Entity Authentication

Securitate informatică

March 18, 2020

Mihai-Lica Pura

Cuprins

- Entity versus Data origin authentication
- Entity authentication definition and classification
- Authenticating with something you have
- Authenticating with something you are
- Authenticating with something you know
- Multi-factor authentication
- Security protocols

Entity versus Data origin authentication

- **Entity authentication** - the assurance that a given entity is involved and currently active in a communication session
- **Data origin authentication** - the assurance that a given entity was the original source of some data (sometimes referred to as message authentication)

Entity authentication

- an entity (human user or machine) must provide **assurance of the declared identity** in real time in order to have access to either physical or virtual resources
- as entity authentication provides identity assurance in "real time", it can only truly be achieved for **an instant in time**

Entity authentication

Classification

- **Unilateral entity authentication** - the assurance of the identity of one entity to another (and not vice-versa)
- **Mutual entity authentication** - both communicating entities provide each other with assurance of their declared identity

Entity authentication

- The assurance of the declared identity is given are by using (a combination of) the following:
  - something that **you have**
  - something that **you are**
  - something that **you know**

Entity authentication

Something that **you have**

- dumb tokens
  - plastic cards with a magnetic stripe
  - tokens with memory chip
- smartcards
  - bank cards
  - public transportation cards
- One Time Password (OTP) tokens
- mobile phones/smart phones SIM cards
  - often in conjunction with:
    - an OTP app
    - cryptographic material (i.e., certificate or a key) residing on the device

Entity authentication

Something that **you are** - **biometrics**

- techniques for human user entity authentication that are based on physical characteristics of the human body

- **enrolment** - a biometric control converts a physical characteristic into a digital template that is stored on a database
- **authentication** - when the user physically presents themselves for entity authentication, the physical characteristic is measured by a reader, digitally encoded, and then compared with the template

Entity authentication

Something that **you are** - **biometrics**

- **static (unchanging)** - measurements include fingerprints, finger vein scans, hand geometry, face recognition, retina scans, iris scans, earlobe geometry, etc.

- **dynamic (changing)** measurements include handwriting measurements, voice recognition, etc.

Dumb tokens

- physical device without a memory that can be used as a type of electronic key
- operate with **a reader that extracts some information from the token** and then indicates whether the information authenticates the entity or not
- the security of the card is based entirely on the difficulty of extracting the information from the token
- it is common to combine the use of a dumb token with another entity authentication technique, such as one based on something you know

Smartcards

- a plastic card that contains a chip, which gives the card a limited amount of **memory and processing power**
- can store secret data more securely
- can engage in cryptographic processes that require some computations to be performed (e.g. challenge/response)
- smartcards have limited memory and processing power, thus restricting the types of operation that they can comfortably perform

Entity authentication

Something that **you know**

- password/passphrase
- PIN
- answers to secret questions (challenge/response questions)

Password authentication

- **Authentication Cheat Sheet**
  https://www.owasp.org/index.php/Authentication_Cheat_Sheet
- **Password Storage Cheat Sheet**
  https://www.owasp.org/index.php/Password_Storage_Cheat_Sheet
- **Forgot Password Cheat Sheet**
  https://www.owasp.org/index.php/Forgot_Password_Cheat_Sheet
- **Choosing and Using Security Questions Cheat Sheet**
  https://www.owasp.org/index.php/Choosing_and_Using_Security_Questions_Cheat_Sheet

Password General Guidelines

- User IDs
  - usernames/userids are case insensitive
  - user names should be unique
  - for high security applications usernames could be assigned and secret instead of user-defined public data
- Implement Proper Password Strength Controls
  - Password Length
  - Password Complexity
  - Password Topologies

Password General Guidelines

- **Password Length**
  - Longer passwords provide a greater combination of characters and consequently make it more difficult for an attacker to guess
  - Minimum length of the passwords should be enforced by the application
  - Passwords shorter than 10 characters are considered to be weak (NIST SP800-132)
  - Passphrases shorter than 20 characters are usually considered weak if they only consist of lower case Latin characters
  - Typical maximum length is 128 characters

Password General Guidelines

- **Password Complexity**
  - should enforce password complexity rules to discourage easy to guess passwords
  - Passwords should, obviously, be case sensitive in order to increase their complexity
  - Password mechanisms should allow virtually any character the user can type to be part of their password, including the space character

Password General Guidelines

- **Password Complexity Example**
  - Password must meet at least 3 out of the following 4 complexity rules
    - at least 1 uppercase character (A-Z)
    - at least 1 lowercase character (a-z)
    - at least 1 digit (0-9)
    - at least 1 special character (punctuation) — do not forget to treat space as special characters too
  - at least 10 characters
  - at most 128 characters
  - not more than 2 identical characters in a row (e.g., 111 not allowed)

Password General Guidelines

- **Password Topologies**
  - patterns of passwords, e.g. **Ullllldd** where:
    - u=uppercase
    - l=lowercase
    - d=digit
    - s=special (other character)
  - Ban commonly used password topologies
  - Force multiple users to use different password topologies
  - Require a minimum topology change between old and new passwords

Password General Guidelines

- **Additional Information**
  - Make sure that every character the user types in is actually included in the password
  - **do not truncate the password** at a length shorter than what the user provided (e.g., truncated at 15 characters when they entered 20)
  - be very clear about what the password policy is - the required policy needs to be explicitly stated on the password change page
  - if the new password doesn't comply with the complexity policy, the error message should describe EVERY complexity rule that the new password does not comply with

Password Storage Cheat Sheet

- Do not limit the character set and set long max lengths for credentials
- Hash the password as one of several steps
- Use a cryptographically strong credential-specific salt

  [protected form] = [salt] + protect([protection func], [salt] + [credential]);

Password Storage Cheat Sheet

- **salt** - fixed-length cryptographically-strong random value
- salt purposes:
  - prevent the protected form from revealing two identical credentials
  - augment entropy fed to protecting function without relying on credential complexity
  - the second purpose makes intractable
    - pre-computed lookup attacks on an individual credential
    - time-based attacks on a population

Password Storage Cheat Sheet

- Impose infeasible verification on attacker
  - the defender needs an acceptable response time for verification of user's credentials during peak use
  - the time required to map <credential> to <protected form> must remain beyond threats' hardware (GPU, FPGA) and technique (dictionary-based, brute force, etc) capabilities
- e.g. use adaptive one-way functions (Argon2, PBKDF2, scrypt, bcrypt, etc.)
  - adaptive one-way functions compute a one-way (irreversible) transform
  - allows configuration of **work factor**
  - underlying mechanisms used to achieve irreversibility and govern work factors (such as time, space, and parallelism) vary between functions

Password Storage Cheat Sheet

```
//STEP 1 Create the salt value with a cryptographic PRNG:
byte[] salt;
new RNGCryptoServiceProvider().GetBytes(salt = new
byte[SALT_SIZE]);

//STEP 2 Create the Rfc2898DeriveBytes and get the hash value:
var pbkdf2 = new Rfc2898DeriveBytes(password, salt, 10000);
byte[] hash = pbkdf2.GetBytes(20);

//STEP 3 Combine the salt and password bytes for later use:
byte[] hashBytes = new byte[SALT_SIZE + 20];
Array.Copy(salt, 0, hashBytes, 0, SALT_SIZE);
Array.Copy(hash, 0, hashBytes, SALT_SIZE, 20);

//STEP 4 Turn the combined salt+hash into a string for storage
string savedPasswordHash = Convert.ToBase64String(hashBytes);
```

Password Storage Cheat Sheet

```
/* Fetch the stored value */
string savedPasswordHash = database.Password;

/* Extract the bytes */
byte[] hashBytes =
Convert.FromBase64String(savedPasswordHash);

/* Get the salt */
byte[] salt = new byte[SALT_SIZE];
Array.Copy(hashBytes, 0, salt, 0, SALT_SIZE);

/* Compute the hash on the password the user entered */
var pbkdf2 = new Rfc2898DeriveBytes(user.Password, salt, 10000);
byte[] hash = pbkdf2.GetBytes(20);

/* Compare the results */
for (int i = 0; i < 20; i++)  if (hashBytes[i + SALT_SIZE] !=
hash[i])  return false;
```

Other types of information

- **geolocation and time** may be additionally included in the authentication process
- but always in CONJUNCTION with at least one of the three factors mentioned above and NEVER alone
- e.g. geolocation and time data may be used to restrict remote access to an entity's network in accordance with an individual's work schedule
- but the remote access method must still require authentication using at least one factor

Multi-factor authentication

- provides a **higher degree of assurance of the identity** of the entity attempting to access a resource (physical location, computing device, network or a database)
- creates a **multi-layered mechanism** that an unauthorized entity would have to defeat in order to gain access
- PCI DSS requires MFA to be implemented as defined in Requirement 8.3 and its sub-requirements
  - Guidance column of the standard includes:
  - "Multi-factor authentication requires an individual to present **a minimum of two separate forms of authentication** (as described in Requirement 8.2), before access is granted."

Multi-factor authentication

- the authentication mechanisms used for MFA should be **independent of one another** such that:
  - access to one factor does not grant access to any other factor
  - the compromise of any one factor does not affect the integrity or confidentiality of any other factor

Multi-factor authentication

- e.g. **no independence** - the same set of credentials (e.g., username/password) is used:
  - as an authentication factor
  - and also for gaining access to an e-mail account where a secondary factor (e.g., one-time password) is sent
- e.g. **no independence** - the same set of credentials is used for both:
  - protecting a software certificate stored on a laptop (something you have)
  - logging in to the laptop (something you know)

Multi-factor authentication

- issue with **authentication credentials embedded into the device** is a potential loss of independence between factors
- i.e. physical possession of the device can grant access to:
  - a secret (something you know)
  - as well as a token (something you have) such as the device itself, or a certificate or software token stored or generated on the device

Out-of-band authentication

- authentication processes where **authentication mechanisms are conveyed through different networks or channels**
- authentication factors are conveyed through a single device/channel
  - entering credentials via a device that also receives, stores, or generates a software token
  - a malicious user who has established control of the device has the ability to capture both authentication factors

Out-of-band authentication

- transmission of a one-time password (OTP) to a smartphone has traditionally been considered an effective out-of-band method
- if **the same phone** is then used to submit the OTP (e.g. via a web browser) the effectiveness of the OTP as a secondary factor is **effectively nullified**
- e.g. accessing the Internet banking account using the smartphone on which one receives the OTP

- NIST currently permits the use of SMS, but they have advised that **out-of-band authentication using SMS or voice has been deprecated** and may be removed from future releases of their publication

Out-of-band authentication

Problems using SMS for out-of-band authentications

- Dan Goodin, **Database leak exposes millions of two-factor codes and reset links sent by SMS** - 11/16/2018 https://arstechnica.com/information-technology/2018/11/millions-of-sms-texts-in-unsecured-database-expose-2fa-?amp=1
- Natasha Bernal, **Metro Bank hit by cyber attack used to empty customer accounts**, The Telegraph - 01/02/2019 https://www.telegraph.co.uk/technology/2019/02/01/metro-bank-hit-cyber-attack-used-empty-customer-account amp/

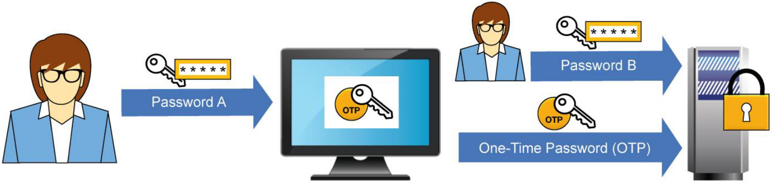
Multi-step vs. Multi-Factor authentication

- "multi-step" authentication
  - e.g. an entity submits credentials (e.g. username/password) that, once successfully validated, lead to the presentation of the second factor for validation (e.g. biometric)
- **no prior knowledge of the success or failure of any factor should be provided to the individual until all factors have been presented**
- if an unauthorized user can deduce the validity of any individual authentication factor
  - the overall authentication process becomes a collection of subsequent, single-factor authentication steps
  - even if a different factor is used for each step

Multi-step and Multi-Factor authentication

- a remote user entering credentials to log in to their corporate laptop
- he/she then initiates a VPN connection to the organization's network using a combination of:
  - credentials
  - a physical smartcard or hardware token

Multi-factor authentication scenarios



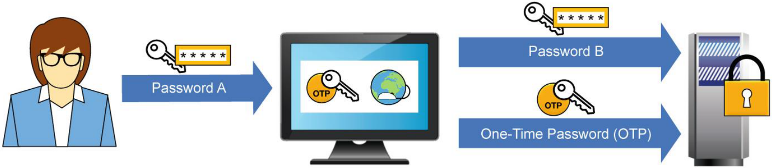
Multi-factor authentication scenarios

- an individual uses **one set of credentials (password A)** to:
  - log in to a device
  - and also to access a software token stored on the device
- the individual then establishes a connection to the corporate network, providing:
  - **a different set of credentials (password B)**
  - and **the OTP** generated by the software token as authentication

Multi-factor authentication scenarios

- **to ensure the independence of authentication factors**:
  - **the software token** ("something you have") **is embedded into the physical device** in such way that it cannot be copied or used on any other device
  - physical security over the device becomes a required security control as **proof of possession of the device**
- if access to software token is merely a reflection of the ability to login into the device (either locally or remotely)
  - the overall authentication process is a usage of "something you know" twice

Multi-factor authentication scenarios



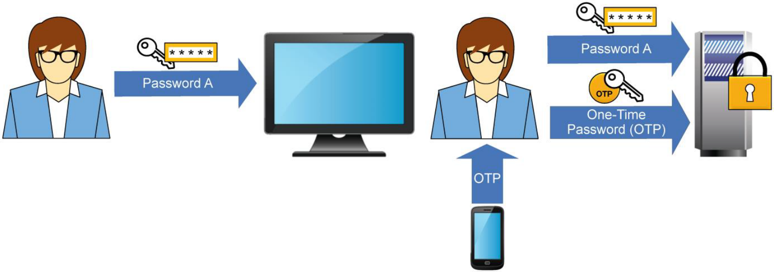
Multi-factor authentication scenarios

- an individual uses **one set of credentials (password A)** to:
  - log in to a device
  - and also to access a software token stored on the device
- the individual then establishes a connection to the corporate network by:
  - launching a browser window that **pre-populates a different set of credentials** (e.g., cached on the device or using password manager)
  - and **the OTP** generated by the software token as authentication

Multi-factor authentication scenarios

- **no independence between authentication factors**:
- a single set of credentials (Password A) provides access to both factors (password B and software token)

Multi-factor authentication scenarios



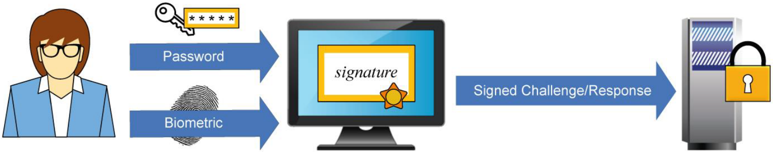
Multi-factor authentication scenarios

- the individual uses **one set of credentials** (e.g., username/password) to log in into the computer
- the connection to the corporate network requires both:
  - **the initial set of credentials**
  - **an OTP generated by a software token residing on a mobile device**

Multi-factor authentication scenarios

- the independence of the authentication factors is maintained by **the software token residing on the mobile phone** ("something you have")
- even though the individual uses the same password ("something you know") to authenticate both to the laptop and the corporate network
- **if the mobile device is also used to initiate the connection to the corporate network** (instead of the PC)
  - additional security controls would be needed to demonstrate independence of the authentication mechanisms

Multi-factor authentication scenarios



Multi-factor authentication scenarios

- the individual uses **multi-factor authentication** (e.g. password and biometric) to log in to a smartphone or a laptop
- to establish a connection to the corporate network he/she then provides a **single authentication factor** (e.g. a different password, digital certificate, or signed challenge-response)

Multi-factor authentication scenarios

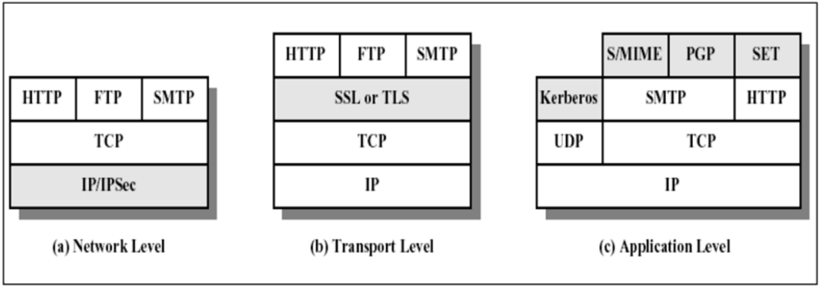
- additional controls may be needed **to prevent an unauthorized party from gaining constructive use** of the "trust" established between the device and the corporate network
- e.g. a malicious user executes a process on the device:
  - that allows them to interact with the corporate network
  - but **without having knowledge of the password or biometric** used by the legitimate user

Security protocols

- a **protocol** is a series of steps carried out by two or more entities
  - HTTP, TCP, SMTP, etc.
- a **security protocol** is a protocol that runs in an untrusted environment and tries to achieve one or more security objectives
  - academic: Needham-Schroeder-Lowe, PAKE, EKE, etc.
  - industrial: S/MIME, PGP, SSH, Kerberos, SSL/TLS, IPSec, etc.

- a **security protocol (cryptographic protocol or encryption protocol)** is an abstract or concrete protocol that performs a security-related function and applies cryptographic methods, often as sequences of cryptographic primitives (*Wikipedia*)

Security protocols



Security protocols

Passwordless authentication

- OAuth (Open Authorization)
- OpenId
- SAML (Security Assertion Markup Language)
- FIDO (The Fast Identity Online)

Certificate authentication

- SSL/TLS (Secure Sockets Layer/Transport Layer Security)

OAuth

- allows an application to authenticate against a server as a user
  - without requiring passwords
  - or any third party server that acts as an identity provider
- it uses a token generated by the server
- provides how the authorization flows most occur, so that a client, such as a mobile application, can tell the server what user is using the service
- use and implement OAuth 1.0a or OAuth 2.0
- OAuth1.0 has been found to be vulnerable to session fixation

OAuth

- OAuth 2.0
  - relies on HTTPS for security
  - is currently used and implemented by API's from companies such as Facebook, Google, Twitter and Microsoft
- OAuth1.0a
  - is more difficult to use because it requires the use of cryptographic libraries for digital signatures
  - since OAuth1.0a does not rely on HTTPS for security it can be more suited for higher risk transactions

OAuth 2.0

- 3 main players in an OAuth transaction:
  - the user
  - the consumer
  - the service provider
- the OAuth Love Triangle

OAuth 2.0

- exemplification scenario:
  - the user - Joe
  - the consumer - Bitly
  - the service provider - Twitter
- Joe would like Bitly to be able to post shortened links to his Twitter stream

OAuth 2.0

Step 1 – The User Shows Intent

- Joe (User): "Hey, Bitly, I would like you to be able to post links directly to my Twitter stream."
- Bitly (Consumer): "Great! Let me go ask for permission."

OAuth 2.0

Step 2 – The Consumer Gets Permission

- Bitly: "I have a user that would like me to post to his stream. Can I have a **clientID** and a **clientSecret**?"

- Twitter (Service Provider): "Sure. Here's the **clientID** and the **clientSecret**."

- the clientID and the clientSecret are used to prevent request forgery
- the consumer uses them so that the service provider can verify it is actually coming from the consumer application

OAuth 2.0

Step 3 – The User Is Redirected to the Service Provider

- Bitly: "OK, Joe. I'm sending you over to Twitter so you can approve. Take this **token** with you."
- Joe: "OK!"
- Bitly directs Joe to Twitter for authorization
- Bitly could pop up a window that looked like Twitter but was really phishing for Joe's username and password
- always be sure to verify that the URL you're directed to is actually the service provider (Twitter, in this case)

OAuth 2.0

Step 4 – The User Gives Permission

- Joe: "Twitter, I'd like to authorize this **request token** that Bitly gave me."
- Twitter: "OK, just to be sure, you want to authorize Bitly to do X, Y, and Z with your Twitter account?"
- Joe: "Yes!"
- Twitter: "OK, you can go back to Bitly and give him this **authorization code** corresponding to the **request token** you gave me."
- Twitter marks the request token as "good-to-go"
- when the consumer requests access, it will be accepted
- so long as it's signed using their shared secret

OAuth 2.0

Step 5 – The Consumer Obtains an Access Token

- Bitly: “Twitter, can I exchange this **authorization code** for an **access token**?”
- Twitter: “Sure. Here's your **access token**.”

OAuth 2.0

Step 6 – The Consumer Accesses the Protected Resource

- Bitly: "I'd like to post this link to Joe's stream. Here's my **access token**!"
- Twitter: "Done!"

OAuth2

https://cloudsignatureconsortium.org/

- ```
  GET https://www.domain.org/oauth2/authorize?
  response_type=code&
  client_id=[CLIENT_ID]&
  redirect_uri=[REDIRECT_URI]&
  scope=service&
  state=12345678&
  lang=en-US&
  appName=Cloud\%20Signature\%20Service
  ```
- ```
  HTTP/1.1 302 Found
  Location: [REDIRECT_URI]?code=FhkXf9P269L8g&
  state=12345678
  ```

OAuth2

- ```
  POST https://service.domain.org/csc/v0/oauth2/token
  Content-Type: application/json
  {
  "grant_type": "authorization_code",
  "code": "FhkXf9P269L8g",
  "client_id": "test",
  "client_secret": "password"
  }
  ```
- ```
  HTTP/1.1 200 OK
  {
  "access_token": "4/CKN69L8gdSYp5_pwH3XlFQZ3ndFhkXf9P2_Ti
  "refresh_token": "_TiHRG-bA H3XlFQZ3ndFhkXf9P24/CKN69L8g
  "token_type": "Bearer",
  "expires_in": 3600
  }
  ```

OAuth2

- `POST https://service.domain.org/csc/v0/credentials/list`
  `Authorization: Bearer 4/CKN69L8gdSYp5_pwH3XlFQZ3ndFhkXf9`
- `HTTP/1.1 200 OK`

  ```
  {
  "credentialIDs":
  [
  "GX0112348",
  "HX0224685"
  ]
  }
  ```

- and so on

OpenId

- an HTTP-based protocol, using JSON
- uses **identity providers to validate that a user is who he says he is**
  - allows the user to **re-use a single identity** given to a trusted OpenId identity provider
  - he/she can **be the same user in multiple websites**
  - **without the need to provide any website the password**, except for the OpenId identity provider
- **has been well adopted** - some of the well known identity providers for OpenId are Stack Exchange, Google, Facebook and Yahoo

SAML

- **uses identity providers** (like OpenId), but it is XML-based and provides more flexibility
- isn't only initiated by a service provider, but **it can also be initiated from the identity provider**
- this allows the user to navigate through different portals while still being authenticated without having to do anything, making the process transparent
- e.g. SAP ERP, SharePoint, etc.

OpenId vs. SAML

- OpenId is considered a secure and often better choice than SAML **for non-enterprise environments**
  - as long as the identity provider is of trust
- SAML is often the choice for **enterprise applications**
  - because there are few OpenId identity providers which are considered of enterprise class
  - meaning that the way they validate the user identity doesn't have high standards required for enterprise identity

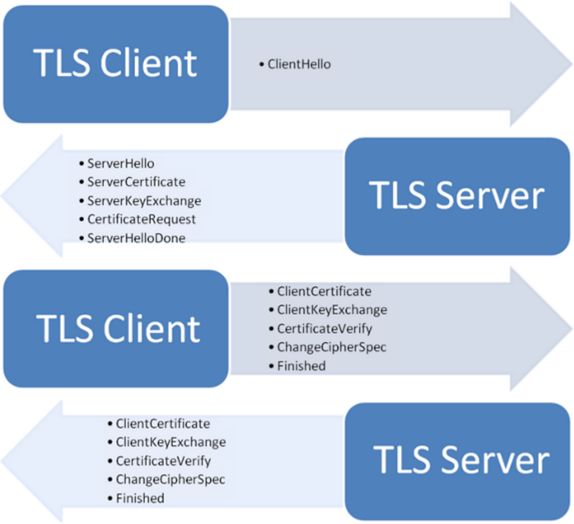
FIDO

- **Universal Authentication Framework (UAF) protocol**
  - passwordless authentication
  - takes advantage of existing security technologies present on devices for authentication
    - fingerprint sensors,
    - cameras (face biometrics),
    - microphones (voice biometrics),
    - Trusted Execution Environments(TEEs),
    - Secure Elements(SEs), etc.
  - plugs-in these device capabilities into a common authentication framework
  - works with **native applications and web applications**
  - based on a **public key cryptography challenge-response** model

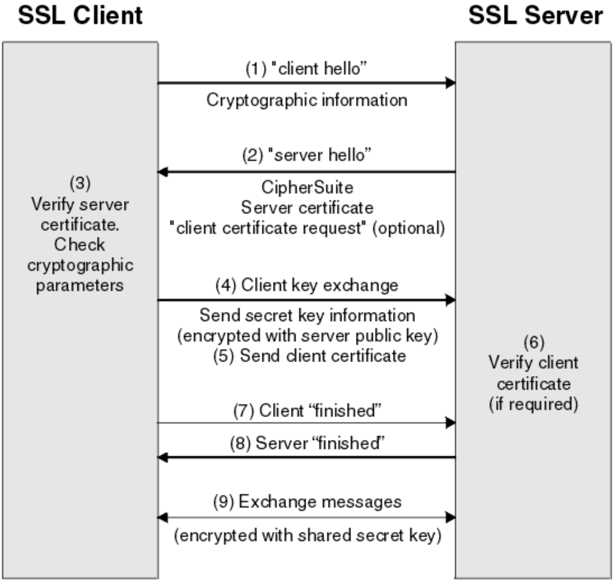
FIDO

- **Universal Second Factor (U2F) protocol**
  - based on a **public key cryptography challenge-response** model
  - allows the addition of a second factor to existing password-based authentication
  - augments password-based authentication using a hardware token (typically USB) that stores cryptographic authentication keys and uses them for signing
  - the user can use the same token as a second factor for multiple applications
  - works with **web applications**
  - provides **protection against phishing** by using the URL of the website to lookup the stored authentication key

SSL/TLS


TLS Client
• ClientHello
• ServerHello
• ServerCertificate
• ServerKeyExchange
• CertificateRequest
• ServerHelloDone
TLS Server
TLS Client
• ClientCertificate
• ClientKeyExchange
• CertificateVerify
• ChangeCipherSpec
• Finished
• ClientCertificate
• ClientKeyExchange
• CertificateVerify
• ChangeCipherSpec
• Finished
TLS Server

SSL/TLS



SSL Client
SSL Server
(1) "client hello"
Cryptographic information
(2) "server hello"
CipherSuite
Server certificate
"client certificate request" (optional)
(3)
Verify server certificate.
Check cryptographic parameters
(4) Client key exchange
Send secret key information
(encrypted with server public key)
(5) Send client certificate
(6)
Verify client certificate
(if required)
(7) Client "finished"
(8) Server "finished"
(9) Exchange messages
(encrypted with shared secret key)

SSL/TLS

client hello

- the **SSL or TLS version**
- the **CipherSuites** supported by the client (in the client's order of preference)
- a **random byte string** that is used in subsequent computations
- may include the data compression methods supported by the client

SSL/TLS

server hello

- the **CipherSuite** chosen by the server from the list provided by the client
- the **session ID**
- another **random byte string**
- the **server's digital certificate**
- **"client certificate request"**
  - if the server requires a digital certificate for client authentication
  - includes a list of the types of certificates supported
  - and the Distinguished Names of acceptable Certification Authorities

SSL/TLS

client key exchange

- the client **verifies the server's digital certificate**
- the client sends the random byte string that enables both the client and the server to compute the secret key to be used for encrypting subsequent message data
- the random byte string itself is **encrypted with the server's public key**

SSL/TLS

if the server has sent a "client certificate request"

- the client sends a **random byte string encrypted with the client's private key**
- together with the **client's digital certificate**
- or a "no digital certificate alert"
- this alert is only a warning, but with some implementations the handshake fails if client authentication is mandatory

SSL/TLS

if a certificate has been received from the client

- the server **verifies the client's certificate**

SSL/TLS

client finished

- the client sends the server a "finished" message
- it is **encrypted with the secret key**
- indicates that the client part of the handshake is complete

SSL/TLS

server finished

- the server sends the client a "finished" message
- it is **encrypted with the secret key**
- indicates that the server part of the handshake is complete

SSL/TLS

- for the duration of the SSL or TLS session, the server and client can now exchange messages that are **symmetrically encrypted with the shared secret key**

SSL/TLS

- **Transport Layer Security (TLS) Parameters**
  http://www.iana.org/assignments/tls-parameters/tls-parameters.xhtml
- **Check cipher suits used/supported by your browser**
  https://cc.dcsec.uni-hannover.de/
- **Decrypting TLS Browser Traffic With Wireshark** – sometimes needed for debug purposes
  https://jimshaver.net/2015/02/11/decrypting-tls-browser-traffic-with-wireshark-the-easy-

SSL/TLS


Acasa
https://www.anaf.ro/anaf/internet/ANAF/acasa/!ut/p/a1/04_Sj9CPykssy0xPLMnMz0vMAfGjzOI9DD3MPlwsjLwsHF2dDBydLfxCLZyBAu7GQAWRQAUGOICjASH94fpR-JS4uxhDFeCxwks_...
ANAF
Agenţia Naţională de Administrare Fiscală
GHIŞEUL.Ro
Servicii Web
Depunere Declaraţie Unică
Autentificare Username
Autentificare Certificat
Cod IBAN Cont TVA
Spaţiu Privat Virtual
Schimbă-ţi casa de marcat! E obligatoriu. Află cum!
Ghidul privind rezidenţa fiscală a persoanelor fizice
Procesul de selecţie a persoanelor fizice cu venituri mari care prezintă risc fiscal
ANAF > Acasa
Sâmbătă, 8 decembrie 2018, în intervalul orar 7⁰⁰ - 9³⁰, portalul ANAF va fi indisponibil, din cauza unor operaţiuni de mentenanţă. Ne cerem scuze pentru eventuale inconvenienţe create şi vă mulţumim pentru înţelegere.
NOUTĂŢI
În M.O. nr. 1005 din 27.11.2018 a fost publicat OPANAF nr. 2937/2018 pentru aprobarea Metodologiei de distribuire a sumelor plătite de contribuabili persoane fizice în contul unic şi de stingere a obligaţiilor fiscale.
CĂUTARE
ASISTENŢĂ CONTRIBUABILI
CALL CENTER: 031.403.91.60
PERSOANE FIZICE
01 Legislaţie
PERSOANE JURIDICE
01 Legislaţie
SERVICII ONLINE
FORMULAR DE CONTACT
01 Declaraţii electronice
ENG
17:13
06/12/2018

SSL/TLS

```
TLSv1.2 Record Layer: Handshake Protocol: Client Hello
    Content Type: Handshake (22)
    Version: TLS 1.0 (0x0301)
    Length: 512
    Handshake Protocol: Client Hello
        Handshake Type: Client Hello (1)
        Length: 508
        Version: TLS 1.2 (0x0303)
        Random: a98b0fcdeca75da08e78b1922472d15b6a784c18ce04e386...
        Session ID Length: 32
        Session ID: f8a6080db4cd61e478c0489c806049efda4b15fe711807c6...
        Cipher Suites Length: 34
        Cipher Suites (17 suites)
            Cipher Suite: Reserved (GREASE) (0x8a8a)
            Cipher Suite: TLS_CHACHA20_POLY1305_SHA256 (0x1303)
            Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
            Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)
            Cipher Suite: TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca9)
            Cipher Suite: TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca8)
            Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)
            Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
            Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c)
            Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
            Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)
            Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
            Cipher Suite: TLS_RSA_WITH_AES_128_GCM_SHA256 (0x009c)
            Cipher Suite: TLS_RSA_WITH_AES_256_GCM_SHA384 (0x009d)
            Cipher Suite: TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)
            Cipher Suite: TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)
            Cipher Suite: TLS_RSA_WITH_3DES_EDE_CBC_SHA (0x000a)
        Compression Methods Length: 1
        Compression Methods (1 method)
            Compression Method: null (0)
```

SSL/TLS

```
⌄ TLSv1.2 Record Layer: Handshake Protocol: Server Hello
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 91
  ⌄ Handshake Protocol: Server Hello
      Handshake Type: Server Hello (2)
      Length: 87
      Version: TLS 1.2 (0x0303)
    > Random: 8fe8104bef1987574d076637a2b0ccc5caea4015bd317a17...
      Session ID Length: 32
      Session ID: 12481462668d8947fb10adf6ced531cf027a8a6e3f2f754a...
      Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)
      Compression Method: null (0)
      Extensions Length: 15
    > Extension: renegotiation_info (len=1)
    > Extension: ec_point_formats (len=2)
    > Extension: extended_master_secret (len=0)
```

SSL/TLS

```
v TLSv1.2 Record Layer: Handshake Protocol: Certificate
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 3666
  v Handshake Protocol: Certificate
      Handshake Type: Certificate (11)
      Length: 3662
      Certificates Length: 3659
    v Certificates (3659 bytes)
        Certificate Length: 1653
      > Certificate: 3082067130820559a003020102021020060516706004 1d59... (id-at-commonName=*.anaf.ro,id-at-organizatic
        Certificate Length: 1169
      > Certificate: 3082048d30820375a003020102020f2006051670030d545d... (id-at-commonName=certSIGN Enterprise CA Clas
        Certificate Length: 828
      > Certificate: 3082033830820220a003020102020620060516700230 0d06... (id-at-organizationalUnitName=certSIGN ROOT C
```

SSL/TLS

```
  ⌄ TLSv1.2 Record Layer: Handshake Protocol: Server Key Exchange
      Content Type: Handshake (22)
      Version: TLS 1.2 (0x0303)
      Length: 333
    ⌄ Handshake Protocol: Server Key Exchange
        Handshake Type: Server Key Exchange (12)
        Length: 329
      ⌄ EC Diffie-Hellman Server Params
          Curve Type: named_curve (0x03)
          Named Curve: secp256r1 (0x0017)
          Pubkey Length: 65
          Pubkey: 04d64548fc9784104990ed171ddaf1338db4da2f4d1e7594...
        ⌄ Signature Algorithm: rsa_pkcs1_sha256 (0x0401)
            Signature Hash Algorithm Hash: SHA256 (4)
            Signature Hash Algorithm Signature: RSA (1)
          Signature Length: 256
          Signature: 6507b42638ce5091006a2d59756d8f4f0901d3aa0191989e...
⌄ Secure Sockets Layer
  ⌄ TLSv1.2 Record Layer: Handshake Protocol: Server Hello Done
      Content Type: Handshake (22)
      Version: TLS 1.2 (0x0303)
      Length: 4
    ⌄ Handshake Protocol: Server Hello Done
        Handshake Type: Server Hello Done (14)
        Length: 0
```

SSL/TLS

```
v TLSv1.2 Record Layer: Handshake Protocol: Certificate Request
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 3968
  v Handshake Protocol: Certificate Request
      Handshake Type: Certificate Request (13)
      Length: 3927
      Certificate types count: 3
    v Certificate types (3 types)
        Certificate type: RSA Sign (1)
        Certificate type: DSS Sign (2)
        Certificate type: ECDSA Sign (64)
      Signature Hash Algorithms Length: 24
    > Signature Hash Algorithms (12 algorithms)
      Distinguished Names Length: 3895
    > Distinguished Names (3895 bytes)
v Secure Sockets Layer
  v TLSv1.2 Record Layer: Handshake Protocol: Encrypted Handshake Message
      Content Type: Handshake (22)
      Version: TLS 1.2 (0x0303)
      Length: 48
      Handshake Protocol: Encrypted Handshake Message
```

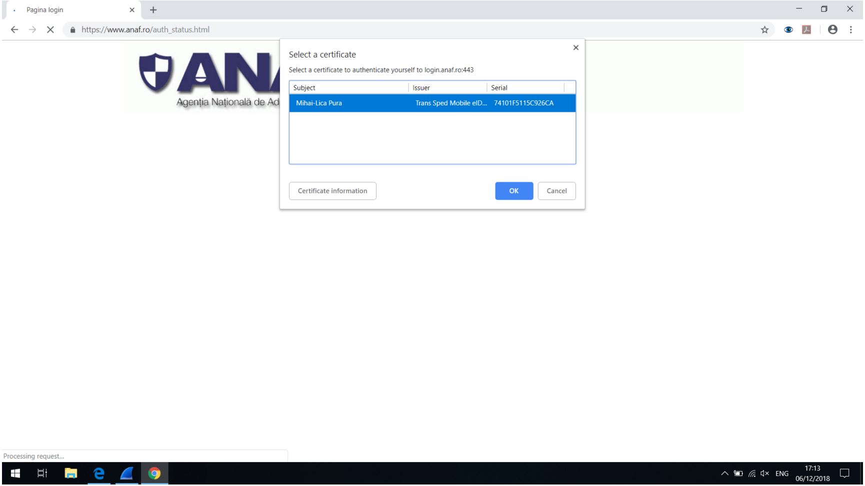
SSL/TLS

```
v Signature Hash Algorithms (12 algorithms)
  v Signature Algorithm: rsa_pkcs1_sha256 (0x0401)
      Signature Hash Algorithm Hash: SHA256 (4)
      Signature Hash Algorithm Signature: RSA (1)
  v Signature Algorithm: SHA256 DSA (0x0402)
      Signature Hash Algorithm Hash: SHA256 (4)
      Signature Hash Algorithm Signature: DSA (2)
  v Signature Algorithm: ecdsa_secp256r1_sha256 (0x0403)
      Signature Hash Algorithm Hash: SHA256 (4)
      Signature Hash Algorithm Signature: ECDSA (3)
  v Signature Algorithm: rsa_pkcs1_sha384 (0x0501)
      Signature Hash Algorithm Hash: SHA384 (5)
      Signature Hash Algorithm Signature: RSA (1)
  v Signature Algorithm: SHA384 DSA (0x0502)
      Signature Hash Algorithm Hash: SHA384 (5)
      Signature Hash Algorithm Signature: DSA (2)
  v Signature Algorithm: ecdsa_secp384r1_sha384 (0x0503)
      Signature Hash Algorithm Hash: SHA384 (5)
      Signature Hash Algorithm Signature: ECDSA (3)
  v Signature Algorithm: rsa_pkcs1_sha512 (0x0601)
      Signature Hash Algorithm Hash: SHA512 (6)
      Signature Hash Algorithm Signature: RSA (1)
  v Signature Algorithm: SHA512 DSA (0x0602)
      Signature Hash Algorithm Hash: SHA512 (6)
      Signature Hash Algorithm Signature: DSA (2)
  v Signature Algorithm: ecdsa_secp521r1_sha512 (0x0603)
      Signature Hash Algorithm Hash: SHA512 (6)
      Signature Hash Algorithm Signature: ECDSA (3)
  v Signature Algorithm: rsa_pkcs1_sha1 (0x0201)
      Signature Hash Algorithm Hash: SHA1 (2)
      Signature Hash Algorithm Signature: RSA (1)
  v Signature Algorithm: SHA1 DSA (0x0202)
```

SSL/TLS

```
Distinguished Names (3895 bytes)
    Distinguished Name Length: 32
    Distinguished Name: (id-at-commonName=ANAF Offline RootCA)
        RDNSequence item: 1 item (id-at-commonName=ANAF Offline RootCA)
    Distinguished Name Length: 105
    Distinguished Name: (id-at-countryName=RO,id-at-organizationName=DigiSign S.A,id-at-organ
        RDNSequence item: 1 item (id-at-commonName=DigiSign Qualified Root CA v2)
        RDNSequence item: 1 item (id-at-organizationalUnitName=DigiSign Root CA)
        RDNSequence item: 1 item (id-at-organizationName=DigiSign S.A)
        RDNSequence item: 1 item (id-at-countryName=RO)
    Distinguished Name Length: 106
    Distinguished Name: (id-at-countryName=RO,id-at-organizationName=DigiSign S.A,id-at-organ
        RDNSequence item: 1 item (id-at-commonName=DigiSign Qualified Public CA)
        RDNSequence item: 1 item (id-at-organizationalUnitName=DigiSign Public CA)
        RDNSequence item: 1 item (id-at-organizationName=DigiSign S.A)
        RDNSequence item: 1 item (id-at-countryName=RO)
    Distinguished Name Length: 110
    Distinguished Name: (id-at-commonName=Trans Sped QCA,id-at-organizationalUnitName=Trans S
        RDNSequence item: 1 item (id-at-countryName=RO)
        RDNSequence item: 1 item (id-at-organizationName=Trans Sped SRL)
        RDNSequence item: 1 item (id-at-organizationalUnitName=Trans Sped Certification Author
        RDNSequence item: 1 item (id-at-commonName=Trans Sped QCA)
    Distinguished Name Length: 112
    Distinguished Name: (id-at-commonName=Trans Sped Qualified CA II,id-at-organizationalUnit
        RDNSequence item: 1 item (id-at-countryName=RO)
        RDNSequence item: 1 item (id-at-organizationName=Trans Sped SRL)
        RDNSequence item: 1 item (id-at-organizationalUnitName=Individual Subscriber CA)
        RDNSequence item: 1 item (id-at-commonName=Trans Sped Qualified CA II)
    Distinguished Name Length: 109
```

SSL/TLS


Pagina login
https://www.anaf.ro/auth_status.html
Agenția Națională de Ad
Select a certificate
Select a certificate to authenticate yourself to login.anaf.ro:443
Subject
Issuer
Serial
Mihai-Lica Pura
Trans Sped Mobile eID...
74101F5115C926CA
Certificate information
OK
Cancel
Processing request...
ENG
17:13
06/12/2018

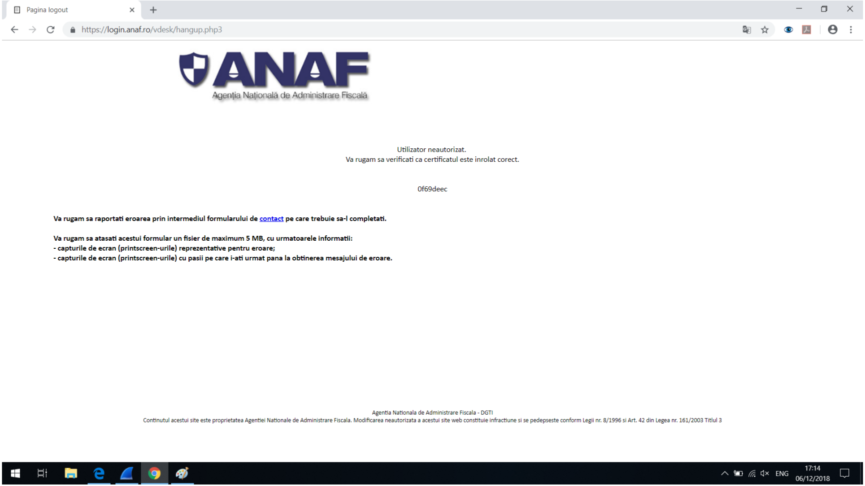
SSL/TLS

```
v TLSv1.2 Record Layer: Handshake Protocol: Multiple Handshake Messages
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 3088
  v Handshake Protocol: Certificate
      Handshake Type: Certificate (11)
      Length: 2698
      Certificates Length: 2695
    v Certificates (2695 bytes)
        Certificate Length: 1464
      > Certificate: 308205b43082049ca003020102020874101f5115c926ca30... (id-at-commonName=Mihai-Lica Pura,id-at-seria
        Certificate Length: 1225
      > Certificate: 308204c5308203ada003020102020a611e6e4c0000000000... (id-at-commonName=Trans Sped Mobile eIDAS QCA
  v Handshake Protocol: Client Key Exchange
      Handshake Type: Client Key Exchange (16)
      Length: 66
    v EC Diffie-Hellman Client Params
        Pubkey Length: 65
        Pubkey: 0484ff7f17696c521f9885d002aa7f5eeca09f46a685cab4...
  v Handshake Protocol: Certificate Verify
      Handshake Type: Certificate Verify (15)
      Length: 260
    > Signature Algorithm: rsa_pkcs1_sha256 (0x0401)
      Signature length: 256
      Signature: cc56a1c15654af71854ce6fa90570c89f3f5a428d598d4f4...
```

SSL/TLS

```
⌄ TLSv1.2 Record Layer: Handshake Protocol: Finished
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 64
  ⌄ Handshake Protocol: Finished
      Handshake Type: Finished (20)
      Length: 12
      Verify Data
```

SSL/TLS


Pagina logout
https://login.anaf.ro/vdesk/hangup.php3
ANAF
Agenția Națională de Administrare Fiscală
Utilizator neautorizat.
Va rugam sa verificati ca certificatul este inrolat corect.
0f69deec
Va rugam sa raportati eroarea prin intermediul formularului de contact pe care trebuie sa-l completati.
Va rugam sa atasati acestui formular un fisier de maximum 5 MB, cu urmatoarele informatii:
- capturile de ecran (printscreen-urile) reprezentative pentru eroare;
- capturile de ecran (printscreen-urile) cu pasii pe care i-ati urmat pana la obtinerea mesajului de eroare.
Agentia Nationala de Administrare Fiscala - DGTI
Continutul acestui site este proprietatea Agentiei Nationale de Administrare Fiscala. Modificarea neautorizata a acestui site web constituie infractiune si se pedepseste conform Legii nr. 8/1996 si Art. 42 din Legea nr. 161/2003 Titlul 3
ENG
17:14
06/12/2018

References

- Keith Martin McCrea, **Introduction to Cryptography and Security Mechanisms**
  www.isg.rhul.ac.uk/static/msc/teaching/ic2/slides05/Unit2.8.ppt
- PCI Security Standards Council, **Multi-Factor Authentication**, February 2017
  https://www.pcisecuritystandards.org/pdfs/Multi-Factor-Authentication-Guidance-v1.pdf
- DRAFT **NIST Special Publication 800-63B Digital Authentication Guideline** (accessed: November 07, 2016)
  https://pages.nist.gov/800-63-3/sp800-63b.html

References

- Rob Sobers, **What is OAuth? Definition and How it Works**
  https://www.varonis.com/blog/what-is-oauth/
- Aaron Parecki, **OAuth 2 Simplified**
  https://aaronparecki.com/oauth-2-simplified/
- Ton van Deursen, **Introduction to Security Protocols**
  http://satoss.uni.lu/members/patrick/1stsxxm.pdf
- IBM Knowledge Center, **An overview of the SSL or TLS handshake**
  https://www.ibm.com/support/knowledgecenter/SSFKSJ_7.1.0/com.ibm.mq.doc/sy10660_.htm

References

- Matthew Green, **On the (provable) security of TLS**
  https://blog.cryptographyengineering.com/2012/09/06/on-provable-security-of-tls-part-1/
  https://blog.cryptographyengineering.com/2012/09/28/on-provable-security-of-tls-part-2/
- Alvaro Castro-Castilla, **Traffic Analysis of an SSL/TLS Session**
  http://blog.fourthbit.com/2014/12/23/traffic-analysis-of-an-ssl-slash-tls-session
- Kaushal Kumar Panday, **SSL Handshake and HTTPS Bindings on IIS**
  https://blogs.msdn.microsoft.com/kaushal/2013/08/02/ssl-handshake-and-https-bindings-on-iis/