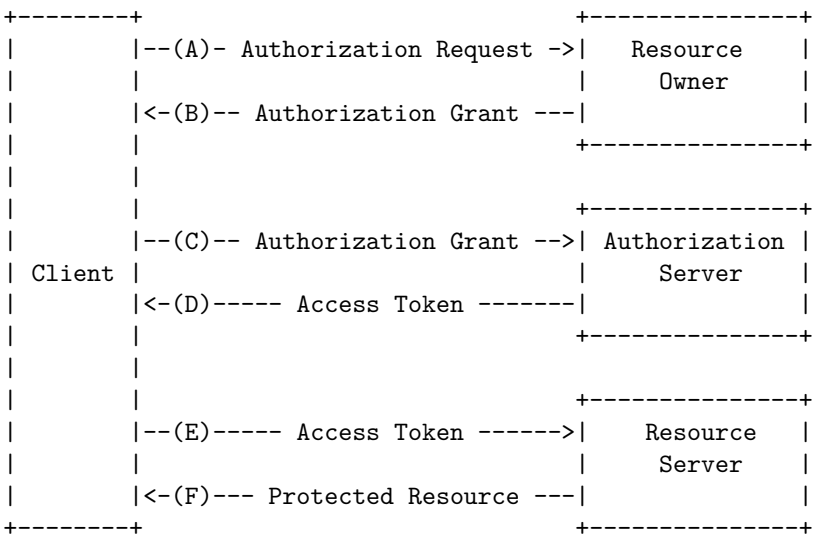
- Proposed Standard RFC 6749 (see https://datatracker.ietf.org/doc/html/rfc6749)
- Introduces an Authorization Server

OAuth 2.0: Differences

- Role Separation: Authorization Server
- 4 Different Protocol Flows:
  - Authorization Code (OAuth 1)
  - Implicit (no client authentication)
  - Owner Password CredentialsCredentials are directly handled by the client. This gives the client full access to the resource server.
  - Client CredentialsClient uses own access to access resources. No owner authorisation required.
- Bearer Tokens
- Short-Lived Tokens/Long-Lived authorizations

(source http://hueniverse.com/2010/05/introducing-oauth-2-0/)

OAuth 2.0: Flow



(source: RFC 6749)

+———+ +———————+ ||–(A)- Authorization Request →| Resource ||||Owner ||
|←(B)– Authorization Grant —|||| +———————+ |||| +———————+ ||–(C)– Authorization Grant –>| Authorization ||Client ||Server |||←(D)—– Access Token ———-|||| +———————+ |||| +———————+ ||–(E)—– Access Token ——>| Resource ||||Server |||←(F)— Protected Resource —|| +———+ +———————+

10

Software Security

10.1 Vulnerabilities

Vulnerability

Vulnerability: an exploitable weakness

- unintended "feature"
- 0-day: not publicly known vuln

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.

Vulnerability Example

- 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

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

- CWE 697: Incorrect Comparison
- CWE 713: OWASP Top Ten 2007 Category A2 - Injection Flaws

The Bug

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)) \{
```

```
                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.

Evaluation of user address

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

Exploit

Netcat to SMTP-Server

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

10.2 Vulnerability Patterns

10.2.1 Buffer Overflow

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

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

Buffer Overflow Code

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

```
}

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

10.2.2 Unbounded Copy

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

Unbounded Copy

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

Figure 10.1: If `s` exceeds the buffer thats not good. [http://guidanceshare.com/wiki/Buffer_Overflow_Vulnerability_Pattern 2022-07-19]

10.2.3 Non-Null-Terminated String

Missing 0-Term in String

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

Figure 10.2: Missing Null-Termination [http://guidanceshare.com/wiki/Buffer_Overflow_Vulnerability_Pattern 2022-07-19]

10.2.4 Source-sized copy

Source-sized Copy

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

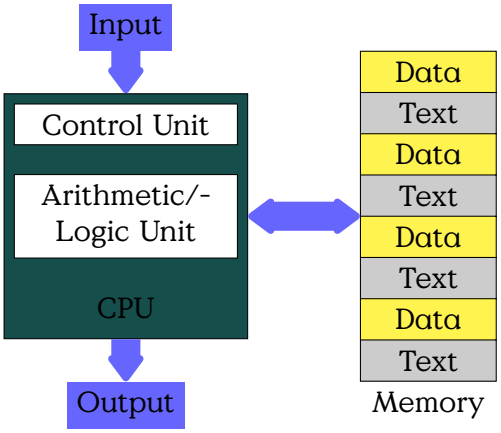
Figure 10.3: Always check for the amount of memory available. [http://guidanceshare.com/wiki/Buffer_Overflow_Vulnerability_Pattern 2022-07-19]

User-Defined Buffer-Size

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

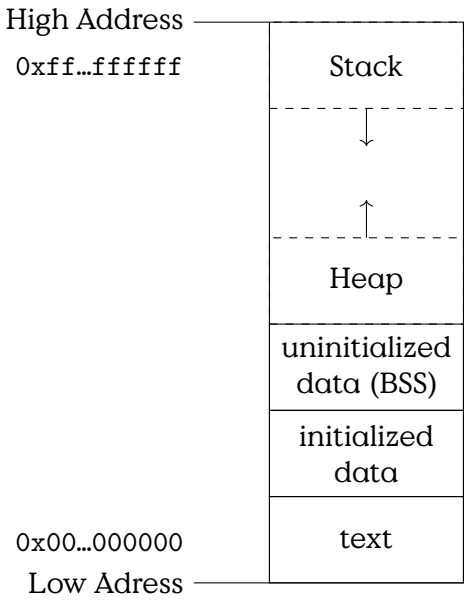
Figure 10.4: Do not let the user decide on how large his string his. [http://guidanceshare.com/wiki/Buffer_Overflow_Vulnerability_Pattern 2022-07-19]

Von Neumann Architecture



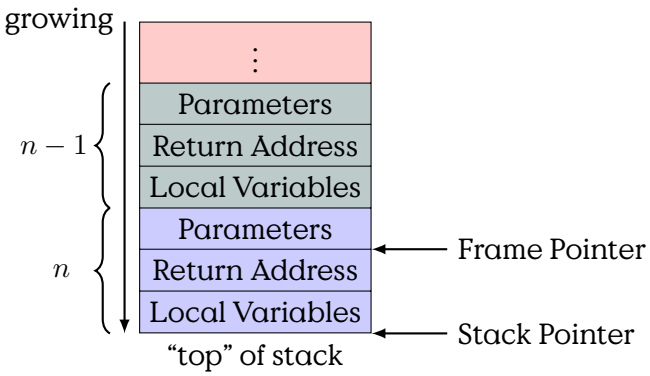
- Code/Data Share Memory
- Process Separation

Process Memory



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

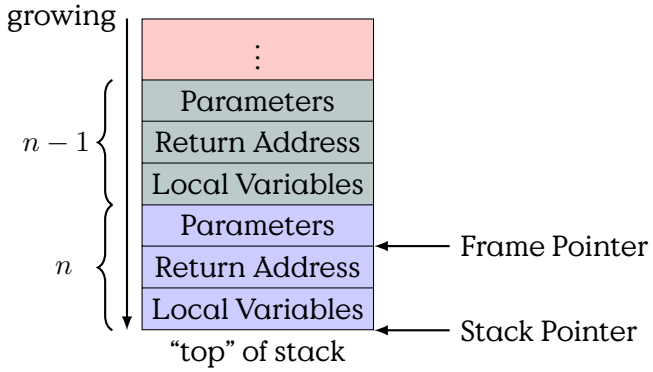
Stack Frames



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

Calling Convention

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



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

Simple Call in C

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

void caller() {
  int a = 1;
  int b = 2;
```

```
  int c = 3;
  int sum;
  sum = callee(a,b,c);
}
```

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

Compiled to Assembler

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

```
caller:
  pushl   %ebp
  movl    %esp, %ebp
  subl    $28, %esp
  movl    $1, -4(%ebp)
  movl    $2, -8(%ebp)
  movl    $3, -12(%ebp)
  movl    -12(%ebp), %eax
  movl    %eax, 8(%esp)
  movl    -8(%ebp), %eax
  movl    %eax, 4(%esp)
  movl    -4(%ebp), %eax
  movl    %eax, (%esp)
  call    callee
  movl    %eax, -16(%ebp)
  leave
  ret
```

`LEAVE` is synonymous to

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

`CALL` is synonymous to

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

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

HowTo Shellcode

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

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

Shellcode Example

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

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

The source code has been compiled using:

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

Calc Machine Code

`objdump -d calc.S`:

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

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

Calc Extract

```
hexdump -C calc.S
00000000  7f 45 4c 46 01 01 01 00  00 00 00 00 00 00 00 00  |.ELF............|
00000010  01 00 03 00 01 00 00 00  00 00 00 00 00 00 00 00  |................|
00000020  b8 00 00 00 00 00 00 00  34 00 00 00 00 00 28 00  |........4.....(.|
00000030  09 00 06 00 55 89 e5 83  ec 10 c7 45 fc 03 00 00  |....U......E....|
00000040  00 c7 45 f8 04 00 00 00  8b 45 f8 8b 55 fc 01 d0  |..E......E..U...|
00000050  c9 c3 00 00 00 47 43 43  3a 20 28 44 65 62 69 61  |.....GCC: (Debia|
```

Code from `0x3a` until `0x4f`

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

Calc Encode

Code from `0x3a` until `0x4f`

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

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

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

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

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

Position inside Memory?

```
1  _start:
2   jmp short ender
3  starter:
4   pop ebx                  ;get the address of the string
5   xor eax, eax
6
7   mov [ebx+7 ], al         ;put a NULL where the N is
8   mov [ebx+8 ], ebx        ;put the address of the string
9   mov al, 11               ;execve is syscall 11
10  lea ecx, [ebx+8]         ;load the address of string
11  int 0x80                 ;call the kernel, execve
12
13 ender:
14  call starter
15  db '/bin/shNAAAABBBB'
```

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

10.2.6 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[2], but I've heard LLVM[3] 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. 2 — Transform the following C-code into usable shellcode, test it by directly executing a pointer to a string.

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

10.2.7 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>
```

10.2.8 Integer Overflow

Arithmetic 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] */
```

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

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

```
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].

Integer Fails

- Arithmetic Overflows
- Sign Overflows (MAXINT+1 = 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

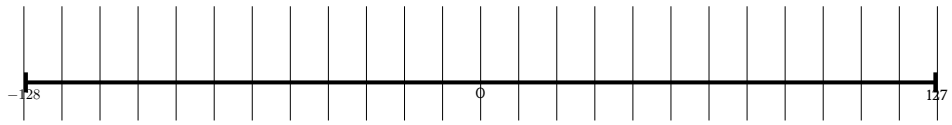
This section provides an introduction to Integer Overflow vulnerabilities. The root cause of the problem is the wrap-around of the implementation of integer operations in (all) processors. As a result most programming languages implement integers not as numbers of arbitrary value, but as a number between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

The unintended effect of integer overflows can range from memory corruption to logic failures. The former, memory corruption, is only valid for programming languages that allow to handle process memory addresses directly, like C/C++/C#. Logic failures can occur in most other

languages like Java, Python and such. Some compilers provide flags that enable overflow protection code to be included, but this obviously comes at some computational cost.

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

Integer Overflow



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

Arithmetic Overflow Example

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

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

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

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

[source: phrack 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].

Integer Vulnerability Patterns

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

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

but

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

Results:

- Memory Access
- Faulty Branching

10.2.9 Incompatible Types

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

Incompatible Range

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

10.2.10 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[4], 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.

Type Casting Types

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

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

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

Sign extension upcast

Upcasting a positive value:

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

Upcasting a negative value:

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

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

Binary Operations Fail

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

Figure 10.6: XOR will cast both operators to `int`. [**Howard:2009:DSS:1594832**]

10.2.11 Arithmetic Overflow

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

Arithmetic Overflow

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

Figure 10.7: Overflowing `bytesRead` while reading from unsecure source. [http://guidanceshare.com/wiki/Integer_Overflow_Vulnerability_Pattern, 2022-07-19]

10.2.12 Mistaken Operator Precedence

You are trying your best to

Mistaken Operator Precedence

```
void PartialRecordBytes(int maxGet)
{
  int counter = 0;
  //trying to allocate 3/4 of the size
  //but operator precedence makes an overflow possible
```

```
  int partialMaxGet = maxGet*3/4;
  char buf[partialMaxGet];
  while (counter < partialMaxGet)
  {
    counter += getFromInput(buf+counter);
  }
 }
```

10.3 Fixes and Non-Fixes

Arithmetic Overflow Non-Test

Compiler optimizes to `true`

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

Figure 10.8: [**Howard:2009:DSS:1594832**]

Filed and disclosed as CVE-2019-19307[5], 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:

```
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;
```

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

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`, .

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.

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/470df6dca29d8c7319d587f10cde 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.)

10.3.1 Format String

To demonstrate a simple formatstring attack try "`Bob %x %x`" as `argv[1]` in the code of formatstring[6]:

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

```
  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 ;
}
```

10.4 Injection

SQL-Injection

Take a look at the example code.

Vulnerable Example

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

Escape: ")

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

Escape: "

SQL Statement Cheat

Auswahl von Mengen:

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

Vereinigung von Mengen:

<Select_Statement1> UNION <Select_Statement2>

Conditions:

AND, OR, <row>=<val>

regexps, functions...

10.5 Basic Security Measures

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

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

Input Validation

Example (whitelisting in Ruby)

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

Output Encoding

Encode HTML Output in Ruby

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

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

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

HTML-Entities:

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

HTML Output Encoding

```
out.println(this.header()+
        "<h1>;XSS UserList" + "</h1>;\n" + "<ul>\n");
while(users.hasNext()) {
    User user = users.next();
    /* Output Encoding, you are not doing it! */
    out.println("<li>"+ Encode.forHtmlContent(user.getUname())
      + ":" + Encode.forHtmlContent(user.getPass()) +
    "</li>");
}
out.println("</ul>\n"+ this.footer());
```

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

Security Mindset

> "any person can invent a security system so clever that he or she can't think of how to break it" (interpretation by Cory Doctorow in 2004) [7]

Solitaire

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

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

Secure Development Guidelines

- OWASP SAMMSecurity Assurance Maturity Model
  - Security Practices for:
    - Governance
    - Design

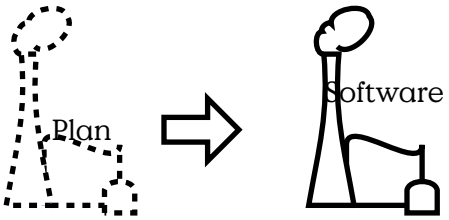
[7] The original quote read "Anyone, from the most clueless amateur to the best cryptographer, can create an algorithm that he himself can't break." and has been phrased in 1998 [https://www.schneier.com/blog/archives/2011/04/schneiers_law.html]

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

10.5.1 Systemdesignprinzipien

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

Software Entwicklung



Basic engineering phases:

- Concept development
- Engineering development
- Post-development

[MITRE: Systems Engineering Guide]

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

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

Table 10.1: Saltzer/Schroeder Security Design Principles [source: http://msdn.microsoft.com/en-us/magazine/cc163519.aspx, 2013-09-26]

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

See Table 10.1.

Saltzer/Schroeder

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

Secure Coding Practise

1. Validate Input
2. Heed compiler warnings
3. Architect and design for security policies
4. Keep it simple
5. Default deny
6. Adhere to the principle of least privilege
7. Sanitize data sent to other systems
8. Practise defence in depth
9. Use effective quality assurance technique
10. Adopt a secure coding standard

Bonus:

- Define security requirements
- Model threats

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

Open Web Application Security Project (OWASP) provides a exhaustive checklist for secure coding.

Secure Coding Practise

1. Input Validation
2. Output Encoding
3. Authentication and Password Management (includes secure handling of credentials by external services/scripts)
4. Session Management
5. Access Control

6. Cryptographic Practices
7. Error Handling and Logging
8. Data Protection
9. Communication Security
10. System Configuration
11. Database Security
12. File Management
13. Memory Management
14. General Coding Practices

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

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

Input Validation

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

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

Input Validation

Example (whitelisting in Ruby)

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

Output Encoding

Encode HTML Output in Ruby

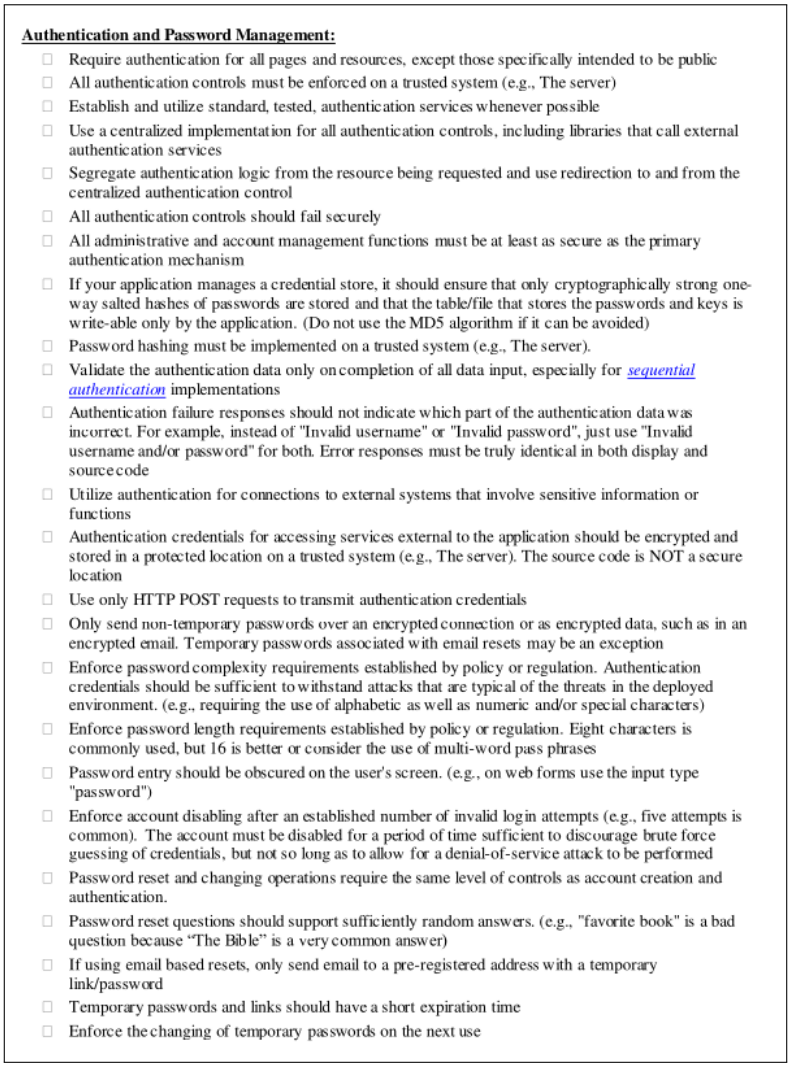
```
output = 'bla<script>alert()</script>'

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

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

HTML-Entities:

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



Authentication and Password Management:

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

Figure 10.9: Example page from the OWASP Secure Codinc Practise

11

WebSicherheit

11.1 Domain Name System (DNS)

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

Domain Name System (DNS)

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

DNS Resource Records

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

Usually DNS lookups are executed in the background unseen by the user. For administration and debugging purposes it is useful to explore avail-

able records. Many Linux installations have the program `dig` available for this purpose.

DNS Lookup

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

Further interesting parameters

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

11.2 Domain Name System Security Extensions (DNSSEC)

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

DNSSEC

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

DNSKEY contains (zone) key

RRSIG contains signature

A DNSKEY Resource Record (RR) stores a public key for a given DNS Zone, e.g. Figure 11.1.

RRSIG Format

DNSKEY RR

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

Figure 11.1: DNSKEY RR Example

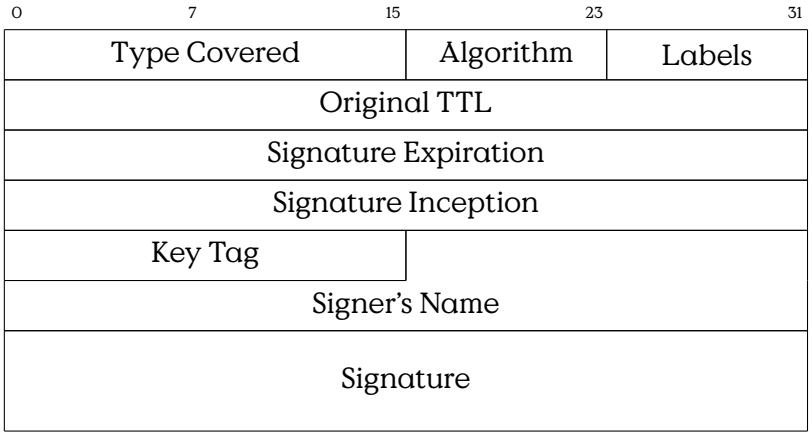


Figure 11.2: RRSIG Format

RRSIG example

```
host.example.com. 86400 IN RRSIG A 5 3 86400 20030322173103 (
                                20030220173103 2642 example.com.
                                oJB1W6WNGv+ldvQ3WDG0MQkg5IEhjRip8WTr
                                PYGv07h108dUKGMeDPKijVCHX3DDKdfb+v6o
                                B9wfuh3DTJXUAfI/M0zm0/zz8bW0Rznl8O3t
                                GNazPwQKkRN20XPXV6nwwfoXmJQbsLNrLfkG
                                J5D6fwFm8nN+6pBzeDQfsS3Ap3o= )
```

Figure 11.3: RRSIG example

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

Figure 11.4: 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

11.3 DNS-based Authentication of Named Entities (DANE)

RFC 6698 defines DNS-based Authentication of Named Entities (DANE), a standard for binding x.509 certificates to DNS records to secure communication using TLS. DNS-based Authentication of Named Entities (DANE) introduces TLSA resource records to DNS. DANE can thus be used to secure transport connection. DANE basically provides certificate constraints atop of Domain Name System Security Extensions (DNSSEC). [**RFC6698**]

DANE

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

DANE Lookup

11.4 Email Security

Email, the Simple Mail Transfer Protocol (SMTP) goes back to 1981, is the single fundermental communication in the Internet. The standard has been extended and redrafted, its heritance goes back to the ARPA Network Text Messages, many times. There are — at least — six Request for Comments (RFCs) with the title "Simple Mail Transfer Protocol". Email addresses often are used as primary identifier to register at various service providers. The authority over an Email postbox is commonly used to authenticate these identities. An overview of the existing overall structure is described in Internet Mail Architecture [**rfc5598**].

SMTP provides no security features. Authenticity of principals or secrecy of contents are not provided. Basically the protocol allows everybody to send or relay messages from and to everybody. And, while this renders the protocol comparatively simple and extensible, experience shows, that

— without additional security technology — SMTP can und is easily misused. Unsolicitated mass emails make up a large portion of all messages, phishing emails are a mayor attack vector and most communication can easily be evesdropped on.

A first collection of techniques common or less common to secure email.

- HashCash **??**
- DomainKeys Identified Mail (DKIM): Authenticates the domain part auf an email using credentials distributed in TXT-Entries in DNS
- S/MIME: Format to include security-related parts in emails
- PGP: Certification and cryptographic protocol for end-to-end encryption and authentication
- X.509: like PGP, but based on PKI 6.2
- Greylisting
- SpamFiltering (Bayesian)
- Sender Policy Framework (SPF)
  - Authorizes Domain of either MailFrom or EHLO
  - List of authorised MX
  - Published in DNS TXT RR
- DKIM (DomainKey Identification)
  - Authentication of sending Server,
  - Key in DNS Records
- Domain-based Message Authentication, Reporting and Conformance (DMARC)
  - Policies published by Sender MTA
  - via DNS
  - Uses further protocols for sender/relay authentication
  - Defines under which conditions an email would pass as legitimate
    * SPF and/or DKIM
    * Identifier Alignment [**rfc7489**]
      - IP aligns with Sender Domain
      -

Domain-based Message Authentication, Reporting and Conformance (DMARC) Example

```
$ dig _dmarc.hs-bremerhaven.de TXT
[…]
_dmarc.hs-bremerhaven.de. 21600 IN TXT "v=DMARC1;
                                        p=reject;
                                        sp=reject;
                                        pct=100;
                                        rua=mailto:dmarc@hs-
bremerhaven.de;
                                        ruf=mailto:dmarc@hs-
bremerhaven.de;
                                        adkim=r;
                                        aspf=r"
[…]
```

Figure 11.5: Example DMARC TXT RR

v=DMARC1 Version

p=reject Domain Policy (none/quarantine/reject)

sp=reject Subdomain Policy

pct=100 Percentage of "bad" emails to apply policy to

rua= ... Aggregate reporting URI

ruf= ... Failure reporting URI

adkim=r DomainKeys Identified Mail (DKIM) allignment mode (relaxed/strict)

aspf=r Sender Policy Framework (spf) allignment mode (relaxed/strict)

12

Threat Modelling

Wir behandeln Angriffsmodellierung hier ausschließlich mit dem Ziel den Schutz gegen Angriffe zu verbessern. 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.

12.1 Risk

Im Rahmen eines Sicherheitsprozesses, wie dem in Abb. 14.4 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.

Angriffsmodellierung

Ziele

- 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.

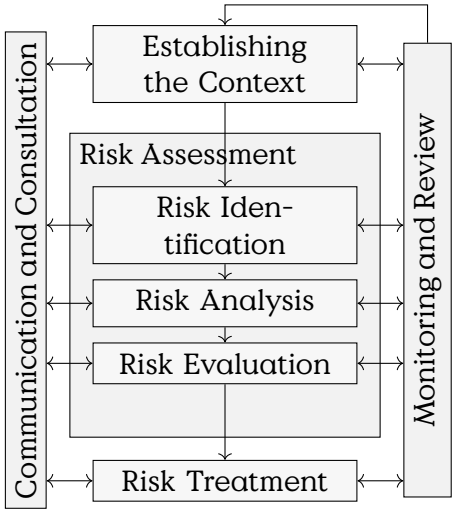


Abbildung 12.1: Risikomanagementprozess nach ISO/IEC 31000

Risk is a term denoting a potential for damaging events. As this concerns future events, uncontrollable actors, unknown factors and complex interdependencies, estimation of risk is always riddled with uncertainty. A number of tools and simplifications have been developed to operationalise risk in order to enable descision-making.

The broadest and fundamental simplification is the general risk equation that defines risk as an expectation of likelyhood and amount of damage,

$$R = P \times I, \tag{12.1}$$

where R denotes "risk", P a likelyhood estimate of I impact (the amount of damage).

Threat Levels

VERY HIGH	Existential Consequences for Organisation or Large Parts of Society
HIGH	Non-Functional Large Areas, Significant Impairment, Affects Third Parties
NORMAL	Impairment of Processes
none/low	No significant consequences

Table 12.1: Threat Levels

Threat levels are determined with respect to the consequences of a security breach. Very high levels are used if failure has existential consequences for an organisation or large parts of society or economy. High levels are used if damages render key areas of an organisation non-functional, the damage leads to significant impairment of the organisation or affects third parties. Normal levels are used if any level of impairment is to be expected on security fails. If no impairment is to be expected, no threat level is assigned, i. e., the component/objective is of no importance to security. (Table 14.1)

A risk matrix is a simplified model of the generic risk equation.

Risk Matrix

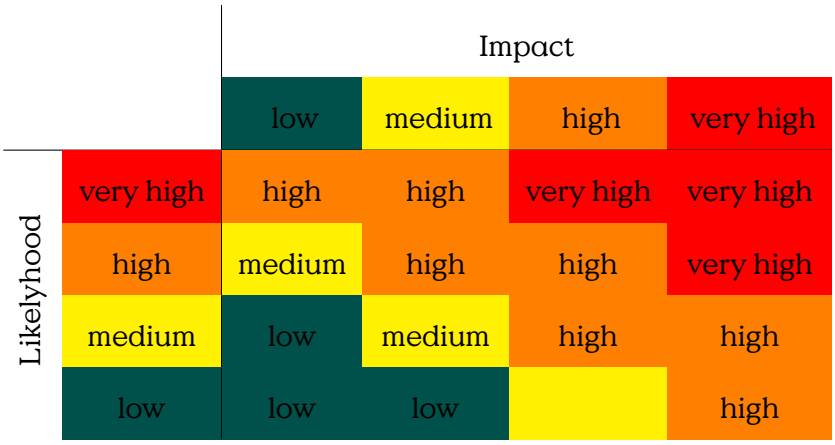


Figure 12.2: Example Risk Matrix

12.2 Elemente der Angriffsmodellierung

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

Komponenten von Angriffsmodellen

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

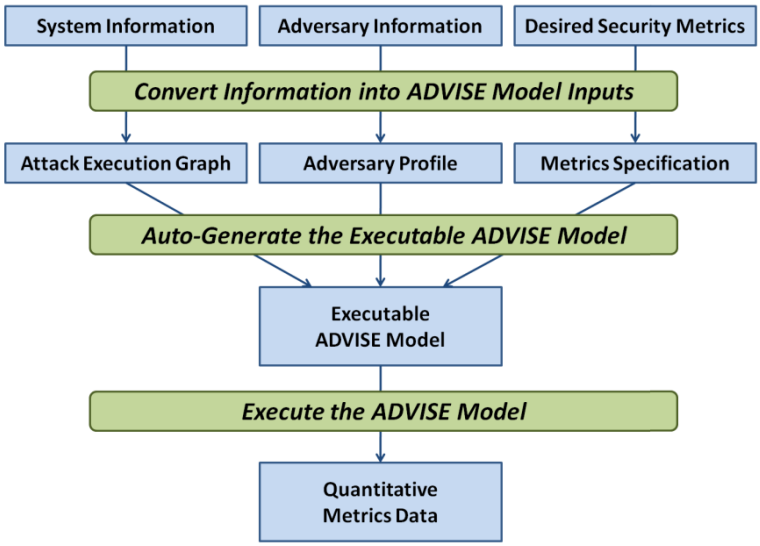


Figure 12.3: ADVISE methodology [**LeMay2011aeg**]

Ich möchte hier unterschiedliche Stufen der Angriffsmodellierung unterscheiden.

Konzepte der Angriffsmodellierung

- Angriffspfade:
  - Attack Tree [**Saltzer**, **Schneier**] siehe Abb. **??**

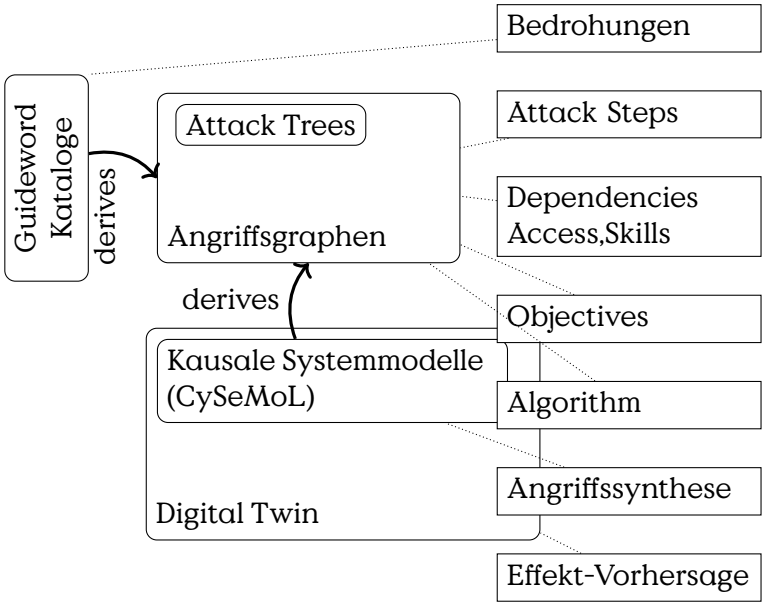


Figure 12.4: Zusammenhänge unterschiedlicher Konzepte der Angriffsmodellierung

  - 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 (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

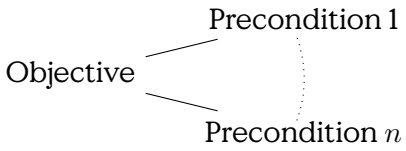
12.3 Attack Trees

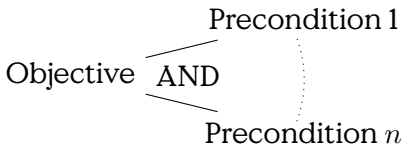
Attack Trees are a (if not the) fundamental method to systematically explore attack vectors and preconditions to attack objectives. They can be used to explore different possibilities to reach an objective, quantify the relative costs of different attack path or the minimum costs of an attacker to breach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

Attack Tree Components





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

Costs are one factor to estimate the probability of an attack. Other influencing factors are motivation of an attacker, e. g., by estimating the

profit an attacker can make from a successful attack. Aggregating this into a single measure provides a frequency or general probability of an attack.

Attack Tree Annotation

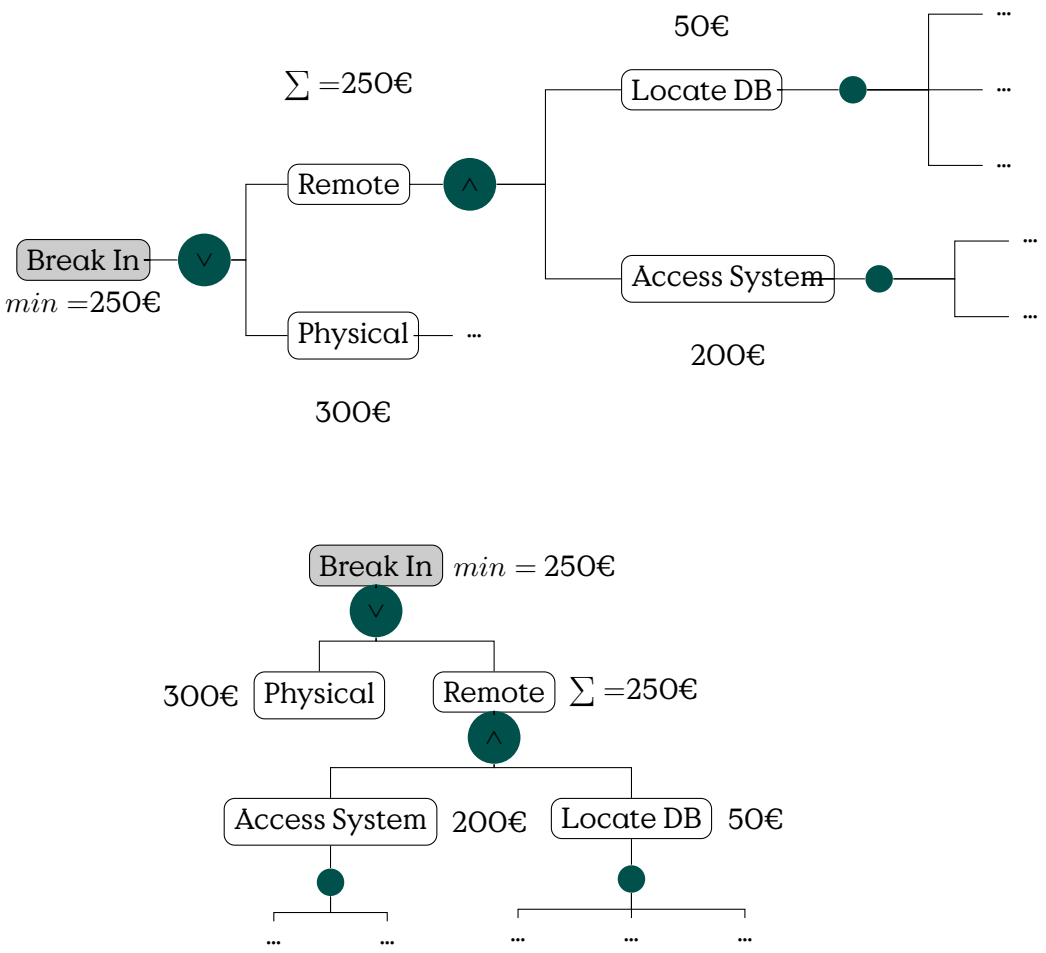


Figure 12.5: Example Attack Tree

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

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

12.4 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 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 12.6, 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.

Erstellung von AEGs

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 12.7 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).

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.

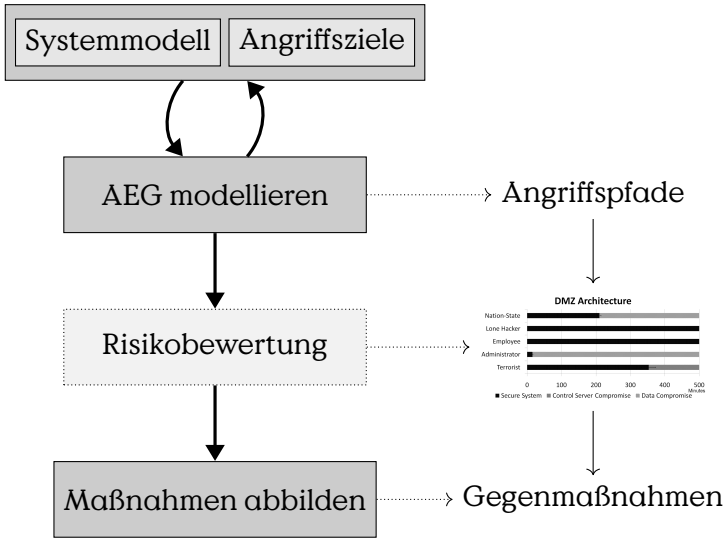


Abbildung 12.6: Prozess zur Erstellung von Angriffsgraphen

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.

Attack Execution Graph (AEG)

Komponenten

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

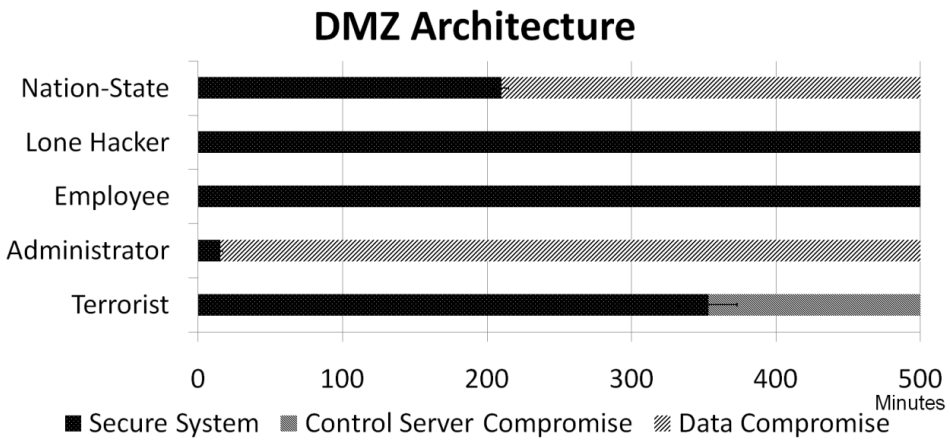


Abbildung 12.7: AEG-result, time-to-compromise for different threat agent models. [**LeMay2011aeg**]

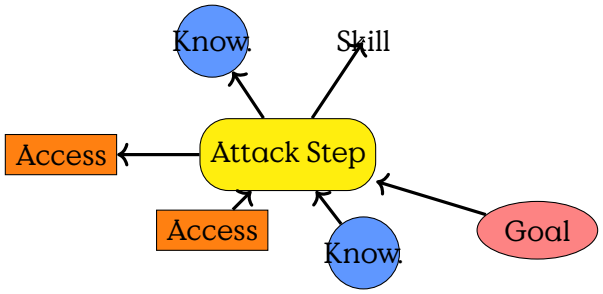


Abbildung 12.8: Knotenklassen und Kanten in AEG

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. 12.15)

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 Ausführungsreihenfolge von Angriffsschritten. In der Regel sind AEG nicht zyklisch. (Abb. 12.9)

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.)

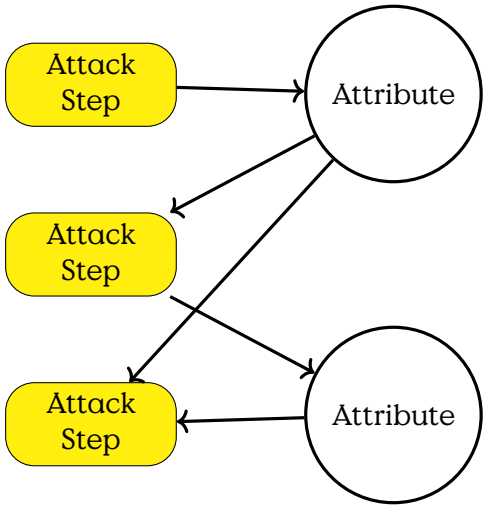


Abbildung 12.9: Die Knotenmengen in AEG sind paarig, d.h. es gibt Kanten nur zwischen Angriffsschritten und anderen Knoten. Es gibt keine Kanten zwischen Skill, Knowledge, Access, Goal-Knoten, und keine Kanten zwischen Angriffsschritten.

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.

Attack Step

Kantenspezifikation

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 12.10 stehen jeweils für eine oder mehrere Bedingungen (Skill, Knowledge, Access).

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. 12.11). Oder, dass eine Vorbedingung

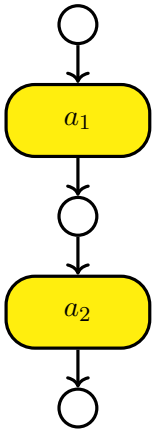


Abbildung 12.10: Building a Sequence of Attack Step

zwei unterschiedliche Angriffsschritte ermöglicht, welche dann wieder in disjunkte Pfade münden (Verzweigung B in Abb. 12.12).

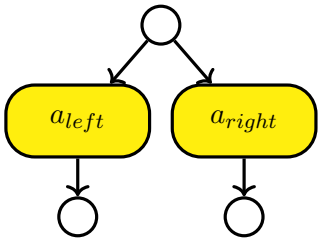


Abbildung 12.11: Verzweigung durch zwei mögliche Angriffsschritte

Als letzten Punkt müssen wir schauen, wie Disjunktion und Konjunktion von Vorbedingungen in AEG dargestellt werden können. (Abb. 12.13 und 12.14)

Beispiel Wasserversorgung

High-Level Beispiel

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

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

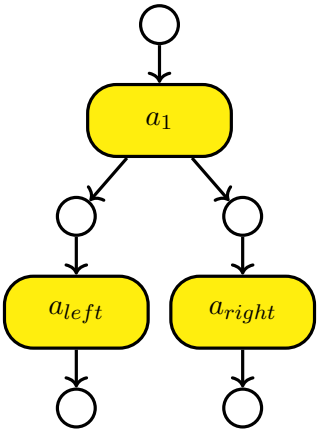


Abbildung 12.12: Verzweigung durch zwei Ergebnisse

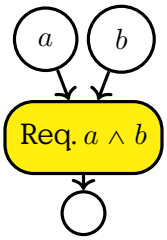


Abbildung 12.13: Konjunktion der Vorbedingungen eines Angriffsschrittes ist die normale Behandlung der Bedingungen

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

State Transition $E_i : X \times O_i \to X$. 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.

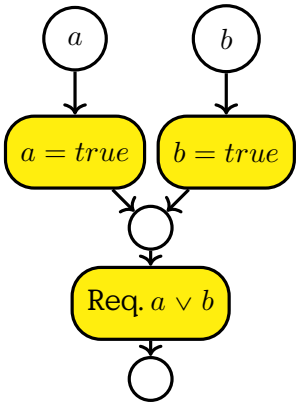


Abbildung 12.14: Disjunktion der Vorbedingungen eines Angriffsschrittes ist die normale Behandlung der Bedingungen

Evaluation of a given 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.

Ausführung

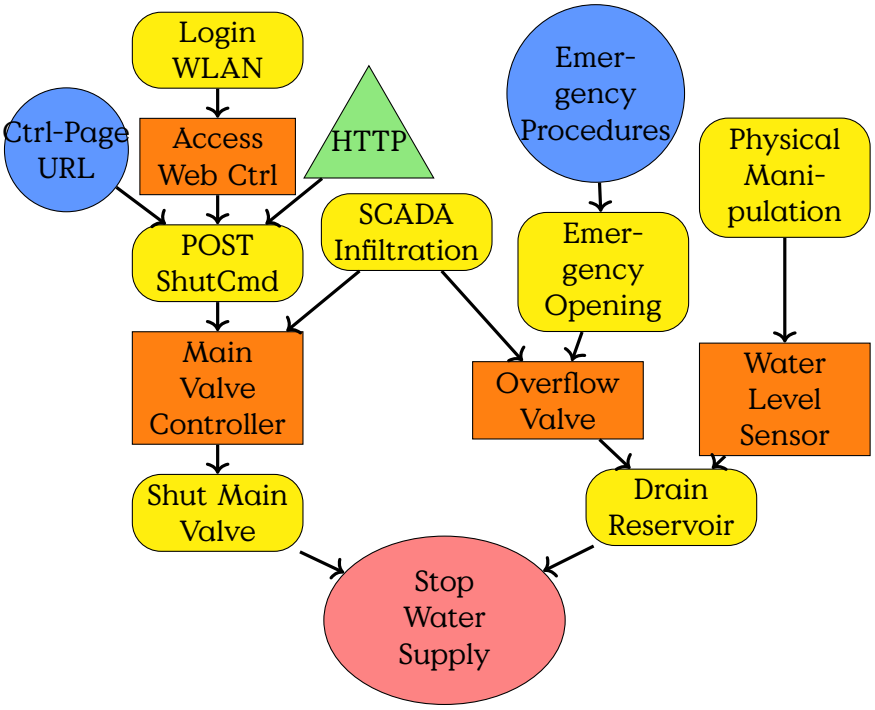
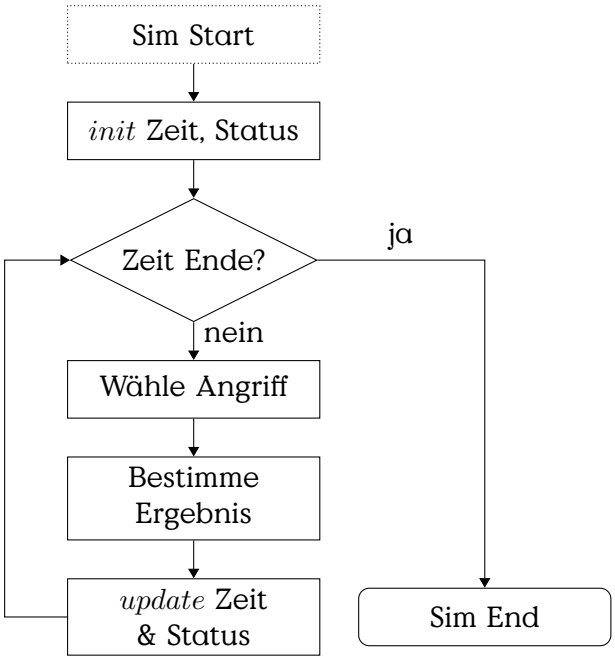


Abbildung 12.15: Modellierung der Angriffswege die zur Abschaltung der Wasserversorgung führen können als AEG

Abbildung 12.16: Informelle Beschreibug des Ablaufs der Auswertung



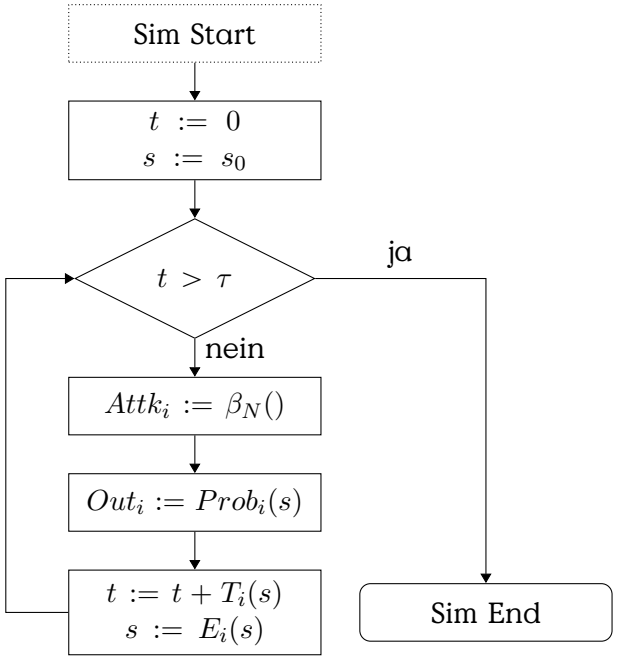


Abbildung 12.17: Schematische Darstellung des formalen Ablaufs

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.

Angreiferprofil

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.

$L: S \rightarrow [0,1]$ (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

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

Tabelle 12.2: Beispiele für Angreiferprofile [**LeMay2011aeg**]

oder Detectability des einzelnen Schrittes. Essentiell hat er einen Planungshorizont von $N = 1$.

Short-Sighted Adversary

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

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**, **casey2015extendedthreattaxonomy**].

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:

Graphgenerierung:

- Ermittlung der möglichen Angriffsschritte a_i in Zustand s

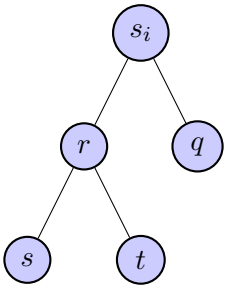


Abbildung 12.18: Zustandsbaum ausgehend vom aktuellen Zustand s_i

- 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.

Long-Range-Planning Adversary (cont.)

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

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.

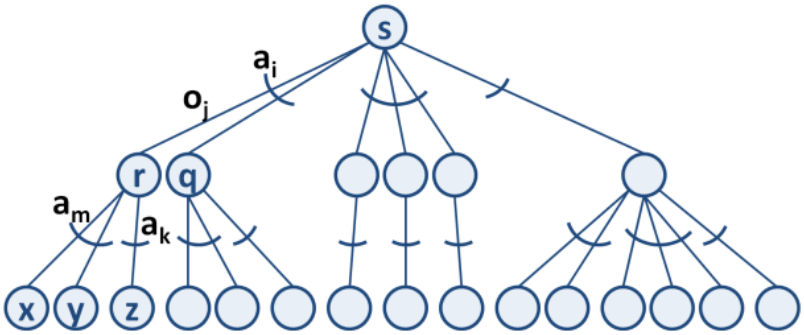


Abbildung 12.19: State Look-Ahead Tree ausgehend vom Zustand s

12.4.1 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. 12.20)

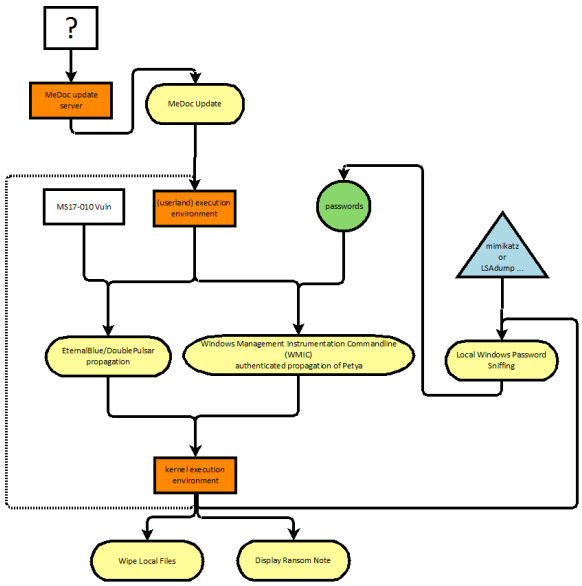


Abbildung 12.20: AEG describing the flow of the Petya Incidents 2017

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. **??**)

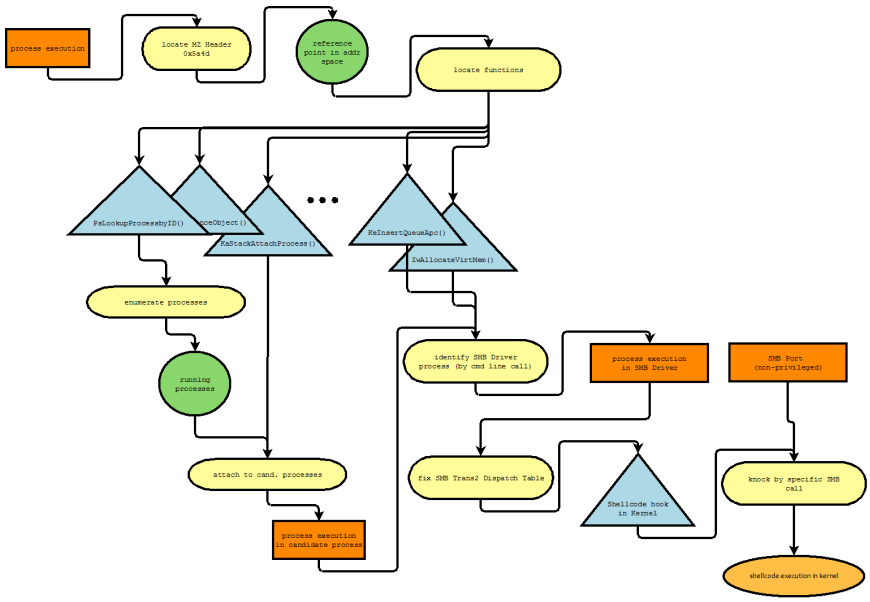


Abbildung 12.21: AEG for DoublePulsar Attack

12.5 Vulnerability Quantification

12.5.1 Vulnerability Levels

You can distinguish vulnerabilities by where they are in a simplified development model consisting of protocol, implementation and configuration. The mayor difference between these two layers, with respect to vulnerabilities, is the area of effect and the effort required to fix — in general.

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

of the protocol.

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

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

Vulnerability Levels

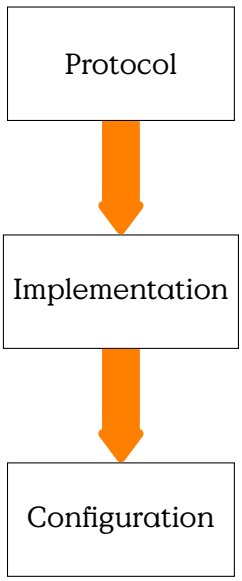


Figure 12.22: Hierarchy of devops parts from the perspective of vulnerabilities

12.5.2 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

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

Table 12.3: Vulnerabilities at different DevOps levels.

webpage[1]

Common Vulnerability Scoring System (CVSS)

Figure 12.23: 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
- Environmental Metric Group for quantification of environmental security requirements and and modified base metrics

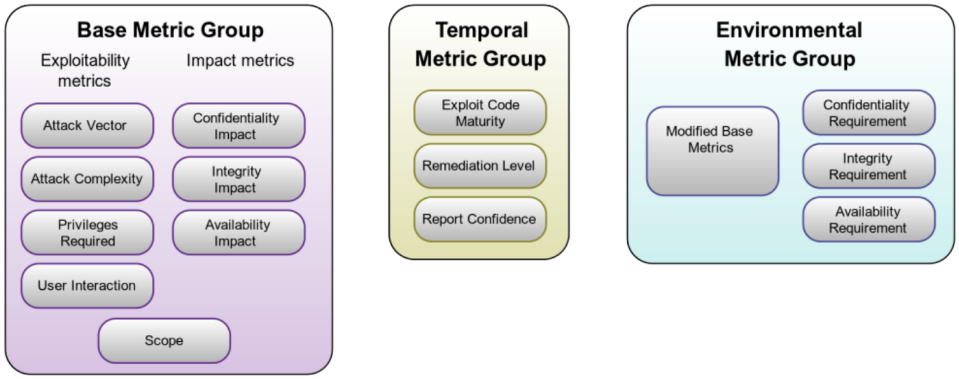
Common Vulnerability Scoring System

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

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 12.4: CVSS Severity Scores

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 Metric Evaluation

`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)

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

12.6 Attack Patterns

12.6.1 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.

CAPEC

https://capec.mitre.org/

12.6.2 MITRE ATT&CK (ATT&CK) Framework

The MITRE ATT&CK (ATT&CK) Framework provides a tool for structured expression of 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.

MITRE ATT&CK (ATT&CK)

12.6.3 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.

Threat Agent Types

Civil Activist	**Government Spy**
Radical Activist	**Internal Spy**
Anarchist	**Irrational Individual**
Competitor	**Legal Adversary**
Corrupt Government Official	**Mobster**
Cybervandal	**Sensationalist**
Data Miner	**Terrorist**
Disgruntled Employee	**Thief**
Government Cyberwarrior	**Vendor**

INTEL Threat Agent Library

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

12.7 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. **[Schol2004EntwicklungeinerMe**

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.

Preparation

- Laboratory

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

Pentest Process

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

Assignment Scope

Get a Permission Slip/Contract!

- Objectives
- Target Systems
- Time Frame

- Mode of Operation
- Team Requirements

Preparation

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

Reconnaissance

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

Gain and Extend Access

Work through CAPEC, OWASP,...

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

Report!

See Section 12.7.2.

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:

Pentest Tools

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

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

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

12.7.1 Guidewords

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 12.5: 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.

STRIDE:Common Attacks

12.7.2 Schwachstellenbewertung

DREAD

The DREAD risk assessment model has been developed by Microsoft. It provides a separation of different topics that, individually, could be as-

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

sessed to generate an overall threat level for individual threats.

DREAD — Risk Assessment Model

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

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

The severity of individual threats is evaluated by determining a risk level for each individual DREAD category (Table 12.6) and summarising them as provided by example in Table 12.7.

DREAD

The severity level of every threat depends on the

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

The threat evaluation is summarised in a Threat Description Table (Table 12.8) 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.

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]

Common Vulnerability Scoring System (CVSS)

- Metric for Severity
- Communication on Vulnerabilities
- Base Score

[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 12.6: 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 12.7: Example DREAD Threat Level Evaluation

Thread Description

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

Table 12.8: 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

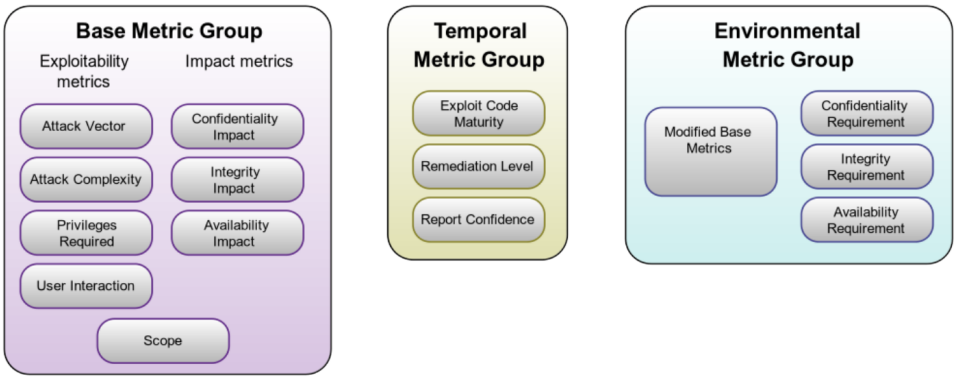
Table 12.9: CVSS Severity Scores

Figure 12.24: Example CVSS Value.

- Temporal Score allows to assess temporary variable factors, i. e., maturity of available exploit implementations
- Environmental Metric Group for quantification of environmental security requirements and and modified base metrics

Common Vulnerability Scoring System

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 Metric Evaluation

`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)

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

13

Security Law

Disclaimer

Dieser Abschnitt ist keine Rechtsberatung sondern bietet nur einen Überblick über ausgewählte Gesetze und Normen.

Sicherheitsgesetze

- NIS Directive [**NIS-Directive-2016**] Europäische Gesetznorm zur Einführung von nationaler Cyber-Sicherheits-Gesetzgebung
- NIS-2 Directive [**NIS-2-directive-2022**]
  - nationale Umsetzung bis 2024-10-18
  - Kritische Infrastrukturen: +Supply-Chains
  - Stärkung Aufsichts-/Ermittlungsbefugnisse
- Gesetz zur Erhöhung der Sicherheit informationstechnischer Systeme (IT-Sicherheitsgesetz – ITSiG, 2015, Artikelgesetz), beeinflusst (uA)
  - Telekommunikationsgesetz (TKG)
  - Telemediengesetz (TMG)
  - Gesetz über das Bundesamt für Sicherheit in der Informationstechnik (BSI-Gesetz – BSIG) [**BSIG2015**]
  - Gesetz über die Elektrizitäts- und Gasversorgung (Energiewirtschaftsgesetz – EnWG)
  - ...

- IT-Sicherheitskatalog gemäß § 11 Absatz 1a Energiewirtschaftsgesetz

- Verordnung zur Bestimmung Kritischer Infrastrukturen nach dem BSI-Gesetz (BSI-Kritisverordnung – BSI-KritisV) [**BSI-KritisV2016**, **BSI-KritisV2017**]

IT-Sicherheitsgesetz (2015)

- Artikelgesetz

- Telemedien/Telekommunikation nach "Stand der Technik" sichern

- Vorsatz oder fahrlässig:

  - Bußgeld bis 50k€ (KRITIS: 100k€)

  - ggF. persönliche Haftung

- BSiG § 8c: Sicherheitsanforderungen digitale Dienste

  - Maßnahmen: Sicherheit, Kontinuität, Erkennung,...

  - Meldepflicht von Vorfällen

Gesetz
zur Erhöhung der Sicherheit informationstechnischer Systeme
(IT-Sicherheitsgesetz)*

Vom 17. Juli 2015

Der Bundestag hat das folgende Gesetz beschlossen:

Artikel 1

Änderung des
BSI-Gesetzes

Das BSI-Gesetz vom 14. August 2009 (BGBl. I S. 2821), das zuletzt durch Artikel 3 Absatz 7 des Gesetzes vom 7. August 2013 (BGBl. I S. 3154) geändert worden ist, wird wie folgt geändert:

1. § 1 wird wie folgt gefasst:

„§ 1

Bundesamt für
Sicherheit in der Informationstechnik

Der Bund unterhält ein Bundesamt für Sicherheit in der Informationstechnik (Bundesamt) als Bundesoberbehörde. Das Bundesamt ist zuständig für die Informationssicherheit auf nationaler Ebene. Es untersteht dem Bundesministerium des Innern."

2. Dem § 2 wird folgender Absatz 10 angefügt:

„(10) Kritische Infrastrukturen im Sinne dieses Gesetzes sind Einrichtungen, Anlagen oder Teile davon, die

1. den Sektoren Energie, Informationstechnik und Telekommunikation, Transport und Verkehr, Gesundheit, Wasser, Ernährung sowie Finanz- und Versicherungswesen angehören und
2. von hoher Bedeutung für das Funktionieren des Gemeinwesens sind, weil durch ihren Ausfall oder ihre Beeinträchtigung erhebliche Versorgungsengpässe oder Gefährdungen für die öffentliche Sicherheit eintreten würden.

Die Kritischen Infrastrukturen im Sinne dieses Gesetzes werden durch die Rechtsverordnung nach § 10 Absatz 1 näher bestimmt."

3. § 3 wird wie folgt geändert:

a) Absatz 1 Satz 2 wird wie folgt geändert:

aa) In Nummer 2 werden die Wörter „zur Wahrung ihrer Sicherheitsinteressen erforderlich ist" durch die Wörter „erforderlich ist, sowie für Dritte, soweit dies zur Wahrung ihrer Sicherheitsinteressen erforderlich ist" ersetzt.

bb) In Nummer 15 werden die Wörter „kritischen Informationsinfrastrukturen" durch die Wörter „Sicherheit in der Informationstechnik Kritischer Infrastrukturen" und der Punkt am Ende durch ein Semikolon ersetzt.

cc) Die folgenden Nummern 16 und 17 werden angefügt:

„16. Aufgaben als zentrale Stelle im Bereich der Sicherheit in der Informationstechnik im Hinblick auf die Zusammenarbeit mit den zuständigen Stellen im Ausland, unbeschadet besonderer Zuständigkeiten anderer Stellen;

17. Aufgaben nach den §§ 8a und 8b als zentrale Stelle für die Sicherheit in der Informationstechnik Kritischer Infrastrukturen."

b) Folgender Absatz 3 wird angefügt:

„(3) Das Bundesamt kann Betreiber Kritischer Infrastrukturen auf deren Ersuchen bei der Sicherung ihrer Informationstechnik beraten und unterstützen oder auf qualifizierte Sicherheitsdienstleister verweisen."

4. Die Überschrift von § 4 wird wie folgt gefasst:

* Notifiziert gemäß der Richtlinie 98/34/EG des Europäischen Parlaments und des Rates vom 22. Juni 1998 über ein Informationsverfahren auf dem Gebiet der Normen und technischen Vorschriften und der Vorschriften für die Dienste der Informationsgesellschaft (ABl. L 204 vom 21.07.1998, S. 37), zuletzt geändert durch Artikel 26 Absatz 2 der Verordnung (EU) Nr. 1025/2012 des Europäischen Parlaments und des Rates vom 25. Oktober 2012 (ABl. L 316 vom 14.11.2012, S. 12).

Das Bundesgesetzblatt im Internet: www.bundesgesetzblatt.de | Ein Service des Bundesanzeiger Verlag www.bundesanzeiger-verlag.de

IT-Sicherheitsgesetz 2.0

- BSI Gesetz (BSiG)
  - Einführung ISBÖFI (Infrastruktur im besonderem öffentlichen Interesse)
    - e. g., Rüstungsindustrie
  - Anzeigepflicht beim Einsatz kritischer Komponenten.Nach §2 Abs. 13 solche Komponenten die wesentlich für die Erbringung der Dienste in kritischen Infrastrukturen sind. In einer strengen Auslegung können das sehr viele Komponenten sein. Das BMI hat Erlaubnisvorbehalt.
  - Geheimhaltung von Schwachstellen, wobei das BSI hier auf

Weisung des BMI handeln müsste. Zerstört nach Ansicht vieler Experten das Vertrauensverhältnis von White-Hat Sicherheitsforschern und BSI.

- ...

Gesetz über das Bundesamt für Sicherheit in der Informationstechnik (BSiG)

BSiG

Stand 2015

- §§ 4 und 8b: Zentrale Meldestelle (Bund, KRITIS)
- § 5: Gefahrenabwehr für Bund TK
- § 5a: Incidence Response (Bund, KRITIS) in herausgehobenen Fällen
- § 7a: Produktuntersuchung
- § 8c: Sicherheits und Meldepflicht
- § 9: Zertifizierung, Personen und Systeme
- § 14: Bußgelder 50k/100k€

13.1 German Law

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

Hackerparagraph

§ 202 StGB: Ausspähen

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

tet 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.

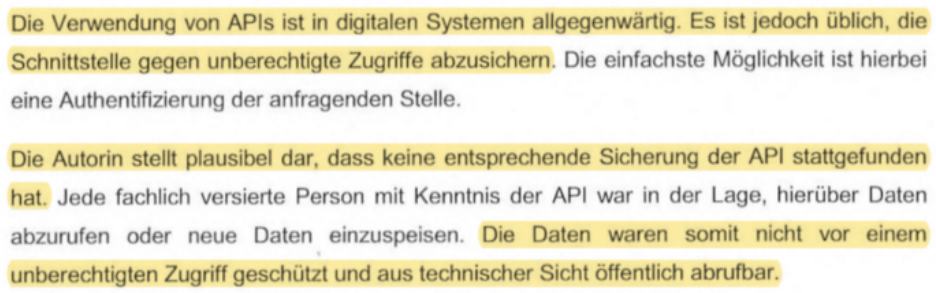
(4) In besonders schweren Fällen des Absatzes 2 ist die Strafe Freiheitsstrafe von sechs Monaten bis zu zehn Jahren. Ein besonders schwerer Fall liegt in der Regel vor, wenn der Täter

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.

2021 wurde Lilith Wittmann von der damals an der Regierung befindlichen Partei polizeilich angezeigt, weil sie eine Sicherheitslücke in deren Wahlkampf-App CDU-Connect gemeldet hatte. In ihrem Blog dokumentiert die Sicherheitsforscherin den Prozess bis zur Einstellung des Verfahrens. Die Ankläger waren auf die Sicherheitslücke aufmerksam geworden, weil die Sicherheitsforscherin selbst sie, entsprechend der Vorgehensweise Responsible Disclosure, vor der Veröffentlichung der Schwachstelle informiert hat.

Der Fall Wittmann vs. CDU-Connect



Die Verwendung von APIs ist in digitalen Systemen allgegenwärtig. Es ist jedoch üblich, die Schnittstelle gegen unberechtigte Zugriffe abzusichern. Die einfachste Möglichkeit ist hierbei eine Authentifizierung der anfragenden Stelle.

Die Autorin stellt plausibel dar, dass keine entsprechende Sicherung der API stattgefunden hat. Jede fachlich versierte Person mit Kenntnis der API war in der Lage, hierüber Daten abzurufen oder neue Daten einzuspeisen. Die Daten waren somit nicht vor einem unberechtigten Zugriff geschützt und aus technischer Sicht öffentlich abrufbar.

Figure 13.1: Aktenvermerk zur von Lilith Wittmann gemeldeten Sicherheitslücke in CDU-Connect. Aus der Verfahrensakte, Quelle: Wittmann

Die Ermittlungen der Polizei und Staatsanwaltschaft haben dann ergeben, dass es sich alleine deshalb nicht um eine Straftat nach §202 StGB handeln kann, weil die Daten technisch "nicht vor unberechtigtem Zugriff geschützt", sondern "öffentlich abrufbar" waren.

Im Nachgang wird eine Bewertung des Datenschutzes erwartet um Einzuschätzen inwieweit die öffentliche Bereitstellung persönlicher Daten durch die Anwendung CDU-Connect gegen die Datenschutzgrundverordnung verstoßen hat.

13.2 European Law

General Data Protection Regulation [**dsgvo2016**], Datenschutz-Grundverordnung (DSGVO)

DSGVO

Rechtmäßigkeit Jede Verarbeitung von Daten ist durch ein Gesetz erlaubt. Insbesondere nur, wenn mindestens eine der Bedingungen

aus Art. 6 DSGVO erfüllt ist.

Treu und Glauben Verfahren nach "Verkehrssitte", Zuverlässigkeit, Aufrichtigkeit und Rücksichtnahme im Handeln, im Sinne eines Vertrages handelnd

Transparenz Nachvollziehbarkeit und Offenheit

Zweckbindung

Datenminimierung

Richtigkeit

Speicherbegrenzung

Integrität und Vertraulichkeit

Rechenschaftspflicht

Datenschutz-Informationen

Nach Art. 5 (1) a) DSGVO soll Datenverabreitung grundsätzlich transparent, i. e., "in einer für die betroffene Person nachvollziehbare Weise" stattfinden. Das wird dann in Art. 12 genauer ausgeführt. Dort wird eine "einfache und klare Sprache" ebenso vorgeschrieben, wie eine "leicht zugängliche Form". Es ist also nicht ausreichend ein die Informationen in einem dunklen Keller, "in einem verschlossenen Aktenschrank, in einem unbenutztem Klo" mit einem Schild "Vorsicht, bissiger Leopard!"[1] auszulegen.

Die Informationen müssen leicht zu finden sein und eingesehen werden können bevor eine Verarbeitung von persönlichen Daten stattfindet.

Was gehört in solche Informationen hinein?

Wer? Verarbeitet die Daten? Wer ist verantwortlicher Ansprechpartner?

Daten Welche Daten werden konkret verarbeitet?

Zweckbindung Was ist der Zweck der Datenerhebung und -verarbeitung?

Speicherung Für welchen Zeitraum und wie werden die Daten gespeichert und verarbeitet.

Weitergabe
- Wohin und an wen werden ggF. Daten zur Weiterverarbeitung weitergegeben?
- Was ist der Zweck der Weiterverarbeitung?

[1] Frei nach Douglas Adams: Per Anhalter durch die Galaxis

Beschwerden? An welche Stellen können sich Betroffene wenden. Datenschutzbeauftragte von Organisation, Land und Bund.

13.3 International Law

Begriff: Aktive Cyberabwehr

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

3 Stufen:

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

Staatliche Cyberangriffe

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

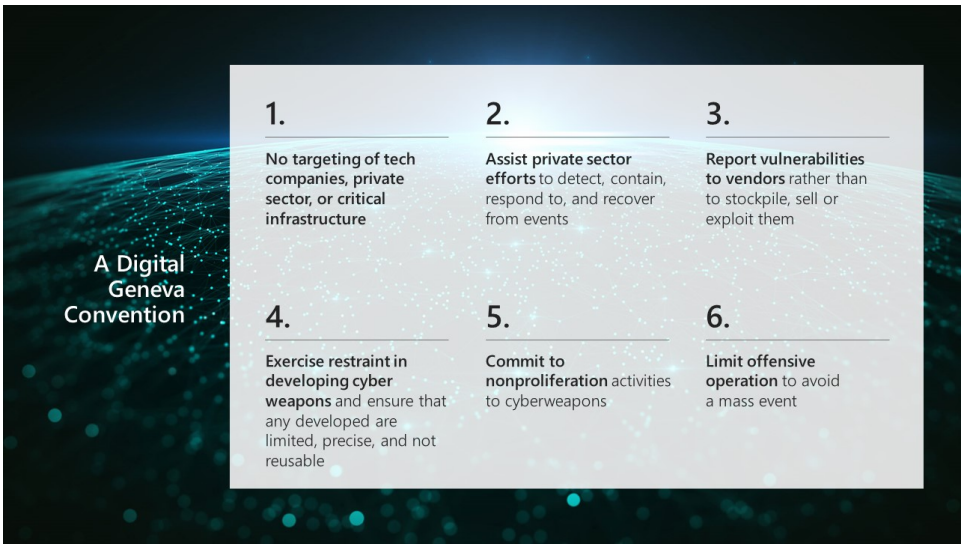
Hackback

- Target Location
  - IP-Address

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

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

Digital Geneva Convention



[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

13.4 Ethics

Der Chaos Computer Club (CCC) e.V. hat sich seit 1981 als Sprachrohr der "Komputerfrieks" und fachlich-moralische Instanz in der Computersicherheit in Deutschland etabliert. Vertreterinnen des Vereins treten als Expertinnen in Ausschüssen des Bundestages. Der jährlich veranstaltete Chaos Communication Congress wird umfassend von der Presse reflektiert und ist mittlerweile nicht nur unter Experten bekannt.

In Anlehnung an das Buch "Hackers" von Stephen Levy hat der CCC die Hackerethik formuliert, in der ein Grundverständnis des Umgangs mit (Informtions-)Technik in der Gesellschaft formuliert werden.

CCC Hackerethik

- 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]

Es lässt sich diskutieren, wie weit diese Ethik die Gesellschaft geprägt hat. In der neueren Zeit sind Gesetze wie das Informationsfreiheitsgesetz und die DSGVO entstanden.

14

Security Management

Abbildung 14.1 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.

Attacker - Owner — Assets

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

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

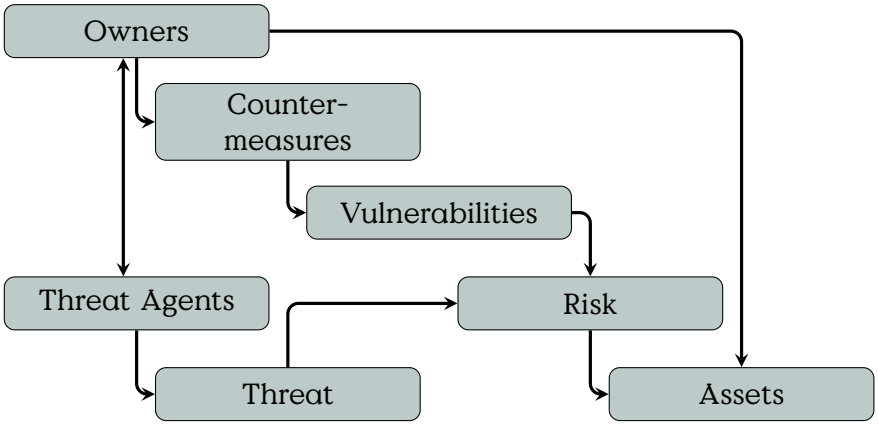


Figure 14.1: Security concepts and relationships [**stallings2008computer**][**CommonCriteria**]

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.

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.

Security Chain

14.1 Rechtlicher & staatlicher Rahmen

Sicherheitsgesetze

- NIS Directive [**NIS-Directive-2016**] Europäische Gesetznorm zur Einführung von nationaler Cyber-Sicherheits-Gesetzgebung
- NIS-2 Directive [**NIS-2-directive-2022**]
  - nationale Umsetzung bis 2024-10-18
  - Kritische Infrastrukturen: +Supply-Chains
  - Stärkung Aufsichts-/Ermittlungsbefugnisse
- Gesetz zur Erhöhung der Sicherheit informationstechnischer Systeme (IT-Sicherheitsgesetz – ITSiG, 2015, Artikelgesetz), beeinflusst (uA)

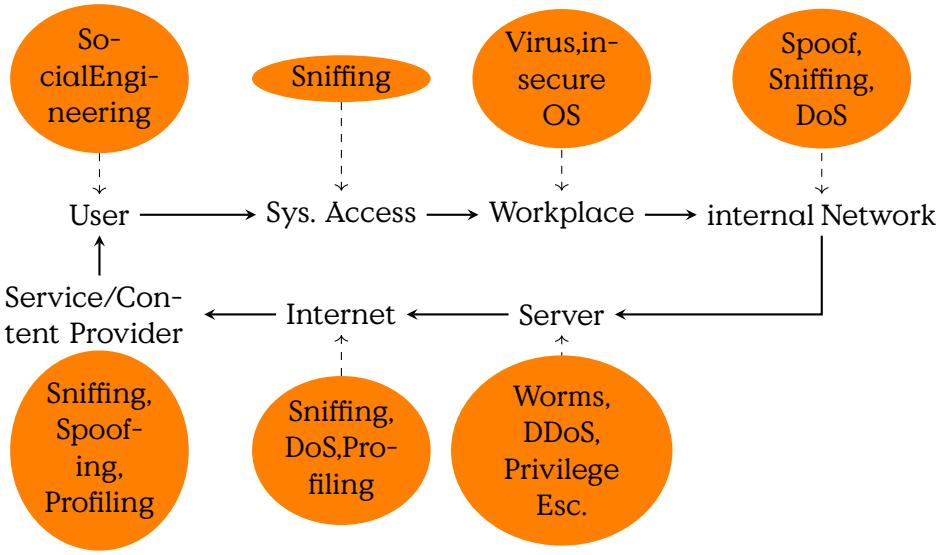


Figure 14.2: Chain of system security-relevant components. If the security of a single one is compromised, the whole system is jeopardised.[Eckert 2011, p.40]

  - Telekommunikationsgesetz (TKG)
  - Telemediengesetz (TMG)
  - Gesetz über das Bundesamt für Sicherheit in der Informationstechnik (BSI-Gesetz – BSIG) [**BSIG2015**]
  - Gesetz über die Elektrizitäts- und Gasversorgung (Energiewirtschaftsgesetz – EnWG)
  - ...
- IT-Sicherheitskatalog gemäß § 11 Absatz 1a Energiewirtschaftsgesetz
- Verordnung zur Bestimmung Kritischer Infrastrukturen nach dem BSI-Gesetz (BSI-Kritisverordnung – BSI-KritisV) [**BSI-KritisV2016**, **BSI-KritisV2017**]

IT-Sicherheitsgesetz (2015)

- Artikelgesetz
- Telemedien/Telekommunikation nach “Stand der Technik” sichern
- Vorsatz oder fahrlässig:

  - Bußgeld bis 50k€ (KRITIS: 100k€)
  - ggF. persönliche Haftung
- BSiG § 8c: Sicherheitsanforderungen digitale Dienste
  - Maßnahmen: Sicherheit, Kontinuität, Erkennung,...
  - Meldepflicht von Vorfällen

Gesetz
zur Erhöhung der Sicherheit informationstechnischer Systeme
(IT-Sicherheitsgesetz)*

Vom 17. Juli 2015

Der Bundestag hat das folgende Gesetz beschlossen:

Artikel 1

Änderung des
BSI-Gesetzes

Das BSI-Gesetz vom 14. August 2009 (BGBl. I S. 2821), das zuletzt durch Artikel 3 Absatz 7 des Gesetzes vom 7. August 2013 (BGBl. I S. 3154) geändert worden ist, wird wie folgt geändert:

1. § 1 wird wie folgt gefasst:

„§ 1

Bundesamt für
Sicherheit in der Informationstechnik

Der Bund unterhält ein Bundesamt für Sicherheit in der Informationstechnik (Bundesamt) als Bundesoberbehörde. Das Bundesamt ist zuständig für die Informationssicherheit auf nationaler Ebene. Es untersteht dem Bundesministerium des Innern."

2. Dem § 2 wird folgender Absatz 10 angefügt:

„(10) Kritische Infrastrukturen im Sinne dieses Gesetzes sind Einrichtungen, Anlagen oder Teile davon, die

1. den Sektoren Energie, Informationstechnik und Telekommunikation, Transport und Verkehr, Gesundheit, Wasser, Ernährung sowie Finanz- und Versicherungswesen angehören und
2. von hoher Bedeutung für das Funktionieren des Gemeinwesens sind, weil durch ihren Ausfall oder ihre Beeinträchtigung erhebliche Versorgungsengpässe oder Gefährdungen für die öffentliche Sicherheit eintreten würden.

Die Kritischen Infrastrukturen im Sinne dieses Gesetzes werden durch die Rechtsverordnung nach § 10 Absatz 1 näher bestimmt."

3. § 3 wird wie folgt geändert:
   a) Absatz 1 Satz 2 wird wie folgt geändert:
      aa) In Nummer 2 werden die Wörter „zur Wahrung ihrer Sicherheitsinteressen erforderlich ist" durch die Wörter „erforderlich ist, sowie für Dritte, soweit dies zur Wahrung ihrer Sicherheitsinteressen erforderlich ist" ersetzt.
      bb) In Nummer 15 werden die Wörter „kritischen Informationsinfrastrukturen" durch die Wörter „Sicherheit in der Informationstechnik Kritischer Infrastrukturen" und der Punkt am Ende durch ein Semikolon ersetzt.
      cc) Die folgenden Nummern 16 und 17 werden angefügt:
      „16. Aufgaben als zentrale Stelle im Bereich der Sicherheit in der Informationstechnik im Hinblick auf die Zusammenarbeit mit den zuständigen Stellen im Ausland, unbeschadet besonderer Zuständigkeiten anderer Stellen;
      17. Aufgaben nach den §§ 8a und 8b als zentrale Stelle für die Sicherheit in der Informationstechnik Kritischer Infrastrukturen."
   b) Folgender Absatz 3 wird angefügt:

„(3) Das Bundesamt kann Betreiber Kritischer Infrastrukturen auf deren Ersuchen bei der Sicherung ihrer Informationstechnik beraten und unterstützen oder auf qualifizierte Sicherheitsdienstleister verweisen."

4. Die Überschrift von § 4 wird wie folgt gefasst:

* Notifiziert gemäß der Richtlinie 98/34/EG des Europäischen Parlaments und des Rates vom 22. Juni 1998 über ein Informationsverfahren auf dem Gebiet der Normen und technischen Vorschriften und der Vorschriften für die Dienste der Informationsgesellschaft (ABl. L 204 vom 21.07.1998, S. 37), zuletzt geändert durch Artikel 26 Absatz 2 der Verordnung (EU) Nr. 1025/2012 des Europäischen Parlaments und des Rates vom 25. Oktober 2012 (ABl. L 316 vom 14.11.2012, S. 12).

Das Bundesgesetzblatt im Internet: www.bundesgesetzblatt.de | Ein Service des Bundesanzeiger Verlag www.bundesanzeiger-verlag.de

IT-Sicherheitsgesetz 2.0

- BSI Gesetz (BSiG)
  - Einführung ISBÖFI (Infrastruktur im besonderem öffentlichen Interesse)

    - e. g., Rüstungsindustrie

  - Anzeigepflicht beim Einsatz kritischer Komponenten.Nach §2 Abs. 13 solche Komponenten die wesentlich für die Erbringung der Dienste in kritischen Infrastrukturen sind. In einer strengen Auslegung können das sehr viele Komponenten sein. Das BMI hat Erlaubnisvorbehalt.

  - Geheimhaltung von Schwachstellen, wobei das BSI hier auf Weisung des BMI handeln müsste. Zerstört nach Ansicht vieler Experten das Vertrauensverhältnis von White-Hat Sicherheitsforschern und BSI.

- ...

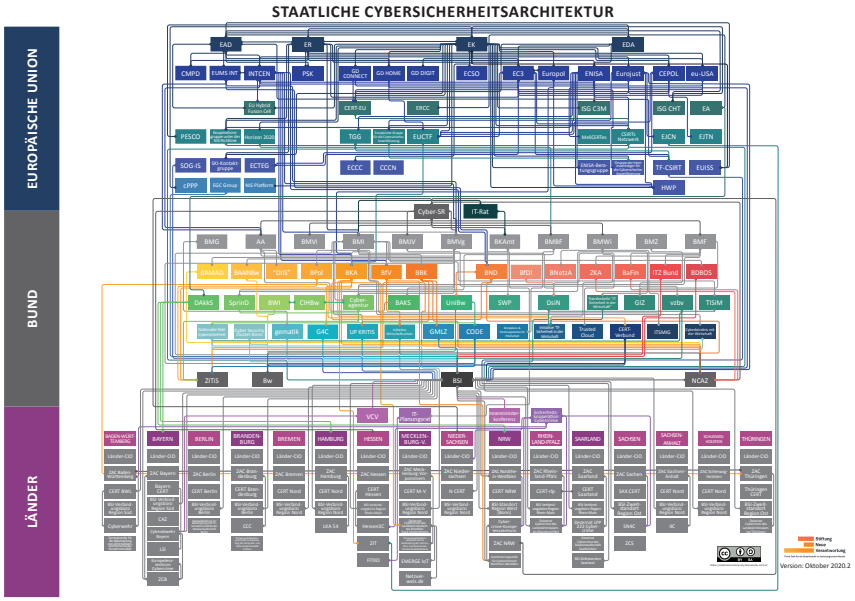
Gesetz über das Bundesamt für Sicherheit in der Informationstechnik (BSiG)

BSiG

Stand 2015

- §§ 4 und 8b: Zentrale Meldestelle (Bund, KRITIS)
- § 5: Gefahrenabwehr für Bund TK
- § 5a: Incidence Response (Bund, KRITIS) in herausgehobenen Fällen
- § 7a: Produktuntersuchung
- § 8c: Sicherheits und Meldepflicht
- § 9: Zertifizierung, Personen und Systeme
- § 14: Bußgelder 50k/100k€

14.1.1 Behördenstrukturen Cyberabwehr in Deutschland



- BSI
  - Nationales Cyber-Abwehrzentrum (Cyber-AZ)
  - Allianz für Cybersicherheit
  - Nationalen IT-Lagezentrum/IT-Krisenreaktionszentrum
- UP-KRITIS (BSI und BKK)
- Zentrale Stelle für Informationstechnik im Sicherheitsbereich (ZITIS)
- Agentur für Cybersicherheit `https://www.bundesregierung.de/breg-de/themen/digital-made-in-de/agentur-fuer-innovation-in-der-cybersicherheit-1`
- Bundeswehr
  - Kommando Cyber- und Informationsraum (CIR)
- Bundeskriminalamt/Landeskriminalämter
- Bundesnachrichtendienst
- Bundes-/Landesverfassungsschutz

14.2 Tasks and Duties of Managment

Tasks and Duties of Managment

- Assumption of overall responsibility for...
- Integrating...
- Managing and maintaining...
- Setting achievable goals
- Weighing up security costs against benefits
- Be role model

This list is defined in [**bsi08InformationSecurityManagment**]

> "An information security management system [...] specifies the instruments and methods that the administration/management level of an institution should use to comprehensibly manage the tasks and activities aimed at achieving information security."
> [**bsi08InformationSecurityManagment**]

Information Security Managment System

- **instruments and methods** of the administration/management level
- to **manage**
- tasks and activities
- to achieve **information security**

[bsi08InformationSecurityManagment]

Components of an ISMS

Risk Management Process ISO/IEC 31000

Threat Levels

Threat levels are determined with respect to the consequences of a security breach. Very high levels are used if failure has existential consequences for an organisation or large parts of society or economy. High levels are used if damages render key areas of an organisation non-functional,

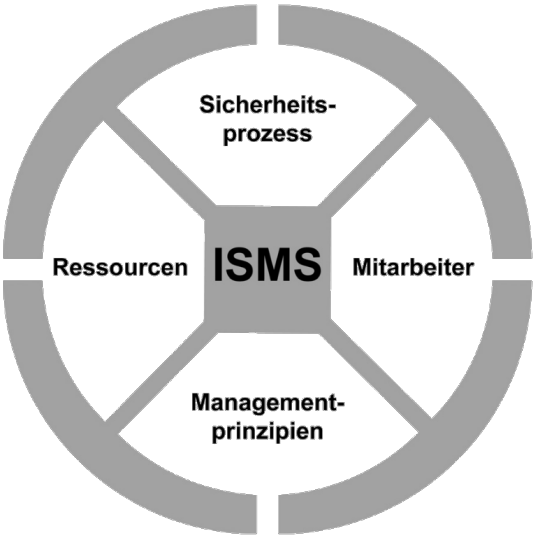


Figure 14.3: Components of an ISMS

VERY HIGH	Existential Consequences for Organisation or Large Parts of Society
HIGH	Non-Functional Large Areas, Significant Impairment, Affects Third Parties
NORMAL	Impairment of Processes
none/low	No significant consequences

Table 14.1: Threat Levels

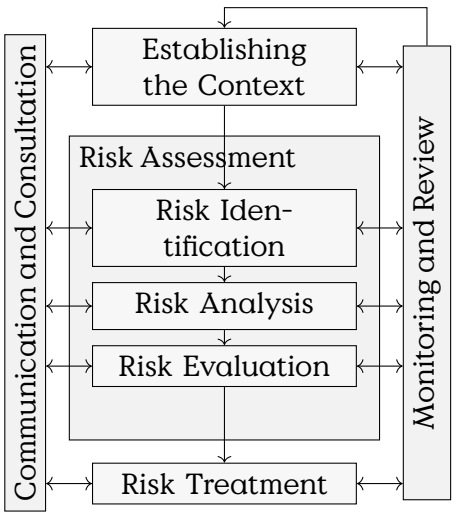


Figure 14.4: Risikomanagementprozess nach ISO/IEC 31000

the damage leads to significant impairment of the organisation or affects third parties. Normal levels are used if any level of impairment is to be expected on security fails. If no impairment is to be expected, no threat level is assigned, i. e., the component/objective is of no importance to security. (Table 14.1)

14.2.1 Sicherheitskataloge und -empfehlungen

Die folgende Auflistung soll einen ersten Überblick über die umfangreiche Landschaft von Schutzkatalogen und Bewertungsmethoden in kritischen Infrastrukturen, aus der Sicht der Stromversorgung geben. Die Kenntnis zumindest dieser Kataloge ist essentiell für die Arbeit in der IT-Sicherheit in Europa, da sie essentiell das Sprachgerüst für die Organisation von Sicherheitsmaßnahmen darstellen.

In der technischen Praxis sind diese Kataloge hilfreich bei der Identifizierung von Schutzlücken und Maßnahmen. Eine wesentliche Funktion ist aber letztlich, dass die Sicherheit einer Organisation einem strukturierten Vorgehen folgt. Diese Vorgehen wird oft unter dem Stichwort Information Security Management System (ISMS) geführt, was in Gänze aber in separaten Vorlesungen behandelt werden muss.

Sicherheitskataloge

Europa/International:

- ENISA

  - Railway Cybersecurity
  - Port Cybersecurity - Good practices for cybersecurity in the maritime sector
  - National Electric Sector Cybersecurity Organization Resource [**Marinos2013**]
- SGIS Toolbox [**M490**] Ausgehend von Use-Cases und einer SGAM-Modellierung werden Anforderungen, Schutzlücken und Maßnahmen aus SGIS-Katalogen abgebildet. Die SGI
- BDEW Whitepaper [**BDEW_White_2018**]
- Empfehlungen des National Cyber Security Center (NCSC)
- BSI-Standards [**bsi200-1**, **bsi200-2**, **bsi200-3**]
- UP KRITIS Best-Practice [**upkritis2017bestpractice**]
- BNetzA
  - IT-Sicherheitskatalog für Strom- und Gasnetze
  - IT-Sicherheitskatalog für Betreiber von Energieanlagen
- ISO/IEC 27k-Standards
  - 27001: ISMS-Specification
  - 27032, 27033, 27034: cyber-, network, application security
  - ISO/IEC TR 27019: Information security management guidelines based on ISO/IEC 27002 for process control systems specific to the energy industry
- IEC 62443 Industrial communication networks - Network and system security

USA:

- NIST Framework for Critical Infrastructures [**NIST2017Framework**] liefert grundlegende Terminologie und Klassifizierung von Maßnahmen, die — leider — nicht mit den Maßnahmenkatalogen in [**NISTIR7628**] übereinstimmt.
- NIST SP 800 Series
- NISTIR 7628 [**NISTIR7628**] Auswahl eines Maßnahmenkatalogs basierend auf modellierten Systemstruktur und Schutzklassen für unterschiedliche Schnittstellen. Anhand der Kritikalität werden Maßnahmen aus einem Maßnahmenkatalog ausgewählt. Bietet eine vollständige Methodik zum Grundschutz, sowie eine Anleitung zum vorgehen für besonders kritische Bereiche.

- NERC Critical Infrastructure Protection (CIP) Standards
- NESCOR Szenarienmethode und -katalogeNutzung der Szenarienmethodik zur Identifizierung von Schutzlücken durch Wargame-Sessions

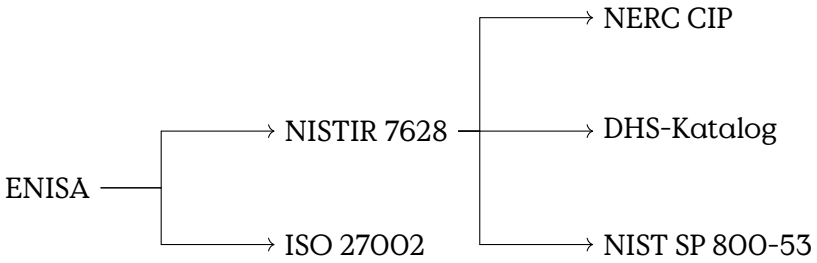


Figure 14.5: Hierarchische Abbildbarkeit von Maßnahmenkatalogen in den unterschiedlichen Richtlinien und Handbüchern. (Oder: wer könnte bei wem abgeschrieben haben?)

Und auch in der Rechtsprechung wird das Thema Sicherheit mittlerweile prominent verhandelt:

14.3 IT-Security Process

> Security is a process, not a product.
>
> Products provide some protection, but the only way to effectively do business in an insecure world is to put processes in place that recognize the inherent insecurity in the products. The trick is to reduce your risk of exposure regardless of the products or patches.
>
> Bruce Schneier:
> Computer Security: Will We Ever Learn? [seen 2020-08-26]

The Deming-Cycle (Figure 14.6) is a general concept for management. It establishes that any management process begins with a plan, that is then executed. After execution good management evaluates the results and decides on actions based on this evaluation. Not only IT-security management comprises a never ending cycle, but in IT-security at least, this has been made explicit by Bruce Schneier:

IT-Security Process

This is valuable to keep in mind if one intends to create secure products.

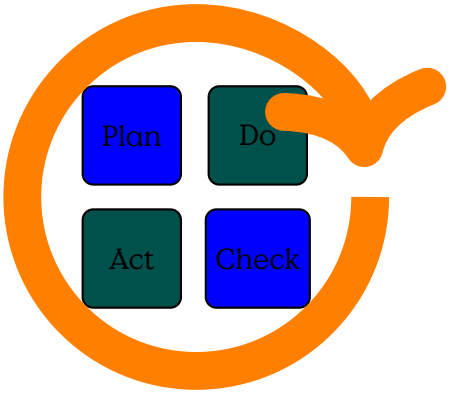
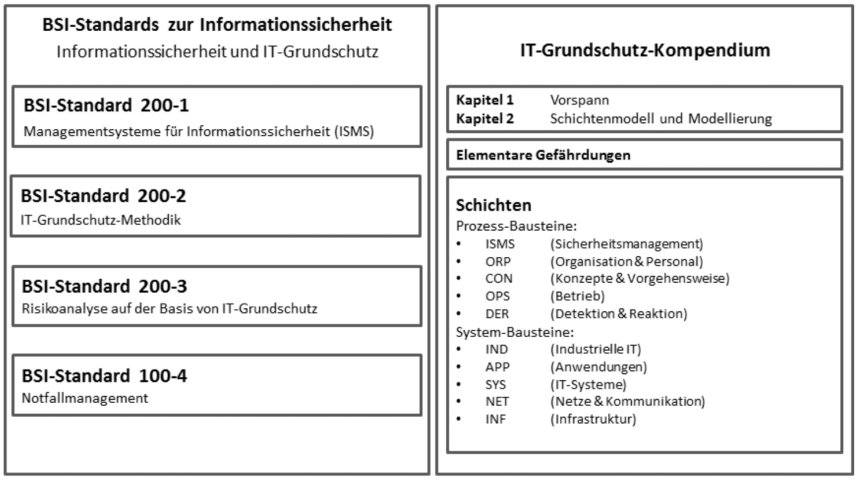


Figure 14.6: PDCA Managment Cycle

The “Bundesamt für Sicherheit in der Informationstechnik” (BSI) has defined a standard procedure for the establishment of an Informatios Security Managment System. The Standard 100 contains four separate works considering the topics managment, Grundschutz (baseline security), Risk Analysis, and Emergency Managment.

- 200-1: Managementsysteme für Informationssicherheit (ISMS)
  - Kompatibel zu ISO 27001
- 200-2: IT Grundschutz Methodik
  - Konkrete Handlungsanleitung für Basis-, Kern-, und Standard-Schutzlevel.
  - Baukastenprinzip für
  - Anleitung am Beispiel
- 200-3: Risikomanagement
- 100-4: Notfallmanagement

BSI-Security Standards



[BSI 200-1, Abb. 1, https://www.bsi.bund.de]

Based on BSI Standard 100-1 and 100-2 we summarise the IT-Security Process. Please refer to [**eckert2011sicherheit**] for an introduction in german language.

0. Initiation
1. Security Concept
2. Implementation
3. Maintenance and Improvement

Security Process

You have to look closely to spot the Deming cycle within the final loop of the security process in Figure 14.7 as the “Check” and “Act” Phases have been squashed into “Aufrechterhaltung und Verbesserung”.

Initiation

Initiation of the Security Process — Overiew

1. Accepting Responsibility (Mgmt)
2. Design and Plan of the Security Process
3. Creation of Security Policy
4. Organisation of Security Process
5. Providing Resources
6. Integration of all Employees

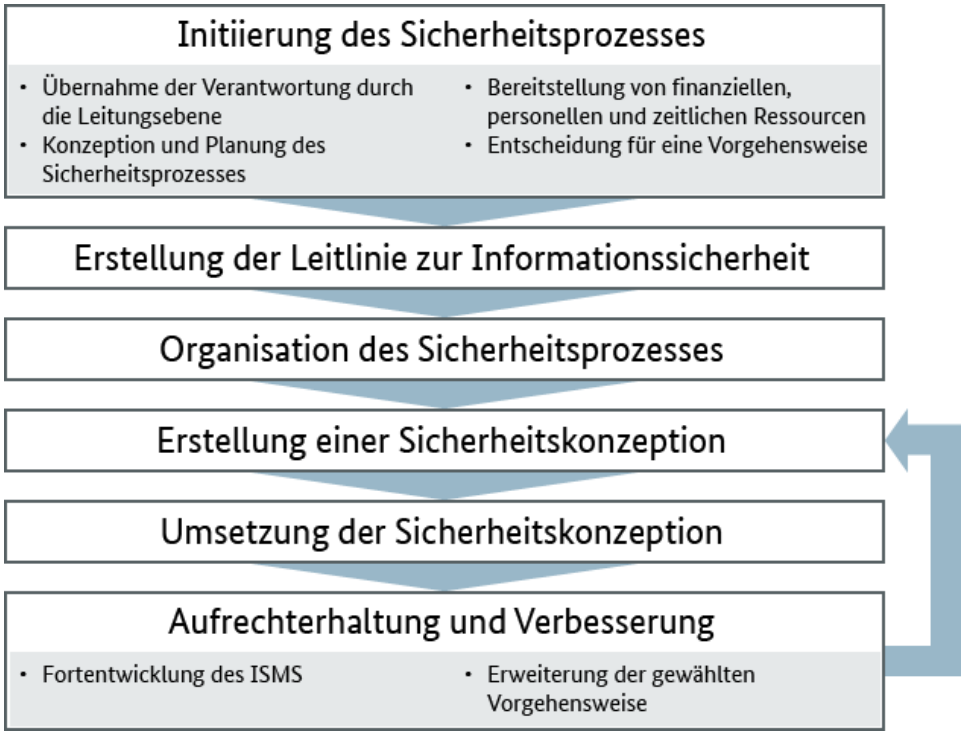


Figure 14.7: Phasen des Sicherheitsprozesses nach BSI 200, https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzSchulung/OnlinekursITGrundschutz2018/Lektion_2_Sicherheitsmanagement/Lektion_2_02/Lektion_2_02_node.html;jsessionid=6FCE91C1D11E131A548B603DB8C89776.1_cid360

Accepting Responsibility by the management:

Initiation

1. Managment is informed about risks and consequences
2. Managment assumes responsibility
3. Managment initiates process within organisation

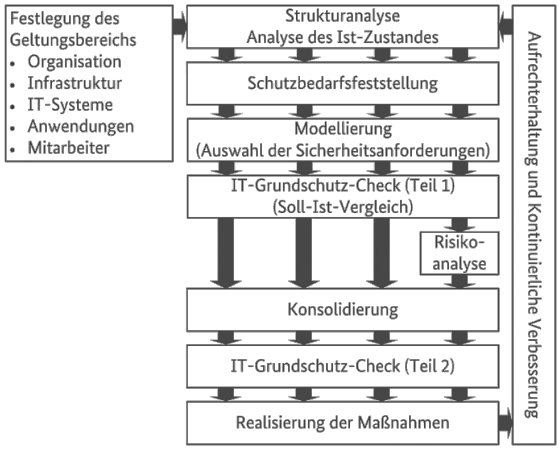
Design and plan of the security process:

Initiation

1. Appoint contact persons for all business processes and specialised tasks

2. Perform a rough assessment of the value of the information, business processes, and specialised tasks
3. Determine the general requirements
4. Estimate the importance of the business processes, specialised tasks, and information
5. Specify the general information security objectives
6. Obtain the agreement of management

BSI 200-2 Standard Absicherung



14.3.1 Security Policy

A security policy must describe in general terms how security is achieved in an organisation. It must describe which assets are to be protected and what kind of resources are utilised in which way for protection. The must describe the objectives and the desired level of protection. (Figure 14.8)

Initiation

1. Obtain a request from management
2. Specify the scope
3. Summon a development group
4. Organise management approval
5. Release the security policy
6. Check the security policy regularly

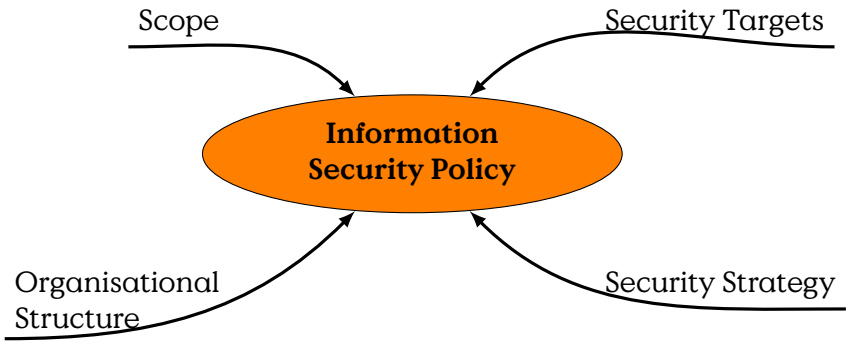


Figure 14.8: Contents of a Security Policy

Security Policy

14.3.2 Security Strategy

A security strategy has to provide answers how a system is secured on every layer. The Security Strategy Stack (Figure 14.9) shows the different layers which have to be considered. From top to bottom the layers provide increasingly detailed realisations of the layer above.

Security Policy: Guidelines and Directives, the top level security requirements definition.

Security Model/Concept: Representation of the IT-Security Infrastructure, Connections between Services, Dependencies, etc.

Security Architecture: structural realisation describing how the model is realised in the real world.

Security Methods: which techniques and services shall be implemented to provide the security requirements as posed by the architecture?

Security Strategy

Security Concept

14.4 Sicherheitsgrundfunktionen

[Dieser Abschnitt ist aus [**eckert2011sicherheit**] entnommen und wird, um Übersetzungsfehler zu vermeiden, in deutscher Sprache präsentiert.]

Die Sicherheitsgrundfunktionen bieten einen Baukasten zur Sicherstellung von grundlegenden Sicherheitseigenschaften und finden im Grund-

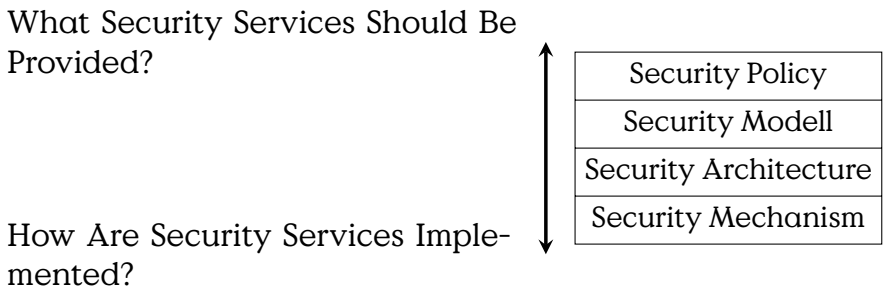


Figure 14.9: Security Strategy Stack

schutz insbesondere dann Anwendung, wenn eine niedrige Schadenshöhe zu erwarten ist. Eine vollständige Bedrohungs- und Risikoanalyse stellt in solchen Fällen eine unnötig hohen Aufwand dar. Grundsätzlich sollte jede Sicherheitsstrategie zumindest Konzepte zur Realisierung der Grundfunktionen enthalten.

Die Sicherheitsgrundfunktionen sind:

Sicherheitsgrundfunktionen

- Identifikation und Authentifikation
- Rechteverwaltung
- Rechteprüfung
- Beweissicherung
- Wiederaufbereitung
- Gewährleistung der Funktionalität

[eckert2011sicherheit]

14.4.1 Identifikation und Authentifikation

Identifikation und Authentifikation

- Abwehr von Maskierungsangriffen (Spoofing)
- Wann und Wer wird authentifiziert?
  - Welche Aktionen erfordern (welche Form der) Identität
  - z.B.: Systemzugang, DB-Zugriff,...
- Fehlerbehandlung

14.4.2 Rechteverwaltung

Rechteverwaltung

- Welche Subjekte auf
- Welche Objekte unter
- Welchen Bedingungen, mit
- Welchen Rechten?
- Rechtegranularität

14.4.3 Rechteprüfung

Rechteprüfung

- Policy Enforcement Points
- Bsp.: `open` file vs. `read` file
- Ausnahmebehandlung bei unauthorisierten Zugriffsversuchen

14.4.4 Beweissicherung

Beweissicherung

- Unabstreitbarkeit/Non-Repudiation herstellen
- Welche Ereignisse Wie protokollieren
- Zugriff auf Protokolle?
- Fälschbarkeit der Protokolle?

14.4.5 Wiederaufbereitung

Wiederaufbereitung

- z.B. Hauptspeicher, Festplatten, USB-Sticks
- Informationen sicher entfernen

14.4.6 Gewährleistung der Funktion

Unterthema Backup:

Gewährleistung der Funktion

> "Nobody is interested in Backups, what everybody wants is Restore." Michi Nagorsnik

> "Only wimps use tape backup: real men just upload their important stuff on ftp, and let the rest of the world mirror it ;)" Torvalds, Linus (1996-07-20)

Kein Backup — Kein Mitleid!