Zwei-Faktor-Authentifikation mit Yubikey-Token

Stefan Schumacher

sicherheitsforschung-magdeburg.de
stefan.schumacher@sicherheitsforschung-magdeburg.de

Chemnitzer Linux-Tage 2016
19.03.2016



Über Mich



Über Mich

- Geek, Nerd, Hacker seit mehr als 20 Jahren
- Berater für Finanzinstitute, Regierungen, Sicherheitsbehörden
- Direktor des Magdeburger Instituts für Sicherheitsforschung
  Forschungsprogramme zur Unternehmenssicherheit
- Herausgeber des Magdeburger Journals zur Sicherheitsforschung
- `www.Sicherheitsforschung-Magdeburg.de`



Authentifizierung

- Authentifizierung: Nachweis einer bestimmten Eigenschaft
- Autorisierung: Einräumung von Rechten
- Bsp: Benutzername/Passwort; Anruf/Parole; Person/Personalausweis;
- Problem: Jeder der Benutzername und Passwort kennt, kann sich als der Benutzer ausgeben
- schwache Passwörter, schlechte Passwortsicherheit
- Stratfor-Hack: Kundendatenbank gestohlen, Passwörter als MD5 gehasht
- Thomas Roth: 1-6 stellige SHA1-Hashes in 50 Minuten/2$ auf Amazon EC2



Zwei-Faktor-Authentifizierung

- Zwei-Faktor-Authentifizierung: Einführung eines 2. Faktors zur Authentifizierung
- Einbrecher kann Passwort kennen, benötigt aber 2. Faktor (Token)
- Schutz vor Shouldersurfing, schwachen Passwörtern, schlechten Datenbanken
- Schutz vor Man-in-the-Middle möglich



Zwei-Faktor-Authentifizierung

- Wissen: Passwort, Pin, TAN
- Besitz: Schlüssel, EC-Karte, Token, Datei
- Sein: Fingerabdruck, Iris, Stimme, Gesicht
- Kombinationen: Benutzername+Passwort+Token; Benutzername+TAN+Token; Smartcard+Token
- Nachteile: Token muss mitgeführt werden und kostet Geld



Zwei-Faktor-Authentifizierung

- Wissen: Passwort, Pin, TAN
- Besitz: Schlüssel, EC-Karte, Token, Datei
- Sein: Fingerabdruck, Iris, Stimme, Gesicht
- Kombinationen: Benutzername+Passwort+Token; Benutzername+TAN+Token; Smartcard+Token
- Nachteile: Token muss mitgeführt werden und kostet Geld



Inhaltsverzeichnis

1 Konsortien

2 Token

3 Standards

4 Anwendung



OATH

- Initiative for Open Authentication (OATH)
- Industriezusammenschluss
- Mitgliedschaft kostet Geld
- HOTP: RFC4226
- TOTP: RFC6238



FIDO

- Industrie-Konsortium mit 200 Mitgliedern, u.a. Google, Paypal, Lenovo, Alibaba, NTT DoCoMo, Samsung, Visa, RSA, Intel, ING, Yubico
- Spezifikationsrahmen u.a. für Biometrie, Trusted Platform Modules, Smart Cards, NFC
- Authentifikation mittels asymmetrischer Kryptographie
- U2F: Universal 2nd Factor, *offener* Standard für 2FA via USB oder NFC
- Integriert in Chrome seit 38, Firefox, Windows 10 und Edge folgen



Inhaltsverzeichnis

1 Konsortien

2 Token

3 Standards

4 Anwendung



Yubikeys



PlugUp Keys





fido U2F Certified

Yubikey Token

- Yubikey Neo: 55€, USB+NFC, OTP, PIV, OpenPGP (2048bit), U2F
- Yubikey 4: 48€, USB, OTP, PIV, OpenPGP (4096bit), U2F, PKCS#11
- Yubikey 4 Nano: 60€
- FIDO U2F Security Key: 17€, nur U2F
- Plug-up U2F: 5€
- Yubikey Neo/4: 2 Slots, müssen programmiert werden



Fähigkeiten

- Public-Key-basiertes Challenge-Response Protokoll
- Phishing und MitM-Schutz
- applikationsspezifische Schlüssel,
- Erkennung geklonter Token
- Beglaubigung von Token



Inhaltsverzeichnis

1 Konsortien

2 Token

3 Standards

4 Anwendung



One Time Passwort (OTP)

- Passwort, welches nur einmal gültig ist
- verfällt nach (missglücktem) Login-Vorgang
- passives Mithören, Replay-Attacken nicht möglich
- MitMA können funktionieren
- Alice und Bob müssen das gültige OTP kennen
- Kennwortliste oder Kennwortgenerator



One Time Passwort (OTP)

Kennwortgeneratoren

- übertragen das *Ergebnis* eines Algorithmus
- Zeitgesteuerte Generatoren
  wohldefiniertes Zeitintervall, Uhren müssen synchron sein
- Ereignisgesteuerte Generatoren
  Ereignis löst Kennwortgenerator aus, keine Uhr/Strom nötig, OTP kann theoretisch ∞ gültig bleiben
- Challenge-Response-gesteuerte Generatoren
  Client berechnet OTP basierend auf Challenge, Challenge sollte zufallsgeneriert sein
- HOTP: HMAC-SHA1 via Zähler und Seed, SHA1 + Key + Message +XOR (definierte Werte)



U2F Login

- Login auf Google.com
- Benutzername und Passwort eingeben
- wenn Passwort korrekt, Token einstecken (oder SMS)
- Kontaktschalter drücken
- Token authentifiziert sich via U2F Server
- Google akzeptiert Login, wenn der Token eingetragen ist
- Browser kann per Cookie aktiviert bleiben



U2F Schema

- Token hat k_{priv} und sendet k_{pub} an Relying Party (RP)
- Schlüsselpaar wird im Token generiert, kann nicht manipuliert werden
- Browser kompiliert Informationen (URI, TLS Channel ID) über HTTP-Verbindung
- Token signiert diese Informationen und schickt sie an RP



U2F Schema

- Token generiert neues Schlüsselpaar mit App ID und Key Handle für jede Registrierung $\rightsquigarrow$ RP weiß nicht, dass Alice und Bob das selbe Token nutzen
- Authentifikationszähler inkrementiert bei jeder Authentifizierung, RP vergleicht Zähler mit gespeichertem Stand $Z_T > Z_{RP}$
- Beglaubigungs-Zertifikat kann vom Hersteller ausgefüllt und auf Token gespeichert werden $\rightsquigarrow$ RP kann nur bestimmte Token zulassen (z.B. Yubikey ja, RSA nein, oder spezielle Yubikeys)



U2F Unterstützung

- Google, Dropbox, Github, PAM,
- Wordpress, Django, Ruby on Rails
- OpenSSH, Login, OpenVPN, FreeRADIUS via PAM
- LastPass, Dashlane, Password Safe, Passpack, Password Tote, pwSafe, KeePass



Inhaltsverzeichnis

1 Konsortien

2 Token

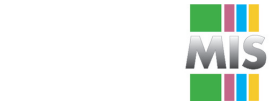
3 Standards

4 Anwendung



Google einrichten

1. Mein Konto,
2. Anmeldung und Sicherheit
3. Bestätigung in zwei Schritten
4. Sicherheitsschlüssel
5. Unter Bestätigungscodes: Handy registrieren und SMS einrichten oder Google Authenticator App einrichten
6. Zusätzlich: Ersatzcodes einrichten (TAN-Liste), ausdrucken und wegschließen



Web.de unterstützt U2F nicht

- OATH-HOTP im Yubikey Personalization Tool einrichten; Secret Key generieren
- KeePass installieren und einrichten, Plugin OtpKeyProv installieren, Secret Key hinterlegen
- sicheres Passwort für KeePass vergeben
- Zufallspasswort (256HEX Bit) generieren und bei Web.de eintragen

  `09137f0ac6627f9e94e2af2342d2610bf20ff0ee929114d27b3629e3823e11ea`
- KeePass mit dem Webbrowser via Plugin verbinden
- KeePass-Datenbank mit Passwort und Token entschlüsseln, Browser holt User/Passwort für Web.de via Plugin aus DB.



Yubico für eigene Dienste

- Plugins existieren u.a. für Wordpress, Django etc.
- PAM-Modul (`yubico-pam`) kann eingebunden werden
- Yubikey-Server existiert als Paket, YubiHSM als Hardwareerweiterung
- Online-Validierung via YubiCloud oder eigenem Validierungsserver
- seit 2.6 Offline-Validierung dank HMAC-SHA1 Challenge-Response
- PAM-Modul installieren und konfigurieren,
  Conf-Datei landet in $HOME, Obacht bei ECryptFS
- SSH (Passwort + Token) per PAM
- OpenVPN und Radius via PAM



yubikey-luks

- System verschlüsselt installieren, mit sehr langem Zufallspasswort
- Das Zufallspasswort bleibt immer aktiviert, LUKS kann auch ohne Yubikey eingebunden werden
- `cryptsetup luksDump /dev/sda2` # Keyslots anzeigen lassen
- standardmäßig gibt es 7 Keyslots $\rightsquigarrow$ 7 Passwörter für ein LUKS



yubikey-luks

- `ppa:yubico/stable` und `ppa:privacyidea/privacyidea-dev` installieren
- `ykpersonalize -2 -ochal-resp -ochal-hmac -ohmac-lt64 -oserial-api-visible`
  # Challenge-Response Authentifikation mit HMAC-SHA1 aktivieren
- `cryptsetup luksAddKey -key-slot 7 /dev/sda2`
- `yubikey-luks-enroll -d /dev/sda5 -s 7 -c` # neues Passwort für LUKS mit Yubikey vergeben



GnuPG und Yubikey

- Neo (2048 Bit) und 4 (4096 Bit) unterstützten OpenPGP
- `ykpersonalize -m82` # OpenPGP-Applet aktivieren
- `gpg -change-pin` # Admin und User-PIN ändern
- Schlüsselpaar auf Yubikey generieren:
- `gpg -card-edit admin generate` # PrivKey bleibt auf Yubikey
- Schlüsselpaar auf Yubikey kopieren:
- `gpg -export-secret-key -a XXX > XXX.asc`
- `gpg -edit-key XXX keytocard` # lokale Schlüssel werden entfernt



- sicherheitsforschung-magdeburg.de
- stefan.schumacher@sicherheitsforschung-magdeburg.de
- sicherheitsforschung-magdeburg.de/publikationen/journal.html



- youtube.de/Sicherheitsforschung
- Twitter: 0xKaishakunin
- Xing: Stefan Schumacher
- ZRTP: 0xKaishakunin@ostel.co