Fachhochschule
Dortmund
University of Applied Sciences and Arts

Informationssicherheit – SoSe 2023
Authentifikation: Grundlagen, Passwörter, Argon2

Prof. Dr. Holger Schmidt
holger.schmidt004[at]fh-dortmund.de

Fachhochschule Dortmund
Fachbereich Informatik
Professur für IT-Sicherheit, Informatik

Themen & Lernziele

Themen

- Authentifikationsmethoden
- *n*-Faktor Authentifikation
- Entropie
- Linux Passwortdatenbanken
- Angriffsarten für Passwörter
- Argon2

Lernziele

Die Studierenden sind in der Lage,

- Grundlagen der Authentifikation, insb. passwortbasierte Authentifikation und zugehörige Angriffsarten zu differenzieren und zu erklären.
- Authentifikationsmethoden hinsichtlich Entropie zu analysieren.
- Funktionalität von Passwortdatenbanken (Salt, Argon2) zu implementieren.

Authentizität

- Im täglichen Leben: Sie authentisieren einen Freund, wenn Sie sich treffen oder miteinander telefonieren.
- Weitere Beispiele: Geheimzahl am Geldautomaten, Grenzkontrolle mit Ausweis
- Überprüfung der **Echtheit (Authentizität)**
- **Einseitige** und **gegenseitige** Authentifikation
- *n*-**Faktor** Authentifikation

Terminologie

- **Authentifikation (Authentifizierung)** beschreibt die Überprüfung, ob jemand derjenige ist, der er vorgibt zu sein.
- **Authentisierung (Authentikation)** beschreibt den gleichen Vorgang aus der Sicht des Überprüften.
- **Authorisierung** hat eine andere Bedeutung (siehe Zugriffs- und Informationsflußkontrolle).
- **Identifikation (Identifizierung)** beschreibt die Angabe eines Namens für ein Individuum.
- Typischerweise passiert zunächst eine Identifikation und dann eine Authentifikation.

Authentifikationsmethoden

- **Wissen** (etwas, was man weiß): Passwörter, Geheimzahlen, persönliche Informationen
- **Besitz** (etwas, was man hat): Ausweis, RFID, Smartcard
- **Eigenschaft** (etwas, was man ist): Unterschrift, Fingerabdruck

Passwörter

Passwörter

- Authentifikation durch Wissen
- Große **praktische Bedeutung**: Betriebssysteme, Web, WLAN-Protokolle WPA2 und WPA3 (Wi-Fi Protected Access), verschlüsselte Festspeicher, Archive, PGP & S/MIME, Schlüsselbund
- Erhebung zu Passwörtern[1]:
  - Basierend auf über 13 Milliarden öffentlich gewordenen Passwörtern
  - Beliebtestes Passwort ist „123456“mit einem Anteil von ca. 8%
  - Die TOP 10 Passwörter machen einen Anteil von knapp 22% aus.

[1] https://sec.hpi.de/leak-checker/statistics, aufgerufen am 28. Juni 2023

Passwortsicherheit

Wie beurteilen Sie die Passwörter

F9GndpVkfB44VdvwfUgTxGH7A8t

und

rE67AjbDCUotaju9H49sMFgYszA

?

Passwörter und Kerckhoffs Prinzip

- Kommt darauf an: Wenn das **zugrunde liegende System** folgendes ist:
  - Wirf eine Münze.
  - „Kopf“ wird zu F9GndpVkfB44VdvwfUgTxGH7A8t
  - „Zahl“ wird zu rE67AjbDCUotaju9H49sMFgYszA
- …dann sind die Passwörter extrem schwach, denn das System erzeugt **nur zwei** mögliche Ausgaben.

Passwörter und Entropie

- Wir können die Anzahl der möglichen Resultate, die ein System ausgeben kann, in **Entropie-Bits** messen.
- Zwei verschiedene mögliche Ausgaben lassen sich durch 1 Bit darstellen, d. h. das zuvor gezeigte System hat nur 1 Bit Entropie.
- Wenn wir die Münze durch einen Würfel ersetzen, dann gibt es sechs verschiedene mögliche Ausgaben und das System hat ungefähr 3 Bits Entropie.
- Jedes Bit Entropie repräsentiert eine **Verdopplung** der Anzahl der Möglichkeiten.
  - 10 Bits Entropie repräsentieren 1024 Möglichkeiten.
  - 11 Bits Entropie repräsentieren 2048 Möglichkeiten.
  - 128 Bits Entropie repräsentieren 340282366920938463463374607431768211456 Möglichkeiten.

Entropie

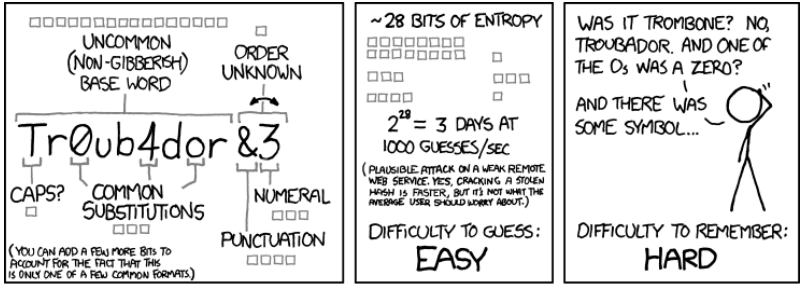
Entropie

$H = L\ log_2 N = L\ \frac{log\ N}{log\ 2}$, wobei
L Länge des Passworts und N Anzahl möglicher Zeichen

Beispiel:

Passwort mit 10 Ziffern, 26 Buchstaben (groß und klein geschrieben) resultiert in $N = 62$ möglichen Zeichen. Bei einer Passwortlänge von $L = 4$ und **vollständig zufälligen Zeichen** ergibt sich $H = 23,8$ Bits.

Faktor Mensch – xkcd Webcomic – Teil 1



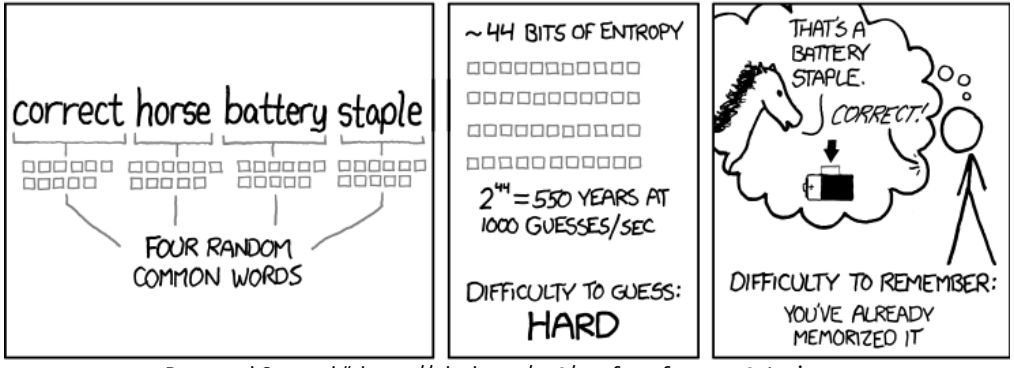
„Password Strength“, https://xkcd.com/936/, aufgerufen am 28. Juni 2023, lizensiert unter CC BY-NC 2.5.o[2]

- Basiswort aus einer Liste von 2^{16} Wörtern (16 Bits Entropie)
- Groß- und Kleinschreibung an Wortanfang (1 Bit Entropie)
- Ziffer aus 0 . . . 9 anhängen (4 Bits Entropie), eines aus 10 Satzzeichen (4 Bits Entropie), 3 Substitutionen/Schreibfehler (3 Bits Entropie)

Von **Menschen generierte Passwörter** sind nicht zufällig.

[2] https://creativecommons.org/licenses/by-nc/2.5/

Faktor Mensch – xkcd Webcomic – Teil 2



„Password Strength“, https://xkcd.com/936/, aufgerufen am 28. Juni 2023, lizensiert unter CC BY-NC 2.5.0[3]

- Basiswort aus einer Liste von 2^{11} ($= 2048$) Wörtern (11 Bits Entropie)

[3] https://creativecommons.org/licenses/by-nc/2.5/

Diceware

Diceware (https://theworld.com/~reinhold/diceware.html, aufgerufen am 28. Juni 2023)

- Basiert auf Listen mit Wörtern, wobei jedem Wort 5 Ziffern aus $\{1, 2, 3, 4, 5, 6\}$ vorangestellt sind
- Die 5 Ziffern dienen als Selektor für 5 Würfelwürfe.
- Jedes Wort erreicht ca. 13 Bits Entropie.
- 4 Diceware Wörter bieten ca. 52 Bits Entropie, d. h. ungefähr so viel wie ein 8 stelliges Passwort aus zufälligen ASCII-Zeichen (berechenbar in weniger als einem Tag).
- 7 Diceware Wörter (oder mehr) aktuell praktisch nicht berechenbar

Weitere Infos zu Passwort- und Schlüssellängen:
https://www.keylength.com/, aufgerufen am 28. Juni 2023

Konstruktion von Passwörtern nach NIST Standards

- NIST zog NIST 800-63, 2004 nach ca. 14 Jahren zurück (leider mitterweile nahezu überall implementiert)
- „Memorized secret verifiers“ gem. **NIST 800-63B, 2017, Sec. 5.1.1.2**
- Minimal 8 Zeichen, maximal 64 Zeichen
- Blacklisting (Wörterbücher, bekannte Passwörter, etc.)
- Keine periodischen Passwörtänderungen
- Keine Anforderungen an Passwortkomplexität (Buchstaben, Zahlen, Sonderzeichen, etc.)

Verwaltung und Nutzung von Passwörtern

Passwörter in Linux – Passwd

- Linux **Benutzerverwaltung** basiert auf Dateien
- `/etc/passwd` ist die zentrale **Benutzerdatenbank**

Ein Eintrag `/etc/passwd` besteht aus folgenden durch : getrennte Komponenten:

Benutzername Benutzername in druckbaren Zeichen

Passwort Hash des Passworts oder *x* für einen Verweis auf `/etc/shadow`

UID Benutzer-ID des Benutzers

GID Gruppen-ID des Benutzers

Info Persönliche Angaben zum Benutzer

Heimatverzeichnis Privates Heimatverzeichnis des Benutzers

Shell Die Shell, die nach der Anmeldung gestartet werden soll.

Passwörter in Linux – Beispiel Passwd

`its_vorlesung:x:1002:1002:its_vorlesung,42,23,0,-:`
`/home/its_vorlesung:/bin/bash`, d. h.

- der Benutzer heißt `its_vorlesung`,
- das Passwort steht in `/etc/shadow`,
- UID und GID sind `1002`,
- `its_vorlesung,42,23,0,-` sind Infos,
- `/home/its_vorlesung` ist das Heimatverzeichnis und
- `/bin/bash` ist die Shell

Passwörter in Linux – Shadow


```
-rw-r--r-- 1 root root 2323 Okt 24 10:16 /etc/passwd
```

- D. h. root darf schreiben, jeder darf lesen
- Dadurch sind Wörterbuchangriffe (dabei werden typische Passwörter gehasht und die Hashes zum Vergleich herangezogen) einfach durchführbar.
- Entsprechend werden Passwörter typischerweise nicht in `/etc/passwd` gespeichert.
- `/etc/shadow` ist die zentrale **Passwortverwaltung**

```
-rw-r----- 1 root shadow 1384 Okt 24 08:07 /etc/shadow
```

- D. h. root darf schreiben, Gruppe shadow darf lesen

Passwörter in Linux – Format Shadow

Ein Eintrag in `/etc/shadow` besteht aus folgenden mit : getrennten Komponenten:

Benutzername Benutzername in druckbaren Zeichen

Passwort **Hash des Passworts**

DOC Tag, an dem das Passwort zuletzt geändert wurde

MinD Minimale Anzahl der Tage, die das Passwort gültig ist

MaxD Maximale Anzahl der Tage, die das Passwort gültig ist

Exp Anzahl Tage die das Passwort trotz Ablauf der MaxD noch gültig ist

Dis Tag bis zu dem das Benutzerkonto gesperrt ist

Res Reserve

Passwörter in Linux – Beispiel Shadow

`its_vorlesung:$6$eJEe0D4/$7gnAYeNKq5MUihn9UtyKq17qj57ZZimeETmiSie1hTojRDBiFHi3/JUP8TYlrxYQx6R6mYrnRrx58LUs0XdGP1:16847:0:99999:7:::`, d. h.

- der Benutzer heißt `its_vorlesung`,
- dann folgt der **Hash des Passworts**,
- zuletzt geändert als 16847 Tage seit dem 1.1.1970 vergangen waren,
- ist minimal 0 Tage und maximal 99999 Tage gültig,
- gewarnt wird 7 Tage vor Ablauf des Passworts und
- die übrigen Werte sind nicht spezifiziert

Passwörter in Linux – Hashing

- Linux verwendet `crypt(3)` (https://man7.org/linux/man-pages/man3/crypt.3.html, aufgerufen am 28. Juni 2023) der Glibc als Hashfunktion
- `crypt(3)` basiert ursprünglich auf DES mit einem Klartext bestehend nur aus 0 als Eingabe; der Geheimtext ist der Hash des Passworts.

Passwörter in Linux – Salt

- Es ist nicht unwahrscheinlich, dass **mehrere Menschen das gleiche Passwort** verwenden.
- Ensprechend würde sich für ein solches mehrfach vergebenes Passwort der gleiche Hash ergeben.
- Daher verwendet `crypt(3)` als Schlüssel das Passwort mit angehangenem zufälligen **salt** Wert, der ebenfalls in der `/etc/shadow` gespeichert wird.

Passwörter in Linux – Beispiel Hashing und Salt

Fachhochschule Dortmund
University of Applied Sciences and Arts

Betrachten wir nochmal das Beispiel:

`its_vorlesung:$` **`6`** `$` **`eJEe0D4/`** `$` **`7gnAYeNKq5MUihn9UtyKq17qj57ZZmeETmiSie1hiTojRDBiFHi3/JUP8TYlrxYQx6R6mYrnRrx58LUsoXdGP1`** `:16847:0:99999:7:::`, wobei

- 6 spezifiziert die verwendete Hashfunktion SHA-512
- `eJEe0D4/` ist der salt Wert
- `7gnAYeNKq5MUih` ...ist der Hash

Passwörter in Linux – Zusammenfassung

- Passwörter werden **nicht im Klartext gespeichert**
- Passwörter werden **separat verwaltet (shadow)**
- Passwörter werden **gesalzen (Benutzer-individuell) und gehasht**
- Passwort-basierte Authentifikation wird basierend auf einem Vergleich von Hash durchgeführt.

Angriffsarten und Argon2

Wiederherstellung von Passwörtern: Online-Angriffe

Ziel

Ausgehend von einem Hash und Salt soll das zugehörige Passwort wiederhergestellt werden.

- **Brute Force**: Berechne nacheinander die Hashes aller möglichen Passwörter, bis der korrekte Hash gefunden ist.
  - Probleme: Laufzeit, Verwerfen von berechneten Hashes
- **Wörterbuch-Angriff**: Berechne nur die Hashes von ausgewählten Passwörtern (z. B. basierend auf den TOP 10-Passwörtern)
  - Problem: Mangelhafte Passwörter-Abdeckung
- Die vorgenannten Angriffe sind **Online-Angriffe** (berechnete Hashes werden verworfen).

Wiederherstellung von Passwörtern: Offline-Angriffe

- Die folgenden Angriffsarten sind heute typischerweise aufgrund **gesalzener Passwörter** nicht mehr praktikabel.
- Berechne nacheinander die Hashes aller möglichen Passwörter. Speichere alle Klartext-Hash-Paare in einer **sortierten Tabelle**.
  - Vorteil: Schnelle, binäre Suche
  - Problem: z. B. bei einer Suche über 77 mögliche Zeichen (z. B. $\{A-Z, a-z, 0-9$ sowie übliche Sonderzeichen$\}$) und einem siebenstelligen Klartext ergeben sich 77^7 Variationen. Ausgehend von MD5, werden pro Klartext-Hash-Paar 16 Byte für den Hash und 7 Byte für den Klartext benötigt und somit insgesamt ca. 335 TB. Bei einem achtstelligen Passwort werden ca. 25 PB benötigt.
- **Rainbow Tables** (Oechslin, 2003) sind Datenstrukturen, die einen **Time-Memory Tradeoff** erlauben.

Praxisrelevanz von Online-Angriffen

- MD5, SHA-1/2/3, etc. sind **universelle und hoch-performante Hashfunktionen** insbesondere zum Nachweis von Datenintegrität.
- In Verbindung mit aktueller CPU/GPU-Performance (**Moore'sches Gesetz**) sind Online-Angriffe praktikabel, selbst wenn gesalzene Passwörter eingesetzt werden.
- Hashcat berechnet z. B. mit einer (!) AMD Radeon RX 6800 XT ca. 50 Milliarden MD5 Hashes pro Sekunde[4]
- Passwort Cracker[5] unterstützen diese Angriffsarten.

[4] https://gist.github.com/epixoip/99085955a1145ff61ec83512a50421a7, aufgerufen am 28. Juni 2023
[5] https://openwall.com/john/ und https://hashcat.net/, aufgerufen am 28. Juni 2023

Härtung gegen Online-Angriffe

- Passwortkonstruktion gem. NIST 800-63B, 2017, Sec. 5.1.1.2 (z. B. keine bekannten Passwörter)
- Mehrfaches Hashing (typischerweise $>$ 10.000 Iterationen)
- **Spezialisierte Hashfunktionen**, z. B. **Argon2**[6] und **SCrypt**[7] bieten Mechanismen zur Anpassung ihrer Performance und können so mit dem Moore'schen Gesetz „mithalten".

[6] https://www.rfc-editor.org/rfc/rfc9106, aufgerufen am 28. Juni 2023
[7] https://www.rfc-editor.org/rfc/rfc7914, aufgerufen am 28. Juni 2023

Argon2

- Parameter[8]:
  - Passwort
  - Salt (16 Byte empfohlen)
  - **Parallelisierung** (Anzahl Threads)
  - **Iterationen** (Ausführungszeit)
  - **Speichernutzung** (KB)
- Adäquate Auswahl Parameter[9]
- Argon2d besonders gehärtet gegen GPU-basierte Angriffe, Argon2i gegen Seitenkanalangriffe
- **Argon2id** hybrider Ansatz, im Allgemeinen empfohlen

[8] https://argon2.online/, aufgerufen am 28. Juni 2023
[9] https://github.com/paragonie/argon2-refiner, aufgerufen am 28. Juni 2023

Fallstricke und Anwendungshinweise

- Benutzer-individuelles Salt nutzen
- Passwortkonstruktion gem. NIST 800-63B, 2017, Sec. 5.1.1.2
- Auf universelle Hashfunktion verzichten, stattdessen Argon2id (oder SCrypt) nutzen

Zusammenfassung

Zusammenfassung

- Grundlagen Authentifikation präsentiert
- Entropie im Zusammenhang mit Passwörtern erklärt
- Linux Passwortdatenbanken und Angriffsarten gelernt
- Argon2 vorgestellt

Weiterführende Literatur

Weiterführende Literatur

- *IT-Sicherheit – Konzepte - Verfahren - Protokolle*, Kapitel 10 von Eckert (2023)
- OWASP Password Storage Cheat Sheet[10]
- Password Hashing Competition[11]

[10] https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html, aufgerufen am 28. Juni 2023

[11] https://www.password-hashing.net/, aufgerufen am 28. Juni 2023

Literatur

- Eckert, C. (2023). *IT-Sicherheit: Konzepte - Verfahren - Protokolle* (11. Aufl.). De Gruyter Oldenbourg. (Siehe S. 38).
- NIST 800-63. (2004). NIST Special Publication 800-63). Electronic Authentication Guideline. https://csrc.nist.gov/CSRC/media/Publications/sp/800-63/ver-10/archive/2004-06-30/documents/sp800-63-v1-0.pdf (siehe S. 17).
- NIST 800-63B. (2017). NIST Special Publication 800-63B. Digital Identity Guidelines – Authentication and Lifecycle Management. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63b.pdf (siehe S. 17, 32, 34).
- Oechslin, P. (2003). Making a Faster Cryptanalytic Time-Memory Trade-Off. In D. Boneh (Hrsg.), *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings* (S. 617–630, Bd. 2729). Springer. (Siehe S. 30).