



Autenticação 2

Smartcards

Não é muito utilizável, porque é necessário ter um leitor específico.

O cartão tem um processador e memória que consegue processar chaves criptográficas. Pode ser utilizado com NFC ou simplesmente por contact.

Muitas vezes o smart card possui o terceiro fator de autenticação que é o pin.

Não é fácil mudar as implementações dos smartcards, porque há um processo de acreditação nas implementações. Por isso, em muitas aplicações, como o cartão cidadão, ainda é usado RSA. Quando as cenas são implementadas ao nível do hardware, é esperado que haja durabilidade.

One-time tokens

É um processador mais simples que o smartcard. Dá-se a pessoa um mecanismo que pode dá um código. Agora além da password que a pessoa configurou há este mecanismo que gera um número.

Protocolo

O protocolo típico é não interativo. Ele usa criptografia simétrica, onde a chave é pré-partilhada com o servidor. Periodicamente o token gera um message authentication code da hora atual. Quando o servidor pede a password ele usa a “janela temporal” para verificar que este é um mac relativamente recente.

Desvantagem

Pode acontecer o phishing e Man-In-The-Middle. Nunca podemos garantir que não ninguém no meio.

O servidor de armazenar chaves secretas para cada utilizador. É um problema claro de escalabilidade.

One-time passcode

Solução para o one-time passcode.

- Utilizar App noutro dispositivo para gerar códigos one-time
- Utilizar app noutro dispositivo para fazer challenge-response

A vantagem é que temos menos um dispositivo para guardar.

A desvantagem é que há menor garantia de independência entre fatores. Ou seja, o telemovel pode ser roubado.

Caso mais simples

Um código é enviado por SMS e introduzido com password.

Muitas vezes utiliza-se para aumentar as garantias de segurança.

Ainda há o problema de que não sabemos se é a pessoa correta que está utilizando o telemóvel.

Conclusão

Ineficiente em cenários críticos, mas tem uma boa relação usabilidade vs segurança em contextos do dia dia.

Biometria

desvantagens: problemas de privacidade, direito ao esquecimento. No futuro se não quisermos ser identificados podemos ser a força.

vantagens: não é transferível, usabilidade ideal, pode dar garantias

Processo

O processo para autenticação da biometria consiste em dois passos: registo e autenticação. O registo consiste em:

- Tirar amostras da biometria
- Construir templates

A autenticação:

- Tirar a amostra bruta
- Fazer match da amostra bruta com templates

Algumas características da amostra precisa estar nos templates.

Ainda temos várias opções para match:

- **Pode-se apenas enviar o caso de match.** Nesta situação o servidor confia no aparelho para permitir o acesso. Precisa-se de alguma forma para demonstrar que este aparelho é confiável (e.g autenticação do hardware). A vantagem é que o servidor não guarda as informações do cliente.
- **Pode-se enviar apenas os dados brutos.** Neste cenário o servidor tem ainda de confiar no aparelho para tirar as amostras. Em todo caso o **servidor armazena todos os dados biométricos** da pessoa.
- **O servidor recebe o template para match (não os dados brutos).** O servidor confia no computador do user para recolher amostras. O servidor pode não armazenar os dados brutos, mas ainda tem o template da biometria.

A taxa de falso positivos para biometrias é bem baixa, mas a taxa de falso negativo são se altas podem atrapalhar a usabilidade.

Ainda pode-se ter **spoofing** de dados biométricos como o replay attack. Pode-se usar fatores biométricos adicionais que garantem que os dados provém de um organismo vivo (e.g verificar temperatura, pulso).

HTTPAuth

Basic access authentication - Wikipedia

In the context of an HTTP transaction, basic access authentication is a method for an HTTP user agent (e.g. a web browser) to provide a user name and password when making a request. In basic HTTP authentication, a request contains a header field in the form of Authorization: Basic , where credentials is the Base64 encoding of [W https://en.wikipedia.org/wiki/Basic_access_authentication](https://en.wikipedia.org/wiki/Basic_access_authentication)

Este tipo de autenticação não garante de nenhum modo a confidencialidade das informações fornecidas.

A resposta do servidor para o acesso de um site, por exemplo, era o pedido de autenticação. O browser mostrava o formulário e como esperado o user deveria fornecer as credenciais. No entanto, as credenciais são apenas codificadas em base64 (**isto não é encriptar a password nem realizar hash!**). A confidencialidade pode ser garantida apenas se o utilizador estiver usando **HTTPS**.


Como a sessão deve ser mantida, estes dados são armazenados na cache do user e enviados para todos os requests enviados ao servidor.

O log-out implicava o browser ser fechado e gerir multiplas contas do mesmo utilizador era complicado.

Token de sessão

Session vs Token Based Authentication - GeeksforGeeks

The Session and Token-based Authentication methods are used to make a server trust any request sent by an authenticated user over the internet. In this way, a user can interact with their account

 <https://www.geeksforgeeks.org/session-vs-token-based-authentication/>



Quando o user faz login o server envia para ele um token de sessão. Este token pode ser armazenada em cookies e é enviado ao server em todos os pedidos.

Pode-se realizar ataques para roubar tokens de sessão (**Session hijacking**). Pode ser conseguido por XSS, falha de logout (token não é invalidado pelo servidor). Uma **mitigação possível** é ligar o token a máquina (e.g endereço ip).

Ainda pode-se fazer **token fixation**. O atacante inicia uma sessão com poucos privilégios e recebe um token. Depois o atacante “convence” a pessoa a fazer login com o mesmo token (e.g um URL com o token embutido). Assim quando o user fizer login e tiver o mesmo token, o atacante passa a ter privilégios elevados. A mitigação para isto é nunca elevar privilégios e nunca fazer login sem criar um token novo.

Perguntas (By Diana Freitas)

O que é HTTPAuth e quais eram as suas limitações?

▼ Resposta

HTTPAuth era a realização do login de um user através do envio das credenciais por meio de HTTP. O problema desta abordagem é que o HTTPAuth não garante confidencialidade. Ele apenas codifica a password em base64. Esta codificação não é uma encriptação nem uma hash. A confidencialidade pode ser garantida pelo envio dessas credenciais em HTTPS. Outro problema é que estas credenciais devem ser enviadas em todos os requests feitos ao servidor.

Outro problema consiste que estas credenciais são guardadas no lado do cliente em cache. O log-out era apenas possível fechando o browser e a administração de multiplas contas de clientes era difícil.

Explique a utilização de Tokens de Sessão nas Sessões Web.

▼ Resposta

Tokens de sessão são gerados pelo servidor e enviados ao user no ato do login. Todo request feito pelo user na sessão contém o token de sessão.

Indique possíveis ataques a tokens de sessão.

▼ Resposta

Há dois tipos de ataques muito comuns: session hijacking e token fixation. O token fixation consiste no atacante fazer login na plataforma e conseguir um token (T). Depois o atacante “convence” outro utilizador a fazer login na plataforma com o mesmo token (T) (e.g fornecer um URL que possui o token embutido). Assim quando a vítima fizer login o atacante terá os mesmos privilégios do outros user.

O session hijacking consistem em roubar o token por meio de XSS, falha de logout etc. Uma mitigação é ligar o user a um endereço ip.