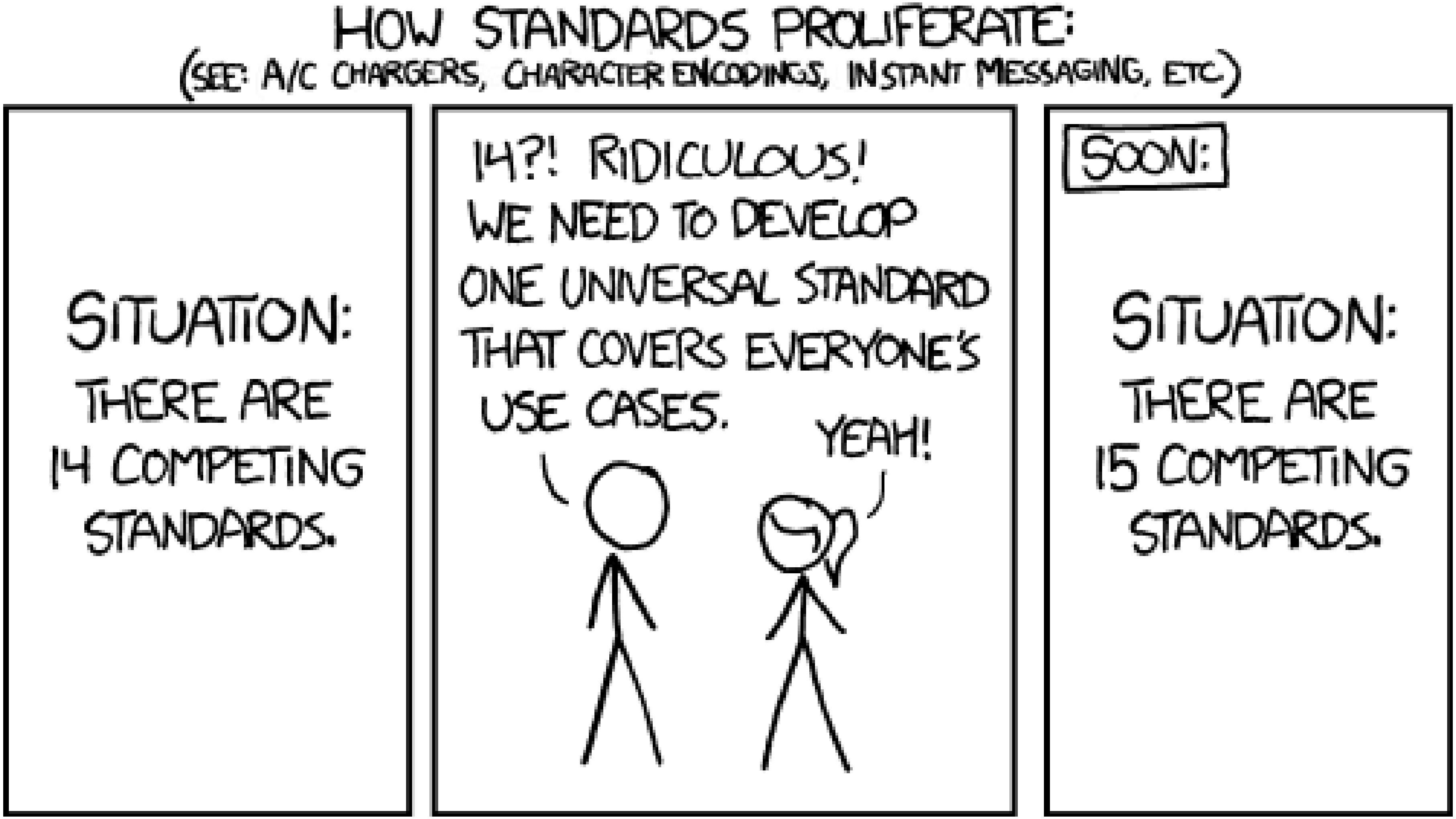
The Authentication Jungle

An overview of all sorts of authentication technologies

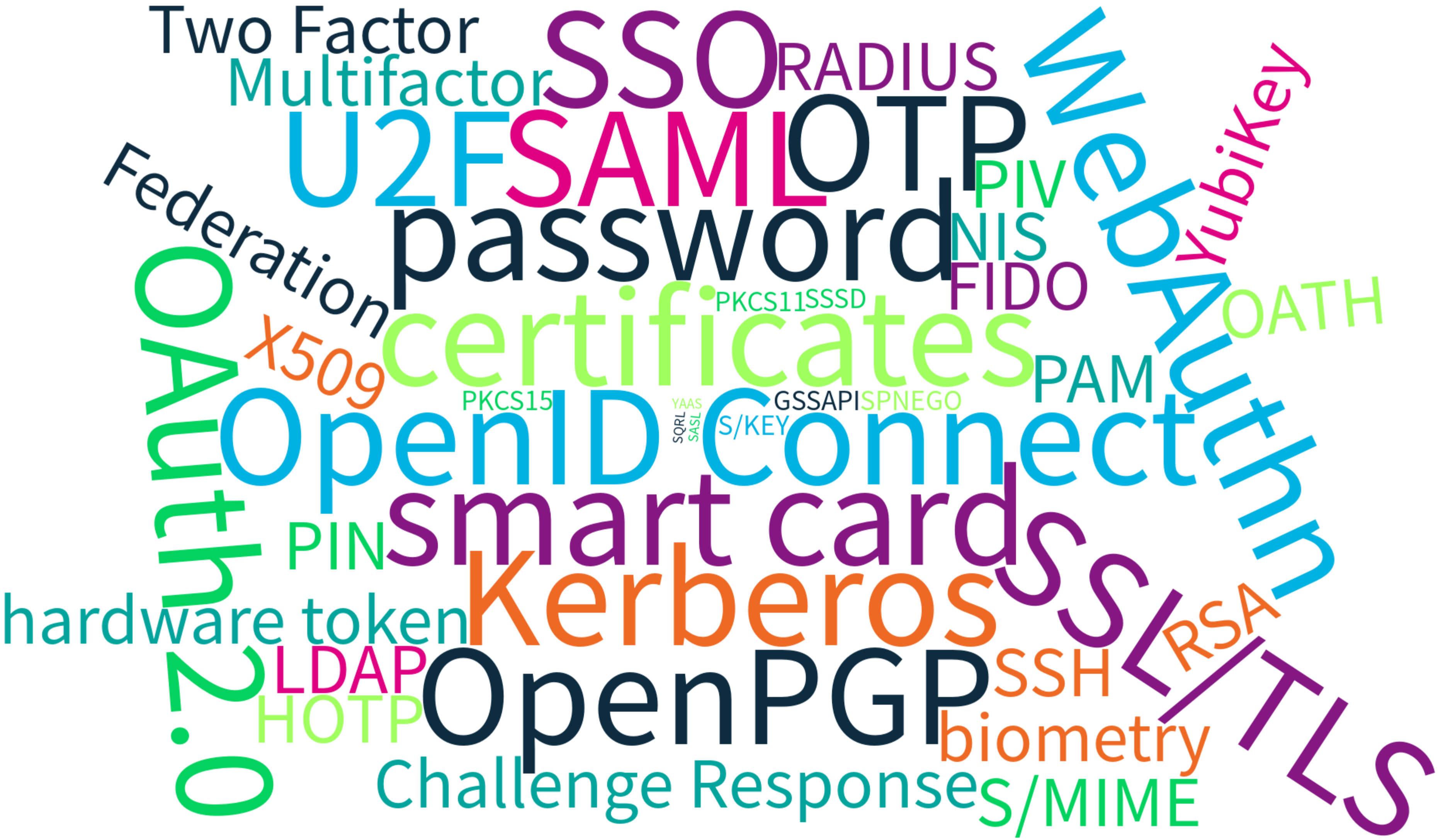
Karol Babioch
Security Engineer
kbabioch@suse.de

New authentication standards ...



HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

Some authentication technologies ...



Two Factor
SSO
RADIUS
WebAuthn
Multifactor
U2F
SAML
OTP
Yubikey
Federation
PIV
password
NIS
OAuth 2.0
FIDO
OATH
PKCS11
SSSD
X509
certificates
PAM
PKCS15
YAAS
SQRL
SASL
S/KEY
GSSAPI
SPNEGO
OpenID Connect
PIN
smart card
SSL/TLS
hardware token
Kerberos
RSA
LDAP
OpenPGP
SSH
HOTP
biometry
Challenge Response
S/MIME

- Authentication theory
- “Simple” authentication schemes
- Centralized authentication schemes
- Federated authentication schemes
- Conclusion

Karol Babioch
Security Engineer
kbabioch@suse.de

Authentication theory

Karol Babioch
Security Engineer
kbabioch@suse.de

What is authentication?

“[…] the act of confirming the truth of an attribute of a single piece of data [...]”

(Wikipedia)

In our context: Mostly concerned about user authentication

→ *Who am I communicating with?*

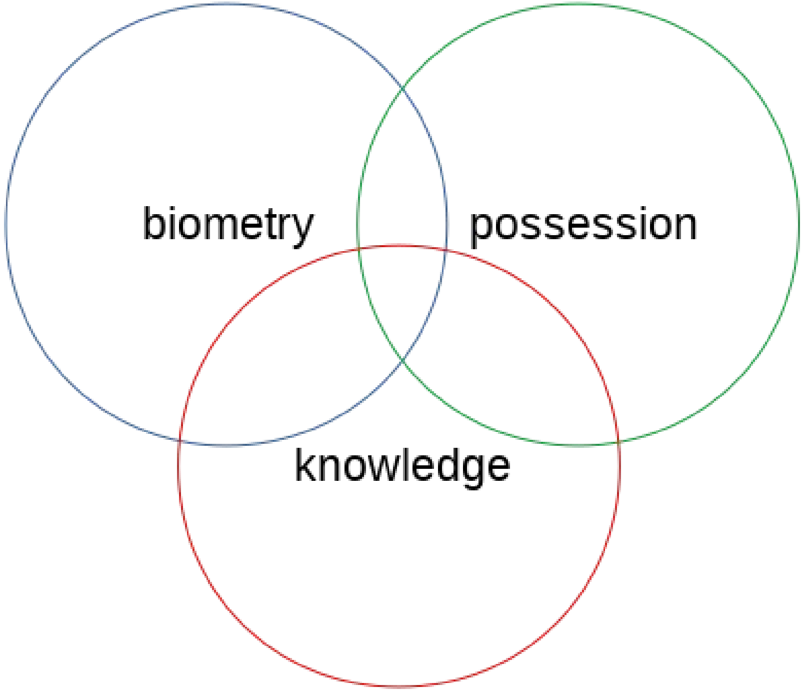
Attributes for authentication

- **Something you know**
  - Secrets (Password, PIN, code, etc.)

- **Something you have**
  - Physical keys
  - Hardware tokens (Smart card, YubiKey, etc.)

  → *Should be difficult to clone*

- **Something you are**
  - Fingerprint
  - Iris
  - Face recognition



Challenges for authentication technologies

- Security

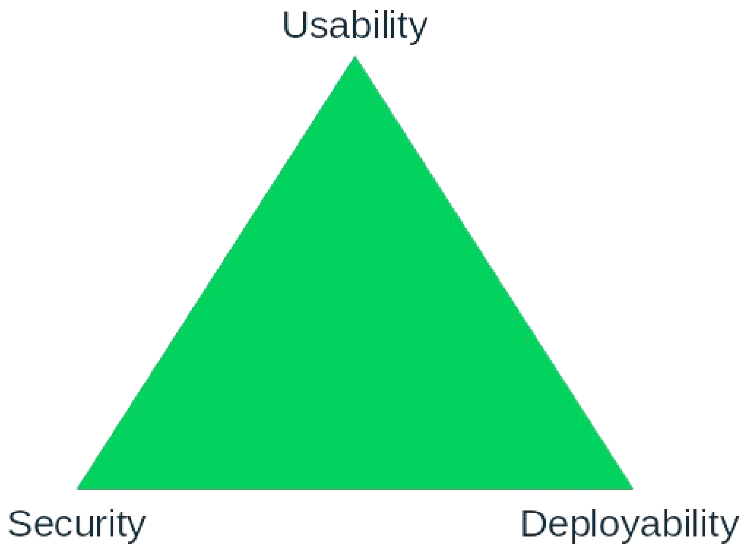
- Resiliency to guessing (brute force, online, offline)
- Resiliency to phishing
- Resiliency to theft
- Resiliency to physical observation
- Resiliency to internal observation
- No trusted third parties
- Explicit user-consent
- Unlinkability

- Usability

- Memorywise effortless
- Scalable for users
- Nothing to carry
- Easy recovery from loss

- Deployability

- Cost per user
- Server compatible
- Browser compatible
- Maturity
- Non proprietary



Authentication vs. Authorization

Authentication (AuthN, A1, Au)

→ *Who am I communicating with?*

Authorization (AuthZ, AuthR, A2, Az)

→ *What am I allowed to do?*

→ **Most of the time: Tightly coupled**

“Simple” authentication schemes

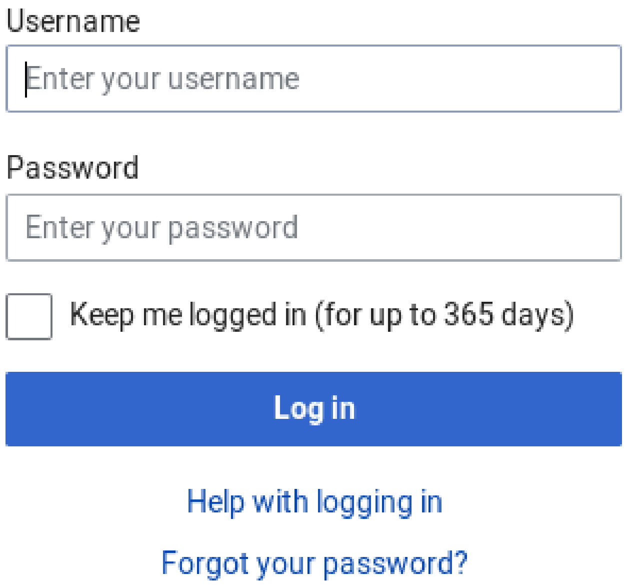
Karol Babioch
Security Engineer
kbabioch@suse.de

Passwords

Karol Babioch
Security Engineer
kbabioch@suse.de

Password-based logins

- Apparently simple to use
- Apparently easy to implement (“string compare”)
- Universal across all domains/contexts
- Recommendations & best practices (NIST, etc.)



Username

Enter your username

Password

Enter your password

□ Keep me logged in (for up to 365 days)

Problems with passwords

- Weak passwords
- Re-usage across different domains/contexts
- Phishing
- Static
- Breaches
- User’s responsibility
  - Chocolate study
  - Easy to remember = Easy to guess

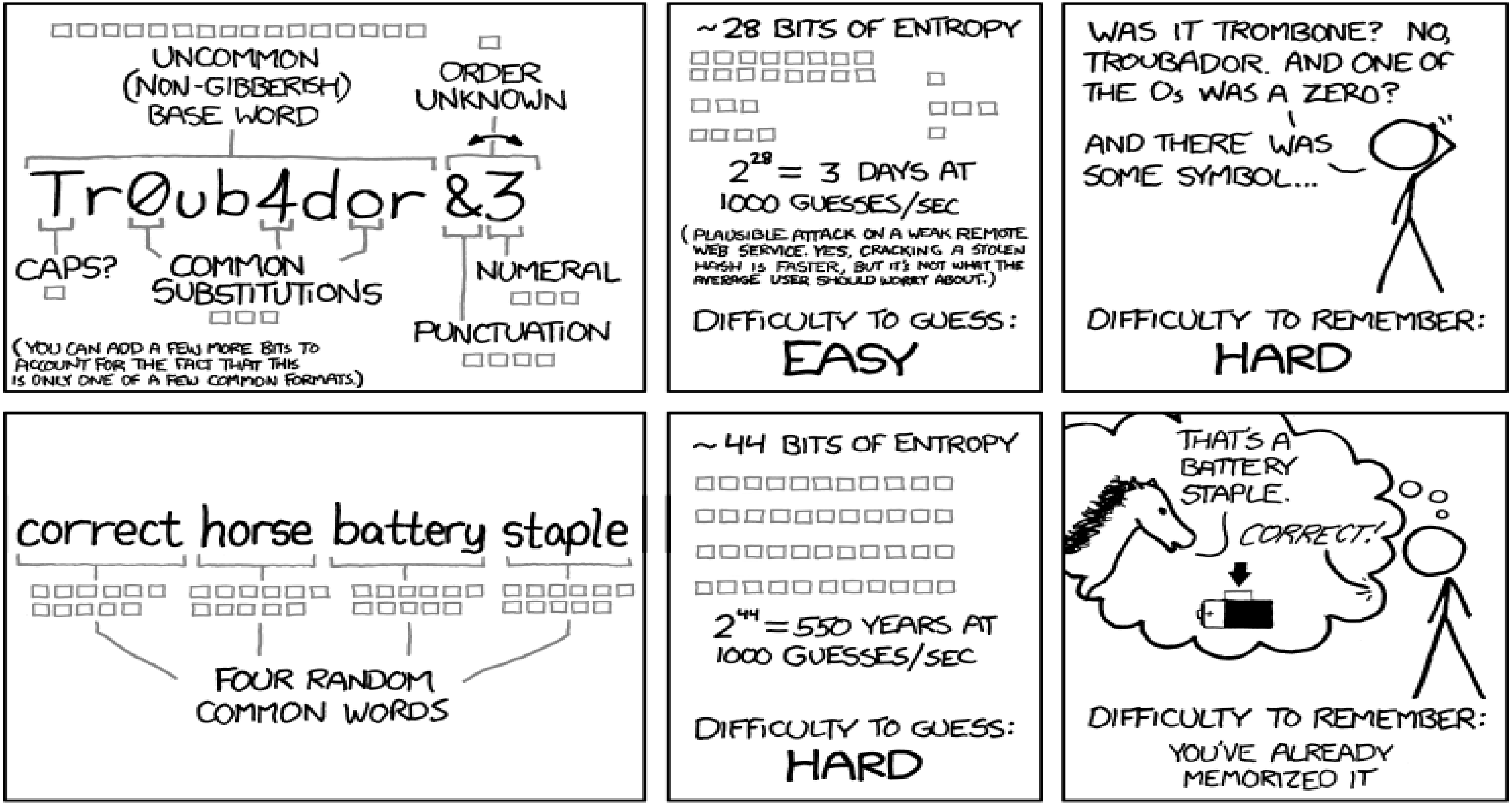
Experts get it wrong

- NIST Special Publication 800-63. Appendix A
  - Originally from 2003
  - Based on no real data (not available)
  - Expiration after x days
  - No re-usage of last x passwords
  - Different character classes: Special character, numbers, big and small caps
  - Example: P@ssW0rd123!

→Users still choose easy-to-guess passwords

- – Less entropy than expected
- – Regular changes bad idea
  - Stolen credentials are used right away (not after x days)
  - weak passwords
  - Workaround: password1 → password2 → password3 → password1

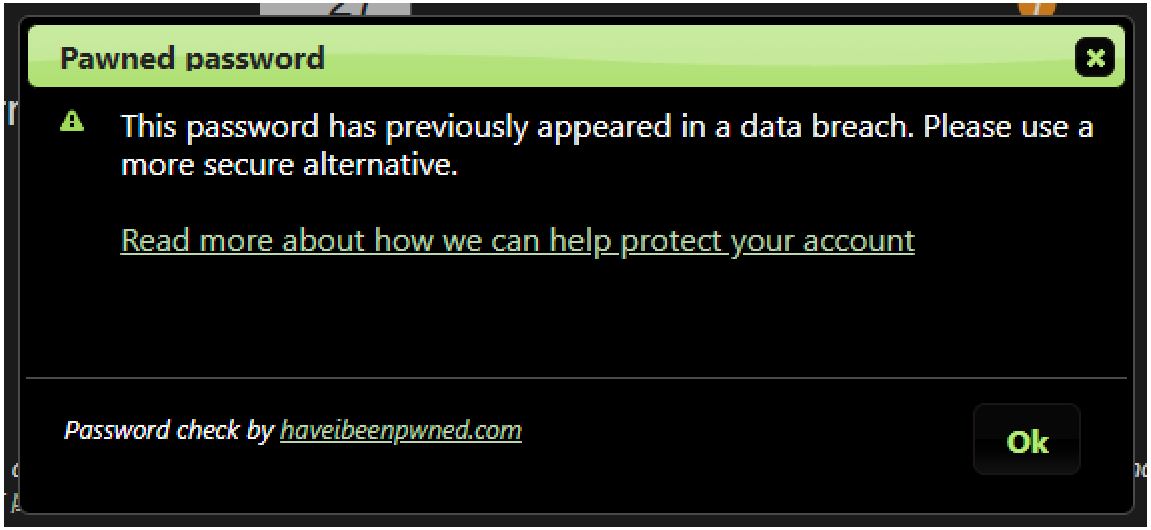
Fun with password strength


UNCOMMON (NON-GIBBERISH) BASE WORD
ORDER UNKNOWN
Tr0ub4dor&3
CAPS?
COMMON SUBSTITUTIONS
NUMERAL
PUNCTUATION
(YOU CAN ADD A FEW MORE BITS TO ACCOUNT FOR THE FACT THAT THIS IS ONLY ONE OF A FEW COMMON FORMATS.)
~28 BITS OF ENTROPY
2^28 = 3 DAYS AT 1000 GUESSES/SEC
(PLAUSIBLE ATTACK ON A WEAK REMOTE WEB SERVICE. YES, CRACKING A STOLEN HASH IS FASTER, BUT IT'S NOT WHAT THE AVERAGE USER SHOULD WORRY ABOUT.)
DIFFICULTY TO GUESS: EASY
WAS IT TROMBONE? NO, TROUBADOR. AND ONE OF THE Os WAS A ZERO?
AND THERE WAS SOME SYMBOL...
DIFFICULTY TO REMEMBER: HARD
correct horse battery staple
FOUR RANDOM COMMON WORDS
~44 BITS OF ENTROPY
2^44 = 550 YEARS AT 1000 GUESSES/SEC
DIFFICULTY TO GUESS: HARD
THAT'S A BATTERY STAPLE.
CORRECT!
DIFFICULTY TO REMEMBER: YOU'VE ALREADY MEMORIZED IT

THROUGH 20 YEARS OF EFFORT, WE'VE SUCCESSFULLY TRAINED EVERYONE TO USE PASSWORDS THAT ARE HARD FOR HUMANS TO REMEMBER, BUT EASY FOR COMPUTERS TO GUESS.

haveibeenpwned.com

- One (of many) password databases based on dumps (> 500 million passwords)
- Search for your account in existing dumps
- Notify when account appears in new dumps
- API / datasets for querying passwords (k-anonymity)
- Should be checked during account creation / password change



Pawned password

This password has previously appeared in a data breach. Please use a more secure alternative.

Read more about how we can help protect your account

Password check by haveibeenpwned.com

Mitigations

- Pro-active password checks during account creation and password changes
- Re-active leak monitoring (i.e. haveibeenpwned.com):
  - Single accounts
  - Whole domain
- Use and encourage password manager
- No annoying limitations for passwords
- Multifactor authentication

- Other authentication schemes
  - Single-Sign-On & Federation

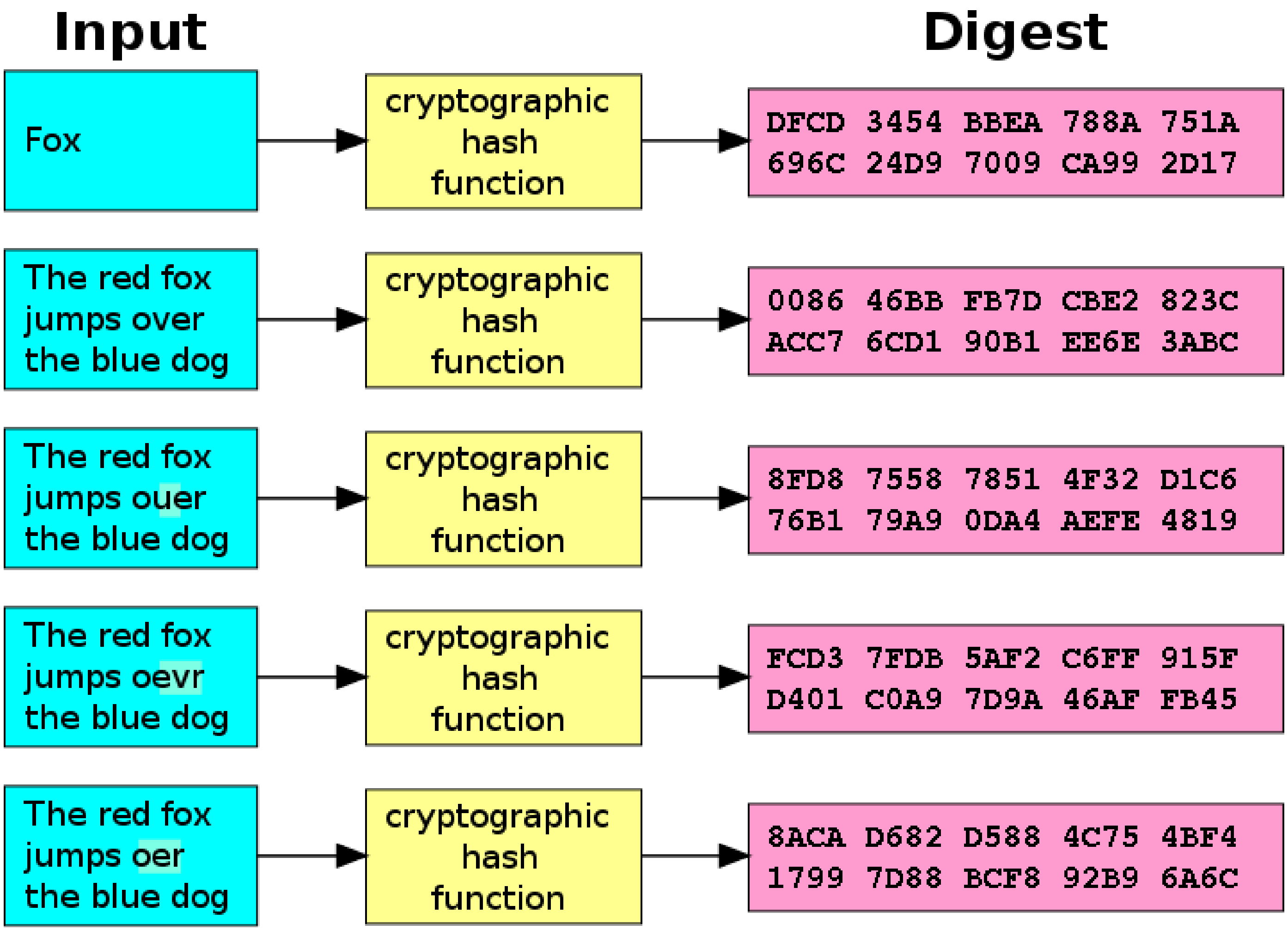
Crypto 101

Karol Babioch
Security Engineer
kbabioch@suse.de

Crypto 101: Cryptographic hash functions

- Returns a (fixed-size) output (“hash-value”) for any input
  - Easy to calculate the hash value value for any given data
  - Computationally difficult to calculate an input with a given hash value
  - Unlikely that two (slightly) different messages have the same hash value
- H(message) → output
- Examples
  - SHA1 (e.g. git)
  - SHA2 (256, 384, 512)
  - SHA3
  - MD5
  - MD4
- Use cases
  - Message integrity
  - Digital signatures
  - Authentication

Crypto 101: Cryptographic hash functions



Input

Digest

Fox → cryptographic hash function → DFCD 3454 BBEA 788A 751A 696C 24D9 7009 CA99 2D17

The red fox jumps over the blue dog → cryptographic hash function → 0086 46BB FB7D CBE2 823C ACC7 6CD1 90B1 EE6E 3ABC

The red fox jumps ouer the blue dog → cryptographic hash function → 8FD8 7558 7851 4F32 D1C6 76B1 79A9 0DA4 AEFE 4819

The red fox jumps oevr the blue dog → cryptographic hash function → FCD3 7FDB 5AF2 C6FF 915F D401 C0A9 7D9A 46AF FB45

The red fox jumps oer the blue dog → cryptographic hash function → 8ACA D682 D588 4C75 4BF4 1799 7D88 BCF8 92B9 6A6C

Crypto 101: HMAC

- Hash-based message authentication code
- Defined in RFC2104
- Any cryptographic hash function can be used
- HMAC(secret, message) → output [hash]

- Examples
  - HMAC-MD5
  - HMAC-SHA256
  - HMAC-SHA3

- Use cases
  - data integrity
  - authentication

Multifactor authentication

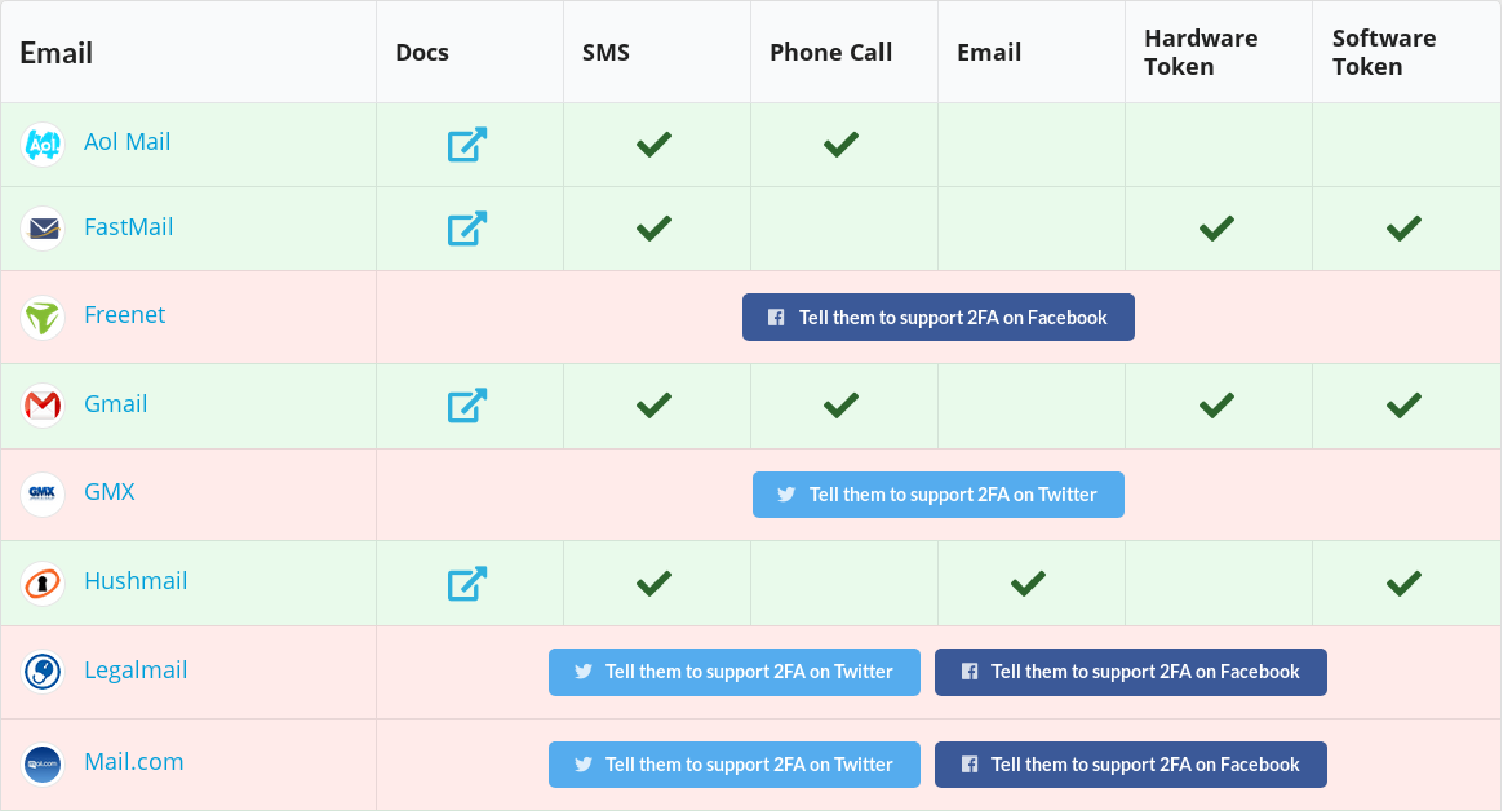
Karol Babioch
Security Engineer
kbabioch@suse.de

Multifactor authentication to the rescue

- Basic idea: Use multiple factors for authentication (passwords is not sufficient)
  - 2FA = Two-factor authentication
  - MFA = Multi-factor authentication
  - Examples:
    - One-Time passwords (OTP)
    - Chip & TAN
    - password & certificate (OpenVPN, etc.)

- Different channels:
  - SMS
  - Smart card (chipTAN)
  - (Smartphone) apps
  - Different devices (Notifications from Google on Android, etc.)
  - Hardware tokens (RSA SecurID, YubiKey, U2F, etc.)

twofactorauth.org



Email	Docs	SMS	Phone Call	Email	Hardware Token	Software Token
Aol Mail	↗	✓	✓			
FastMail	↗	✓			✓	✓
Freenet			Tell them to support 2FA on Facebook			
Gmail	↗	✓	✓		✓	✓
GMX			Tell them to support 2FA on Twitter			
Hushmail	↗	✓		✓		✓
Legalmail		Tell them to support 2FA on Twitter		Tell them to support 2FA on Facebook		
Mail.com		Tell them to support 2FA on Twitter		Tell them to support 2FA on Facebook		

OATH: TOTP & HOTP

- Standardized by OATH (!= OAuth)
- Many software implementations & hardware tokens

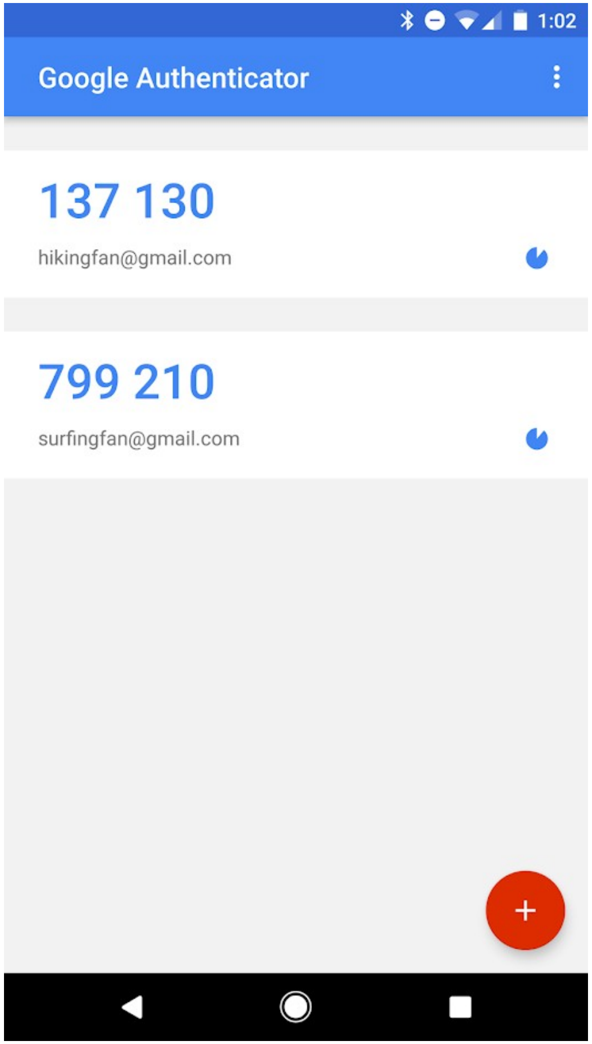
- Requires initial setup to establish shared secret between provider and user
    - e.g. QR code

- TOTP: Time-based OTP
    - Code: HMAC(sharedSecret, timestamp)

- HOTP: Event-based OTP
    - Code: HMAC(sharedSecret, counter)

Soft-token implementations

otpauth://totp/label?secret=secret&issuer=issuer



137 130
hikingfan@gmail.com

799 210
surfingfan@gmail.com

Hardware OTP tokens

- Shared secret is stored in hardware
  → Cannot be duplicated

- Requires enrollment process

- More on hardware tokens → second talk

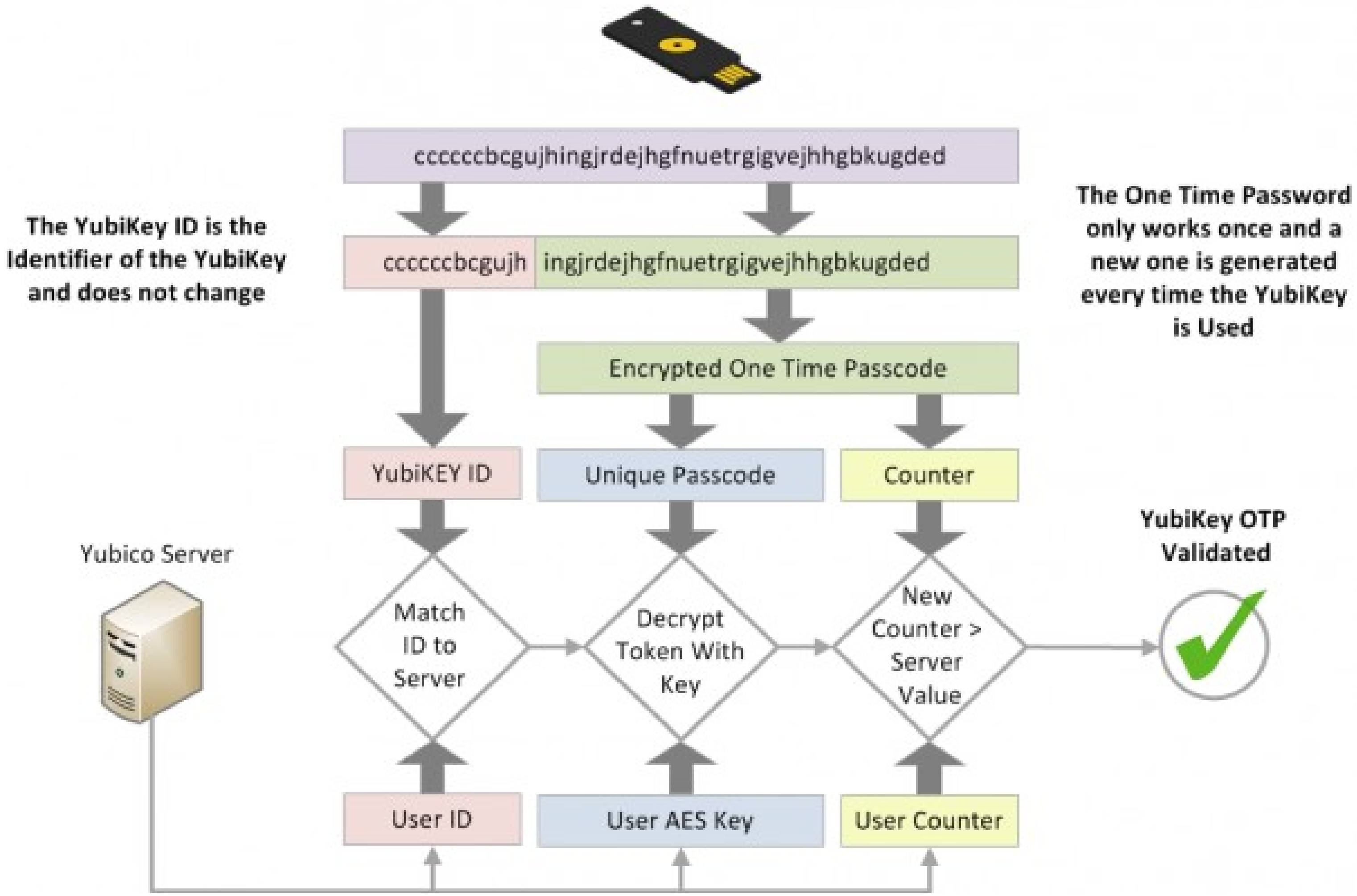
Yubico OTP

- Hardware token with USB interface
- Emulating USB keyboard
- Multiple slots
  - Short push (~ 0.5 sec)
  - Long push (~ 2 sec)
- Push button → User consent

- Supports OATH
  - HOTP
  - TOTP (requires software on host)
  - Yubico OTP

- Many other modes of operation → second talk

Yubico OTP explanation


ccccccbcgujhingjrdejhgfnuetrgigvejhhgbkugded
The YubiKey ID is the Identifier of the YubiKey and does not change
ccccccbcgujh
ingjrdejhgfnuetrgigvejhhgbkugded
The One Time Password only works once and a new one is generated every time the YubiKey is Used
Encrypted One Time Passcode
YubiKEY ID
Unique Passcode
Counter
YubiKey OTP Validated
Yubico Server
Match ID to Server
Decrypt Token With Key
New Counter > Server Value
User ID
User AES Key
User Counter

Problems with multifactor authentication

- Based on shared secret
  → Still something to loose (data breach)
- Trusted third party (in case of RSA, Yubico OTP, etc.)
- Broken fallback routines / recovery processes
- Inconvenient (i.e. smartphone not available, etc.)
- No inherent MitM protection (active attacks, phishing, session hijacking)
- Scales badly
  - Requires setup for each service
  - Requires dedicated key / slot for each service
  - Cost per device

Crypto 101

Karol Babioch
Security Engineer
kbabioch@suse.de

Crypto 101: Symmetric cryptography

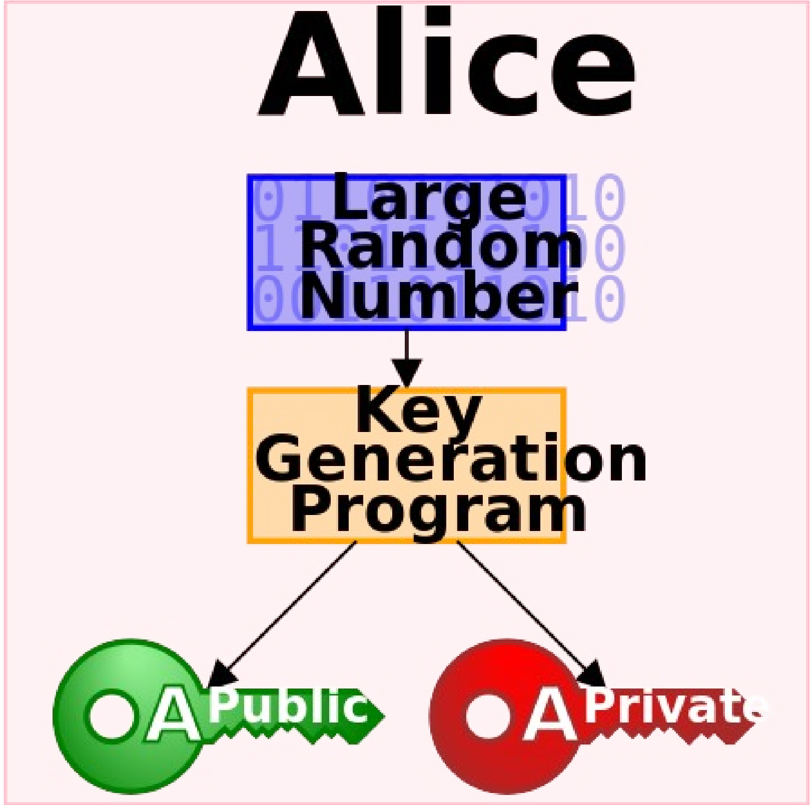
- Encryption and decryption are using the same secret (key)
- Examples:
  - AES
  - DES, 3DES
  - Blowfish
  - Twofish
  - RC4

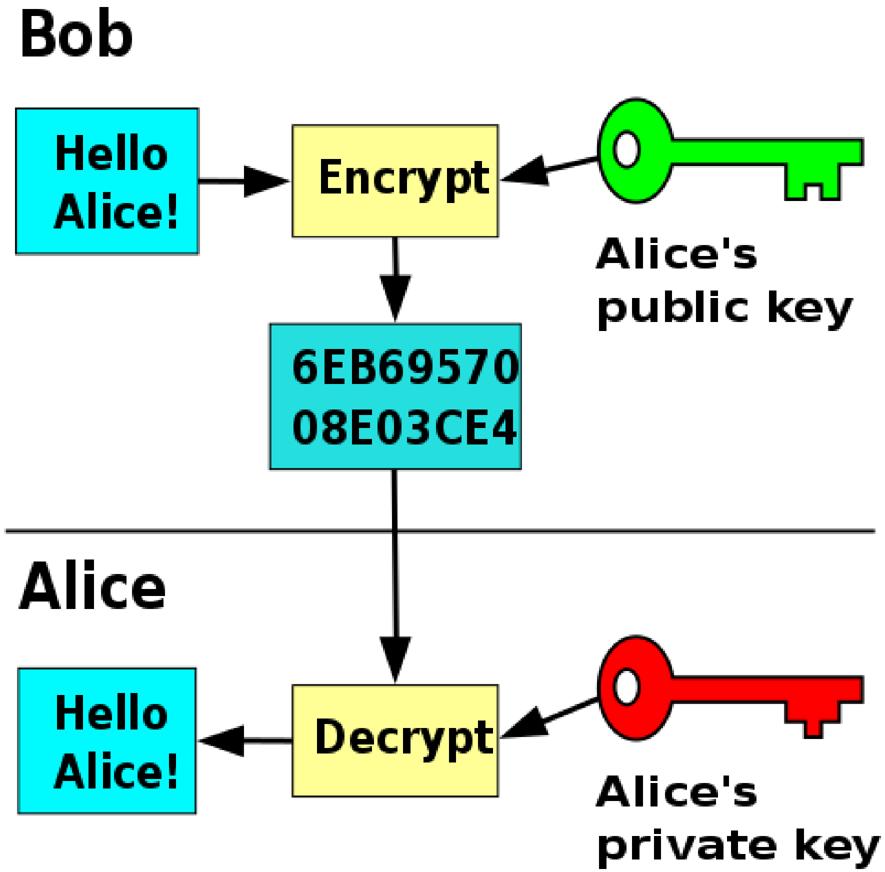
- Block cipher modes:
  - ECB
  - CBC
  - OFB
  - XTS

Crypto 101: Asymmetric Cryptography

- Two keys (referred to as a key pair)
  - Public
  - Private
- Examples:
  - RSA
  - DH (Diffie Hellman)
  - ECC (Elliptic Curve Cryptography)
- Use cases
  - Encryption
  - Authentication
  - Key agreement
  - Signatures
  - Verification
- **Challenge**: Key exchange, authenticity of public keys

Crypto 101: Asymmetric Cryptography





SSL/TLS (X509)

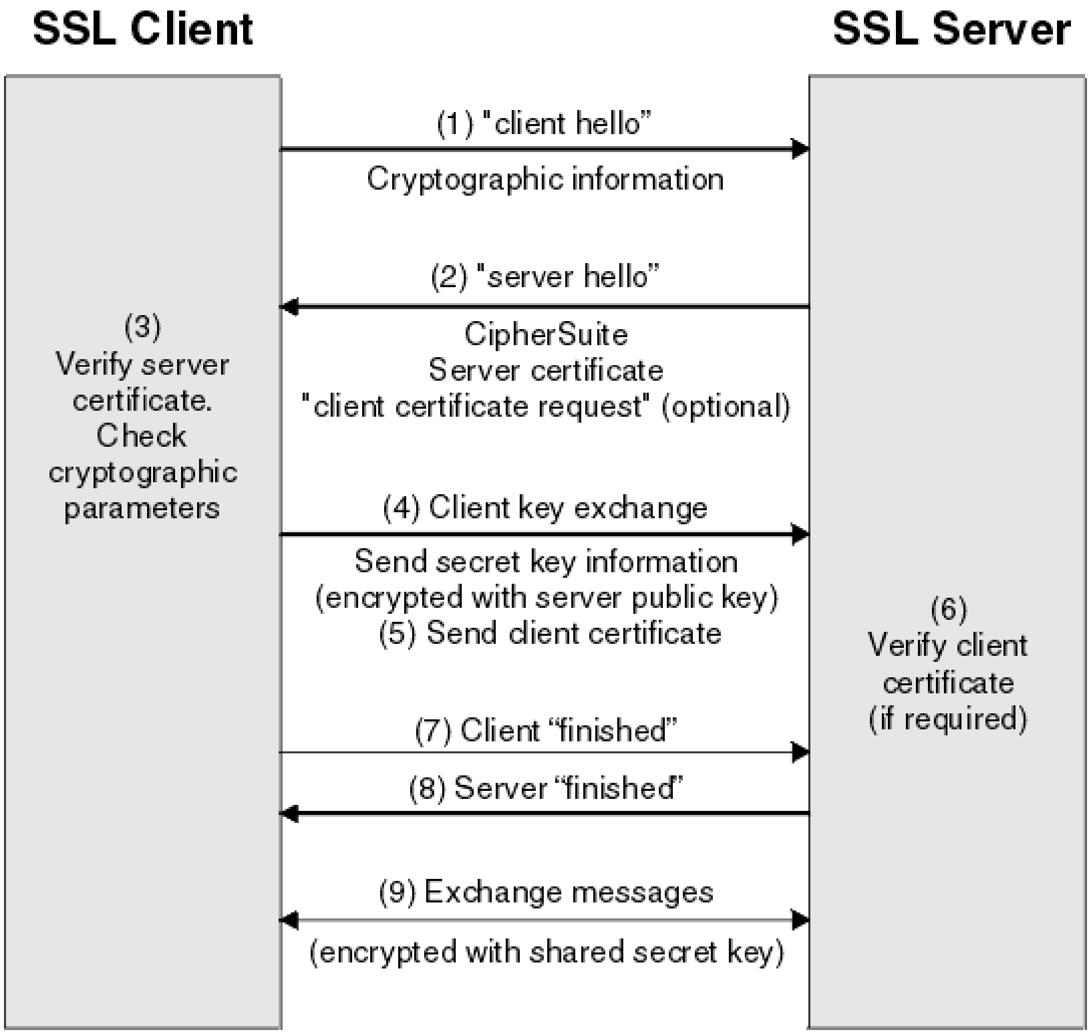
Karol Babioch
Security Engineer
kbabioch@suse.de

SSL/TLS basics

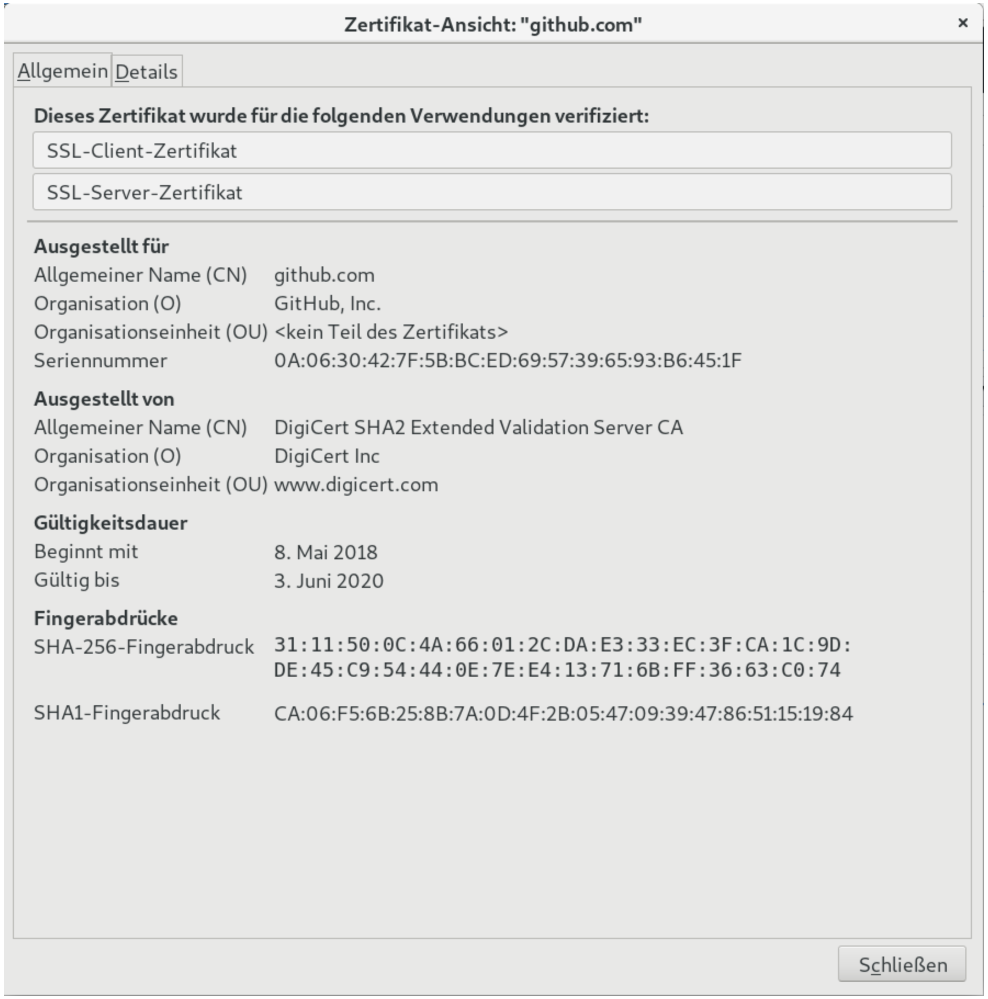
- Prevalent throughout the Internet
- Can basically be used with all protocols (https, ldaps, imaps, etc.)
- Provides confidentiality, integrity, authentication
- Mostly: One-way authentication (Browser)
- Chain of trust: Certificate authority (CA) → … (intermediate CA) … → certificate
- PKI: Public-key infrastructure

- **Interesting to us: Client certificates**
  - Can be offloaded to hardware → Second talk

SSL/TLS handshake



Server certificates



Zertifikat-Ansicht: "github.com"

Dieses Zertifikat wurde für die folgenden Verwendungen verifiziert:

SSL-Client-Zertifikat

SSL-Server-Zertifikat

Ausgestellt für

Allgemeiner Name (CN)	github.com
Organisation (O)	GitHub, Inc.
Organisationseinheit (OU)	<kein Teil des Zertifikats>
Seriennummer	0A:06:30:42:7F:5B:BC:ED:69:57:39:65:93:B6:45:1F

Ausgestellt von

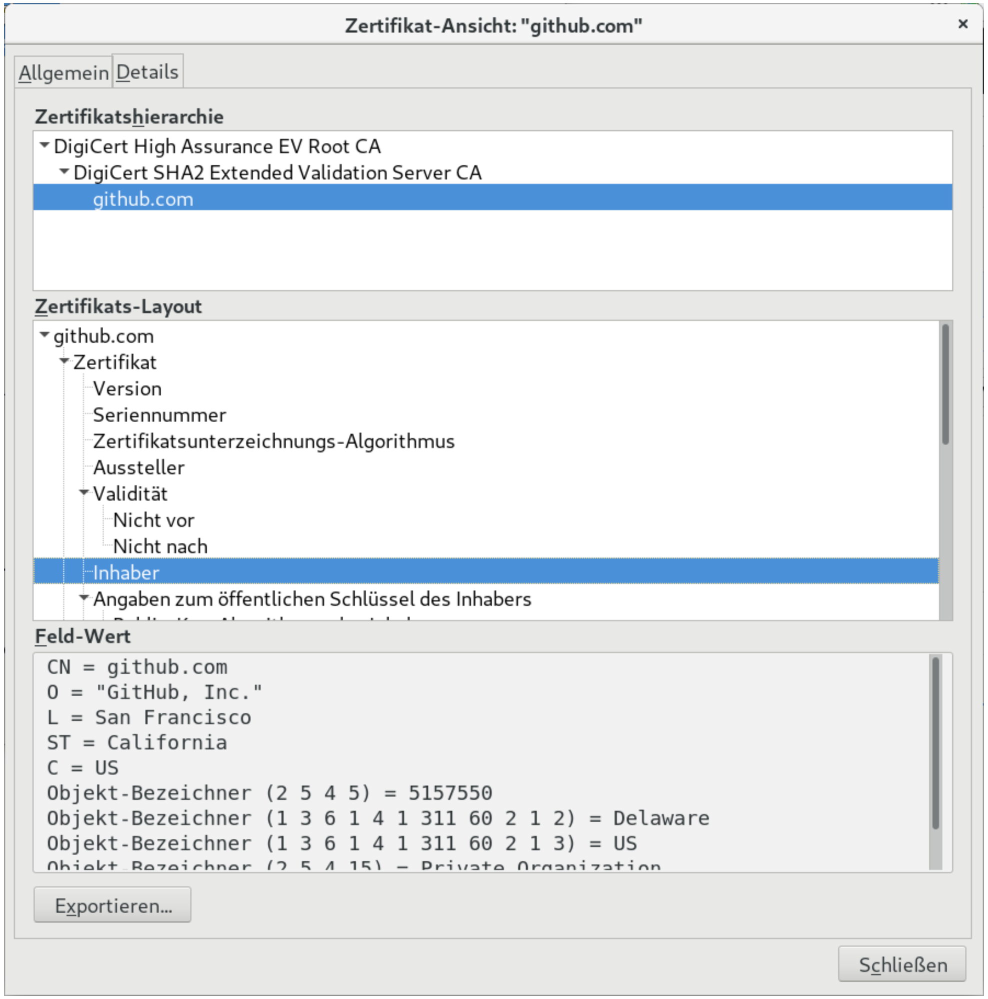
Allgemeiner Name (CN)	DigiCert SHA2 Extended Validation Server CA
Organisation (O)	DigiCert Inc
Organisationseinheit (OU)	www.digicert.com

Gültigkeitsdauer

Beginnt mit	8. Mai 2018
Gültig bis	3. Juni 2020

Fingerabdrücke

SHA-256-Fingerabdruck	31:11:50:0C:4A:66:01:2C:DA:E3:33:EC:3F:CA:1C:9D: DE:45:C9:54:44:0E:7E:E4:13:71:6B:FF:36:63:C0:74
SHA1-Fingerabdruck	CA:06:F5:6B:25:8B:7A:0D:4F:2B:05:47:09:39:47:86:51:15:19:84



Zertifikat-Ansicht: "github.com"

Zertifikatshierarchie

- DigiCert High Assurance EV Root CA
  - DigiCert SHA2 Extended Validation Server CA
    - github.com

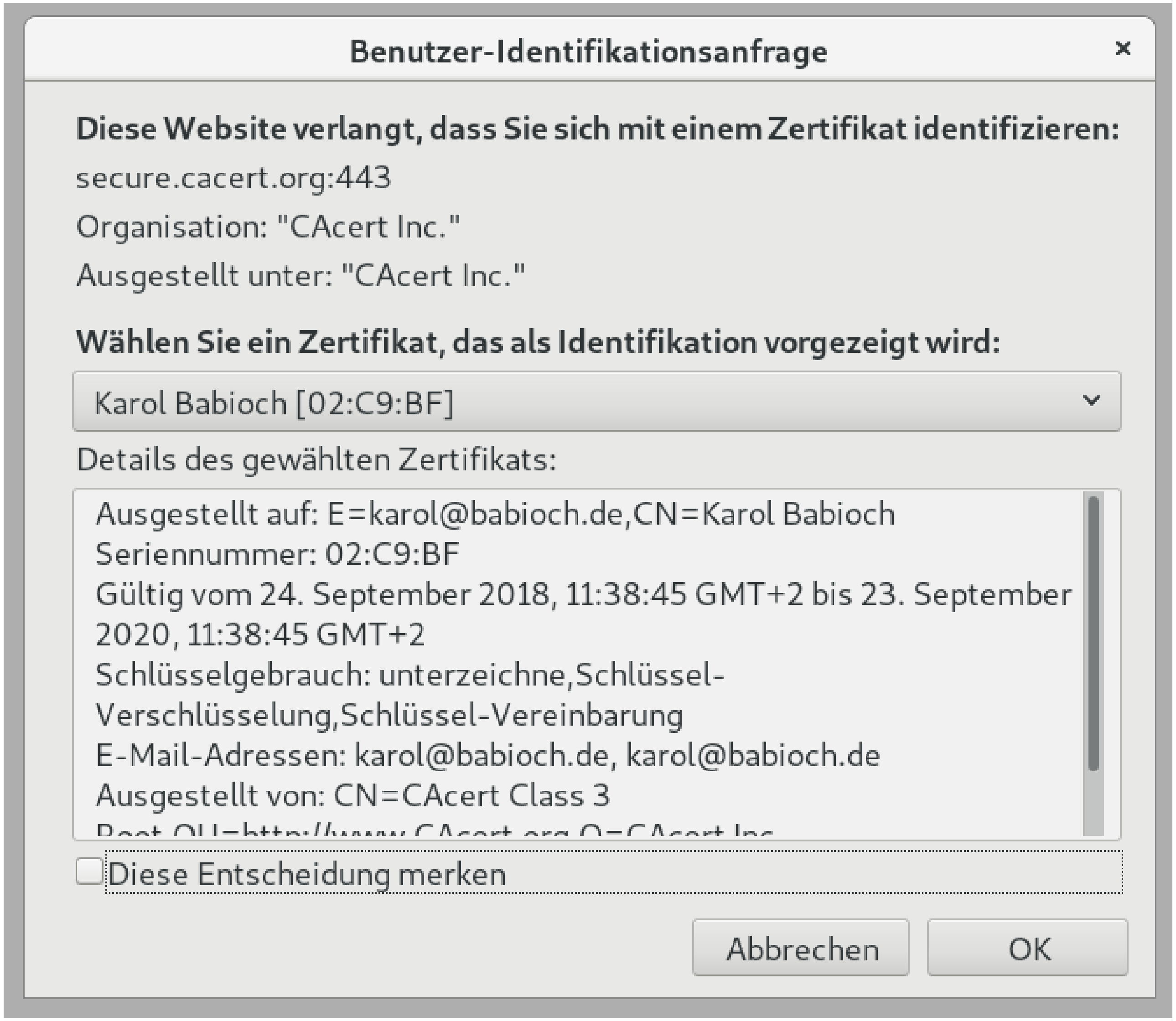
Zertifikats-Layout

- github.com
  - Zertifikat
    - Version
    - Seriennummer
    - Zertifikatsunterzeichnungs-Algorithmus
    - Aussteller
    - Validität
      - Nicht vor
      - Nicht nach
    - Inhaber
    - Angaben zum öffentlichen Schlüssel des Inhabers
      - Public-Key-Algorithmus des Inhabers

Feld-Wert

```
CN = github.com
O = "GitHub, Inc."
L = San Francisco
ST = California
C = US
Objekt-Bezeichner (2 5 4 5) = 5157550
Objekt-Bezeichner (1 3 6 1 4 1 311 60 2 1 2) = Delaware
Objekt-Bezeichner (1 3 6 1 4 1 311 60 2 1 3) = US
Objekt-Bezeichner (2 5 4 15) = Private Organization
```

Client certificates



Diese Website verlangt, dass Sie sich mit einem Zertifikat identifizieren:

secure.cacert.org:443

Organisation: "CAcert Inc."

Ausgestellt unter: "CAcert Inc."

Wählen Sie ein Zertifikat, das als Identifikation vorgezeigt wird:

Karol Babioch [02:C9:BF]

Details des gewählten Zertifikats:

Ausgestellt auf: E=karol@babioch.de,CN=Karol Babioch
Seriennummer: 02:C9:BF
Gültig vom 24. September 2018, 11:38:45 GMT+2 bis 23. September 2020, 11:38:45 GMT+2
Schlüsselgebrauch: unterzeichne,Schlüssel-Verschlüsselung,Schlüssel-Vereinbarung
E-Mail-Adressen: karol@babioch.de, karol@babioch.de
Ausgestellt von: CN=CAcert Class 3
Root,OU=http://www.CAcert.org,O=CAcert Inc.

☐ Diese Entscheidung merken

Certificates

- Many attributes
  - Valid before
  - Valid after
  - Common name
  - Public key
  - Issuer
  - ...

- Binding between key pair and an identity

Problems with SSL/TLS

- General SSL/TLS criticism
  - Trusted Third Party → Every CA can sign anything
  - Broken revocation
  - Key pinning challenging
  - etc., pp.

- Specific to client certificates
  - Support for client certificates (applications, protocols, etc.)
  - Verification of client certificates
  - Handling certificates correctly is challenging
  - Roll your own CA?
  - Privacy concerns (→TLS 1.3?)

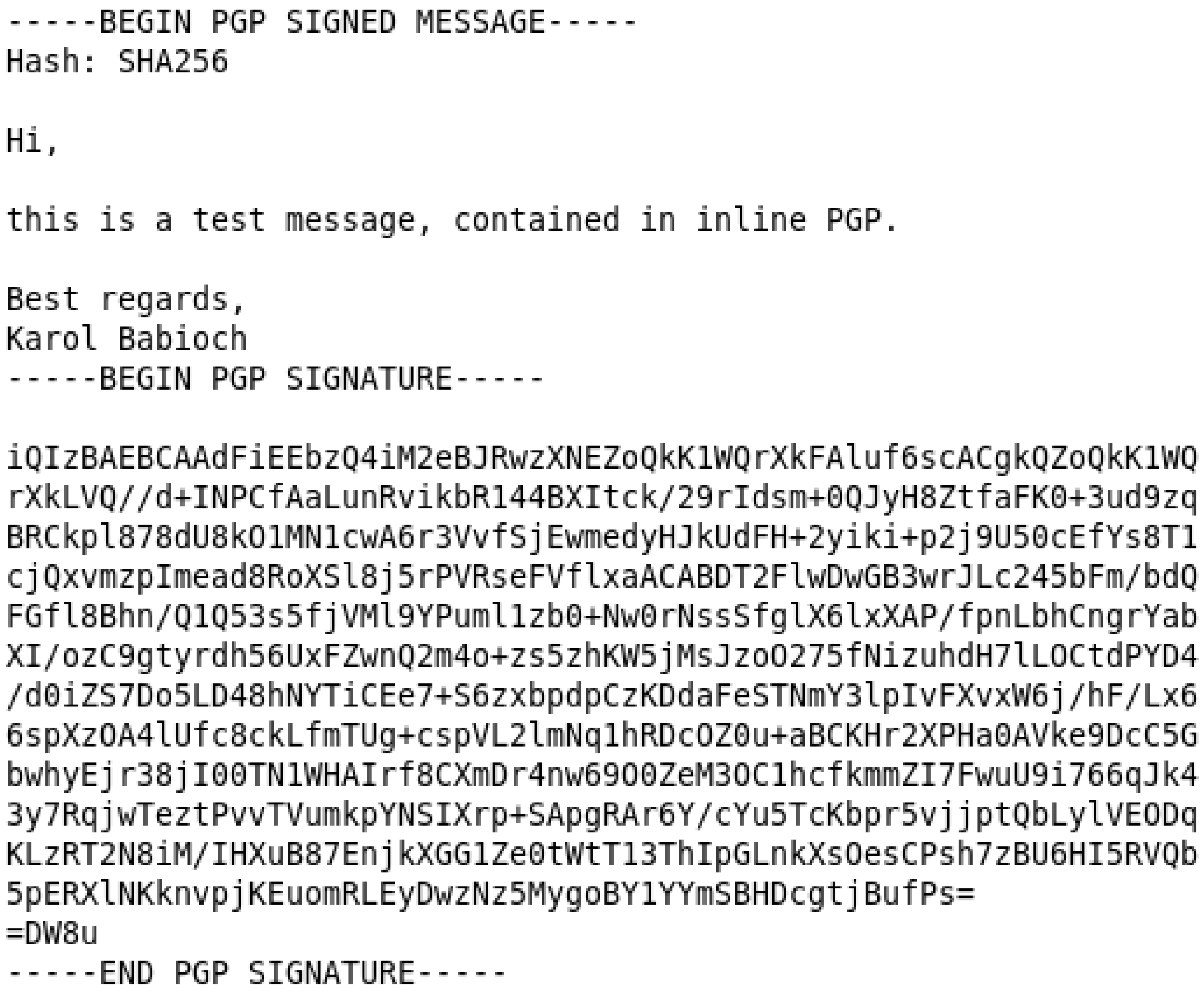
OpenPGP

Karol Babioch
Security Engineer
kbabioch@suse.de

OpenPGP basics

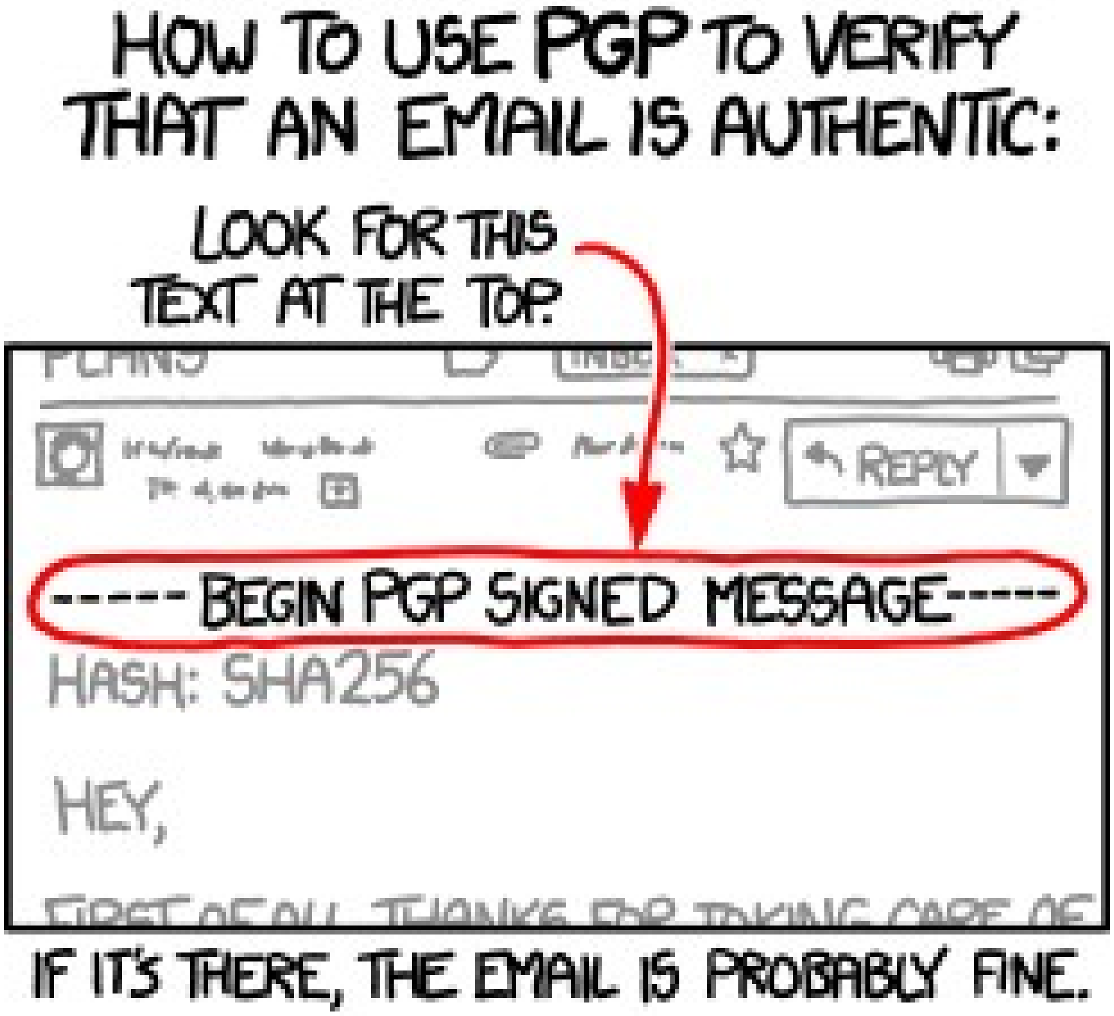
- RFC 4880
- Most widely used implementation: GnuPG (gpg)
- Allows
  - Encryption
  - Signatures / Verification
  - Authentication
- Decentral approach (“web of trust”)
  - Everybody can create key pairs
  - Distribution via keyservers
  - Authentication via keysigning

OpenPGP example



```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA256

Hi,

this is a test message, contained in inline PGP.

Best regards,
Karol Babioch
-----BEGIN PGP SIGNATURE-----

iQIzBAEBCAAdFiEEbzQ4iM2eBJRwzXNEZoQkK1WQrXkFAluf6scACgkQZoQkK1WQ
rXkLVQ//d+INPCfAaLunRvikbR144BXItck/29rIdsm+0QJyH8ZtfaFK0+3ud9zq
BRCkpl878dU8kO1MN1cwA6r3VvfSjEwmedyHJkUdFH+2yiki+p2j9U50cEfYs8T1
cjQxvmzpImead8RoXSl8j5rPVRseFVflxaACABDT2FlwDwGB3wrJLc245bFm/bdQ
FGfl8Bhn/Q1Q53s5fjVMl9YPuml1zb0+Nw0rNssSfglX6lxXAP/fpnLbhCngrYab
XI/ozC9gtyrdh56UxFZwnQ2m4o+zs5zhKW5jMsJzoO275fNizuhdH7lLOCtdPYD4
/d0iZS7Do5LD48hNYTiCEe7+S6zxbpdpCzKDdaFeSTNmY3lpIvFXvxW6j/hF/Lx6
6spXzOA4lUfc8ckLfmTUg+cspVL2lmNq1hRDcOZ0u+aBCKHr2XPHa0AVke9DcC5G
bwhyEjr38jI00TN1WHAIrf8CXmDr4nw69O0ZeM3OC1hcfkmmZI7FwuU9i766qJk4
3y7RqjwTeztPvvTVumkpYNSIXrp+SApgRAr6Y/cYu5TcKbpr5vjjptQbLylVEODq
KLzRT2N8iM/IHXuB87EnjkXGG1Ze0tWtT13ThIpGLnkXs0esCPsh7zBU6HI5RVQb
5pERXlNKknvpjKEuomRLEyDwzNz5MygoBY1YYmSBHDcgtjBufPs=
=DW8u
-----END PGP SIGNATURE-----
```

OpenPGP problems (1)


HOW TO USE PGP TO VERIFY THAT AN EMAIL IS AUTHENTIC:
LOOK FOR THIS TEXT AT THE TOP:
REPLY
-----BEGIN PGP SIGNED MESSAGE-----
HASH: SHA256
HEY,
FIRST OF ALL, THANKS FOR TAKING CARE OF
IF IT'S THERE, THE EMAIL IS PROBABLY FINE.

OpenPGP problems (2)

- Very inconvenient and difficult to use
  - Snowden vs. Glenn Greenwald
- Web of Trust
  - Trust models (pgp, classic, tofu, tofu+pgp, direct, always, auto)
  - Keysigning parties → Crypto nerd overkill
  - Mail addresses are often not verified
- Keys are lost all of the time
- Unlimited lifetime → Bad practice
- Revocation
- Fake keys
- Key handle collision (short handles)
- Autocrypt !?!

→ In daily communication: Utterly broken (in my opinion)

- Good for automated signing and verification
  - Can be part of supply chain security
  - Software distribution

WebAuthn

Karol Babioch
Security Engineer
kbabioch@suse.de

WebAuthn

- New emerging standard (W3C Candidate Recommendation, 7 August 2018)
- Supported by major browsers
- Derived from work previously done by FIDO Alliance (UAF, U2F)
- Mostly backwards-compatible with U2F
- Single factor or additional factor
- JavaScript-based API
- Allows for public-key cryptography in the browser through standardized API
  - Nothing to loose for service providers!

WebAuthn basics

- Server → Relying party (RP)
  - Generates and delivers JavaScript
- Browser
  - Processes JavaScript → Forwards request to authenticator
  - Acts as “proxy” between Authenticator and RP
- Authenticator
  - hardware token (USB, Bluetooth, NFC, etc.)
  - Software / operating system (e.g. Windows Hello (?))

WebAuthn steps

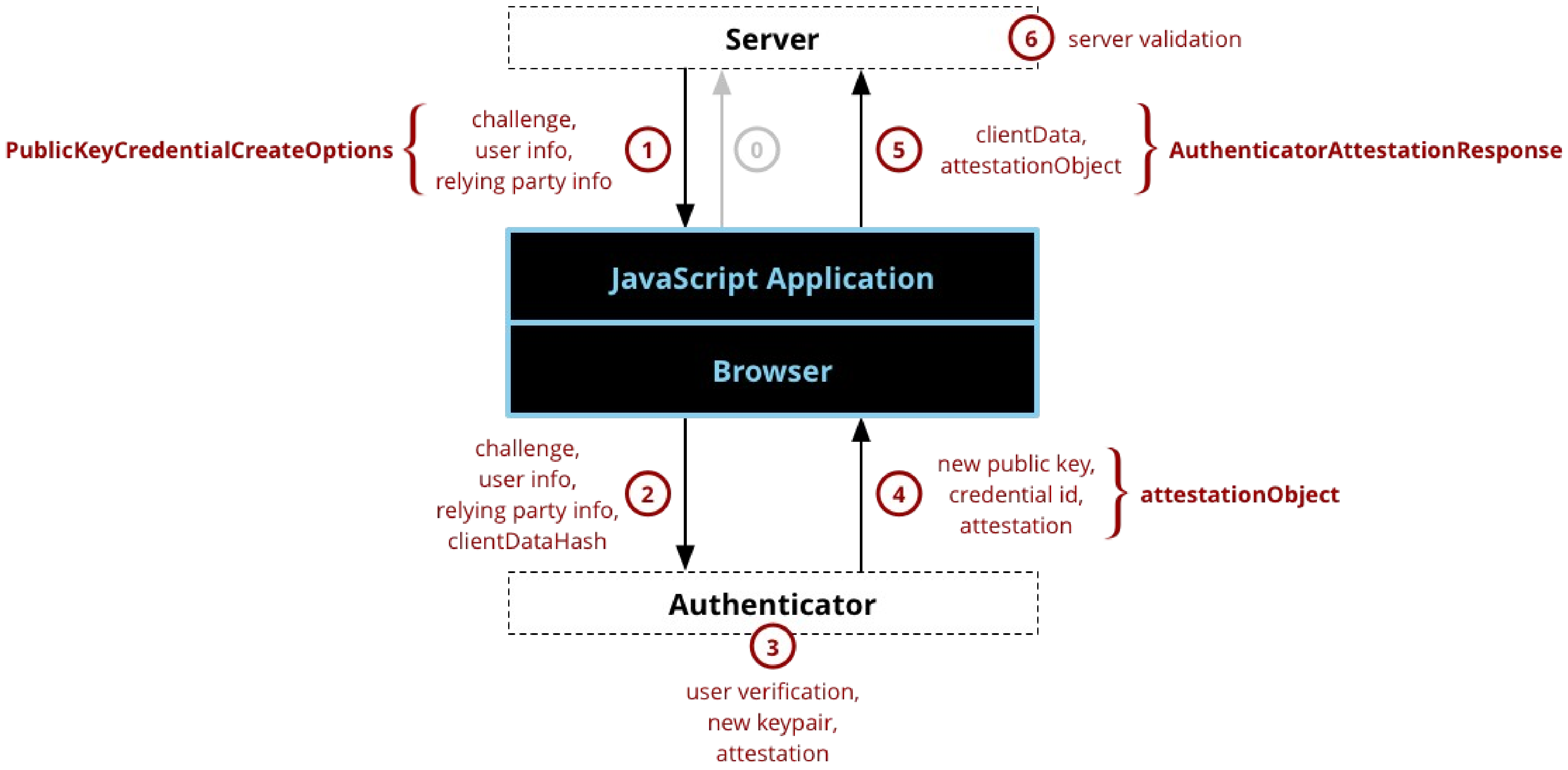
1.) Registration

- Create and register new public key

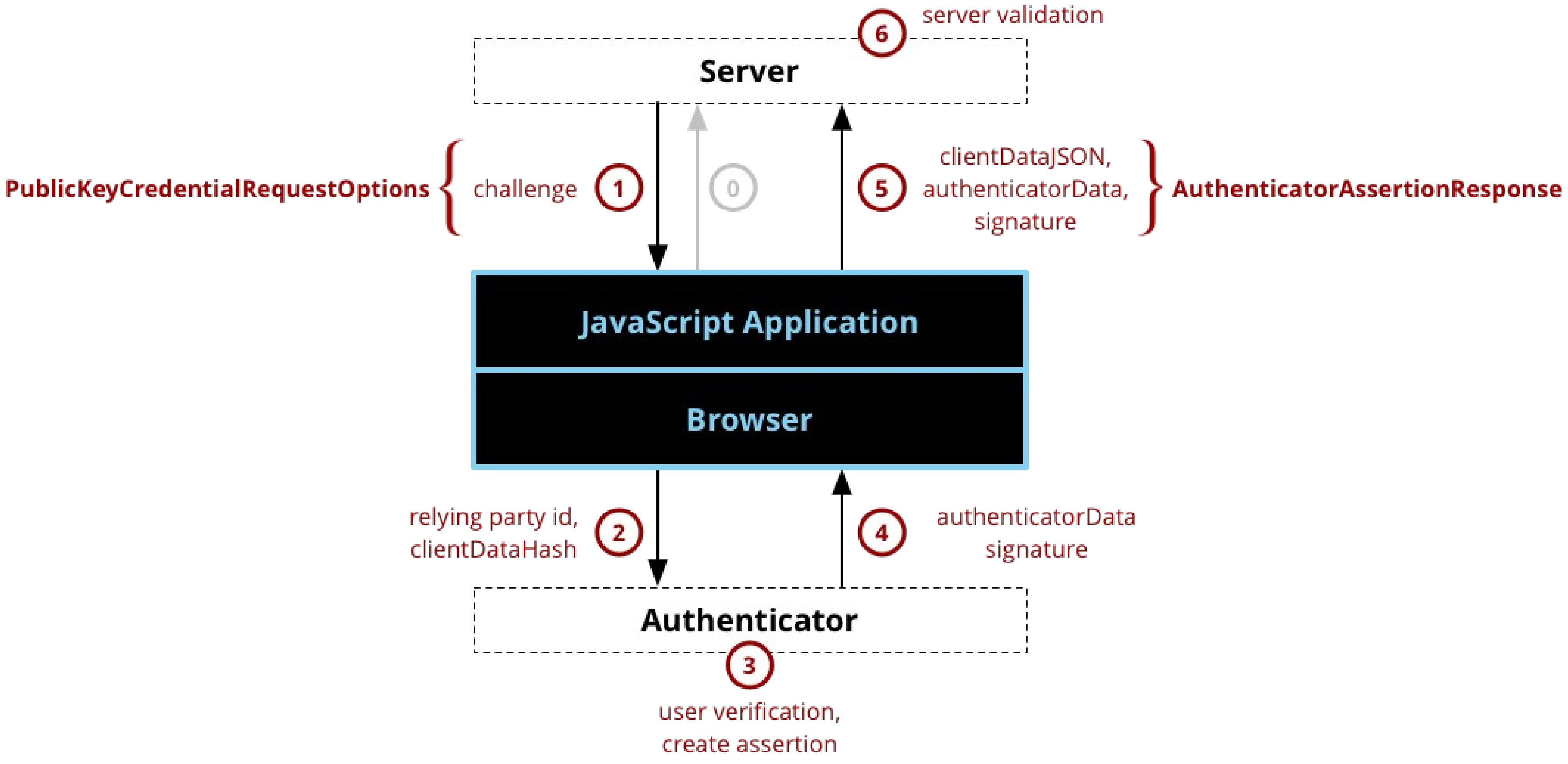
2.) Authentication

- Use previously registered public key to sign a challenge

WebAuthn registration

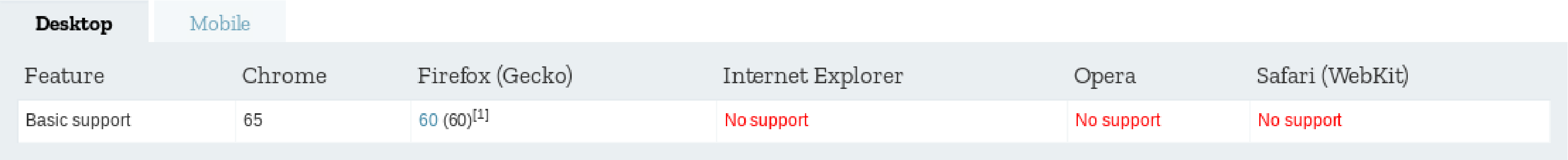


WebAuthn authentication

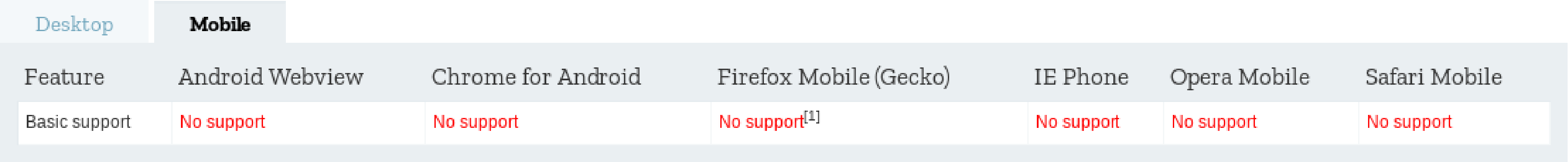


WebAuthn browser support

- Browser support



Feature	Chrome	Firefox (Gecko)	Internet Explorer	Opera	Safari (WebKit)
Basic support	65	60 (60)[1]	No support	No support	No support



Feature	Android Webview	Chrome for Android	Firefox Mobile (Gecko)	IE Phone	Opera Mobile	Safari Mobile
Basic support	No support	No support	No support[1]	No support	No support	No support

WebAuthn challenges / problems

- Adoption, adoption, adoption
  - Browser support
  - Users
  - Servers & application

- Security concerns due to weak cryptography in standard (beginning of Aug 2018)
  - RSA: PKCS1v1.5 padding
  - ECC: ECDAA

→ https://paragonie.com/blog/2018/08/security-concerns-surrounding-webauthn-don-t-implement-ecdaa-yet

WebAuthn demo

- https://webauthn.bin.coffee/
- http://webauthndemo.appspot.com/
- https://webauthn.org/

→ More on this (FIDO2/U2F) → Second talk

FIDO2 / U2F → Second talk

Karol Babioch
Security Engineer
kbabioch@suse.de

Central authentication schemes

Karol Babioch
Security Engineer
kbabioch@suse.de

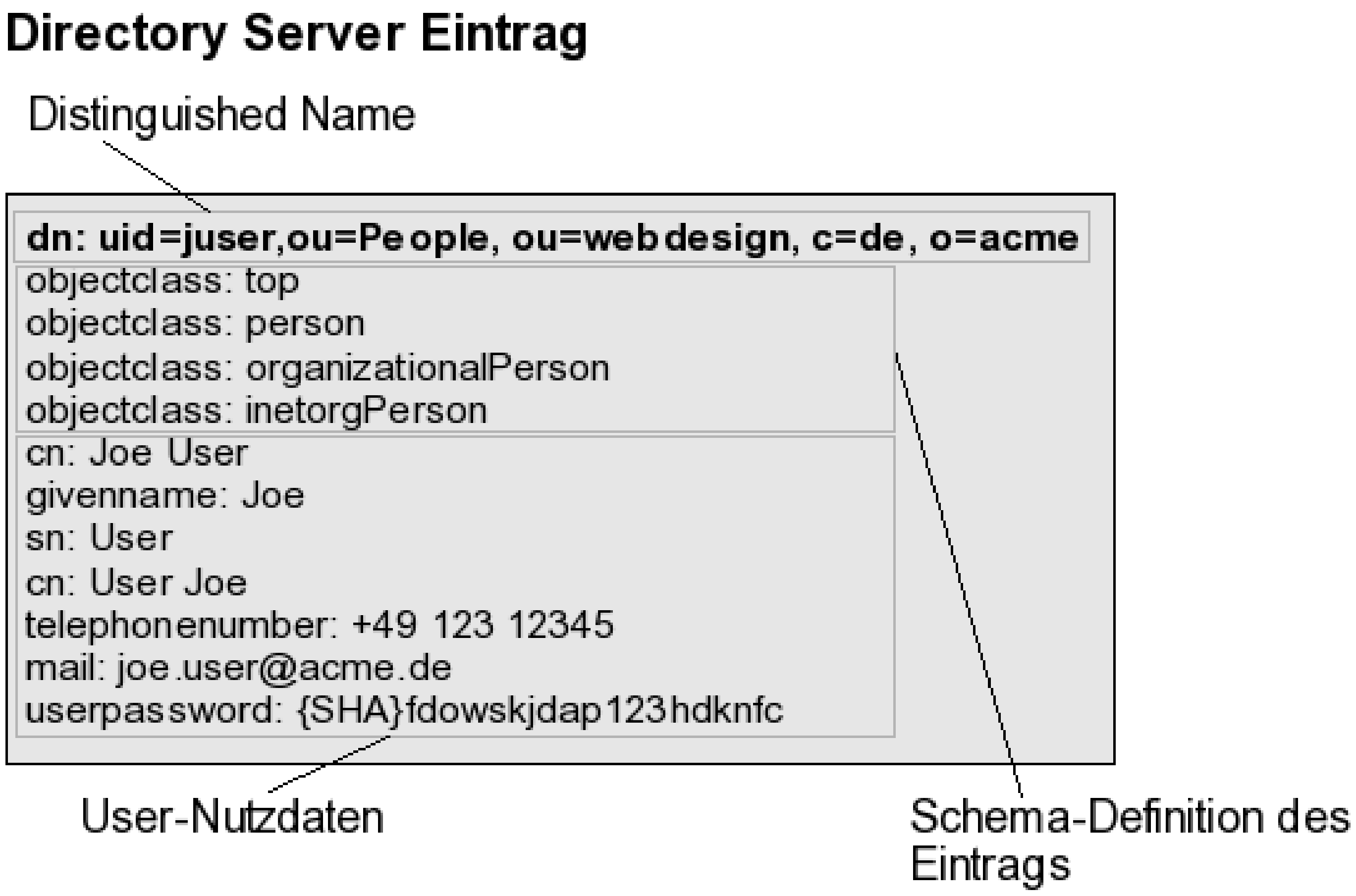
LDAP

Karol Babioch
Security Engineer
kbabioch@suse.de

LDAP

- Lightweight Directory Access Protocol
- Based on X500 (!= X509)
- Directory service (protocol & data format, etc.)
  → Not an authentication protocol
- Central directory
  – Containing (among other things) user information
  → Can be used for authentication
- Used by many applications & appliances, etc.
- Terminology
  – Distinguished Name (DN) → Username
  – Bind → Authentication
- In most cases: Based on username & password → Same problems

LDAP example



Directory Server Eintrag

Distinguished Name

```
dn: uid=juser,ou=People, ou=webdesign, c=de, o=acme
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetorgPerson
cn: Joe User
givenname: Joe
sn: User
cn: User Joe
telephonenumber: +49 123 12345
mail: joe.user@acme.de
userpassword: {SHA}fdowskjdap123hdknfc
```

User-Nutzdaten

Schema-Definition des Eintrags

LDAP problems

- Central, but no Single-Sign-On (SSO)
- Requires LDAP understanding (protocol, structure, hierarchies, etc.)
- Old and “rusty”
  - Legacy password schemes, etc.
  - Un-encrypted by default
- Requires setup by administrator / operator

→ Does not scale for users

- In fairness: Also supports other authentication schemes (SASL, Kerberos)

Federated authentication

Karol Babioch
Security Engineer
kbabioch@suse.de

Kerberos

Karol Babioch
Security Engineer
kbabioch@suse.de

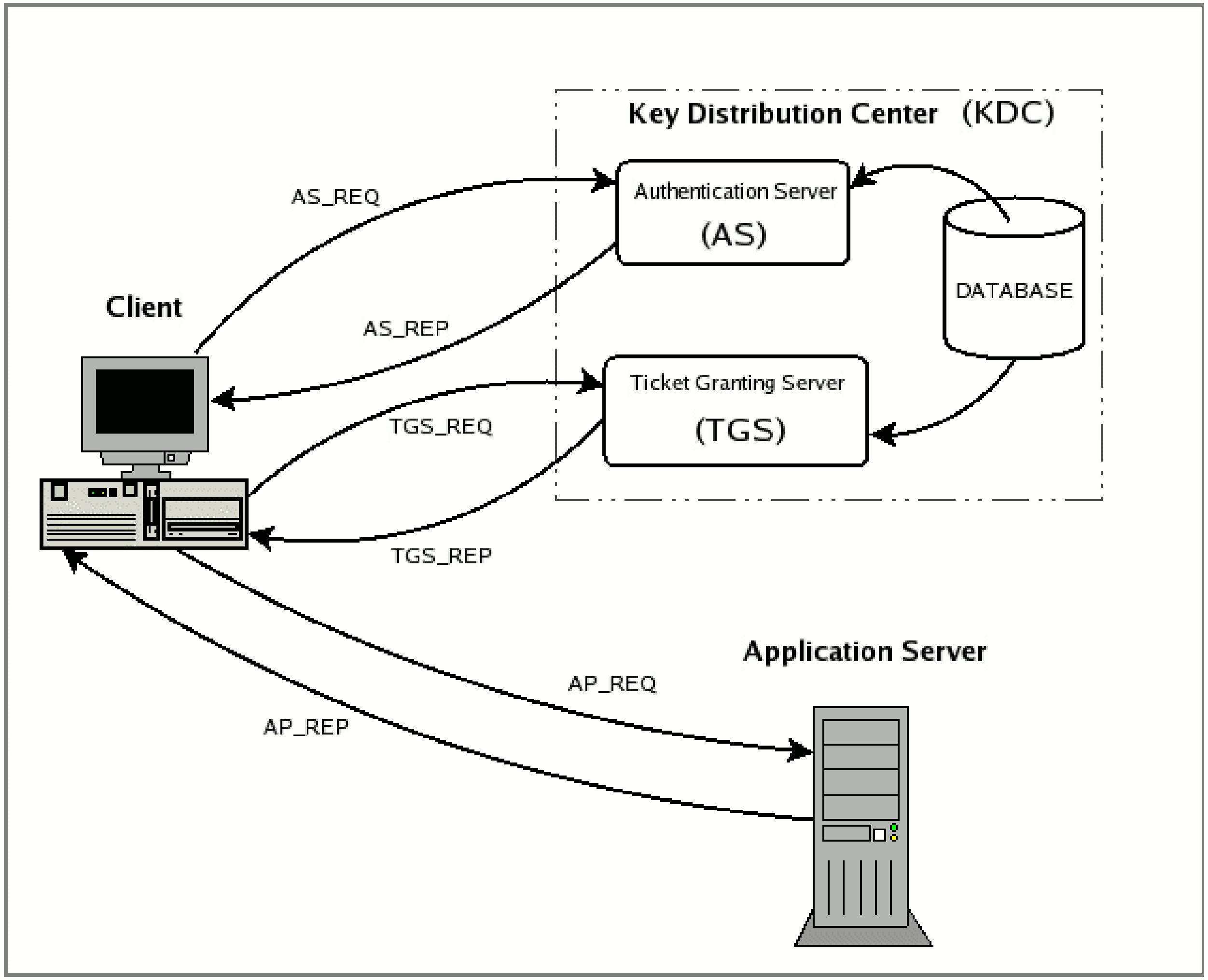
Kerberos

- Originally developed by MIT in the 80’s
- Designed for Single-Sign-On
- Many implementations (e.g. Microsoft, MIT Kerberos, etc.)
- Current version: Kerberos 5

- Basic idea (“tickets”)
  - Ticket-granting ticket (TGT, “master” ticket) can be obtained from central server (KDC)
  - TGT to get any additional tickets for services
  - Service tickets for individual services

- Tickets are short-lived, can be renewed and are mostly managed automatically in credential caches, and keytabs

Kerberos architecture



Kerberos problems / challenges

- Based upon shared secrets
  - Can be mitigated somewhat by PKINIT and OTP
  - TGTs are the key to the kingdom
    - Mitigation: Short life-time and renewal
      - Only files on your machine
      - Machines can be compromised

- KDC contains all of the keys (un-encrypted!)

- Requires application support (“Kerberized”)
  - Provided via GSSAPI (e.g. SSH, NFS, Firefox, Chrome, etc.)

- Requires initial setup (domain-specific)
  - Good within corporate network
  - Scales badly with many domains, etc.

SAML

Karol Babioch
Security Engineer
kbabioch@suse.de

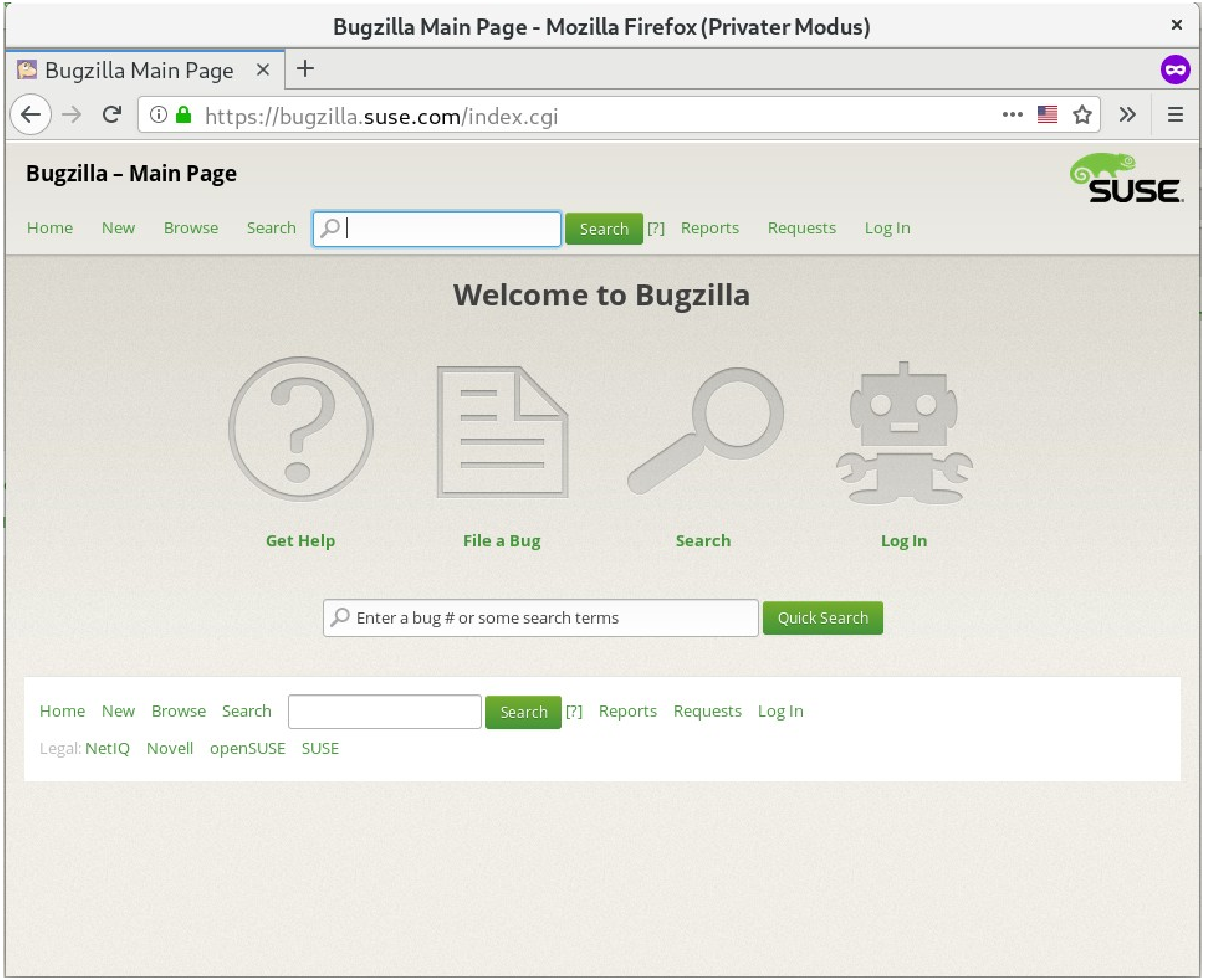
SAML basics

- Security Assertion Markup Language
- Current version: 2.0
- Standardized in 2005 by OASIS
- XML-based
- Mostly used in academic and enterprise environments
- “Assertions” are passed between entities

- Identity Providers (IdP) → Central service that authenticates users
  - Can use all sorts of mechanisms: Passwords, IPs, Kerberos, etc.

- Service Providers (SP) → Services that rely on IdP for authentication
  - Does not care how IdP performs authentication, just “consumes” assertions

SAML basics



Bugzilla Main Page - Mozilla Firefox (Privater Modus)

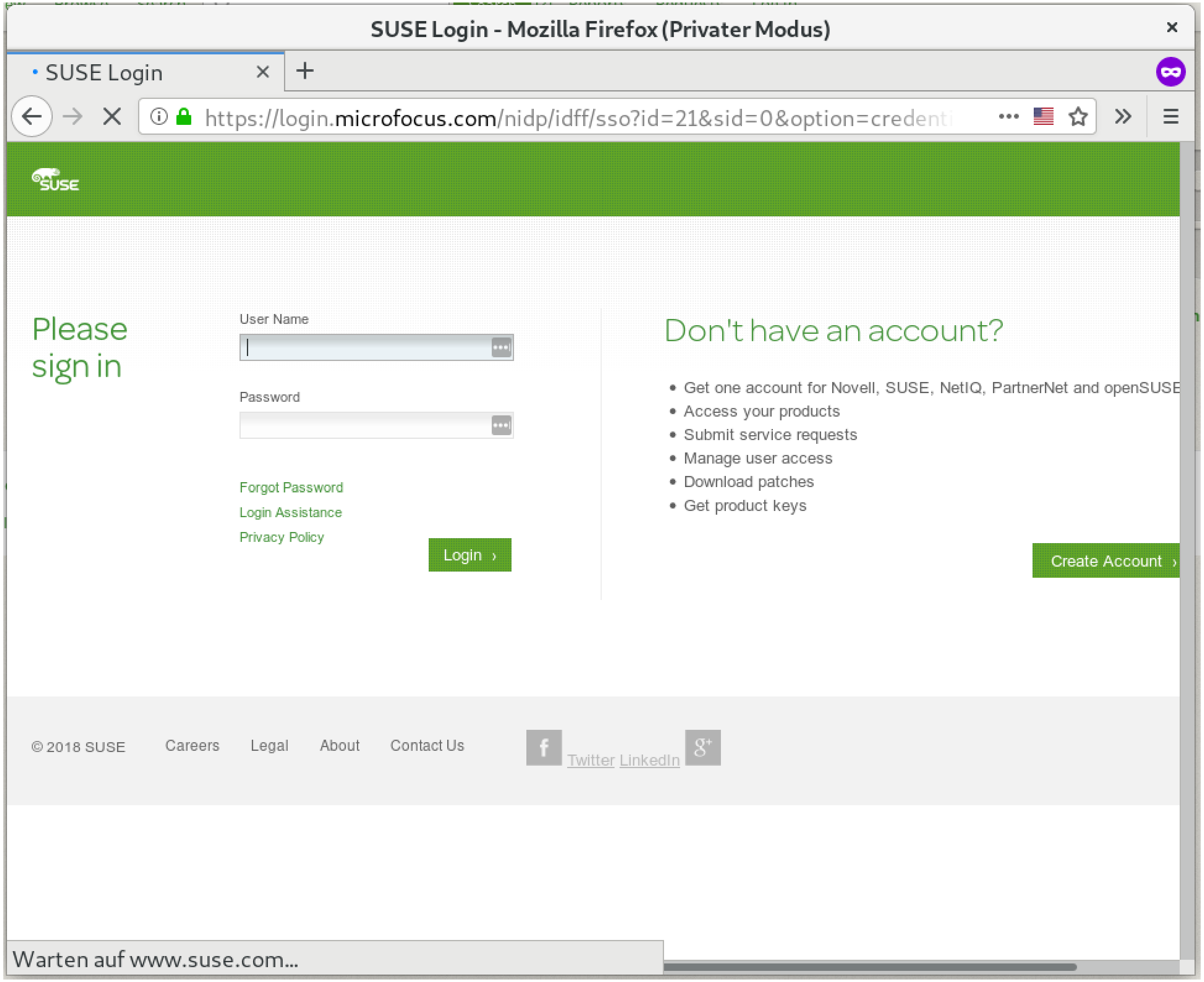
Bugzilla Main Page

https://bugzilla.suse.com/index.cgi

Bugzilla – Main Page

Welcome to Bugzilla

SAML basics



SUSE Login - Mozilla Firefox (Privater Modus)

https://login.microfocus.com/nidp/idff/sso?id=21&sid=0&option=credenti

Please sign in

User Name: |

Password:

Forgot Password
Login Assistance
Privacy Policy

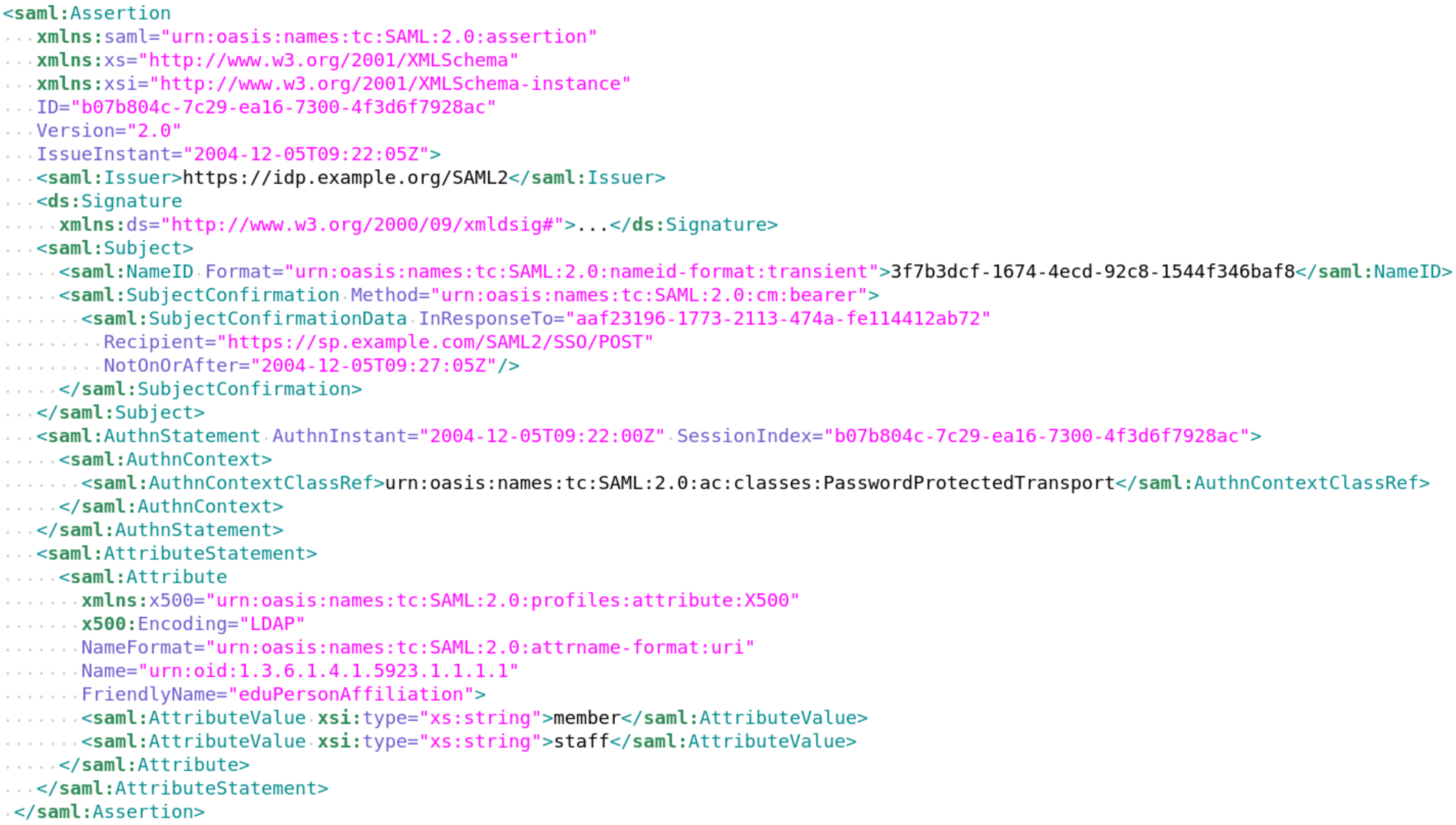
Don't have an account?

- Get one account for Novell, SUSE, NetIQ, PartnerNet and openSUSE
- Access your products
- Submit service requests
- Manage user access
- Download patches
- Get product keys

© 2018 SUSE

Warten auf www.suse.com...

SAML example



```
<saml:Assertion
   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   ID="b07b804c-7c29-ea16-7300-4f3d6f7928ac"
   Version="2.0"
   IssueInstant="2004-12-05T09:22:05Z">
   <saml:Issuer>https://idp.example.org/SAML2</saml:Issuer>
   <ds:Signature
     xmlns:ds="http://www.w3.org/2000/09/xmldsig#">...</ds:Signature>
   <saml:Subject>
     <saml:NameID Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient">3f7b3dcf-1674-4ecd-92c8-1544f346baf8</saml:NameID>
     <saml:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
       <saml:SubjectConfirmationData InResponseTo="aaf23196-1773-2113-474a-fe114412ab72"
         Recipient="https://sp.example.com/SAML2/SSO/POST"
         NotOnOrAfter="2004-12-05T09:27:05Z"/>
     </saml:SubjectConfirmation>
   </saml:Subject>
   <saml:AuthnStatement AuthnInstant="2004-12-05T09:22:00Z" SessionIndex="b07b804c-7c29-ea16-7300-4f3d6f7928ac">
     <saml:AuthnContext>
       <saml:AuthnContextClassRef>urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport</saml:AuthnContextClassRef>
     </saml:AuthnContext>
   </saml:AuthnStatement>
   <saml:AttributeStatement>
     <saml:Attribute
       xmlns:x500="urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500"
       x500:Encoding="LDAP"
       NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
       Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.1"
       FriendlyName="eduPersonAffiliation">
       <saml:AttributeValue xsi:type="xs:string">member</saml:AttributeValue>
       <saml:AttributeValue xsi:type="xs:string">staff</saml:AttributeValue>
     </saml:Attribute>
   </saml:AttributeStatement>
 </saml:Assertion>
```

SAML architecture

- Core

  → Description of syntax, semantic, etc.

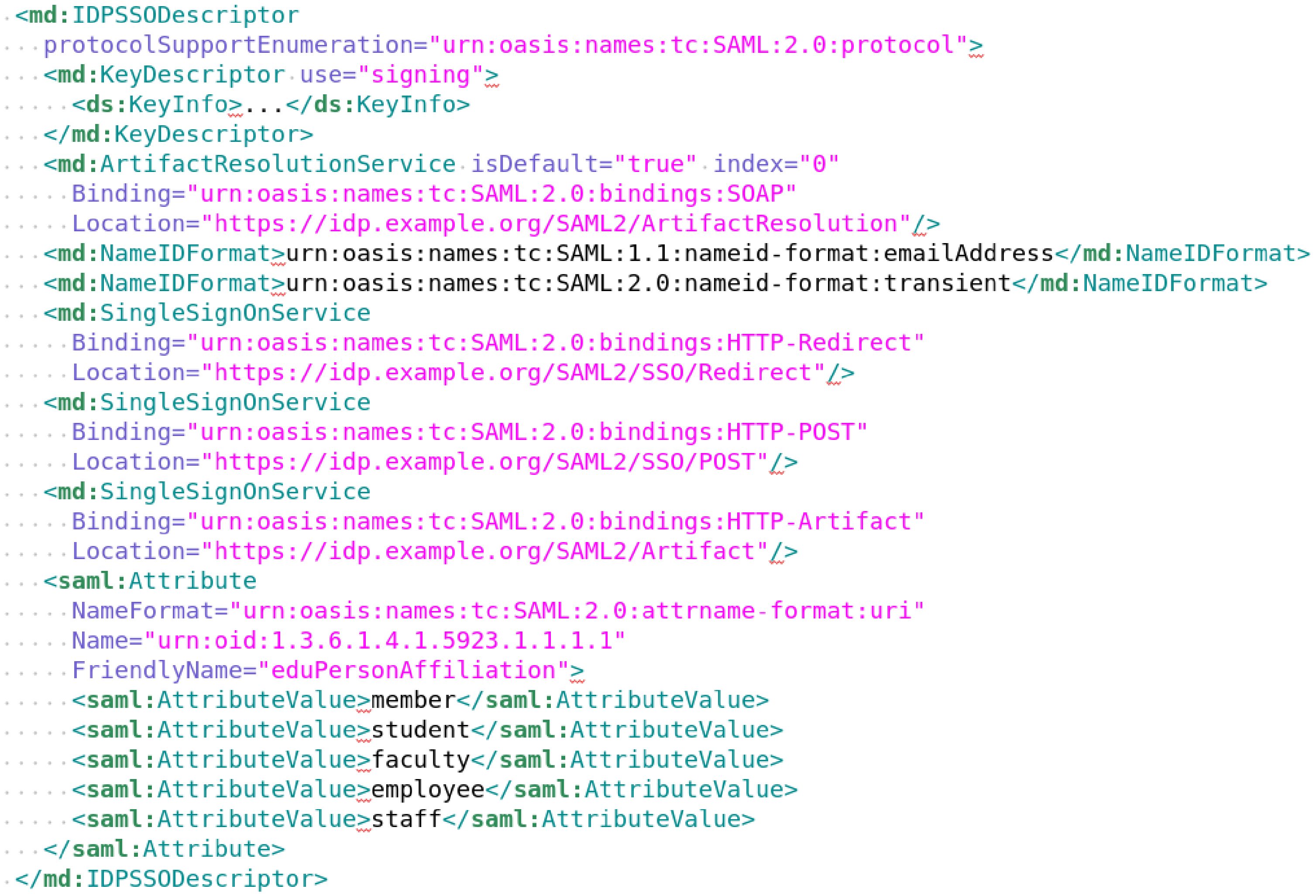
- Bindings
  – HTTP Redirect, HTTP POST, HTTP Artifact, SOAP, PAOS

  → Means of transportation of SAML messages

- Profiles
  – Web Browser SSO Profile
  – Enhanced Client or Proxy (ECP) Profile
  – Single Logout Profile

- Metadata
  – Description of URL endpoints, signing & encryption keys, etc.

SAML example metadata



```xml
 <md:IDPSSODescriptor
   protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
   <md:KeyDescriptor use="signing">
     <ds:KeyInfo>...</ds:KeyInfo>
   </md:KeyDescriptor>
   <md:ArtifactResolutionService isDefault="true" index="0"
     Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
     Location="https://idp.example.org/SAML2/ArtifactResolution"/>
   <md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress</md:NameIDFormat>
   <md:NameIDFormat>urn:oasis:names:tc:SAML:2.0:nameid-format:transient</md:NameIDFormat>
   <md:SingleSignOnService
     Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
     Location="https://idp.example.org/SAML2/SSO/Redirect"/>
   <md:SingleSignOnService
     Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
     Location="https://idp.example.org/SAML2/SSO/POST"/>
   <md:SingleSignOnService
     Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact"
     Location="https://idp.example.org/SAML2/Artifact"/>
   <saml:Attribute
     NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
     Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.1"
     FriendlyName="eduPersonAffiliation">
     <saml:AttributeValue>member</saml:AttributeValue>
     <saml:AttributeValue>student</saml:AttributeValue>
     <saml:AttributeValue>faculty</saml:AttributeValue>
     <saml:AttributeValue>employee</saml:AttributeValue>
     <saml:AttributeValue>staff</saml:AttributeValue>
   </saml:Attribute>
 </md:IDPSSODescriptor>
```

SAML challenges

- Not universal → Requires application support
  - → Many libraries are available
- Requires initial setup (metadata exchange)
- Requires maintenance (key rollovers, etc.)
- No useful auto discovery (only within a domain)

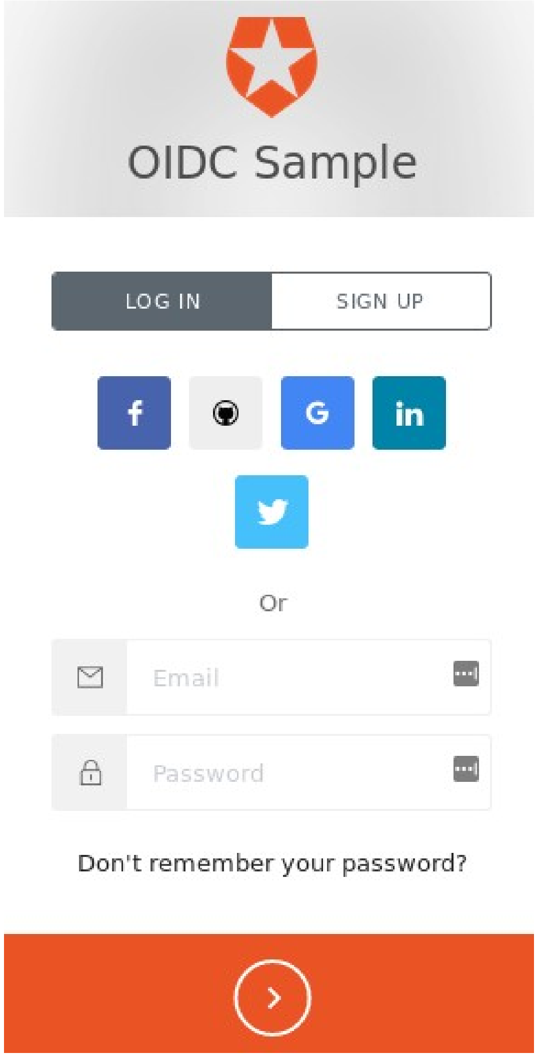
OpenID Connect

Karol Babioch
Security Engineer
kbabioch@suse.de

OpenID Connect

- Published 2014 (by the OpenID Foundation)
- Based on OAuth 2.0
  → “Abuses” authorization for authentication
- Allows Single-Sign-On (SSO)
- Feature-wise similar to SAML
  - REST-API
  - JSON data
  → Easy to consume (web applications, apps on smartphones, etc.)
- Terminology
  – Relying Party (RP)
  – Identity Provider (IdP)

OpenID Connect



OIDC Sample

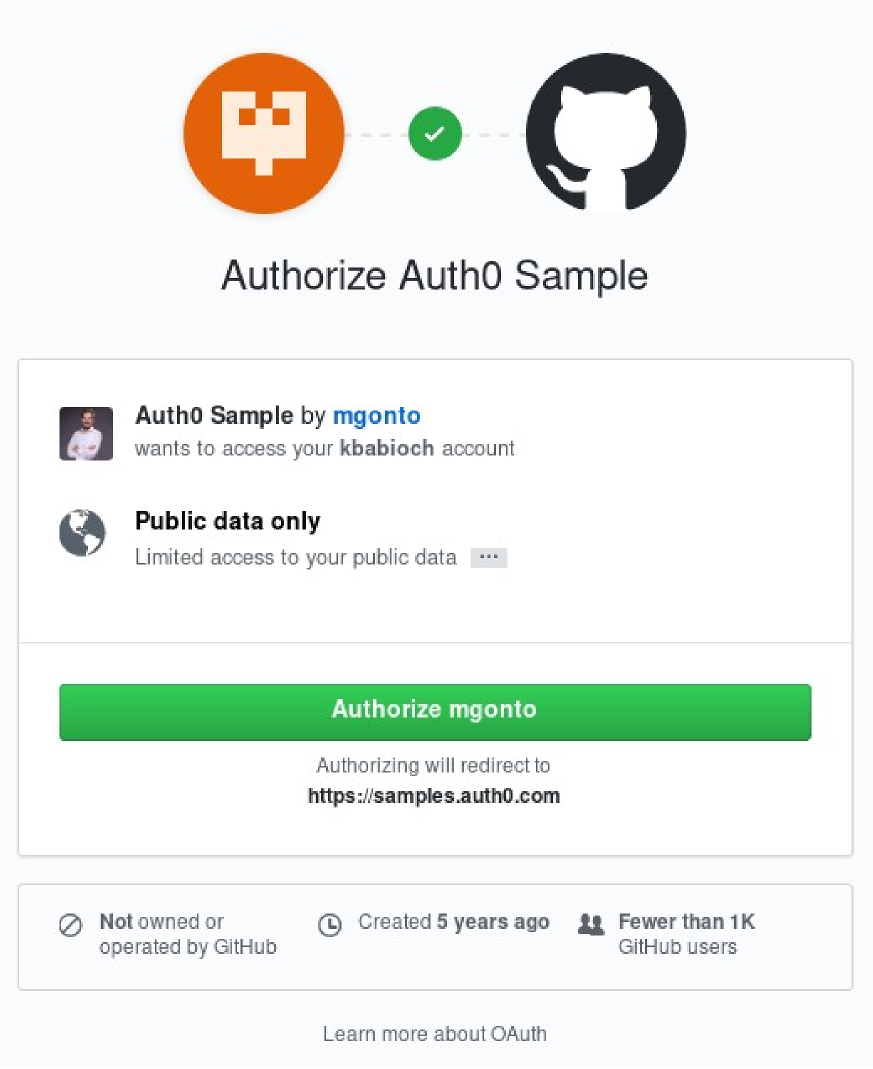
LOG IN | SIGN UP

Or

Email

Password

Don't remember your password?



Authorize Auth0 Sample

Auth0 Sample by **mgonto**
wants to access your **kbabioch** account

Public data only
Limited access to your public data ···

Authorize mgonto

Authorizing will redirect to
https://samples.auth0.com

Not owned or operated by GitHub

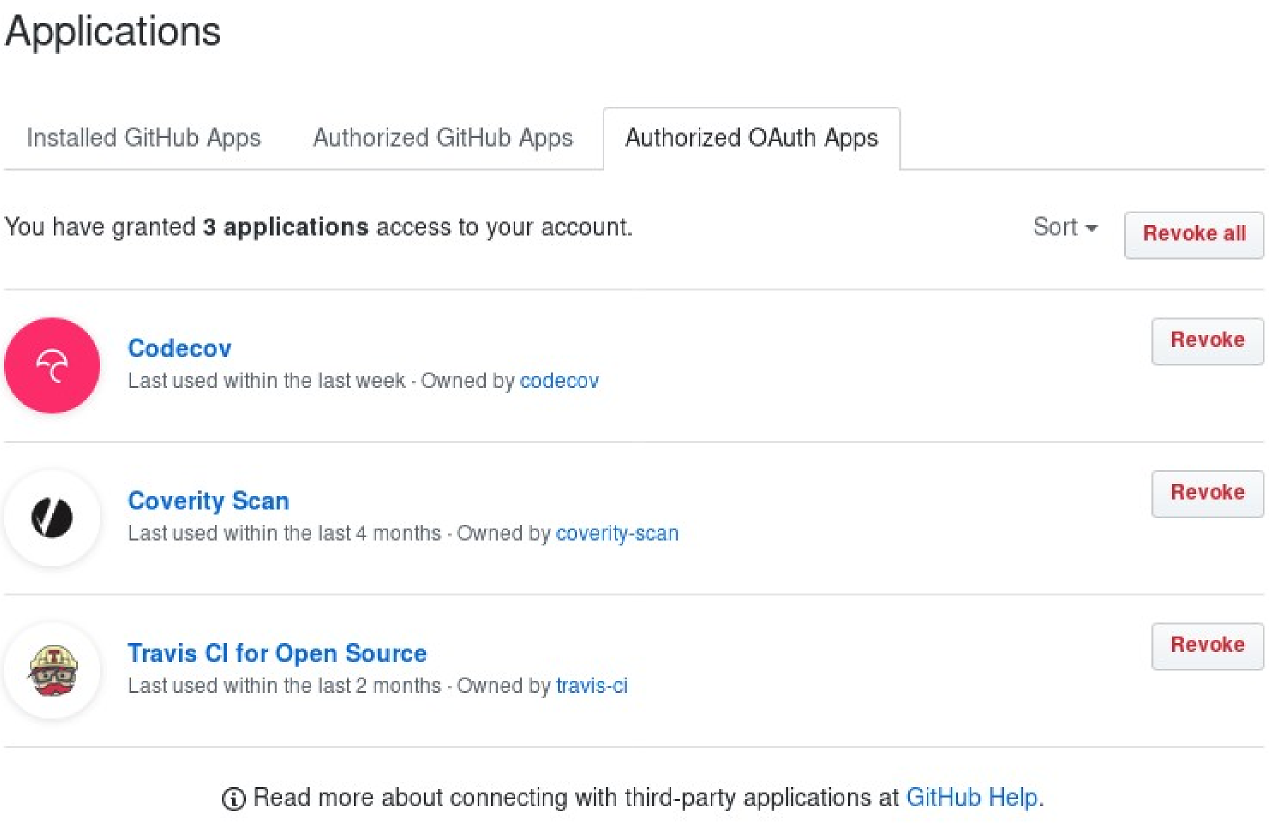
Created **5 years ago**

Fewer than 1K GitHub users

Learn more about OAuth

OpenID Connect tokens

- Authorization tokens are managed by the user
  → Access can be revoked



Applications

You have granted **3 applications** access to your account.

Codecov
Last used within the last week · Owned by codecov

Coverity Scan
Last used within the last 4 months · Owned by coverity-scan

Travis CI for Open Source
Last used within the last 2 months · Owned by travis-ci

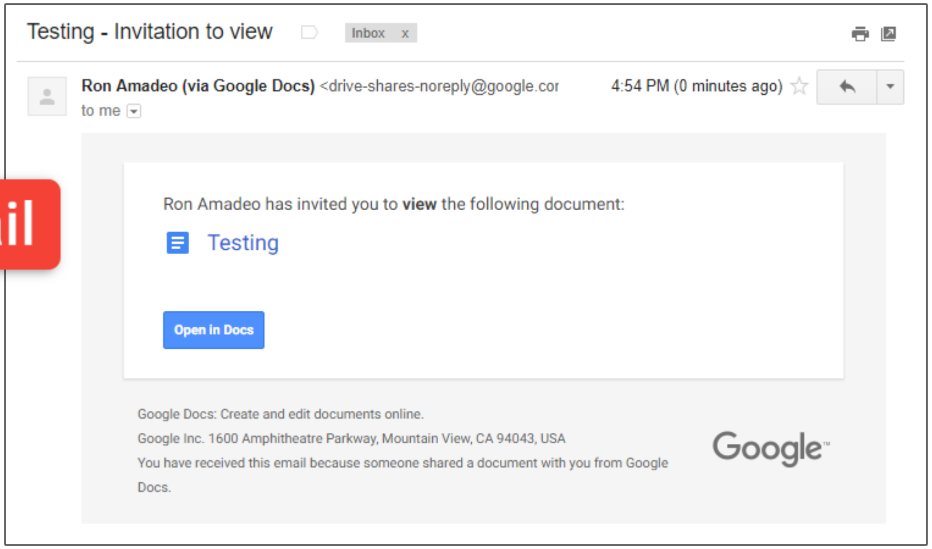
ⓘ Read more about connecting with third-party applications at GitHub Help.

OpenID Connect challenges

- Not universal → Requires application support
  - → Many libraries are available
- Privacy concerns?
- “Phishing” is still possible with OAuth 2.0
  - → There have been “worms”
- No signing / encryption between service provider and identity provider
  - → “Only” TLS for transport
- Check tokens regularly :-)

OAuth 2.0 “phishing”

Real E-mail



Testing - Invitation to view

Ron Amadeo (via Google Docs) <drive-shares-noreply@google.cor

to me

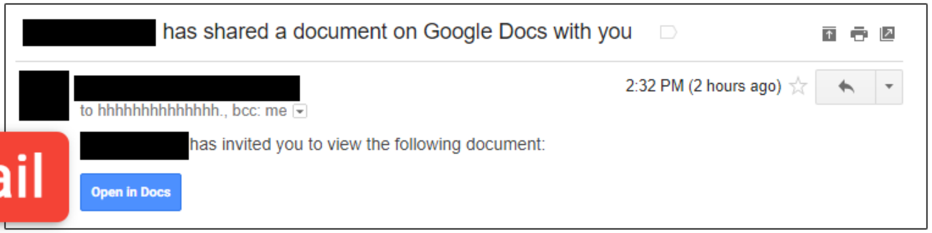
4:54 PM (0 minutes ago)

Ron Amadeo has invited you to **view** the following document:

Testing

Google Docs: Create and edit documents online.
Google Inc. 1600 Amphitheatre Parkway, Mountain View, CA 94043, USA
You have received this email because someone shared a document with you from Google Docs.

Fake E-mail



[redacted] has shared a document on Google Docs with you

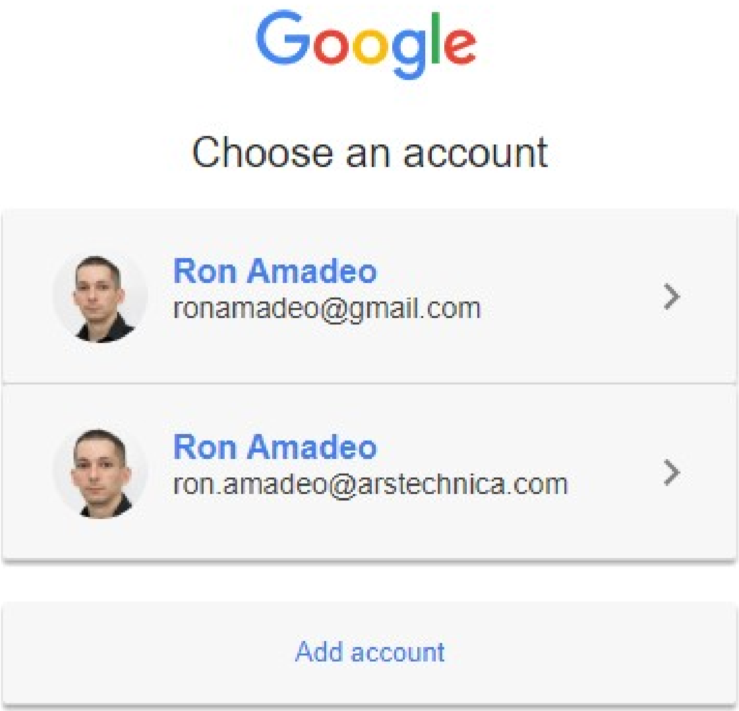
[redacted] [redacted]

to hhhhhhhhhhhhhh., bcc: me

2:32 PM (2 hours ago)

[redacted] has invited you to view the following document:

OAuth 2.0 “phishing”



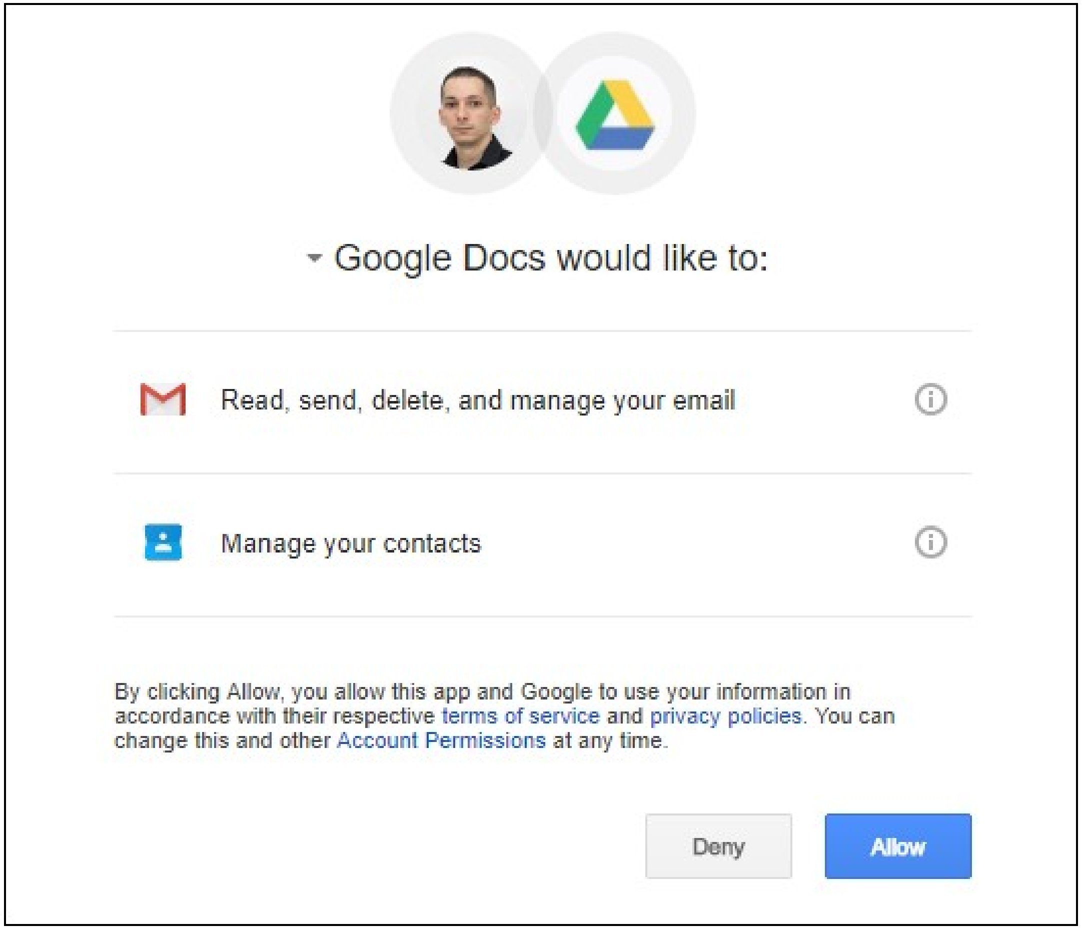
Google

Choose an account

Ron Amadeo
ronamadeo@gmail.com

Ron Amadeo
ron.amadeo@arstechnica.com

OAuth 2.0 “phishing”



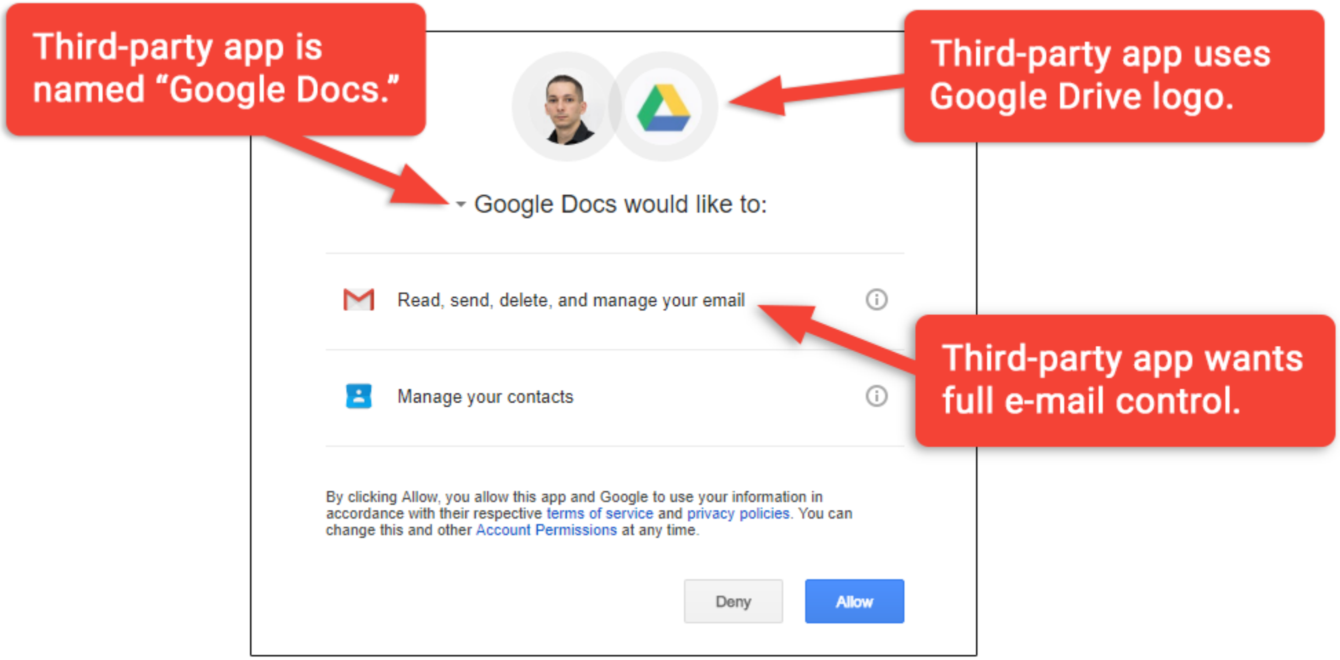
Google Docs would like to:

Read, send, delete, and manage your email

Manage your contacts

By clicking Allow, you allow this app and Google to use your information in accordance with their respective terms of service and privacy policies. You can change this and other Account Permissions at any time.

OAuth 2.0 “phishing”



Third-party app is named “Google Docs.”

Third-party app uses Google Drive logo.

Third-party app wants full e-mail control.

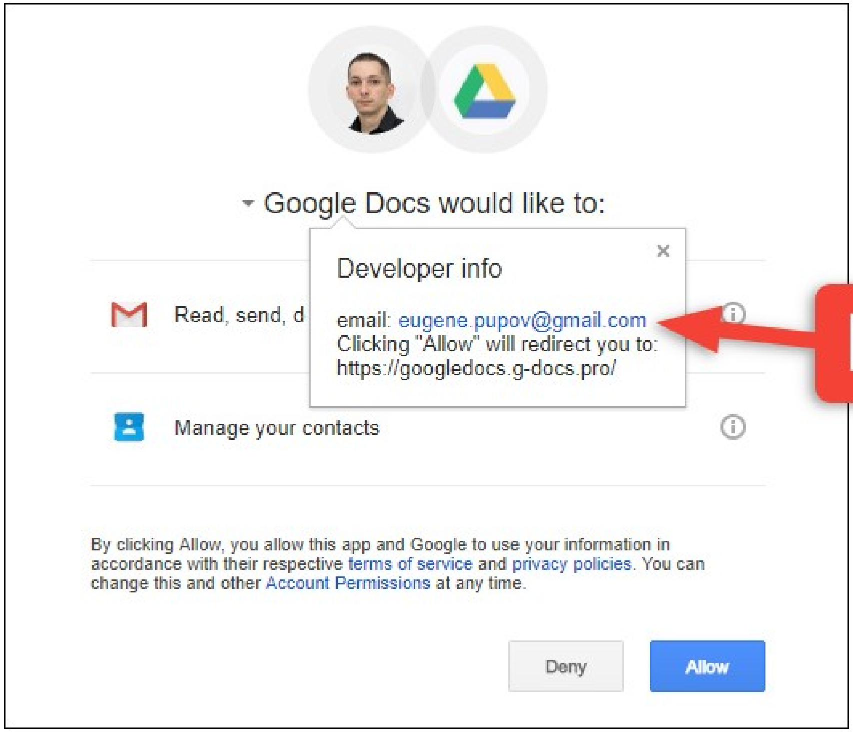
Google Docs would like to:

Read, send, delete, and manage your email

Manage your contacts

By clicking Allow, you allow this app and Google to use your information in accordance with their respective terms of service and privacy policies. You can change this and other Account Permissions at any time.

OAuth 2.0 “phishing”



Google Docs would like to:

Developer info

email: eugene.pupov@gmail.com
Clicking "Allow" will redirect you to: https://googledocs.g-docs.pro/

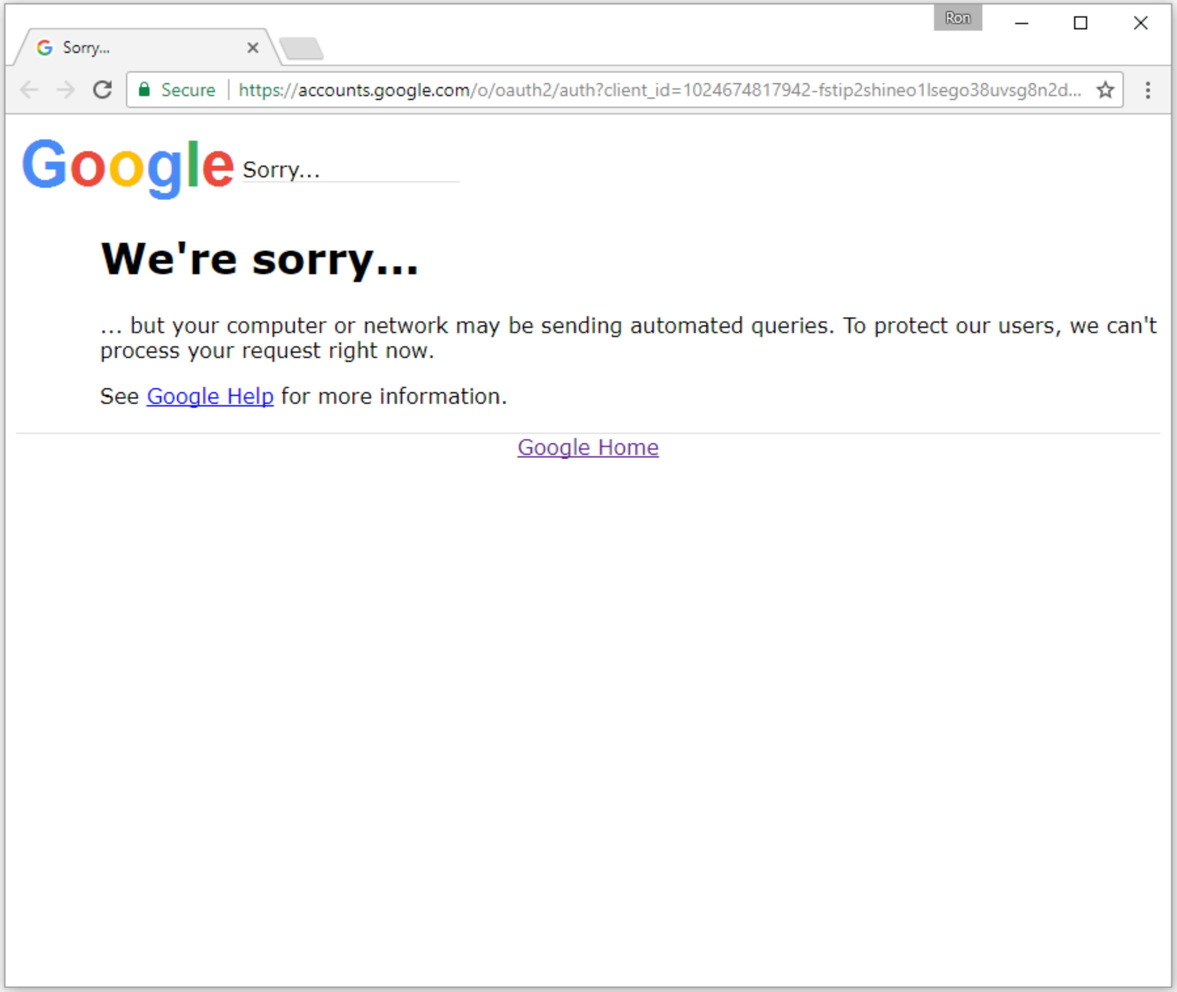
Read, send, d

Manage your contacts

By clicking Allow, you allow this app and Google to use your information in accordance with their respective terms of service and privacy policies. You can change this and other Account Permissions at any time.

Not Google

OAuth 2.0 “phishing”


Sorry...
Ron
Secure
https://accounts.google.com/o/oauth2/auth?client_id=1024674817942-fstip2shineo1lsego38uvsg8n2d...
Google
Sorry...

We're sorry...

... but your computer or network may be sending automated queries. To protect our users, we can't process your request right now.

See Google Help for more information.

Google Home

Conclusion

Karol Babioch
Security Engineer
kbabioch@suse.de

Take-away messages

- Enable two factor authentication where-ever possible
  - Annoy / blame service providers that do not yet support it
- Use password manager
  - teach your friends and family how to use them
- Use OAuth 2.0 (OpenID Connect) where-ever possible?
- Check tokens regularly, re-evaluate if still needed ...

SUSE®

We adapt. You succeed.