M183 Applikationssicherheit Implementieren # 4

By Jürg Nietlispach

Recap # 3?

Recap # 3

1. XSS-Attack Fake-Login
2. Clickjacking-Attack Fake-Login with IFrame

Praktisch ohne Javascript-Vorwissen in 30’ ready!

Securing Web Application Logins?

Assumption: Attacker knows Username & Password from XSS (or Injections, or Brute-Force or, …)

Securing Web Application Logins?

Add additional Security-Layer (2-Factor-Authentication: Something You Know + Something you have)?

- …

Securing Web Application
Login 1

Add additional Security-Layer (2-Factor-Authentication: Something You Know + Something you have):

- One-Time-Password (OTP)
- Time-Based One-Time-Password (TOTP)
- PIN-Token (e-banking),
- Unique-Hardware-Token, …

Securing Web Application
Login 2

Alterative / Third-Party Authentication Methods, Authentication as a Service, Single Sign on etc.: Next Week!

2-Factor Auth using OTP (SMS / Email)

1. User enters credentials
2. Backend-System generates a Token / One Time Password
3. Backend-System sends the Token (Email, SMS)
4. User enters Token and can enter the System

2-Factor Auth using OTP (SMS / Email)

Constraints / Consequences?

- Token?
- System?
- Setup?

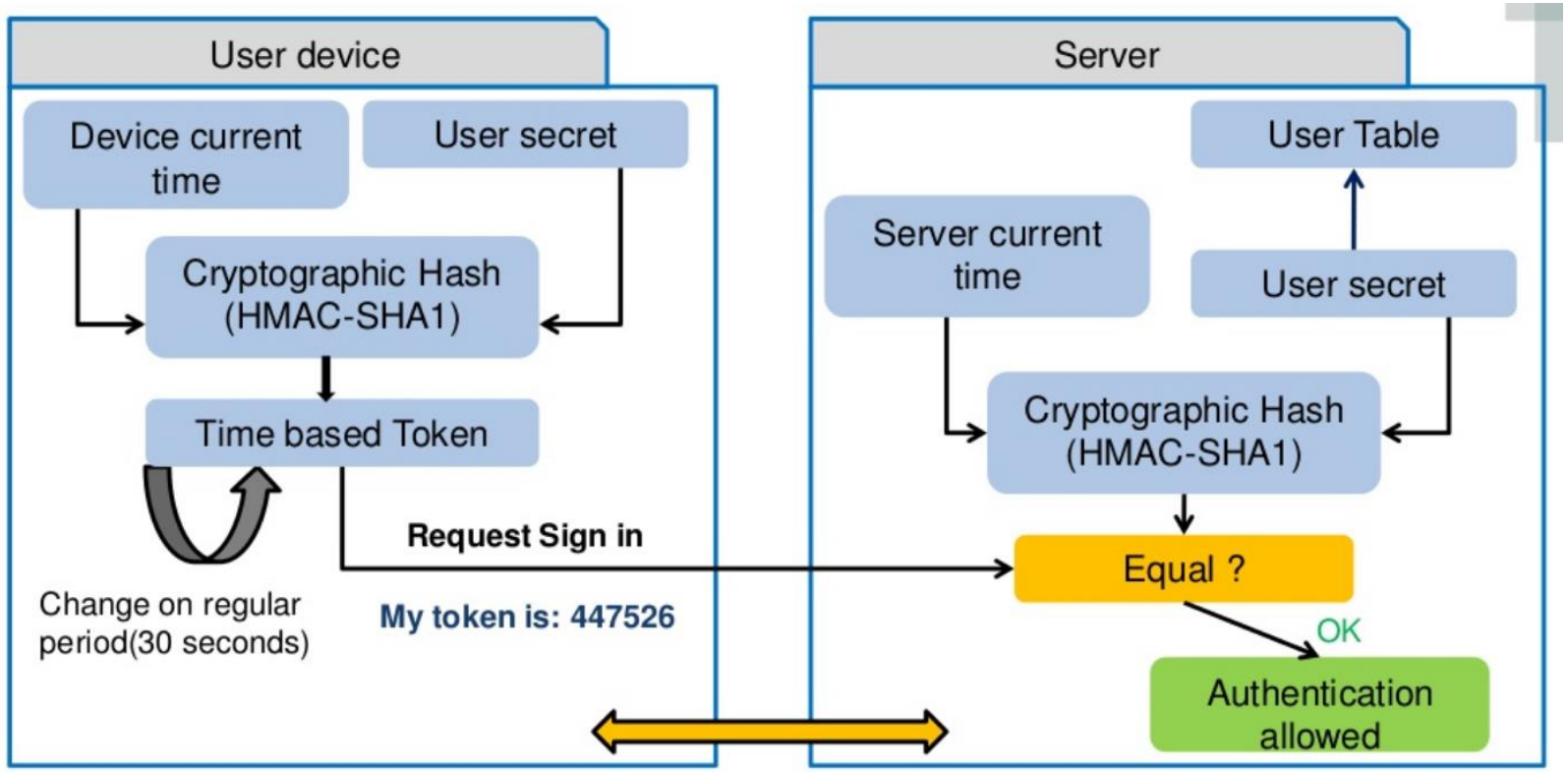
2-Factor Auth using OTP (SMS / Email)

Constraints / Consequences?

- Token: 4-8 Characters (ASCII)?
- System: consider downtime SMS/Email-Gateway?
- Setup: forgot Device/Smartphone?

2-Factor Auth using TOTP (Google-Authenticator)

1. Device and Backend-System agree on a Secret Key (User – Secret, via QR-Code)
2. Device generates Token on Device Current Time
3. Server generates Token on Server Current Time
4. Server compares Tokens and the user can enter the System.



2-Factor Auth using TOTP

Constraints / Consequences?

2-Factor Auth using TOTP

Constraints / Consequences:

1. Shared (initial) Key between Device and Server (per User).
2. Shared Key can be stolen.
3. Moving Factor: Current Time, Token changes in regular Intervals.
4. Clocks have to be perfectly in sync (via Google-Time-Servers, or ETHZ)!

TOTP-Algorithm Step 1

https://tools.ietf.org/html/rfc6238

1) Initialization

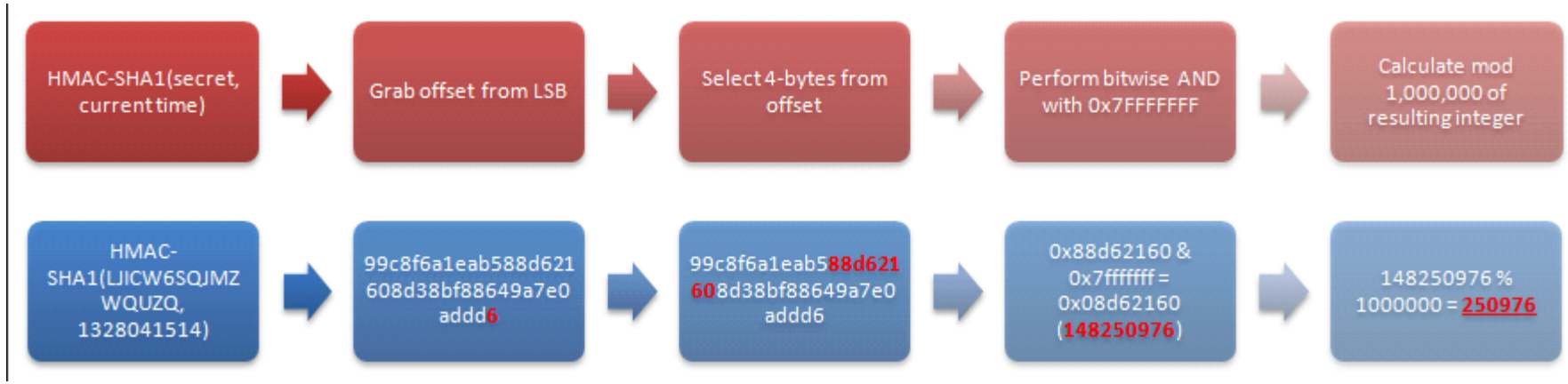
- Generate a key, K, which is an arbitrary byte string, and share it securely with the client.
- Agree upon a T0, the Unix time to start counting time steps from, and an interval, TI, which will be used to calculate the value of the counter C (defaults are the Unix epoch as T0 and 30 seconds as TI)
- Agree upon a cryptographic hash method (default is SHA-1)
- Agree upon a token length, N (default is 6)

TOTP-Algorithm Step 2

2) Token generation

- Calculate C as the number of times TI has elapsed after T0.
- Compute the HMAC hash H with C as the message and K as the key (the HMAC algorithm is defined in the previous section, but also most cryptographical libraries support it). K should be passed as it is, C should be passed as a raw 64-bit unsigned integer.
- Take the least 4 significant bits of H and use it as an offset, O.
- Take 4 bytes from H starting at O bytes MSB, discard the most significant bit and store the rest as an (unsigned) 32-bit integer, I.
- The token is the lowest N digits of I in base 10. If the result has fewer digits than N, pad it with zeroes from the left.

TOTP-Algorithm Example



Setup Übungen / Unterricht

1. Konzentriertes Arbeiten / Mitmachen
2. Ruhiges Arbeiten (bzgl. Lautstärke und anderen Störfaktoren, Gänge zum Brünneli / Toilette, Smartphones etc.)
3. Tipp: Zeit effizient nutzen!
4. Gänge zum Brünneli & Toilette -> Pausen!
5. Schluss der Stunde (autonom): alle .git-Repos mit allen bisherigen Übungen!

Übungen 2-Factor Auth

1. OTP via SMS / Email
2. TOTP via Google Authenticator

Übung 2-Factor Auth: SMS

Setup SMS-Gateway

1. Account bei Nexmo (nexmo.com) erstellen
2. Demo-SMS an eigene Nummer senden (via CURL an das API)

Applikation (Localhost C#)

1. Login-Maske (Username & Password)
2. Bei Form-Submission check von Username & Passwort. Stimmen diese:
    1. OTP generieren (8-ASCII-Zeichen) und per API senden
    2. (Drittes) Input-Feld für OTP anzeigen
3. Form-Sumission check von Usernamen, Passwort & OTP

Übung 2-Factor Auth: Email

Setup Email-Gateway

1. Account bei Mailgun (mailgun.com) erstellen und aktivieren
2. Sich selber als «authorized» Recipient adden
3. Demo-Email an eigene Email senden (via CURL an das API)

Applikation (Localhost C#)

1. Login-Maske (Username & Password)
2. Bei Form-Submission check von Username & Passwort. Stimmen diese:
    1. OTP generieren (8-ASCII-Zeichen) und per API senden
    2. (Drittes) Input-Feld für OTP anzeigen
3. Form-Sumission check von Usernamen, Passwort & OTP

Übung 2-Factor Auth: TOTP

Google Authenticator

1. Aus App-Store downloaden

Applikation (Localhost C#) - Keyaustausch

1. TOTP-Function / Library suchen und installieren
2. Secret generieren
3. QR-Code generieren
4. Mit Google-Authenticator-App scannen

Applikation (Localhost C#) - Authentifizierung

1. Bei Form-Submission check von Username & Passwort. Stimmen diese:
    1. (Drittes) Input-Feld für OTP anzeigen
2. Form-Sumission check von Usernamen, Passwort & TOTP