Klassische Sicherheitsprinzipien

Dozent: Prof. Dr. Michael Eichberg
Kontakt: michael.eichberg@dhbw.de
Version: 1.2.1

Folien: [HTML] https://delors.github.io/sec-klassische-sicherheitsprinzipien/folien.de.rst.html
[PDF] https://delors.github.io/sec-klassische-sicherheitsprinzipien/folien.de.rst.html.pdf
Kontrollfragen: https://delors.github.io/sec-klassische-sicherheitsprinzipien/kontrollfragen.de.rst.html
Fehler melden: https://github.com/Delors/delors.github.io/issues

Klassische Sicherheitsprinzipien

(Jerome Saltzer and Michael Schroeder, 1975)

Principle of Economy of Mechanism (aka Principle of Simplicity):
Die Sicherheitsmechanismen sollten so einfach wie möglich sein.

Principle of Economy of Mechanism (aka Principle of Simplicity):
dies fördert zum Beispiel die Korrektheit der Implementierung/Anwendung, da diese schneller verstanden wird und auch einfacher getestet werden kann. Weiterhin reduziert es die Angriffsfläche.

Principle of Fail-Safe Defaults:
Standardmäßig sollte der Zugriff auf Ressourcen verweigert werden.

Principle of Complete Mediation:
Zugriffsanfragen eines Subjekts auf ein Objekt werden jedes Mal vollständig auf ihre Zulässigkeit hin überprüft.

Principle of Complete Mediation:
Bei der Entwicklung einer Serveranwendung besagt das *Principle of Complete Mediation* somit, dass bei jeder Anfrage zu überprüfen ist, ober der Nutzer die entsprechenden Rechte besitzt.

Principle of Least Authority (aka POLA)/ Principle of Least Privilege:
Jedes Programm und jeder Benutzer sollte nur die für seine Aufgabe unbedingt notwendigen Rechte besitzen.

Principle of Separation of Privilege:
Ein System sollte in mehrere POLA konforme Komponenten unterteilt sein. Sollte eine Komponente kompromittiert sein, dann sind die Möglichkeiten des Angreifers dennoch begrenzt.

(Eng verwandt mit dem POLA.)

Das *Principle of Separation of Privilege* wird dann eingehalten, wenn ein Angreifer, der eine Komponente kompromittiert hat, nicht die Rechte erhält, die notwendig sind, um das gesamte System zu kompromittieren. Zum Beispiel ist es für Überweisungen notwendig diese auf zwei verschiedene Arten zu autorisieren.

Achtung!

Privilege Separation (für Programme) sollte nicht mit dem hier beschriebenen Prinzip verwechselt werden. *Privilege Separation* liegt zum Beispiel dann vor, wenn ein Programm in zwei Teile aufgeteilt ist und ein Teil - zum Beispiel zum Zugriff auf Betriebssystemressourcen wie Sockets oder bestimmte Dateien - erhöhte Rechte benötigt als der Rest vom Programm. In diesem Fall erfolgt dann der Austausch zwischen den beiden Teilen über eine wohldefinierte, minimale Schnittstelle, die die Rechte des ersten Teils auf das notwendige Minimum beschränkt.

Principle of Least Common Mechanism:
Die Sicherheitsmechanismen sollten über Nutzer (hier insbesondere Programme, die andere Programme nutzen) hinweg möglichst wenig Gemeinsamkeiten haben.

Beispiel - Principle of Least Common Mechanism (~ 🏴 *Grundsatz des kleinsten gemeinsamen Mechanismus*`)

Das Prinzip besagt zum Beispiel, dass die Mechanismen, die von mehreren Benutzern verwendet werden oder von dem mehrere Nutzer abhängen, minimiert werden sollten.

Das Prinzip kann/sollte auf ganz verschiedenen Ebenen angewendet werden:

- Z. B. sollten keine gemeinsamen Speicherbereiche verwendet werden in denen möglicherweise sicherheitsrelevantes Material vorgehalten wird. Es ist deswegen z. B. sinnvoll - wenn möglich - auf Implementierungen im Kernel zu verzichten und statt dessen auf User-Space-Implementierungen zu setzen.

TCP Connection Hijacking Angriffe werden bzw. wurden z. B. durch die Implementierung des TCP Stacks im Kernel ermöglicht ($\Leftrightarrow$ „Principle of Least Common Mechanism“).

- Z. B. sollten keine geteilten Passworte verwendet werden, um sich gegenüber einem System zu authentifizieren. (Dies bezieht sich sowohl auf die Passwörter einer Person als auch auf Passwörter über Personen und Systemgrenzen hinweg!)

Principle of Open Design (vgl. Kerckhoffs Prinzip):
Die Sicherheit des Systems sollte nicht von der Geheimhaltung der Sicherheitsmechanismen abhängen (sondern nur vom Schlüssel).

Principle of Psychological Acceptability:
Die Sicherheitsmechanismen sollten einfach zu verstehen und zu benutzen sein.

Principle of Isolation:
Die Sicherheitsmechanismen sollten so entworfen sein, dass Fehler in einem Teil des Systems nicht die Sicherheit des gesamten Systems gefährden; d. h. die einzelnen Komponenten sollten möglichst unabhängig voneinander sein und nur über wohldefinierte Schnittstellen miteinander kommunizieren und entsprechende Sicherheitsüberprüfungen durchführen.

Beispiel - Principle of Isolation:

1. Virtuelle Maschinen können genutzt werden, um Anwendungen in einer isolierten Umgebung auszuführen. Ein Angreifer, der eine Anwendung kompromittiert hat, kann somit nicht auf andere Anwendungen oder das Betriebssystem zugreifen.
2. Typischerweise kommuniziert zum Beispiel ein Basebandchip (WIFI, LTE, 5G, ...) mit dem Betriebssystem über eine minimale Schnittstelle über die nur Nachrichten übermittelt werden können, die leicht auf ihre Korrektheit überprüft werden können. Insbesondere erfolgt kein direkter Zugriff auf den Speicher des Betriebssystems.

Einen Angreifer ist es somit ggf. möglich den Basebandchip anzugreifen und ggf. zu kompromittieren, aber er kann nicht direkt auf das Betriebssystem zugreifen und Nachrichten, die bereits auf Betriebssystem oder Anwendungsebene verschlüsselt werden, sind weiterhin sicher.

Principle of Modularity:
Die Sicherheitsmechanismen sollten so entworfen sein, dass sie unabhängig voneinander implementiert und geprüft werden können.

Principle of Layering:
Die Sicherheitsmechanismen sollten in Schichten organisiert sein.

Beispiel für ein Schutzsystem für Netzwerke, dass mehrere Schichten verwendet:

- einfache (und effiziente) Paketfilter auf unterster Ebene
- zustandsbehaftete Paketfilter auf der nächsten bzw. der Anwendungsebene

Principle of Least Astonishment:
Die Sicherheitsmechanismen sollten so entworfen sein, dass sie keine Überraschungen für die Benutzer bereithalten.

Übung

0.1. Principle of Open Design

Benennen Sie ein historisches Verschlüsselungsverfahren, das gegen das *Principle of Open Design* verstoßen hat.

0.2. Verletzung

Stellen Sie sich vor, dass Sie als Pin (z. B. für ein Tablet) folgende Zahl verwenden wollen, diese aber abgelehnt wird (Leerzeichen dienen nur der Lesbarkeit):

```
3671 1197 4769
```

Während als Pin das folgende Passwort akzeptiert wird:

```
1364 7964 1364
```

Wie bewerten Sie dies?

Hinweis

Schauen Sie sich ggf. ein Pinpad an.

Übung

0.3. Browser

Der Chrome-Browser (zum Beispiel) unterstützt die so genannten Isolierung von besuchten Webseiten. Bei dieser werden Seiten von verschiedenen Websites in unterschiedliche Prozesse aufgeteilt.

Welches Prinzip bzw. welche Prinzipien wird/werden hier umgesetzt?

0.4. Quantum Algorithmen für die Verschlüsselung

Zukünftige Verschlüsselungsalgorithmen, z. B. solche die auch im Zeitalter der Quantencomputer noch sicher sein sollen, werden häufig im Rahmen von offenen Wettbewerben entwickelt bzw. ausgesucht. Wie bewerten Sie dieses Vorgehen?

4

Übung

0.5. Rechte von im Hintergrund laufenden Prozessen auf Servern

Es ist üblich, dass für Prozesse, die auf Servern im Hintergrund laufen, extra Nutzerkonten eingerichtet werden.

- Warum ist dies so?
- Welche Rechte sollten diese „Nutzer“ bekommen?
- Was sollte weiterhin beachtet werden?