



Escola
Superior
de Redes
RNP

Teste de Invasão de Aplicações Web

Capítulo 3

Teste do mecanismo de autenticação

- **Apresentar as principais vulnerabilidades às quais estão sujeitos os mecanismos de autenticação de usuários, bem como as técnicas que podem ser empregadas para detectá-las e explorá-las.**

- **Fatores e tecnologias de autenticação, enumeração de usuários, autenticação com múltiplos fatores, ataques de força bruta, ataques de dicionário, rainbow tables, política de senhas fortes.**

- **Introdução**
- **Tecnologias de autenticação empregadas em aplicações**
- **Descoberta de vulnerabilidades e exploração**
- **Contramedidas**

Um requisito importante de segurança da informação, que deve ser satisfeito por aplicações web, é a autenticação de entidades.

Tal processo envolve a interação entre o reclamante e o verificador, que visam, respectivamente, comprovar a própria identidade e averiguar as provas fornecidas pelo primeiro.

Neste contexto, o reclamante pode utilizar um ou mais dos fatores, abaixo enumerados, para provar o que deseja:



Algo que se sabe.



Algo que se tem.



Algo que se é.

Quando mais de um fator é empregado no processo de autenticação, diz-se que ele é multifator.

Em biometria, é possível utilizar mais de uma característica física ou comportamental do usuário, para autenticá-lo, o que é chamado de autenticação biométrica multimodal.

A autenticação de usuários é o mecanismo de segurança mais evidente para aqueles que utilizam uma aplicação, pois ele é explícito e obrigatório



Exercício de Fixação 1

Autenticação de usuário

1. Que fatores podem ser utilizados na autenticação de um usuário?



Exercício de Nivelamento 1

Autenticação de entidades

- ▲ O que se entende por autenticação de entidades?
- ▲ Que tecnologias de autenticação você encontra nas aplicações web que utiliza?

O mecanismo de autenticação de usuários mais comumente empregado por aplicações web consiste no fornecimento de identificador e senha.

Esta abordagem pode ser implementada por meio das seguintes tecnologias:



Autenticação HTTP.



Autenticação integrada ao Windows.



Autenticação por formulários.

Um problema decorrente das soluções baseadas em senhas é que elas podem ser capturadas por software malicioso (*keylogger*).

Isto levou muitas empresas a adotarem teclados virtuais.

A resposta criminosa foi a criação de *screenloggers*, capazes de gravar regiões da tela ao redor do ponto clicado com o mouse, bem como as coordenadas desta posição.

Por isso, alguns teclados virtuais permutam as posições dos elementos e escondem o símbolo de uma tecla, quando o mouse passa por cima dela.

01	9780	09	7464	17	9806	25	1487	33	8386	41	6432
02	3489	10	5110	18	3712	26	4647	34	2431	42	3325
03	8208	11	0404	19	6107	27	7285	35	8293	43	2815
04	9318	12	0392	20	2163	28	4855	36	0505	44	6111
05	0163	13	2663	21	3258	29	4203	37	6109	45	5940
06	6558	14	2300	22	5078	30	3859	38	9977	46	3569
07	7199	15	8420	23	4219	31	8705	39	9346	47	8822
08	9984	16	0809	24	5418	32	9300	40	9734	48	4046

(a)



(b)

Figura 3.2 - Tokens: (a) Cartela de senhas. (b) Dispositivo síncrono de senhas dinâmicas baseado em horário.



Figura 3.3 - Aplicação da Receita Federal, que permite acesso via e-CPF.

Diversas informações interessantes podem ser obtidas nas fases de mapeamento e de reconhecimento de aplicações web.

Em alguns casos, são descobertos identificadores válidos de usuários, que servem de ponto de partida para um ataque de força bruta, por exemplo.

Em outros cenários mais favoráveis, as informações permitem acesso direto às áreas protegidas do sistema.

Outra fonte de credenciais válidas são comentários deixados pelos programadores, no código HTML apresentado ao usuário.



Exercício de Nivelamento 2

Vulnerabilidades

- ▶ Que tipos de vulnerabilidades podem ser encontrados em mecanismos de autenticação?

▲ Exemplo: arquivo “accounts.txt”

```
'admin', 'adminpass', 'Monkey!!!  
'adrian', 'somepassword', 'Zombie Films  
Rock!!!  
'john', 'monkey', 'I like the smell of  
confunk  
'ed', 'pentest', 'Commandline KungFu  
anyone?'
```

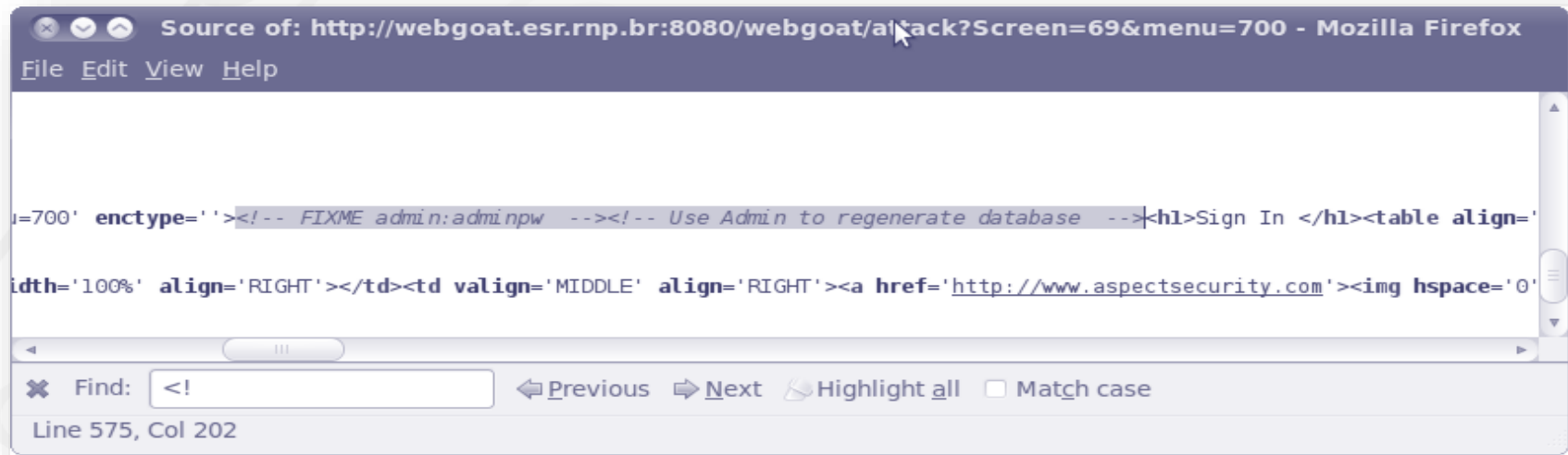


Figura 3.8 - Credenciais contidas em comentário no código HTML.

Sistemas e plataformas, normalmente, são distribuídos com algumas contas padronizadas, cujas senhas são conhecidas publicamente.

Caso aquelas não sejam desativadas ou as senhas não sejam alteradas, um atacante pode, muito facilmente, obter acesso não autorizado ao sistema.

É normal encontrar contas previsíveis como “admin”, “administrator”, “root”, “system”, “test”, “teste”, “test123” e “guest”.

As senhas, por sua vez, costumam ser vazias, iguais aos próprios identificadores ou palavras comuns, como “password”, “pass123” e “senha”.

Usuário e senha padronizados

Plataforma/Sistema	Versões	Usuário	Senha
Apache Tomcat	Diversas	admin	admin
Apache Tomcat	Diversas	admin	tomcat
Apache Tomcat	Diversas	tomcat	tomcat
Microsoft SQL Server	2000, 2005	sa	-
MySQL	Todas	root	-
Oracle Database	Diversas	scott	tiger
Oracle Database	7, 8i	sys	change_on_install
Oracle Database	< 10	system	manager
phpMyAdmin	Todas	root	-

Figura 3.9 - Exemplos de conta e senha padronizadas.

É importante ter à disposição um conjunto de identificadores válidos de usuários, para se testar outras potenciais vulnerabilidades.

Uma vulnerabilidade na aplicação consiste em se exibir ao usuário mensagens de erro muito específicas, quando a autenticação não é realizada com sucesso.

A chave para o sucesso desta técnica depende de se conseguir discernir a situação em que a conta é inválida daquela em que ela existe, mas a senha informada não é a esperada.

Abaixo, estão listados exemplos de características que podem variar e não serem visualmente perceptíveis:



Código HTML



Título da página



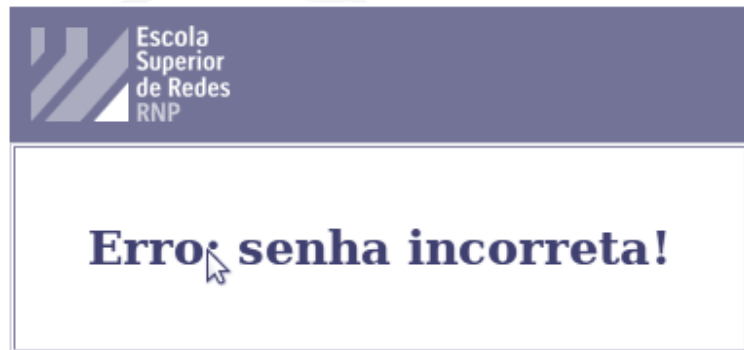
Tempo de resposta



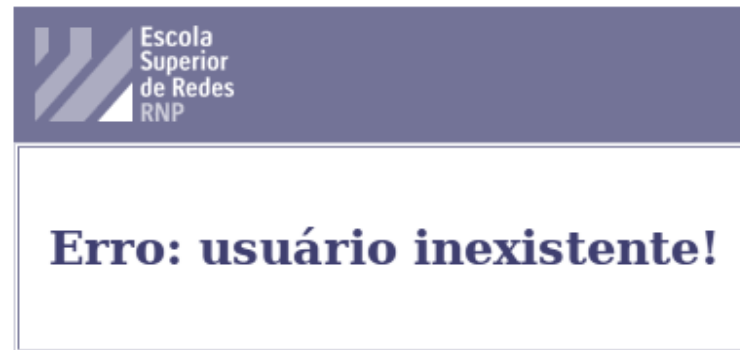
Cabeçalhos da resposta

Em um teste de invasão real, é fundamental recorrer à automatização.

Enumeração de identificadores de usuário



[Retornar à página de login](#)



[Retornar à página de login](#)



Figura 3.10 - Mensagens de erro não uniformes em caso de tentativa malsucedida de autenticação: (a) Usuário existe, mas senha informada é incorreta. (b) Usuário não existe.

▲ Automatização com curl:

```
~$ curl -s --data  
"userid=admin&senha=admin&Submit1=Login"  
http://form-auth.esr.rnp.br/login.php
```

- ▲ Aplicações web desenvolvidas para a Internet, que permitem que os usuários criem suas próprias contas, possuem uma funcionalidade para auxiliá-los em casos de esquecimento da senha de acesso.

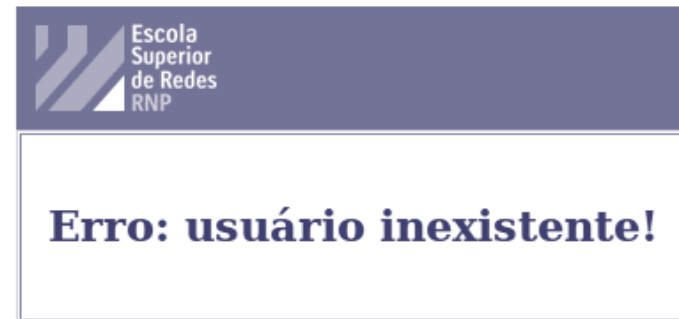


Escola
Superior
de Redes
RNP

Recuperação de senha

Qual a sua cor favorita?

OK



Escola
Superior
de Redes
RNP

Erro: usuário inexistente!

[Retornar à página de login](#)

Figura 3.13 - Mecanismo vulnerável de recuperação de senhas, que possibilita enumeração de usuários: (a) Tela exibida, uando identificador existe. (b) Mensagem de erro informando inexistência de identificador.

Um mecanismo de recuperação de senhas pode estar sujeito a diversos outros problemas de segurança:



Inexistência de mecanismo de bloqueio.



Respostas disponíveis em fontes públicas.



Cadastro de perguntas secretas personalizadas.



Envio da senha original para uma conta pré-cadastrada de correio eletrônico.

Um mecanismo de recuperação de senhas pode estar sujeito a diversos outros problemas de segurança:



Redirecionamento para uma sessão autenticada.



Recuperação de senha por equipe de suporte.



Solicitação da conta de correio eletrônico, para a qual as instruções para recuperação de senha devem ser enviadas, após perguntas secretas serem respondidas corretamente.



Falta de notificação de troca de senha.

Funcionalidade “Lembrar usuário”

Algumas aplicações web disponibilizam uma funcionalidade que permite lembrar o usuário, automaticamente, toda vez que forem utilizadas.

Se o mecanismo é ativado em um computador de uso público, a conta da última pessoa que usou a aplicação é revelada.

Um cenário mais grave resulta quando a aplicação usa o cookie para autenticar o usuário, automaticamente, sem a necessidade de fornecimento de senha.

Uma pequena variação no ataque acima permite que um usuário legítimo escale privilégios na aplicação.

Quando existe a possibilidade de credenciais de acesso serem enviadas para o destino incorreto ou serem capturadas em trânsito, diz-se que são transportadas de maneira insegura.

A falha mais comum nesse sentido consiste no envio das informações em claro ou apenas codificadas.

É comum encontrar na Internet aplicações web que enviam as informações de autenticação, via protocolo HTTPS, mas que fornecem a página para capturá-las, empregando HTTP simples.

Outra vulnerabilidade consiste na passagem dessas informações como parâmetros da URL.

Finalmente, a camada SSL/TLS, provida pelo servidor web, pode estar mal configurada.

Transporte inseguro de credenciais

```
+ GET /basic/ HTTP/1.1\r\n
Host: exemplo.esr.rnp.br\r\n
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2.13) Gecko/20101206 Ubun
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
Accept-Language: en-us,en;q=0.7,pt-br;q=0.3\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
+ Authorization: Basic ZXNydXNlcjplc3Jlc2Vy\r\n
\r\n
```

01a0	2e 37 2c 2a 3b 71 3d 30 2e 37 0d 0a 4b 65 65 70	.7,*;q=0 .7..Keep
01b0	2d 41 6c 69 76 65 3a 20 31 31 35 0d 0a 43 6f 6e	-Alive: 115..Con
01c0	6e 65 63 74 69 6f 6e 3a 20 6b 65 65 70 2d 61 6c	nection: keep-al
01d0	69 76 65 0d 0a 41 75 74 68 6f 72 69 7a 61 74 69	ive..Aut horizati
01e0	6f 6e 3a 20 42 61 73 69 63 20 5a 58 4e 79 64 58	on: Basi c ZXNydX
01f0	4e 6c 63 6a 70 6c 63 33 4a 31 63 32 56 79 0d 0a	Nlcjplc3 Jlc2Vy..
0200	0d 0a	..

Figura 3.15 - Dados de autenticação HTTP Basic capturados em trânsito.

Se a aplicação não tratar erros inesperados, um usuário malicioso pode se autenticar, mesmo sem conhecimento da senha correspondente.

Uma das regras de ouro em desenvolvimento de software seguro determina que, em caso de falhas inesperadas, a aplicação deve sempre se manter em um estado seguro.

Um teste que deve ser executado consiste na indução de erro na aplicação:

- Remoção de parâmetros;
- Submissão de valores maiores que os permitidos;
- Inclusão de parâmetros inválidos;
- Envio de caracteres não permitidos.

```
sComando = "select count(*) into :nCount  
            from usuarios  
            where id = '" + sIdentificador + "' and  
                   senha = '" + sSenha + "'";  
SqlExecute(hSql, sComando);  
if (nCount > 0) print("Usuário autenticado");
```

```
public static boolean auth(String id, String pwd) {  
    String senha = getPwd(id);  
    boolean bAuth = senha != null;  
    try {  
        if (senha != null) {  
            pwd = pwd.toUpperCase();  
            for (int i=0; i<pwd.length() && bAuth; i++) {  
                bAuth = pwd.charAt(i) == senha.charAt(i);  
            }  
        }  
    } catch (Exception e) {  
    }  
    return bAuth;  
}
```

Diversos padrões de segurança, como o PCI DSS, demandam que senhas de usuários sejam trocadas periodicamente.

Muitas aplicações incluem uma funcionalidade para troca de senhas, mas nem todas a implementam de maneira segura.

Um problema muito comum é deixar de pedir a senha atual nesse processo, porque a tela para efetuar essa operação só é acessível a usuários autenticados.

Sistemas que manipulam informações críticas são fortes candidatos a empregar múltiplos fatores de autenticação.

Uma sessão em uma aplicação desse tipo começa, tipicamente, com a autenticação do usuário por meio da senha, o que permite acesso a diversas funcionalidades básicas.

Quando uma operação envolvendo valores financeiros é solicitada, o usuário precisa fornecer novamente a senha e utilizar o segundo fator, para comprovar sua identidade.


Uma implementação segura desse cenário requer que as informações de autenticação já validadas sejam mantidas exclusivamente no servidor e que etapas do processo não possam ser ignoradas.

Exemplos de vulnerabilidades:



Informações sobre as diversas etapas da autenticação são mantidas em parâmetros enviados ao lado cliente, como “tokenValidado=N”.

Exemplos de vulnerabilidades:



Uma aplicação, que usa cartelas de senha como segundo fator de autenticação, mantém, em um campo escondido, o identificador do usuário corrente.

Exemplos de vulnerabilidades:



A rotina que trata uma determinada operação assume que, se foi chamada, todos os fatores de autenticação já foram validados.

Um ataque de força bruta consiste em tentar se autenticar com todas as senhas possíveis, até que o processo seja bem sucedido.

A tarefa pode se estender por um longo período de tempo.

A técnica é extremamente ruidosa e, por isso, é facilmente percebida por dispositivos de detecção de intrusão.

Um ataque desse tipo deve ser utilizado somente como último recurso de um teste de invasão caixa-preta.

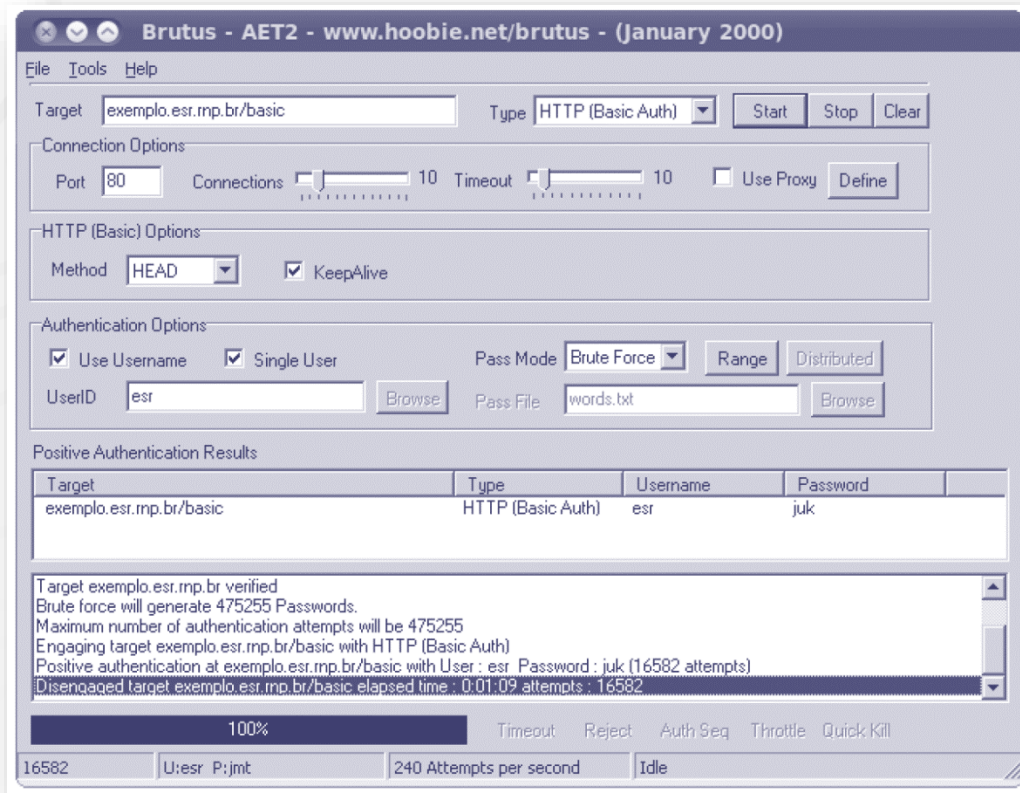


Figura 3.20 - Ataque de força bruta contra Autenticação Basic.

Um ataque de força bruta pode demorar muito tempo para atingir um resultado positivo, se o espaço de senhas for demasiado grande.

Uma abordagem mais inteligente resulta no ataque de dicionário, que seleciona as senhas para teste a partir de uma lista de palavras comuns e variações.

Exemplos de utilitários, que podem ser empregados para realizar ataque de dicionário contra aplicações web, incluem:



Brutus



THC Hydra



Medusa

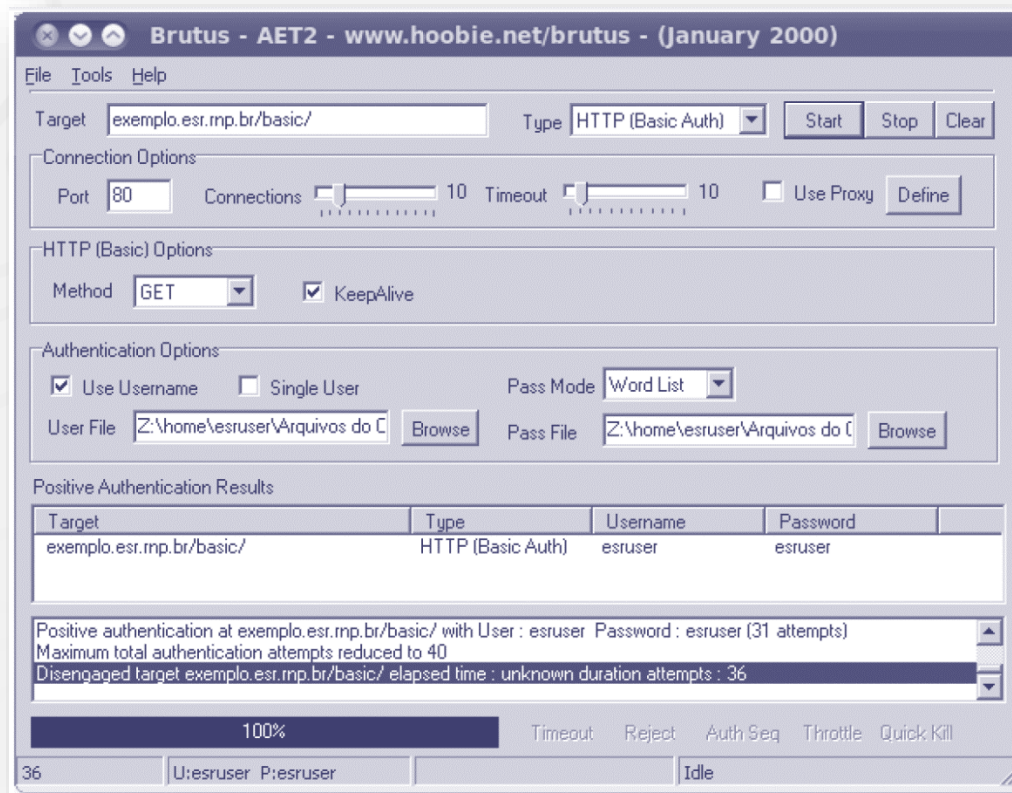


Figura 3.15 - Dados de autenticação HTTP Basic capturados em trânsito.

```
~$ medusa -h exemplo.esr.rnp.br -U ids -P ids -e ns  
-M http -m DIR:basic/ -v 4
```

```
Medusa v1.5 [http://www.foofus.net] (C) JoMo-Kun /  
Foofus Networks <jmk@foofus.net>
```

```
ACCOUNT FOUND: [http] Host: exemplo.esr.rnp.br User:  
esruser Password: esruser [SUCCESS]
```



```
~$ hydra -L ids -P pwds -e ns exemplo.esr.rnp.br  
http-head /digest/
```

```
~$ hydra -L ids -P pwds -e ns form-auth.esr.rnp.br  
http-post-form  
"/login.php:userId=^USER^&senha=^PASS^&Submit1=Log  
in:incorreta"
```

Durante um teste de invasão em aplicações web, não é raro obter senhas a partir de arquivos e bancos de dados.

Para recuperação das senhas, podem ser utilizadas técnicas baseadas em criptoanálise ou calcadas na fraqueza das senhas.

Ataques de dicionário e de força bruta são possíveis.

Um utilitário interessante é o “John the Ripper”.

```
~$ john passwords
```

```
Loaded 3 password hashes with 3 different salts  
(Traditional DES [64/64 BS MMX])
```

```
esruser          (esruser)
```

```
juk              (esr)
```

```
senhad           (hard)
```

```
guesses: 3   time: 0:00:00:14 (3)   c/s: 812046
```

```
trying: secKXy - senhhr
```

Uma abordagem mais elegante, que permite reduzir o consumo de espaço, resume-se no uso de *rainbow tables*.

O elemento central da solução é o *rainbow chain*, que consiste em uma sequência de valores alternados de senhas e *hashes*, calculados a partir de uma senha inicial.

$$s_1 \xrightarrow{h(s_1)} h_1 \xrightarrow{R_1(h_1)} s_2 \xrightarrow{h(s_2)} \dots \xrightarrow{h(s_{t-1})} h_{t-1} \xrightarrow{R_{t-1}(h_{t-1})} s_t$$

Diversas cadeias de mesmo comprimento são geradas, a partir de senhas iniciais diferentes, e somente o primeiro e último elementos de cada uma delas são armazenados.

Seja a i -ésima cadeia parcial a seqüência de valores alternados de hashes e senhas, calculados a partir de $R_i(H)$, com $1 \leq i \leq t - 1$:

$$H \xrightarrow{R_i(H)} S_{i+1} \xrightarrow{h(s_{i+1})} h_{i+1} \xrightarrow{R_{i+1}(h_{i+1})} \dots \xrightarrow{h(s_{t-1})} h_{t-1} \xrightarrow{R_{t-1}(h_{t-1})} S_t$$

Iniciando com $i = t - 1$ e decrescendo até $i = 1$, calculam-se as cadeias parciais, até que a senha final (s_t) seja encontrada como elemento final de uma linha L da tabela.

Neste passo, a cadeia é reconstruída, a partir da senha inicial recuperada da linha L , até h_i , o qual deve ser comparado contra H .

Softwares que podem ser empregados para realizar ataques baseados em rainbow tables:

- RainbowCrack.
- Ferramentas do projeto Free Rainbow Tables.
- Cryptohaze GPU Rainbow Cracker.

▲ Exemplo:

```
~$ rcracki_mt -h 2465d0454ec909560b45b72086604edf  
*.rti
```

Ataques baseados em força bruta, dicionário e *rainbow tables* são executados mais facilmente, quando a aplicação não implementa uma política de senhas fortes.


Itens que devem constar em uma boa política de senhas:

- Comprimento mínimo
- Complexidade
- Troca
- Histórico
- Máximo de trocas por dia
- Bloqueio de contas

Uma política inadequada de bloqueio de contas pode permitir ataques de negação de serviço contra os usuários da aplicação.

Em alguns casos, isso pode ter por objetivo favorecer o atacante, além de causar incômodo à vítima.

Exemplos:




aplicação de leilão eletrônico, que exibe os identificadores dos usuários participantes, em ordem decrescente do lance efetuado.

**Qual é a principal
vulnerabilidade do cenário?**

Mesmo hoje em dia, o maior problema de sistemas de autenticação de usuários não é de origem técnica, mas, sim, de natureza humana.

Algumas vezes, o contratante de um teste de invasão solicita que a postura de segurança de seus colaboradores seja avaliada, como uma maneira de validar a eficácia da política de segurança vigente.

Exemplos de testes que podem ser realizados:




Se não houver um processo formal para redefinição de senhas esquecidas, pode-se ligar para a equipe de suporte, simulando a situação, como se fosse um usuário válido.

Exemplos de testes que podem ser realizados:

Ligar para um usuário, dizendo ser do suporte, e solicitar que ele confirme a senha por telefone, devido a um problema ocorrido na base de autenticação.

Exemplos de testes que podem ser realizados:



Enviar ao usuário de um software malicioso que capture as teclas digitadas ou que permita se conectar remotamente à estação que utiliza.



Exercício de Fixação 2

Vulnerabilidades em mecanismos de autenticação

1. Que tipos de vulnerabilidades podem ser encontrados em mecanismos de autenticação?

**Não permita
identificadores de
usuário repetidos na
aplicação.**

**Remova, das áreas
acessíveis pelo
servidor web, arquivos
que contenham
credenciais de acesso
ao sistema e à infra-
estrutura subjacente.**

Remova, das páginas HTML e dos arquivos de código-fonte, comentários que contenham informações de autenticação.

Desabilite contas pré-definidas da aplicação e das plataformas que a suportam.

Implemente, na aplicação e nas plataformas subjacentes, políticas de senhas fortes.

Para contas criadas automaticamente pela aplicação, atribua uma senha inicial aleatória e obrigue a troca no primeiro acesso.

Informe ao usuário, logo no início de uma sessão, a data e hora da última vez que se conectou com sucesso.

Quando a autenticação falhar, exiba apenas uma mensagem de erro genérica. Adote estratégia similar para mecanismos de recuperação de senha.

Em mecanismos de recuperação de senha:

- **Empregue perguntas secretas, cujas respostas não sejam facilmente dedutíveis ou encontradas na Internet.**
- **Limite o número de tentativas para acerto das questões, para evitar ataques de força bruta.**
- **Em caso de sucesso, envie uma mensagem automática para o endereço de e-mail pré-cadastrado, contendo um link para uma página efêmera individualizada, na qual a nova senha poderá ser definida.**
- **Notifique a alteração ao usuário, por meio de nova mensagem de correio eletrônico.**

**Nunca memorize
credenciais completas
em uma
funcionalidade
“Lembrar Usuário”.**

**Nunca confie em
parâmetros que
podem ser alterados
pelos usuários, para
decidir se estão ou
não autenticados.**

**Nunca submeta
formulários de
autenticação, por meio
do método GET.
Igualmente, não passe
credenciais pela URL,
em processos de
redirecionamento.**

Forneça a página de autenticação ao usuário, somente por meio do protocolo HTTPS.

Configure o lado servidor dos túneis criptográficos, de acordo com as melhores práticas de segurança conhecidas.

Sempre transmita credenciais de acesso por túneis protegidos criptograficamente.

Se um erro inesperado ocorrer, a aplicação deve permanecer em um estado seguro.

Valide a entrada imediatamente antes de processá-la, de modo que o formato esperado seja respeitado.

Não exiba mensagens de erro contendo informações sobre as plataformas e tecnologias empregadas.

Exija sempre o fornecimento da senha atual para efetuar a troca de senha de um usuário.

Em mecanismos de autenticação baseados em múltiplos fatores, sempre verifique que todas as etapas esperadas foram corretamente validadas.

Nunca armazene as senhas em claro.

Não exiba um identificador de usuário, para outras pessoas que não ele próprio.

Registre em trilhas de auditoria todas as tentativas válidas e inválidas de autenticação.

Implante uma política de segurança e conscientize todos os usuários.

Perguntas



**Caderno de
Atividade
3**

1

**Tecnologias de
autenticação**



Caderno de Atividade 3

2

Descoberta de vulnerabilidades e exploração



Teste de Invasão de Aplicações Web

Capítulo 3

Teste do mecanismo de autenticação