OATH-LDAP

OATH-basierte Zwei-Faktor-Authentisierung mit OpenLDAP

Chemnitzer Linux Tage 2018

Zur Person

- Michael Ströder <michael@stroeder.com>, Freiberufler
- Schwerpunkte
  - X.509-basierte PKI, angewandte Verschlüsselung, dig. Signatur
  - Verzeichnisdienste (LDAP etc.), Identity & Access Management
  - Single Sign-On, Zwei-Faktor Authentifizierung
- Open Source / Freie Software
  - Æ-DIR, OATH-LDAP
  - web2ldap

Agenda

- Einführung OATH, HOTP und TOTP
- OTP mit Yubikey
- Warum zur Hölle LDAP?
- Zweistufige OpenLDAP-Architektur
- Flexibles Datenmodell
- Sicheres Enrollment und Anti-Patterns
- Anwendungen und was nicht wirklich geht

Wozu?

- Passwörter gehen durch potentiell unsichere Systeme
- Passwörter können bei Eingabe beobachtet werden
- → mehr Schutz durch Besitz und Wissen
- Regulatorische Vorgaben fordern 2FA
  - Kreditkartensicherheitsstandard PCI DSS v3.2+

OATH -- Open Authentication

- Herstellerunabhängige Standardisierung: http://www.openauthentication.org
- SHA-1 basierte HMAC-Berechnung, optional SHA-2
- IETF: RFCs
  - RFC 4226: HOTP -- zählerbasiert (12/2005)
  - RFC 6238: TOTP -- zeitbasiert (05/2011)
  - RFC 6287: OCRA -- Challenge-Response (06/2011)
- Sicherheitsaspekt: Shared Secrets!

HOTP: An HMAC-Based One-Time Password Algorithm

- Zählerbasiertes Verfahren: Truncate(HMAC-SHA-1(K,C))
- Aktualisierung des Zählers erfordert Schreibzugriffe
- Sicherheitsaspekt:
  HOTP-Wert ist gültig solange nicht validiert!
- Unbedingt bei HOTP-Validierung den Zähler erhöhen!
- Übersprungene HOTP-Werte in gewissen Grenzen tolerieren
- Evtl. Neusynchronisation erforderlich

TOTP: Time-Based One-Time Password Algorithm

- Zeitbasiertes Verfahren: Truncate(HMAC-SHA-1(K,T))
  mit T = (Current Unix time - T0) / X
- Normalerweise keine Schreibzugriffe erforderlich
- Sicherheitsaspekt: TOTP-Wert ist innerhalb des Zeitfensters mehrfach benutzbar!
- TOTP-Wert nach Zeitfenster nicht mehr gültig
- Zeitsynchronisation oder Behandlung von Time-Drift (Schreibzugriff)

OTP mit Yubikey

- Zwei “Slots”, Auslösung durch “Touch”, USB-Tastaturemulation
- OTP-Generierung oder statisches Passwort
- vorbeschlüsselt für Nutzung mit Yubico-Cloud
  - Yubico-OTP (AES-256)
  - Shared Secret bei Yubico → Cloning möglich
- Direkte HOTP-Eingabe über Tastaturemulation
- TOTP nur via “App” und manueller Eingabe

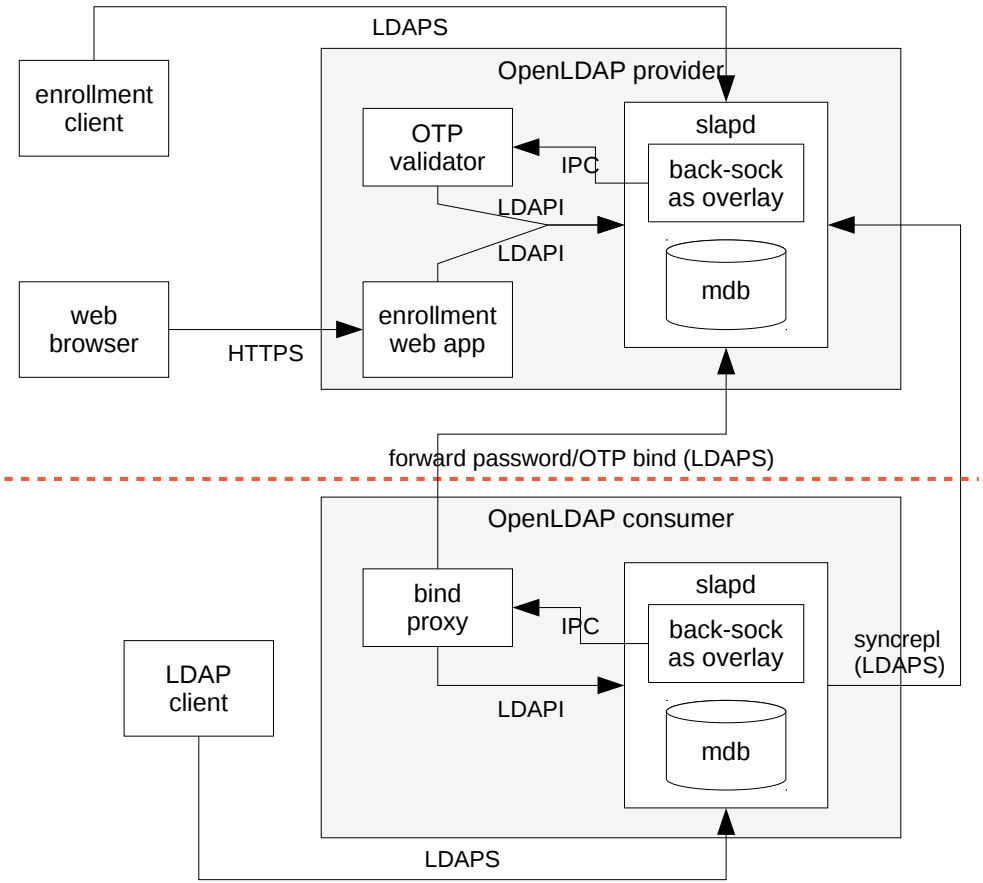
Warum zur Hölle LDAP?

- Anwendungen/Systeme unterstützen LDAP f. Authc
  - Web-Anwendungen
  - Linux-Logins (PAM/NSS)
- In konkretem Projekt:
  - Anwendungen und Systeme bereits integriert via LDAP
  - LDAP-Infrastruktur bereits hochverfügbar
  - separate Cluster-Datenbank erheblicher Aufwand
- → 2FA-Integration direkt via LDAP

OpenLDAP-Integration

- *back-sock* als *Overlay* leitet BIND und COMPARE aus
- Python 2.7 Modulpaket *slapdsock*
- Listener dürfen nie blocken, da sonst OpenLDAP blockt!
- Ggf. 2-stufige Topologie mit r/o-Consumer
- *bind proxy* leitet HOTP-Prüfung an r/w-Provider (Zähler)

Architektur



LDAP Bind Request

- BindRequest (Bind-DN, Passwort + OTP)
- Bind-Requests nutzen quasi alle Anwendungen
- Funktioniert auch mit einem Passwortfeld
- Server-seitige Unterscheidung von Passwort und OTP durch feste OTP-Länge

LDAP Compare Request

- CompareRequest (User-DN, 'oathOTPValue', OTP)
- Für spezielle Anwendungsfälle, z.B. Passwortrücksetzung
- LDAP-Client nicht anonym
- Gesonderte Autorisierung / Proxy-Authentifizierung möglich

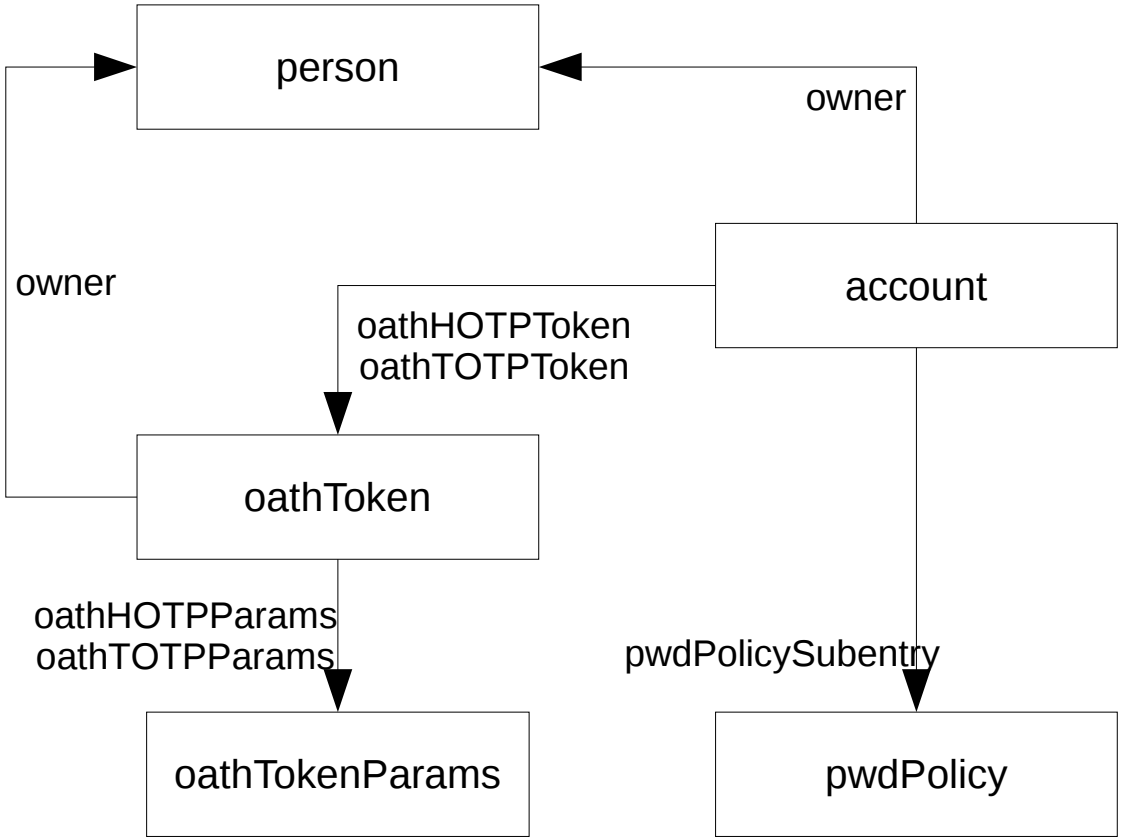
Herausforderungen bei OTP-Validierung

- Admin-Zugriffe auch bei partiell kaputter Infrastruktur
- Weiterleitung an mehrere Provider
- Fail-over und Load-Balancing, Schreibkonflikte vermeiden:
  rotate(providers, hash(token-dn) MOD len(providers))
- HOTP bei Split-Brain ergibt unterschiedliche Zählerstände
  → ggf. Resynchronisierung der Tokens notwendig
- TOTP immun gegen Split-Brain
- Kurzzeitiges Caching ist möglich, aber nur bedingt nützlich

oath-ldap.schema

- Flexible Integration
- Variable Sicherheitsrichtlinien
- Objektklassen (jeweils HOTP und TOTP):
  - Benutzer
  - Tokens
  - Richtlinien / Parameter
- Besitzereintrag nicht notwendigerweise Benutzerkonto

EER-Diagramm



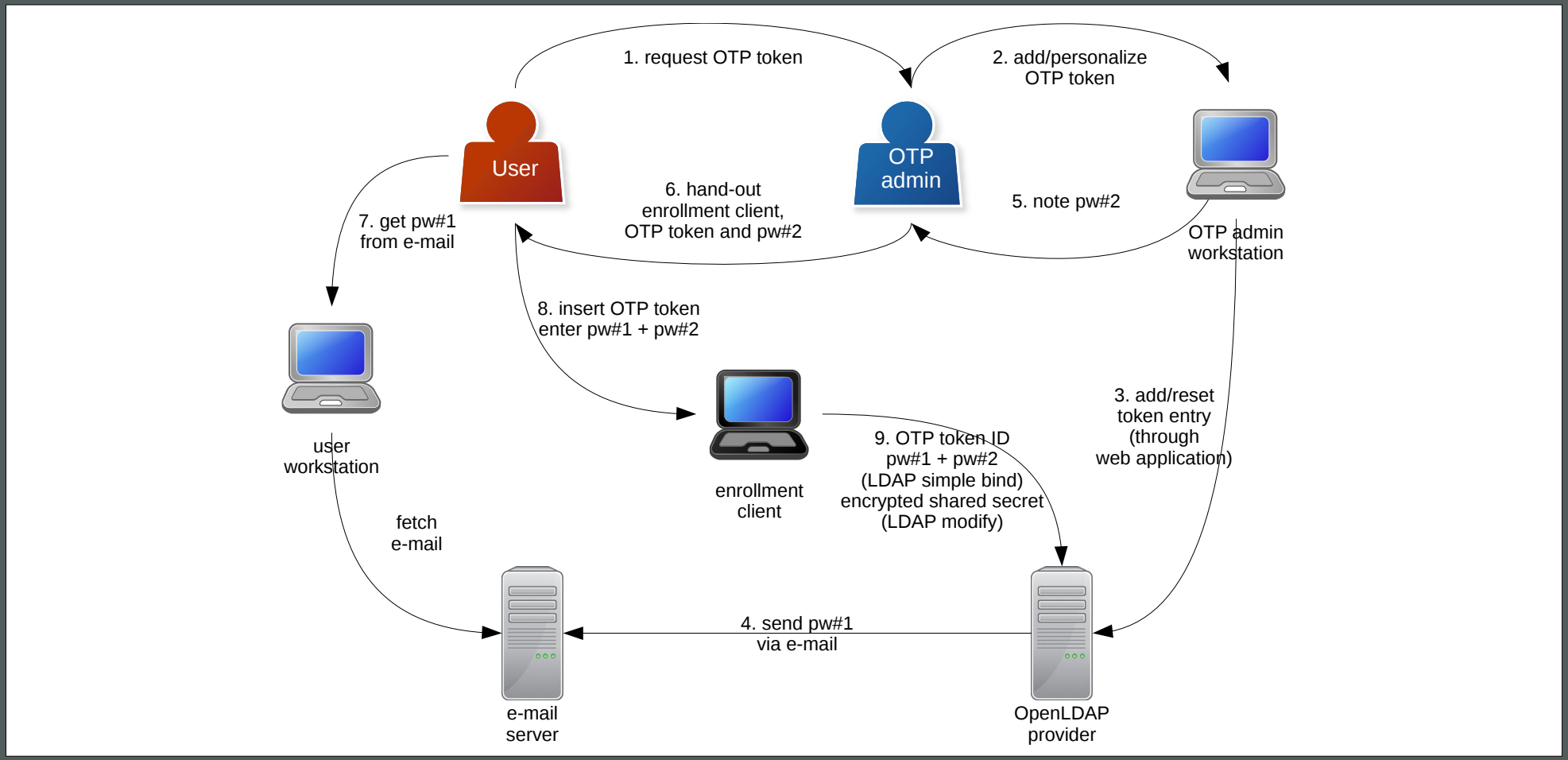
Enrollment -- Sicherheitsanforderungen

- Sichere Schlüsselerzeugung bei Initialisierung
- Benutzerpasswort und OTP-Secret niemals gleichzeitig im Klartext vorhanden (außer im Validator)
- Weder Benutzer noch Admin soll alleine initialisieren können
- Direkte asymmetrische Verschlüsselung des OTP-Secret
- Normale PCs nicht vertrauenswürdig
  → separates Enrollment-Gerät

Enrollment -- Anti-Patterns

- Vorbeschlüsselte OTP-Tokens mit Shared Secrets in der Klaut^H^H^H^H^HCloud
- Benutzer initialisiert nur authentifiziert mit seinem Benutzerpasswort das Token
- Benutzer kann mit Passwort mehrere Tokens initialisieren
  → noch besser für unbemerkte, missbräuchliche Nutzung
- Shared-Secret (Klartext!) wird als QR-Code angezeigt (yuck!)

Enrollment -- Prozess



Anwendungen -- was geht

- Web-Anwendungen mit ordentlicher Sitzungsverwaltung
- Anwendungen mit persistenter Backend-Verbindung:
  - web2ldap :-)
  - Datenbank-GUI-Clients
- vereinzelte SSH-Zugriffe
- Passworteingabe VPN-Verbindungen

Anwendungen -- was nicht geht

- Web-Server mit simpler Basic Authentication ohne Cookies
- Anwendungen mit erneuter Backend-Verbindung je Zugriff:
  - WebMail (z.B. roundcube)
  - naive Datenbank-Clients
- Automatisierte SSH-Zugriffe (ansible, Fabric, etc.)
- WLAN/WIFI/802.1x mit EAP-TTLS/PAP

Anwendungen -- Caching

- Auth-Caching kann Probleme abmildern, z.B.:
  - IMAP-Auth-Caching in dovecot
  - SSH mit *ControlMaster* und ggf. *ControlPersist*
  - SSO-Tokens
  - Caching in OTP-Validator
- Fragwürdig bzgl. Sicherheit und oftmals nicht hilfreich
  - “Re-Login” in aktueller Sitzung?
  - Akzeptable Cache-TTL(s)?

Demo...

Fragen?