

Segurança Computacional

Autenticação

Prof. Carlos Maziero

DInf UFPR, Curitiba PR

Julho de 2019

Conteúdo

- 1 Conceitos básicos
- 2 Usuários e grupos
- 3 Estratégias de autenticação
- 4 Senhas
- 5 Senhas descartáveis
- 6 Estratégias desafio-resposta
- 7 Técnicas biométricas
- 8 Infraestruturas de autenticação
- 9 Kerberos

Conceitos básicos

Conceitos básicos

Autenticação

Provar a identidade das diversas entidades de um sistema computacional

Objetivos:

- Identificar os usuários para o sistema
- Identificar os sistemas para o usuário
- Identificar o sistema para outros sistemas
- Garantir a origem de um software, *driver*, etc

Sessão de trabalho

Etapas típicas do uso de um sistema:

- 1 **Autenticação** do servidor
- 2 Início de sessão (login)
 - 1 **Autenticação** do cliente
 - 2 criação de processo(s)
- 3 Uso do sistema pelos processos
- 4 Fim de sessão (logout, logoff)

Usuários e grupos

Usuários e grupos

Usuários e grupos

UID – User IDentifier

Identificador do usuário no sistema (inteiro ou string)

Em sistemas UNIX, os UIDs estão registrados em `/etc/passwd`

O UID é usado para rotular:

- Entidades (processos, threads, transações)
- Recursos (arquivos, áreas de memória)

Usuários e grupos

Usuários são também classificados em grupos:

GID – Group IDentifier

Identificador do grupo no sistema (inteiro ou string)

Em sistemas UNIX, os GIDs estão registrados em `/etc/group`

Um usuário pode pertencer a mais de um grupo

Credenciais em UNIX

Credenciais

Informações que relacionam um processo ao usuário e grupo

Cada processo tem 3 UIDs:

- 1 UID: UID real, identifica o dono do processo
- 2 EUID: *Effective* UID, usado em decisões de acesso
- 3 SUID: *Saved* UID, usado em situações especiais

Similarmente, cada processo tem três GIDs:

- 1 Real GID, Effective GID, Saved GID

Usuários internos e anônimos

Usuários fictícios que “possuem” recursos públicos:

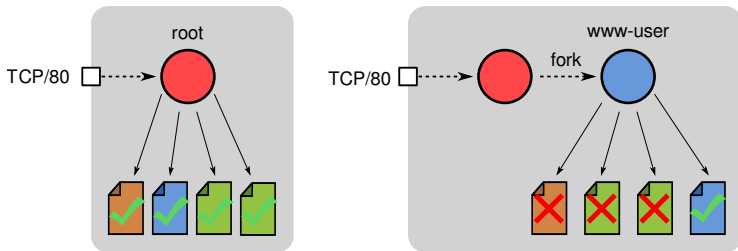
- *guest, nobody, anonymous, ...*

Usuários fictícios responsáveis por sub-sistemas críticos:

- `mail`: sub-sistema de e-mail
- `httpd`: sub-sistema de serviço Web
- `daemon`: diversos serviços de *background*
- `syslog`: gerenciador de registros (*logs*)
- `lp`: sub-sistema de impressão
- ...

Usuários internos e anônimos

Usuários internos são usados para aumentar a segurança



Aplicação do princípio do **privilegio mínimo**

Estratégias de autenticação

Estratégias de autenticação

Três grandes grupos:

SYK - *Something You **K**now*

SYH - *Something You **H**ave*

SYA - *Something You **A**re*

SYK – *Something You Know*

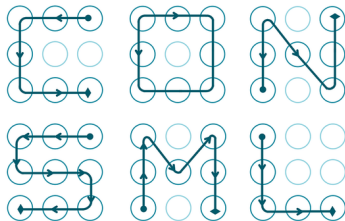
Informações conhecidas pelo usuário (memorizáveis)

Consideradas técnicas de autenticação fracas

Informação é fácil de transferir ou roubar

Exemplos:

- Login & senha
- PIN
- Padrão



SYH – *Something You Have*

Posse de informação complexa ou dispositivo

Mais robustas que as técnicas SYK

Dispositivos materiais podem ser furtados

Exemplos:

- Certificados digitais
- Chaves criptográficas
- Cartões, *tokens*



SYA – *Something You Are*

Características intrínsecas do usuário (biometria)

Técnicas mais complexas de implementar

Consideradas mais robustas que as anteriores

Exemplos:

- Dados biométricos
- Endereços de rede



Autenticação multi-fator

Uso conjunto/complementar de SYK, SYH, SYA

- Sistema militar: senha + íris
- Sistema bancário: senha + cartão

Podem ser usadas de forma gradativa:

- 1 autenticação SYK para acessar o sistema
- 2 autenticação SYH para operações mais sensíveis
- 3 autenticação SYA para operações críticas

Uso crescente na Internet (2FA, U2F, CTAP)

Senhas

Senhas

Senha: informação secreta **memorizada** pelo usuário (SYK)

Procedimento básico:

- 1 A senha (s) é previamente cadastrada no sistema
- 2 Para iniciar a sessão, usuário informa login e senha s'
- 3 Se $s' = s$ usuário pode continuar

Problema: como armazenar as senhas no sistema?

A base de senhas pode ser exposta em um ataque!

Senhas

Solução: armazenar somente o **hash criptográfico** da senha

- 1 Usuário informa sua senha s
- 2 O hash da senha $h(s)$ é armazenado no sistema
- 3 Para iniciar a sessão, usuário informa login e senha s'
- 4 O hash de s' é calculado e comparado com $h(s)$
- 5 Se $h(s') = h(s)$ o usuário está autenticado

Mas... e se a base de *hashes* for exposta? 🙄

Segurança da base de hashes

Hashes criptográficos são difíceis de inverter:

teste	→	d3aa349c8d932ea7
60314c671e469004	→	?????

As senhas “cifradas” com hash não podem ser calculadas

Ataque do dicionário

Cifrar palavras conhecidas (ou combinações delas) e comparar com os hashes das senhas armazenados no sistema

Hashes de dicionários podem ser precomputados!

- <https://crackstation.net>
- <https://hashkiller.co.uk>

Contra medidas:

- Restringir o acesso aos *hashes*
- Impedir o uso de senhas fracas
- **Salgar as senhas...** 😬

“Salgando” senhas...

Sal:

- Número aleatório concatenado à senha (*nonce*)
- Armazenado junto com a senha, na base
- Cada senha pode gerar diversos hashes
- Dificulta ataques de dicionário (inibe pré-cálculo)

No UNIX:

- String com 8 caracteres no conjunto [a-zA-Z0-9./]
- $\sim 2,8 \times 10^{14}$ hashes por palavra do dicionário

“Salgando” senhas...

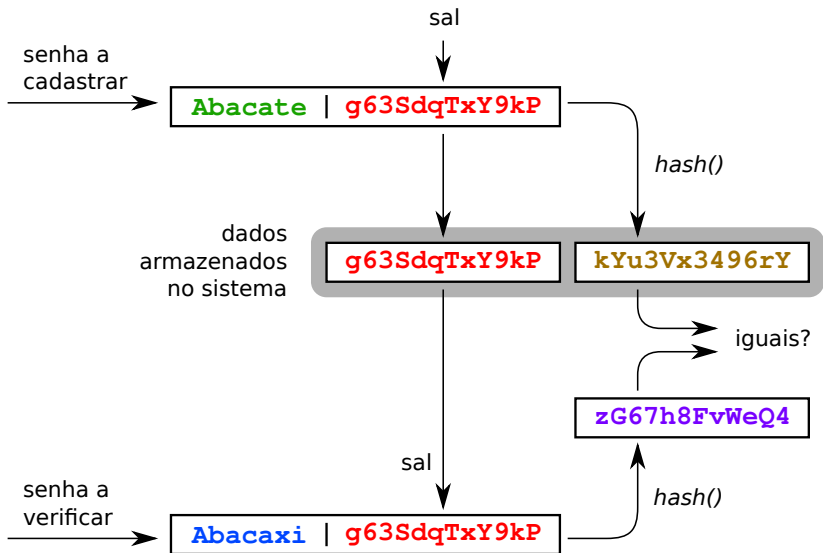
Para cadastrar uma senha s :

- 1 Escolher sal x aleatório (de tamanho fixo)
- 2 Concatenar s e x e calcular hash: $h(x \parallel s)$
- 3 Armazenar sal x e hash juntos na base

Para validar uma senha s' informada pelo usuário:

- 1 ler o valor de sal x armazenado
- 2 Concatenar s' e x e calcular hash: $h(x \parallel s')$
- 3 se $h(x \parallel s') = h(x \parallel s)$ a senha é válida

“Salgando” senhas...



Senhas em UNIX

/etc/passwd: arquivo de usuários

```
1 root:x:0:0:root:/root:/bin/bash
2 daemon:x:1:1:daemon:/usr/sbin:/bin/sh
3 maziero:x:1000:1000:Carlos Maziero,,,:/home/maziero:/bin/bash
```

/etc/group: arquivo de grupos

```
1 cdrom:x:24:daniel,henrique,maziero
2 audio:x:29:pulse,maziero,henrique,daniel
```

/etc/shadow: arquivo de senhas

```
1 root:$6$PjN9/9Fn$i2G2Ki53ed{...}:14816:0:99999:7:::
2 maziero:$6$6cAvNlne$HXxcvd4{...}:14813:0:99999:7:::
```

Senhas descartáveis

Senhas descartáveis

Senhas convencionais são vulneráveis:

- Podem ser escritas em algum meio vulnerável
- Podem ser roubadas sem que o usuário perceba

Proposta: **senha descartável**

- OTP – *One-Time Password*
- Usada uma só vez, depois perde a validade

Implementações:

- Listas de senhas
- Algoritmos (hash chains)
- Security tokens

Listas de senhas

TAN: *Transaction Authentication Numbers*

1	001 342232	002 038234	003 887123	004 545698	005 323241
2	006 587812	007 232221	008 772633	009 123812	010 661511
3	011 223287	012 870910	013 865324	014 986323	015 876876
4	...				

Entregues ao cliente de várias formas:

- Folhas impressas
- SMS sob demanda (mTAN - Mobile TAN)

Muito usados em sistemas bancários

Transaction Authentication Numbers

FRENTE



CHAVE DE SEGURANÇA BRADESCO

Dúvidas ou cancelamento,
ligue para o Fone Fácil Bradesco.
Este cartão é pessoal e intransferível.

Ref.: 0123456789-1

Número de referência

VERSO

Nº Chave	Nº Chave	Nº Chave	Nº Chave	Nº Chave	Nº Chave	Nº Chave
01 111	11 765	21 069	31 456	41 846	51 433	61 422
02 455	12 951	22 081	32 771	42 694	52 246	62 267
03 564	13 623	23 234	33 311	43 484	53 296	63 942
04 921	14 119	24 573	34 925	44 524	54 527	64 969
05 231	15 861	25 993	35 224	45 935	55 369	65 644
06 222	16 323	26 469	36 561	46 281	56 145	66 549
07 996	17 936	27 862	37 919	47 661	57 655	67 516
08 626	18 290	28 363	38 453	48 672	58 261	68 512
09 032	19 666	29 924	39 536	49 156	59 526	69 135
10 580	20 022	30 243	40 714	50 015	60 282	70 232

Nunca forneça mais que um: chave por transação.

Posição Chave de Segurança

Transaction Authentication Numbers



Bradesco

Agência:

Conta: -

2 Ativação do cartão chave.



Nosso sistema está atualizando o Cartão de Segurança de todos os cliente, e identificamos que seu cartão foi desativado. Por favor recadastre seu Cartão de Segurança ativando todas as chaves abaixo.

Nº	Chave	Nº	Chave	Nº	Chave	Nº	Chave	Nº	Chave	Nº	Chave	Nº	Chave
01		11		21		31		41		51		61	
02		12		22		32		42		52		62	
03		13		23		33		43		53		63	
04		14		24		34		44		54		64	
05		15		25		35		45		55		65	
06		16		26		36		46		56		66	
07		17		27		37		47		57		67	
08		18		28		38		48		58		68	
09		19		29		39		49		59		69	
10		20		30		40		50		60		70	

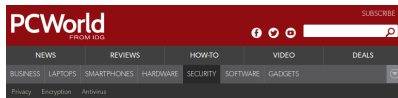


Número de Referência:



Avançar

Mobile TANs



NEWS

Criminals Pay Top Money for Hackable Nokia Phone



By Jeremy Kirk

IDG News Service | APRIL 20, 2009 05:40 AM PT




Personal Tech

€25k for an old Nokia handset?

Scammers pay through the nose for old tech

By Bill Ray 21 Apr 2009 at 13:42

29  SHARE ▼

Scammers are reportedly prepared to pay €25,000 for German Nokia 1100 handsets, on the basis that they can be reprogrammed to intercept SMS messages and thus crack banking security.

Senhas descartáveis por algoritmos

Ideia:

- Gerar senhas únicas por algoritmo
- Devem ser fáceis de gerar, mas difíceis de adivinhar

Estratégias:

- Construir uma cadeia de hashes (Lamport OTP)
- Senha secreta e contador incremental (HOTP)
- Senha secreta e horário atual (TOTP)

O algoritmo OTP de Lamport

Passo 1: criada cadeia de hashes $s_0, s_1, s_2, \dots, s_n$

- s_0 é aleatório e $\forall i > 0 \ s_i = \text{hash}(s_{i-1})$

$$\xrightarrow{\text{random}} s_0 \xrightarrow{\text{hash}} s_1 \xrightarrow{\text{hash}} s_2 \xrightarrow{\text{hash}} \dots \xrightarrow{\text{hash}} s_{n-1} \xrightarrow{\text{hash}} s_n$$

Passo 2: cliente recebe s_0 e servidor recebe s_n

Passo 3: em um novo acesso:

- O cliente informa o valor de s_{n-1} ao servidor
- O servidor compara $\text{hash}(s_{n-1})$ com s_n
- Se forem iguais, o cliente está autenticado
- O servidor armazena s_{n-1} para o próximo acesso

O algoritmo OTP de Lamport

passo	client	rede (Mallory vê)	servidor
1	recebe s_0		recebe s_n
2	calcula s_{n-1}		
3	envia s_{n-1}	s_{n-1}	recebe s_{n-1}
4			$hash(s_{n-1}) = s_n ?$
5			se ok, descarta s_n
...			
i	calcula s_{n-2}		
$i + 1$	envia s_{n-2}	s_{n-2}	recebe s_{n-2}

Garantia: hashes capturados não podem ser reutilizados

Algoritmo HOTP

HMAC-Based OTP:

- HMAC - *Hash-based Message Authentication Code*
- OTP - *One-Time Password*

Princípios:

- Contador compartilhado entre cliente e servidor
- Chave secreta (simétrica) idem
- Chave e contador são cifrados juntos com HMAC
- Algoritmo padronizado pela RFC 4226

Algoritmo HOTP

Entradas:

- Contador c (inicia igual nos dois lados)
- Chave secreta k
- Tamanho de senha d (dígitos)

Cálculo da senha HOTP:

$$s = HOTP(c, k, d) = \text{truncate}(HMAC(c, k)) \bmod 10^d$$

Se o contador dessincronizar entre os dois lados,
pode-se calcular $HOTP(c, k, d)$ para $c \pm 1, c \pm 2, \dots, c \pm s$

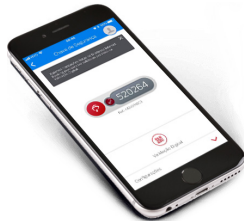
Algoritmo TOTP

TOTP - *Time-based OTP*

Similar ao HOTP:

- Também usa HMAC e uma chave secreta
- Ao invés de um contador, usa o **horário atual**
- Define-se um intervalo de validade T (30s - 60s)

Dispositivo ou
software externo
(*security tokens*)



Estratégias desafio-resposta

Estratégia desafio-resposta

Estratégia genérica muito usada em protocolos de rede:

- O servidor envia um “desafio” único ao cliente
- O cliente resolve o desafio e responde ao servidor
- O servidor confere a resposta
- Opcionalmente, o cliente pode desafiar o servidor

Respostas corretas aos desafios provam autenticidade

Desafios únicos impedem ataques de *replay*

Usado nos protocolos POP, IMAP, LDAP, XMPP, CHAP, ...

Estratégia desafio-resposta

Esquema básico:

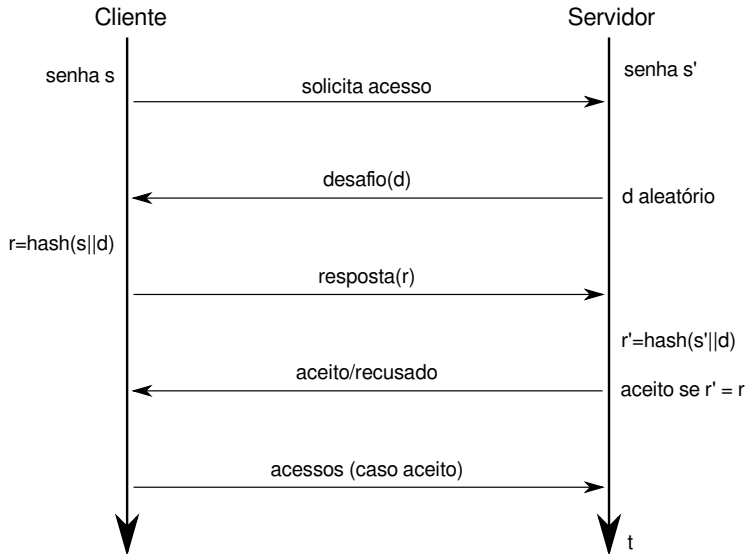
Cliente e servidor compartilham uma senha s

- 1 O servidor envia um aleatório d ao cliente (desafio)
- 2 O cliente recebe d e concatena sua senha s
- 3 O cliente calcula uma resposta $r = \text{hash}(s \parallel d)$
- 4 O cliente envia a resposta r ao servidor
- 5 O servidor recebe r e compara com sua resposta local r'

Vantagens: cada desafio é único, a senha não circula na rede

Problemas: armazenamento da senha em aberto

Estratégia desafio-resposta



Certificados de autenticação

Podem ser usados em ambos os sentidos:

- Autenticar o servidor para o cliente (mais usual)
- Autenticar o cliente para o servidor (bancos)

Podem ser usados em serviços na Internet

- SSL client certificate authentication (Web)
- SSH client authentication (public/private keys)

O certificado pode ser armazenado em *smartcard* ou *pendrive*

Técnicas biométricas

Autenticação biométrica

Biometria

Usar **características biométricas** (físicas ou comportamentais) de um indivíduo para identificá-lo unicamente perante o sistema

Usa características **físicas** ou **comportamentais**



Requisitos das características biométricas

- **Universalidade:** deve estar presente em todos os indivíduos que possam ser autenticados
- **Singularidade** (unicidade): dois indivíduos quaisquer devem apresentar valores distintos para a característica
- **Permanência:** não deve mudar abruptamente no tempo
- **Mensurabilidade:** deve ser facilmente mensurável
- **Desempenho:** permite identificação eficaz e eficiente
- **Coleta:** facilmente medida por um dispositivo
- **Aceitação:** coleta bem aceita pelos indivíduos
- **Robustez:** resistência a ataques de *impersonation*

Características físicas

- impressões digitais
- padrão da íris (parte colorida do olho)
- padrão da retina (vasos sanguíneos no fundo do olho)
- geometria das mãos, do rosto ou das orelhas
- DNA



Fingerprint
Recognition



Iris Recognition



Finger Geometry
Recognition



Biometric
Recognition



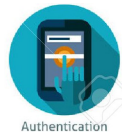
Ear Shape
Recognition



DNA Matching

Características comportamentais

- assinatura (forma e dinâmica)
- padrão de voz
- dinâmica de digitação



Características biométricas mais usadas

Impressões digitais e padrão de íris

- Considerados confiáveis
- Taxas de erro baixas
- Custo de implantação e operação baixo
- Facilidade de coleta
- Boa aceitação em geral



Impressão digital

Histórico:

- Antiguidade: autenticação de contratos (Babilônia, China)
- Séculos 17-18: estudos sobre unicidade das impressões
- 1892: usada para identificar criminoso (Argentina)
- 1901: adotada pela Scotland Yard (UK)

Características:

- Praticamente única
- Facilmente coletável por dispositivo
- Adoção crescente no mundo digital

Tratamento da impressão digital



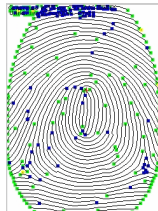
captura



limpeza



thinning















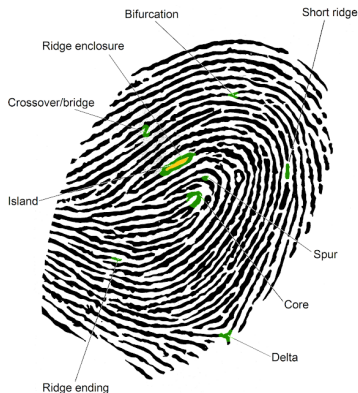
análise

Fonte: “Biometria de Digitais”, GTA-UFRJ

Minúcias em impressões digitais

Basic and composite ridge characteristics (minutiae)

Minutiae	Example	Minutiae	Example
ridge ending		bridge	
bifurcation		double bifurcation	
dot		trifurcation	
island (short ridge)		opposed bifurcations	
lake (enclosure)		ridge crossing	
hook (spur)		opposed bifurcation/ridge ending	



Sistema biométrico

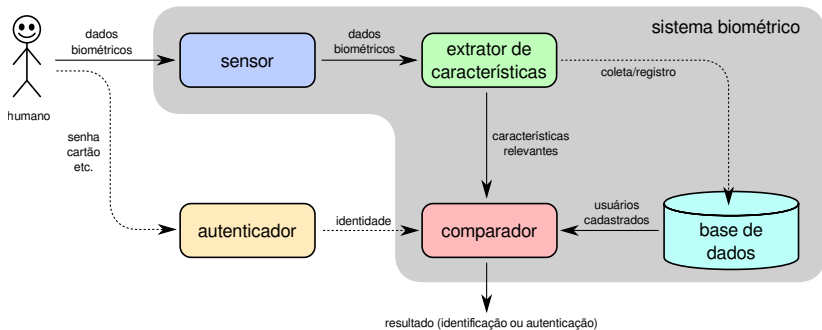
Sistema informático que usa biometria para:

- **Identificação:** identificar uma pessoa
- **Autenticação:** comprovar uma identidade

Elementos de um sistema biométrico:

- **Sensor:** captura os dados biométricos
- **Extrator:** extrai características relevantes dos dados
- **Comparador:** confronta as características com dados armazenados
- **Banco de dados:** infos biométricas registradas no sistema

Sistema biométrico



Modos de operação

- **Coleta:** captura características biométricas dos indivíduos e as cadastra no banco de dados
- **Identificação:** captura características biométricas e tenta associá-las a um indivíduo previamente cadastrado
- **Autenticação:** verifica se as características biométricas correspondem a um indivíduo identificado por outro método (login, etc)

Infraestruturas de autenticação

Infraestruturas de autenticação

Historicamente:

- Cada serviço fazia sua própria autenticação
- Informações de autenticação em bases distintas
- Dificuldades para o usuário
- Dificuldade para a gerência do sistema

Necessidades:

- Unificar as bases de dados de usuários
- Modularizar os métodos de autenticação
- Simplificar a construção de serviços

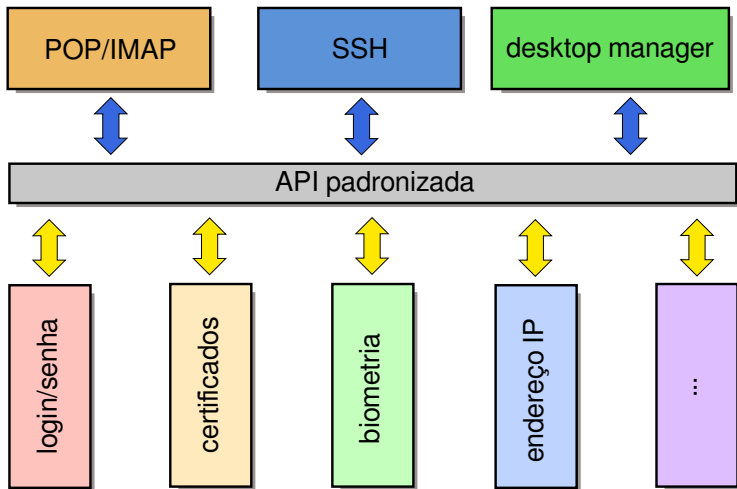
Framework de autenticação

Unifica as técnicas de autenticação, usando os mesmos dados e algoritmos e oferecendo uma interface de programação homogênea

Vantagens:

- Informações de autenticação coerentes entre serviços
- Novas formas de autenticação acessíveis aos serviços
- Simplifica a criação de novos serviços

Infraestruturas de autenticação local



Infraestruturas de autenticação local

- **PAM** – *Pluggable Authentication Modules*
- **XSSO** – *X/Open Single Sign-On* (similar ao PAM)
- **BSD Auth** – usada no OpenBSD
- **GSSAPI** – *Generic Security Services API*
- **SSPI** – *Security Support Provider Interface*
- **JAAS** – *Java Authentication and Authorization Service*
- **U2F** – *Universal 2nd Factor*
- ...

Autenticação em redes

- **SCRAM** – *Salted Challenge Response Auth. Mechanism*
- **NTLM** – *NT LAN Manager*
- **SASL** – *Simple Authentication and Security Layer*
- **CHAP** – *Challenge-Handshake Authentication Protocol*
- **EAP** – *Extensible Authentication Protocol*
- **RADIUS** – *Remote Authentication Dial In User Service*
- **Diameter** – RFC 6733
- **SRP** – *Secure Remote Password protocol*
- **LDAP** – *Lightweight Directory Access Protocol*
- ...

Autenticação na Internet

- X509 PKI
- Kerberos
- OpenID
- OATH
- Shibboleth
- CardSpace/U-Prove
- SAML
- PGP
- SQRL
- WebID
- MSA/LiveID
- WebAuthn
- SPKI/SDSI
- Convergence
- ...

Kerberos

Kerberos

Mitologia grega: guardião dos portões do mundo inferior



Kerberos



Kerberos

Framework de autenticação em rede

- Proposto pelo MIT nos anos 80 (Projeto Athena)

Centraliza autenticação de diversos serviços

- Login, compartilhamentos, impressoras, ...
- Usado em Solaris, Linux, MacOS, Windows, ...

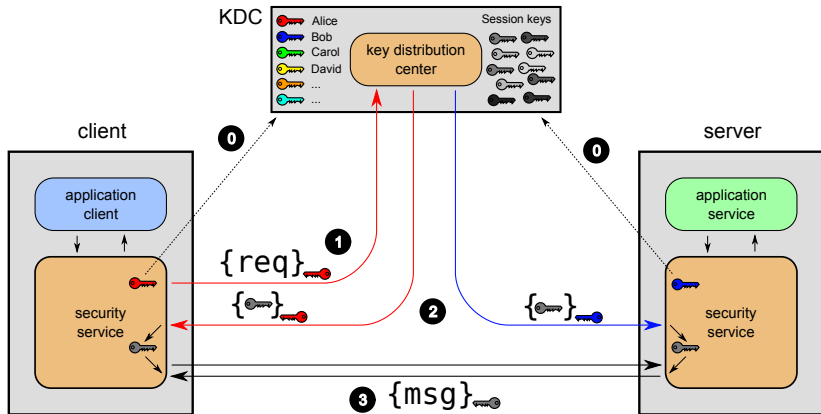
Baseado em *tickets* obtidos pelos clientes

- Usados para acessar os demais serviços da rede
- Cifrados (DE, 3DES, AES) com validade limitada

Kerberos - componentes principais

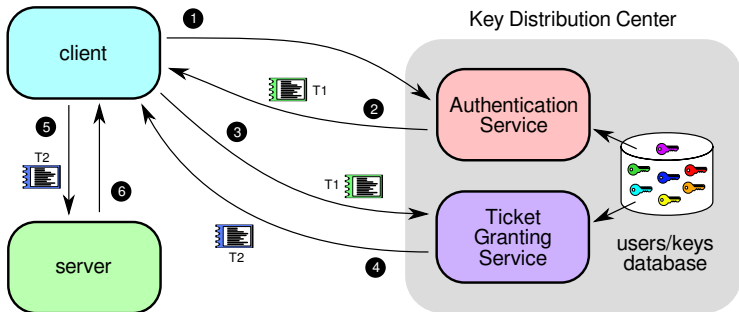
- Clientes (estações de trabalho)
- Serviços de rede
 - login, arquivos, impressão, e-mail
- KDC - *Key Distribution Center*
 - Serviço de Autenticação (AS - *Authentication Service*)
 - Serviço de Tickets (TGS - *Ticket Granting Service*)
 - Base de chaves

Kerberos - visão geral



Kerberos - overview do funcionamento

O cliente se autentica junto ao AS (1) e recebe um ticket de acesso ao TGS (2). A seguir, solicita ao TGS um ticket de acesso ao servidor (3 e 4). Ele enfim se autentica junto ao servidor desejado (5 e 6)



Kerberos - funcionamento

Pressupostos:

- cada cliente c tem sua chave secreta k_c registrada no AS
- cada servidor s tem sua chave k_s registrada no AS
- k_c e k_s são simétricas (cifragem DES)
- cada chave é conhecida só por seu dono e pelo AS

Kerberos passo 1: pedido de autenticação

$$c \xrightarrow{m_1} AS$$

$$m_1 = [c \ tgs \ ts \ n_1]$$

c : identidade do cliente

tgs : identidade do serviço desejado (TGS)

ts : prazo de validade solicitado

n_1 : número aleatório (*nonce*)

Kerberos passo 2: obtém ticket para o TGS

$$c \xleftarrow{m_2} AS$$

- m_2 = $[\{k_{c-tgs} \ n_1\}_{k_c} \ T_{c-tgs}]$
- T_{c-tgs} = $\{c \ tv \ k_{c-tgs}\}_{k_{tgs}}$
- k_{c-tgs} : chave de sessão cliente - TGS
- n_1 : número aleatório (nonce)
- k_c : chave secreta do cliente
- T_{c-tgs} : ticket de acesso ao TGS (TGT)
- c : identidade do cliente
- tv : prazo de validade concedido pelo AS (horas)
- k_{tgs} : chave secreta do TGS

Kerberos passo 3: pede ticket do serviço

$$c \xrightarrow{m_3} TGS$$

$$m_3 = [\{c \ t\}_{k_{c-tgs}} \ T_{c-tgs} \ s \ n_2]$$

c : identidade do cliente

t : data atual

k_{c-tgs} : chave de sessão cliente - TGS

T_{c-tgs} : ticket de acesso ao TGS (TGT)

s : identidade do servidor desejado

n_2 : número aleatório (nonce)

Kerberos passo 4: obtém ticket do serviço

$$c \xleftarrow{m_4} TGS$$

- m_4 = $[\{k_{c-s} \ n_2\}_{k_{c-tgs}} \ T_{c-s}]$
- T_{c-s} = $\{c \ tv \ k_{c-s}\}_{k_s}$
- k_{c-s} : chave de sessão cliente - servidor
- n_2 : número aleatório informado em m_3
- k_{c-tgs} : chave de sessão cliente - TGS
- T_{c-s} : ticket a ser apresentado ao servidor s
- c : identidade do cliente
- tv : validade do ticket
- k_s : chave secreta do servidor s

Kerberos passo 5: acessa o serviço

$$C \xrightarrow{m_5} S$$

m_5 = $[\{c \ t\}_{k_{c-s}} \ T_{c-s} \ request]$
 T_{c-s} = $\{c \ tv \ k_{c-s}\}_{k_s}$ (ticket p/ serviço)
 c : identidade do cliente
 t : data atual
 k_{c-s} : chave de sessão cliente-servidor
 T_{c-s} : ticket de acesso ao servidor
 $request$: pedido de serviço, dependente da aplicação
 tv : validade do ticket
 k_s : chave secreta do servidor s

Kerberos passo 6: resposta do servidor

$$\boxed{C \xleftarrow{m_6} S}$$

$$m_6 = [\{reply\}_{k_{c-s}}]$$

reply : resposta ao cliente, depende da aplicação

k_{c-s} : chave de sessão cliente-servidor

Enquanto T_{c-s} for válido, c pode acessar s diretamente

Enquanto TGT for válido, c pode pedir tickets de outros serviços