2-FACTOR AUTHENTICATION WITH OPENLDAP, OATH-HOTP AND YUBIKEY

Axel Hoffmann

1&1

Biography

- Axel Hoffmann

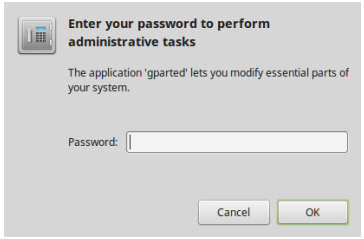
- Linux System Administrator
- 1&1 Mail & Media Dev. & Tech. GmbH
- axel.hoffmann@1und1.de

Introduction – Requirements

- No (or only less) integration effort on servers
- Ensure efficient work with MFA-devices
  - No manual typing of additional codes
  - Integration as Human Interface Device
- No automatic use of the second factor
- No software tokens
- Integration of already LDAP-enabled Appliances and Web GUIs
- High security level for token enrollment
  - Only an owner of a token should be able to use it
  - No pre-generated shared secrets on Tokens
  - Owners&admins haven't access to shared secrets

Introduction – Multi Factor Authentication

- Something that you *know*:
  - secret, password, passphrase, pin, transaction number



- Something that you *own*:
  - token, smartcard, key

- Something that you *are*:
  - eye iris, fingerprint, voice, type speed/behaviour

Introduction – Yubikey I

- Small USB stick-like device
- **Yubikey Standard** has component *YubiKey OTP*
- It is using two slots for
  - generating OATH HOTPs
  - generating Yubico OTPs
  - sending a static password
  - doing challenge-response
- works with default OS HID drivers/modules (simulates keyboard)
- 2 of these functions can be used (1 per slot)

Introduction – Yubikey II

- OATH HOTP, Yubico OTP, static password are typed by button press:
  - Short press, slot 1 is typed
  - Long press, slot 2 is typed

- **Yubikey Neo** has Java Card Applets & NFC:
  - *ykneo-oath*: stores PSKs of HOTPs/TOTPs for Android & Desktop App
  - *Yubico U2F*: is used for FIDO U2F auth
  - *ykneo-openpgp*: uses CCID via USB/NFC
  - *Yubico PIV*: Privilege & Identification Card Interface

Introduction – OATH Standards

Initiative for Open Authentication

- **HOTP**: HMAC-Based One-Time Password

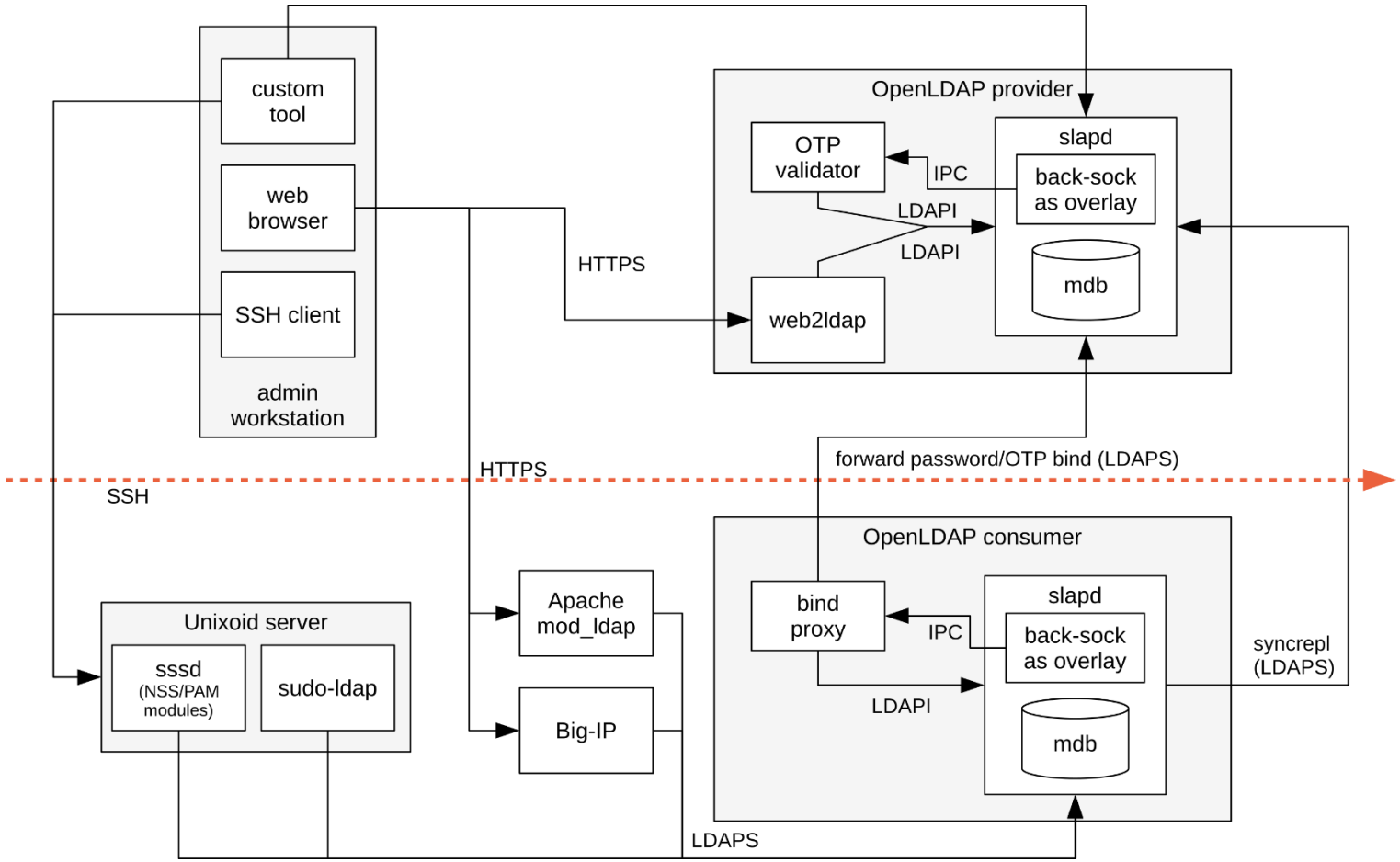
$$\text{TRUNC}(\text{SHA1}(\text{counter}, \text{psk})) \mod 10^{\text{numDigit}}$$

- **TOTP**: Time-Based One-Time Password

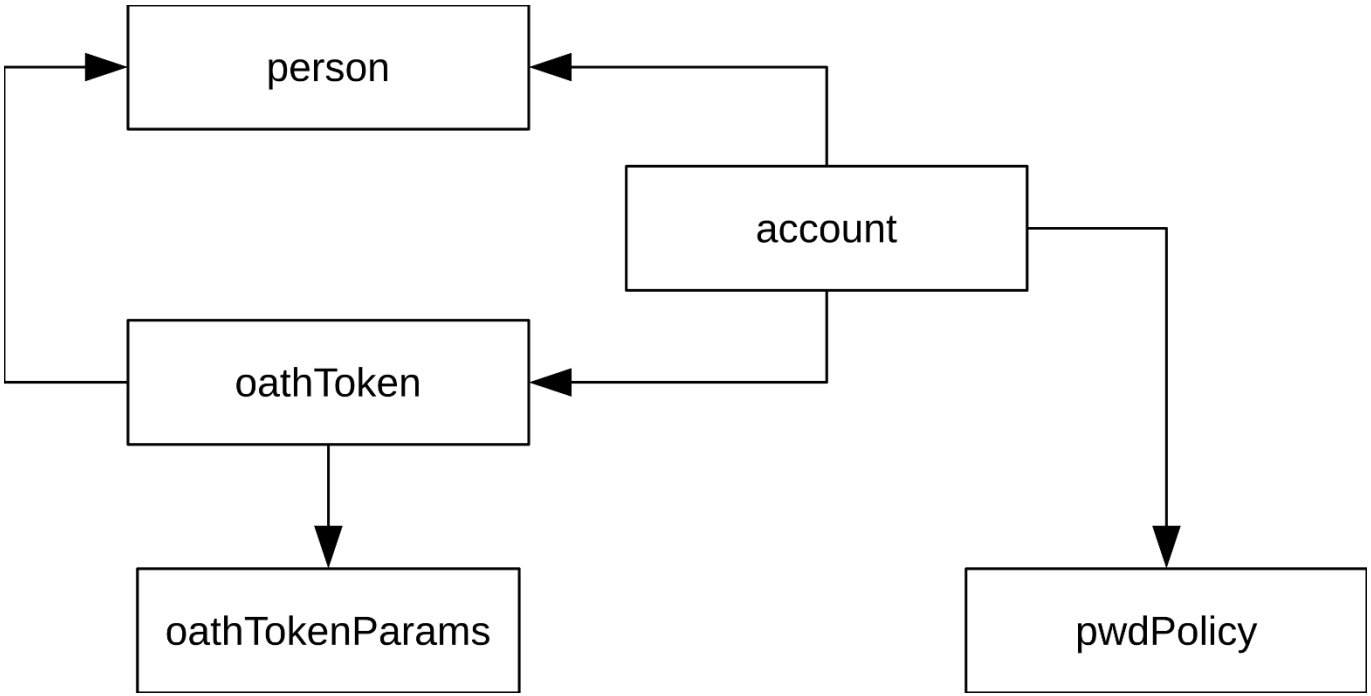
$$\text{counter} = \frac{(\text{UNIXTIME}(\text{now}) - \text{UNIXTIME}(\text{startEpoch}))}{\text{timeSteps}}$$

- **OCRA**: OATH Challenge-Response Algorithm
  - HOTP to calculate response from challenge + psk
  - optional: timeSteps, counter, session information

Implementation – Overview



Implementation – LDAP Structure



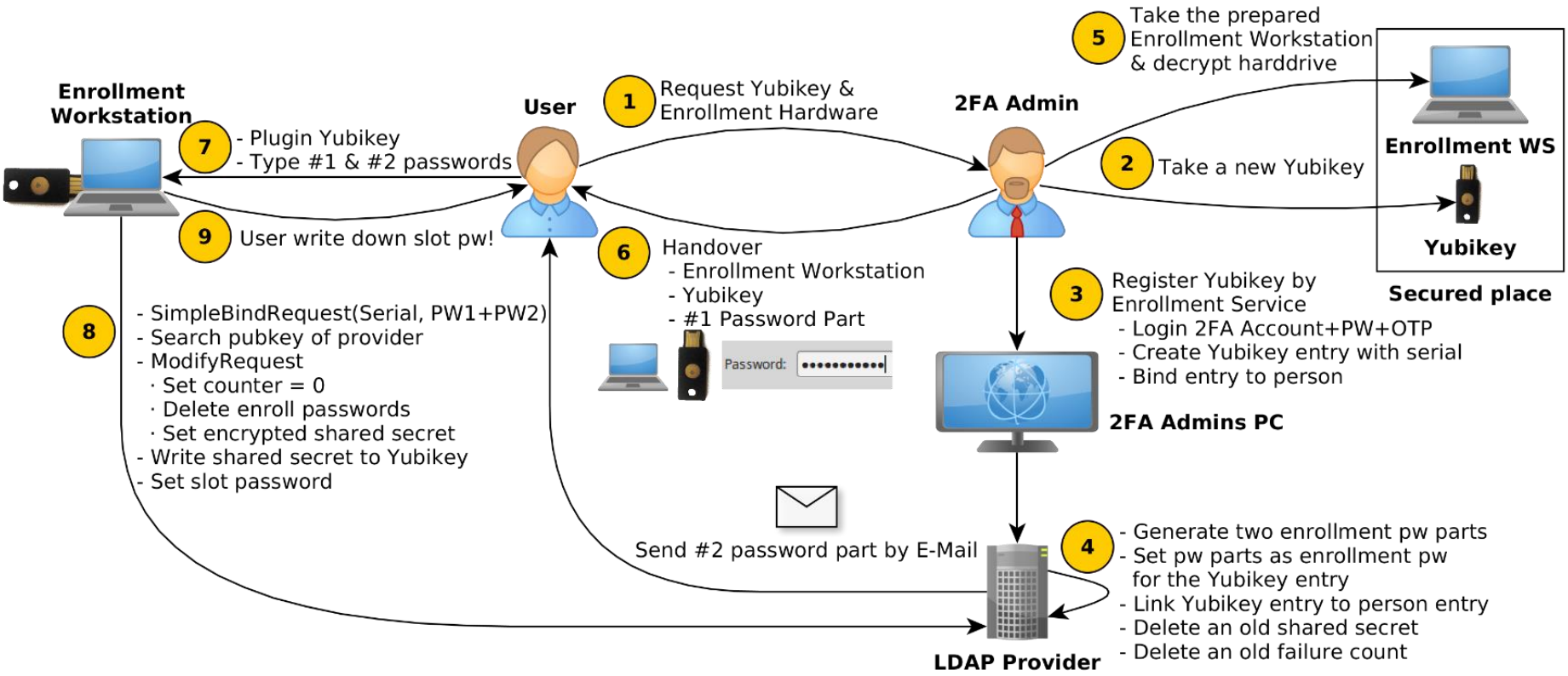
Implementation – Simple Bind Proxy

- Listens to Unix domain socket on consumer
- Back-sock overlay in consumer slapd forwards simple bind requests to Unix domain socket
- Checks if user is HOTP user by requesting consumer slap d over LDAPI
- Forwards password & OTP to providers slapd by LDAPS
- A hash of the bind-DN is used as base to forward bind request every time to the same providers
  - This prevents multi-master replication conflicts
  - But allows load balancing

Implementation – OTP Validator

- Listens to Unix domain socket on consumer
- Back-sock overlay in provider slapd forwards simple bind requests to Unix domain socket
- Checks if user is HOTP user by requesting provider slapd over LDAPI
- Checks password & OTP by reading HOTP
  - counter of HOTP token entry
  - Password hash of users entry
  - Password & OTP policy
  - Decrypting shared secret of users entry
- If OTP is valid, new HOTP counter must be set
  - Value is the counter which led to valid OTP
  - Set a new value to provider is not trivial
  - Insertion with check if new value is greater

Token Enrollment – Token Enrollment Procedure



Token Enrollment – Yubikey Tools and Libraries

- **Graphical Tools**
- *yubikey-personalization-gui*
  - set OTP, challenge-response, static password, NFC, options
  - get serial & firmware version, usb manufacturer&device id
- *neoman*
  - set/get usb mode, (un)installing/show Java Card Applets
  - rename Yubikey Neo, get serial&firmware
- **Command Line Tools**
- *ykpersonalize*
  - cli version of yubikey-personalization-gui + usb mode
- *ykeyneomgr*
  - cli version of neoman + list SmartCard readers, serial&firmware
- *ykinfo*
  - check if slots programmed, get serial&firmware version
- **Libraries**
- *python-yubico*
  - Python, in development, features like yubikey-personalization
- *yubikey-personalization*
  - C, very mature, features like yubikey-personalization, flags

Token Enrollment – Enrollment Service I

- *python-yubico* library was used
- Not a chaotic shell script which calls cli commands
- Runs on hardened dedicated enrollment hardware
- Binds Yubikey to HOTP token entry
- Ensures the secure handling of shared secrets
- Disables all non-used features to prevent attacks by unknown issues and side-channels

Token Enrollment – Enrollment Service II

- Script sequence:
  1. Clear both slots incl. passwords
  2. Read Yubikey serial
  3. Login into LDAP by Yubikey serial and enroll pw
  4. Set USB mode to HID only, disable SmartCard
  5. Set mode of slot 1 to HOTP and write secret
  6. Protect both slots with a user defined password
  7. Write shared secret to LDAP and set counter 0
  8. Switch NFC to unused slot 2

Future Extensions

- SmartCard function by *ykneo-openpgp* applet
  - Extend enrollment service & hw by pgp-agent
  - Enrollment triggers Yubikey to generate PGP key
  - PGP pubkey will be bound token entry
  - PGP key is used instead of SSH key on users workstation
- TOTP hardware token with HID function
  - At the moment there isn't such device
  - A real time clock is needed in token
  - *Problem:* How to synchronise the clock?
  - TOTP is easier to implement on backend side
  - No need to write and synchronise a counter
  - HOTP is easier on token side

End of Presentation

Are there any questions?