Challenge-response authentication

Nordnet

Keep it all public, except the key!

Abstract

Nordnet has designed and implemented a modern scalable secure challenge-response based authentication method. This method allows Nordnet customers to use their smartphone as a safe key holder which eliminates the need for sharing any credentials with Nordnet servers. This authentication method, which uses RSA cryptography with long keys and large space SHA hash function, has been widely used in 3 Scandinavian countries without any security incidents.

Use case:

Customers[1] want to login to Nordnet (web and mobile application) without sharing any form of credentials with anyone, including Nordnet.

The product of that login is something akin to a session (or an access token) using which the customer should be able to use all Nordnet services (after authorization of course).

Customer is not interested in using any 3rd party services during the login process. So the whole **login process** must be handled using Nordnet's web and/or mobile application.

Security requirements:

- **No credential sharing:** Customers don't want to share any credentials with Nordnet. So symmetric cryptography and password based login can not be used. Asymmetric cryptography, however, allows the parties to keep their private keys (credentials) to themselves.
- **Identification:** Nordnet needs to get an identification claim from the customer SSN for example.
- **Authentication:** Nordnet needs to make sure to log the right customer in and validate the identification claim they make. So Nordnet needs to make sure the claimed identity is owned by the one claiming it.
- **Integrity:** The information sent from the customer, must be kept intact before being received by Nordnet.
- **Non replayable:** No one should be able to replay the communication between the customer and Nordnet in order to login again.
- **Available:** The only prerequisites to use this authentication method is having an internet connection, a smartphone and an installation of Nordnet mobile application. The method should work regardless of conditions like time. Also, it should be able to handle all Nordnet daily logins in terms of load, and even more.

UX requirements:

- Customers do not want to be asked for any identifiers, like username, during login.
- Customers want to be able to login to Nordnet web and mobile applications using this method.
- Customers want to use their smartphone as the credential holder.
- Customers want to be the only one who can use their credentials even though their phone might be used by someone else. So a passcode or biometric protection must be required to be able to use the credentials.
- Upon explicit request or misuse, the credentials must be invalidated.

Solution:

Asymmetric challenge-response authentication:
In simple terms, the authentication server sends a challenge to the customer. The customer should respond to that challenge correctly and quickly. The response will be checked using a previously established shared understanding. Also, no challenge will be sent twice and no response will be accepted more than once. So there are two phases involved in this authentication method.

[1] In this document words customer and client are used almost interchangeably. When we say "Nordnet gets an SSO from a customer", it means that the customer's SSO will be communicated to Nordnet through a software client (Nordnet web or mobile application).

Mohammadmahdi.amini@nordnet.se

Challenge-response authentication

Nordnet

A. Enrollment phase:

a) Establish a verified customer identifier (CID) like SSN.

b) Generate a random device identifier (DID) for the customer's (key holder) device.

c) Generate a public key (PUK_C) and private key (PRK_C) pair on the customer's device. This key pair must be of a long enough length and low enough lifetime. Also, PRK_C must be protected by a passcode or biometric check (face/fingerprint).

d) Send CID, DID and PUK_C to the authentication server.

e) Authentication server keeps track of the combination above in a durable data store. Each device can have only one record in this data store.

B. Login phase:

a) Customer sends a login request to the authentication server.

b) Authentication server sends a nonce back. This nonce will expire in 30 seconds. The authentication server should keep track of produced nonces in a durable data store.

c) Customer constructs a message (M_{login}) including the nonce and DID.

d) Customer generates a hash ($Hash(M_{login})$) from M_{login} using a hash function with large enough state space.

e) Customer sings the hash value using PRK_C. Here the customer needs to present PRK_C-Pass or pass the biometric check. The result is $Sign(Hash(M_{login}))$.

f) Customer sends the signed value alongside the M_{login} to the authentication server.

g) Authentication server extracts the nonce from M_{login}. It should be checked against the data store for existence and also lifetime. If the nonce still exists and is valid, we can be sure that the message hasn't been seen before. Otherwise, the login should fail right away.

h) Authentication server extracts DID from M_{login}. Then use it to find the corresponding PUK_C. Now PUK_C is used to verify the signature (decrypt $Sign(Hash(M_{login}))$ in addition to getting access to $Hash(M_{login})$. If the verification goes well, the authentication of the message has been achieved which means that we have established who is trying to login.

i) Now the authentication server creates its own version of the hash value using M_{login} ($Hashserver(M_{login})$). If $Hashserver(M_{login})$ is equal to the $Hash(M_{login})$ gained from signature verification, the integrity of the message is also verified.

j) Now the authentication server can send an access token to the customer as a sign of successful login.

k) Regardless of the result of the login process, any nonce visited by the authentication server, should be wiped from the data store (considered invalid for future login attempts). This satisfies the non replayable security requirement. The wipe of nonce is better done at the end of step g.

Login phase happy flow pseudo code:

Client side:

Get a nonce from authentication server
M_{login} = CID + DID + nonce
hash = $Hash(M_{login})$
signature = $Encrypt(hash, PRK_C)$
request = signature + M_{login}
Send request to authentication server

Server side:

{signature, M_{login}} = request
{CID, DID, nonce} = M_{login}
NonceDataStore.isValid(nonce) ?= TRUE
PUK_C = KeyDataStore.fetchPublicKey(DID)
$Hash_{client}$ = $Decrypt(signature, PUK_C)$
$Hash_{server}$ = $Hash(M_{login})$
$Hash_{client}$?= $Hash_{server}$
return access token (login succeed)

Mohammadmahdi.amini@nordnet.se

Challenge-response authentication

Nordnet implementation:

Here we review some of the notable implementation details which were decided in the context of Nordnet needs.

Enrollment:
The customer needs to install the Nordnet application on a smartphone. Which results in an installation identifier (InsID). InsID is unique per application installation so it can be used as DID. Using InsID, rather than some other more stable identifiers like a MAC address, helps with preserving privacy of customers. This phone, and the installed Nordnet application, act as the key holder.

In order to establish a valid CID, the customer needs to login to the Nordnet application using a different login method. In Scandinavian countries at least one 3rd party (called eId) exists which provides secure digital authentication. When a customer logs in to Nordnet using an eId, Nordnet gets a SSN which can be used to find the corresponding CID from the Nordnet customer database. Using these 3rd parties to verify the identity of the customer during enrollment is equivalent to signing a SSL certificate using a more well known (root) certificate in public key infrastructure. Basically we need to rely on a chain of trust and add our link to the chain.

During enrollment a pair of 4096 bit RSA keys with a lifetime of 2 years (for now) are created and saved in a key store. The key store provided by Android and IOS promises no leakage of keys outside the phone.

Both Android and IOS allow for protecting the access to those keys using a pass code and/or biometric checks. Excessive failed attempts to access those keys, results in them being automatically wiped from the phone.

Also, customers can use the Nordnet website to remove/invalidate any of the keys they have enrolled. They can see a list of phone names from which they can choose one to be invalidated.

Login:
After establishing an enrollment and communicating a public key with the Nordnet backend, a customer can use the aforementioned device (installation) to login to Nordnet web and mobile applications.

Nordnet relies on two nonces, signing nonce and login nonce, as challenge. Both of them are short lived standard randomly produced UUIDs.

Login nonce has been introduced to support login to the Nordnet **web** application using the Nordnet mobile application.

When a user tries to login to the Nordnet web application, the login nonce is not exposed to the mobile application. In fact, login nonce won't leave the web browser at all.

The signing nonce, as the name suggests, is signed by the Nordet mobile application (key holder). When logging into the Nordnet web application, signing nonce is communicated to the Nordnet mobile application using a QR code which can be scanned using the customer's smartphone camera.

The QR code shown on the Nordnet web application changes every 3 seconds to avoid over the shoulder and social engineering attacks. This means that the Nordnet web application requests a new challenge every 3 seconds.

Using login nonce before signing nonce is signed, results in a failure. The hashing function used (so far) is SHA-256.

During the login phase, when everything checks out, the Nordnet backend will generate a JWT (access token). This JWT is sent back to the Nordnet web or mobile application as a sign of successful login.

Anyone who provides a valid login nonce, will get a JWT for the customer who has signed the corresponding signing nonce. The design of the login flow is such that these two people are almost impossible to be different.

Mohammadmahdi.amini@nordnet.se