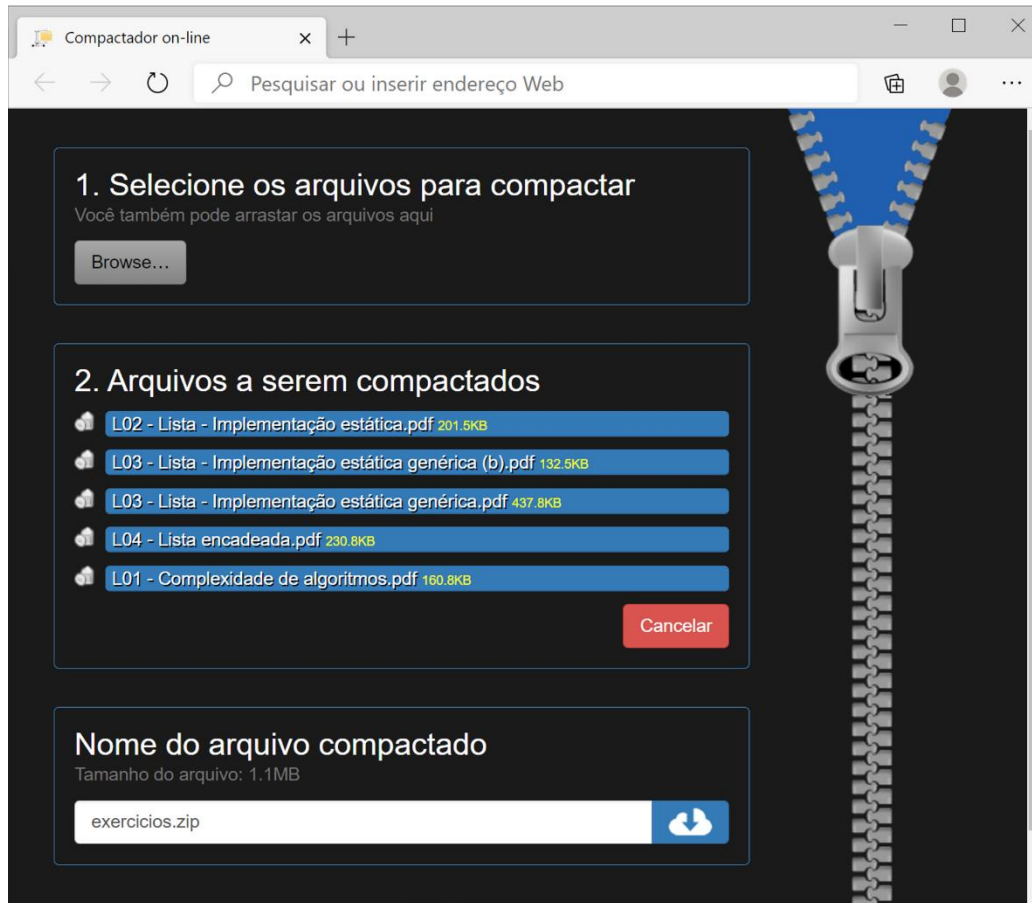


Injeção de comandos do sistema operacional (OS Command Injection)

**CWE-78: Improper Neutralization of
Special Elements used in an OS Command
('OS Command Injection')**

Motivação



- Considerar uma aplicação web para compactar arquivos.
- O usuário faz o upload dos arquivos e informa o nome do arquivo zip que deseja salvar

Motivação

- Para gerar o arquivo, a aplicação recorre ao 7-zip.
- A sintaxe para compactar um arquivo com 7-zip é:

```
> 7z

Usage: 7z <command> [<switches>...] <archive_name> [<file_names>...]

<Commands>
  a : Add files to archive
  b : Benchmark
  d : Delete files from archive
  e : Extract files from archive (without using directory names)
```

- Exemplo:

```
> 7z a fotos.zip foto1.jpg foto2.jpg foto3.jpg
```

```
> 7z a fotos.zip foto1.jpg foto2.jpg foto3.jpg
7-Zip 18.01 (x64) : Copyright (c) 1999-2018 Igor Pavlov : 2018-01-28

Open archive: fotos.zip
--
Path = fotos.zip
Type = zip
Physical Size = 22

Scanning the drive:
3 files, 7951208 bytes (7765 KiB)

Updating archive: fotos.zip

Add new data to archive: 3 files, 7951208 bytes (7765 KiB)

Files read from disk: 3
Archive size: 7872874 bytes (7689 KiB)
Everything is Ok
```

Motivação

- Sendo o *backend* escrito em Java, o fragmento de código que gera o arquivo zip é:

```
public void zipFiles(String outputFile, ArrayList<String> files) {
    String commandLine = getZipApp() + " a " + outputFile;

    for (String file:files) {
        commandLine += " " + file;
    }

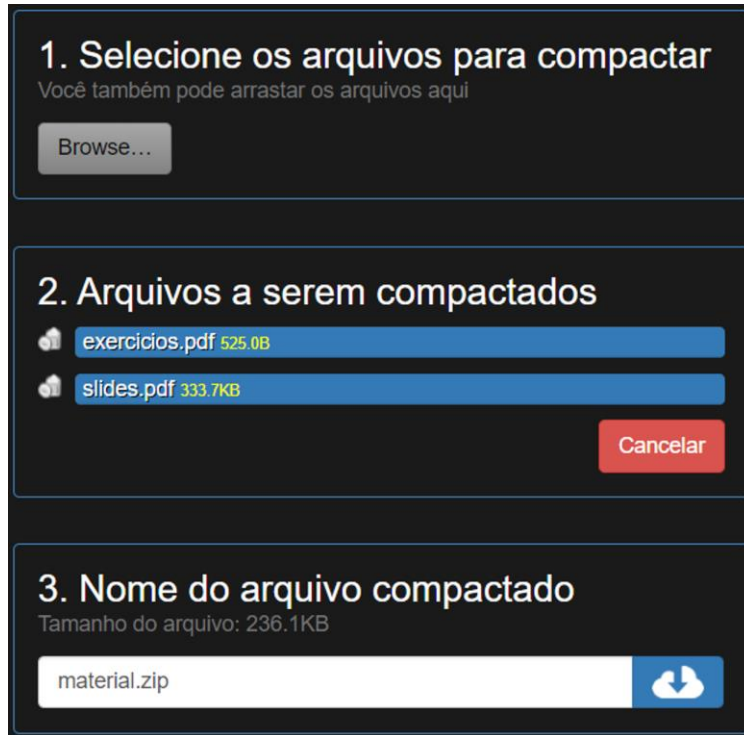
    File defaultDir = (new File(outputFile)).getParent();
    Process p = Runtime.getRuntime().exec(commandLine, null, defaultDir);
    p.waitFor();
}

public String getZipApp() {
    if (Settings.getInstance().hasZipApp()) {
        return Settings.getInstance().zipApp();
    } else {
        return "C:\\Program Files\\7-Zip\\7z.exe";
    }
}
```

- Neste fragmento de código, a aplicação compacta os arquivos, invocando uma aplicação externa

Exemplo 1

Uma entrada possível seria:



The screenshot shows the 7-Zip command line interface with three main sections:

- 1. Selecione os arquivos para compactar**
Você também pode arrastar os arquivos aqui
A "Browse..." button is visible.
- 2. Arquivos a serem compactados**
Two files are listed: "exercicios.pdf" (525.0B) and "slides.pdf" (333.7KB). A red "Cancelar" button is at the bottom right.
- 3. Nome do arquivo compactado**
Tamanho do arquivo: 236.1KB
The filename "material.zip" is entered in the text field, with a "Save" icon (floppy disk) to its right.

Com isso, commandLine seria alimentado com:

```
C:\Program Files\7-Zip\7z.exe a  
material.zip exercicios.pdf  
slide.pdf
```

Separador de comandos

- Nos sistemas operacionais, o interpretador de linha de comandos (*cmd*, *bash*) permite executar vários programas com uma linha de comando.
- Para isso, utiliza-se o separador de comandos
 - No Windows usa-se o caractere &
 - Em sistemas baseados em Unix, usa-se o caractere ;
- Os comandos são executados um após o outro pelo interpretador de linha de comandos

Separador de comandos

- Exemplo:

```
> dir & echo Boa noite
```

- Exemplo de saída:

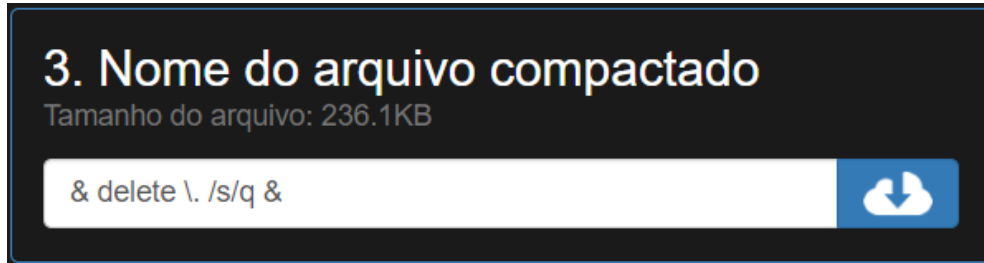
```
O volume na unidade C é OS
O Número de Série do Volume é E6E7-F9D8

Pasta de C:\Dell

08/09/2019  17:56    <DIR>          .
08/09/2019  17:56    <DIR>          ..
08/09/2019  17:56    <DIR>          UpdatePackage
                0 arquivo(s)                0 bytes
                3 pasta(s)  17.149.595.648 bytes disponíveis
Boa noite
```

Exemplo 1

- Uma entrada possível seria:



- Ou seja, seria executada a seguinte linha de comando:

```
> 7z.exe a & delete \. /s/q & exercicios.pdf slides.pdf
```

- Ou seja:

```
7z.exe a
```

```
delete \. /s/q
```

```
exercicios.pdf slides.pdf
```


Definição Injeção de comandos do sistema operacional

- Ocorre quando alguma função da aplicação é executada através de comandos do sistema operacional (*comandos shell*) ou por aplicações externas e sofrem influência da entrada do usuário.
- Nesta situação, quando a entrada de dados contém comandos especiais, é possível executar comandos diretamente do sistema operacional
 - Pode permitir a execução de comandos privilegiados (que o usuário não teria acesso, mas a aplicação possui)
- Também conhecido como *Shell injection*, *Command Injection* ou *OS injection*

Exemplo 2

- Considerar uma aplicação que permite consultar se um determinado ponto TCP/IP está acessível.

Ping a device

Enter an IP address:

Pinging 127.0.0.1 with 32 bytes of data:
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128

Ping statistics for 127.0.0.1:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms

```
<?php
if( isset( $_POST['Submit'] ) ) {
    // obtém a entrada
    $target = $_REQUEST[ 'ip' ];

    // Identifica o sistema e executa o comando ping
    if( striestr( php_uname( 's' ), 'Windows NT' ) ) {
        // Windows
        $cmd = shell_exec('ping' . $target );
    }
    else {
        // *nix
        $cmd = shell_exec( 'ping -c 4 ' . $target );
    }

    // apresenta resultado ao usuário
    echo "<pre>{$cmd}</pre>";
}

?>
```

- Neste caso, a aplicação apresenta a saída do comando ao usuário, dando oportunidade para coletar informações do sistema

Exemplo 2

A vulnerabilidade poderá ser explorada se a entrada contiver, por exemplo:

Enter an IP address:

Submit

Enter an IP address:

Submit

```
Pinging 127.0.0.1 with 32 bytes of data:
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128

Ping statistics for 127.0.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms
    Volume in drive C has no label.
    Directory of C:\xampp\htdocs\exec

06/01/2017  12:08 AM

    06/01/2017  12:08 AM

    06/01/2017  12:08 AM

                help
    06/01/2017  12:08 AM                1,830 index.php
    06/01/2017  12:08 AM

                source
                1 File(s)                1,830 bytes
                4 Dir(s)  129,195,401,216 bytes free
```

Exemplo 3

- Considerar uma aplicação de comércio eletrônico que permite que o usuário consulte o saldo de estoque numa unidade (loja) em particular. Esta informação poderia ser acessada através desta URL

```
https://shopping.com.br/estoque?produtoId=381&lojaId=29
```

- Para fornecer as informações de estoque, a aplicação precisa consultar um sistema legado. Por razões históricas, esta funcionalidade foi implementada através de uma chamada que utiliza o *shell*. A chamada poderia ser:

```
stockrep.pl 381 29
```

- Este comando retorna a situação do estoque para o item especificado
- Utilizar funções de sistemas legados é uma funcionalidade frequentemente utilizada

Exemplo 4

Envio de email a partir do servidor Web

- Considere um web site que permite que o usuário submeta seu feedback.
- A aplicação gera um email para o administrador chamando o programa externo mail

Contact us

We are here to answer any questions you may have about our combadi experiences. Reach out to us and we'll respond as soon as we can.

Even if there is something you have always wanted to experience and can't find it on combadi, let us know and we promise we'll do our best to find it for you and send you there.

NAME: *

EMAIL: *

MESSAGE: *

SEND

EMAIL
info@combad.com

TELEPHONE
+30 6977 447 033

SKYPE
combad

ADDRESS
28A Epidavrou Str.,
15233 Halandri,
Greece.

```
mail -s "Gostei do site" -aFrom:ninguem.com feedback@destino.com
```

Outros meta-caracteres dos sistemas operacionais

SO	Operador	Descrição
Windows	&&	Se o comando que precede && for bem sucedido, o próximo comando será executado
		Somente executa o segundo comando se o primeiro falhar
*Unix		Redireciona a saída do primeiro comando para o segundo comando
	`	Força o shell a interpretar e executar o comando dentro da string
	()	Usado para executar o comando que está numa variável, como \$(var)
	#	Usado para comentar linhas

Evitando Injeção de comandos do SO

- Implementar uma lista branca de caracteres aceitáveis
- Utilizar APIs da linguagem para interagir com o sistema operacional (e evitar chamadas de programas externos)
- Quando for inevitável usar funções do sistema operacional/programas externos, o desenvolvedor deveria garantir que a aplicação esteja executando com privilégios mínimos.

Evitando injeção de comandos do SO

Exemplo

```
// obtém entrada
$target = $_REQUEST[ 'ip' ];
$target = stripslashes($target);

// Divide o IP em 4 octetos
$octet = explode( ".", $target );

// Verifica se cada octeto é um número inteiro
if( ( is_numeric($octet[0] ) ) && ( is_numeric($octet[1] ) )
    && ( is_numeric($octet[2] ) ) && ( is_numeric($octet[3] ) ) {

    $target = $octet[0] . '.' . $octet[1] . '.' . $octet[2] . '.' . $octet[3];

    // Identifica SO
    if ( strstr( php_uname( 's', 'Windows NT' ) ) {
        // Windows
        $cmd = shell_exec( 'ping ' . $target );
    }
    else {
        $cmd = shell_exec( 'ping -c 4' . $target );
    }

    // saída para o usuário
    echo "<pre>{$cmd}</pre>";
}
else {
    echo '<pre>Erro: Você não forneceu um IP válido</pre>';
}
```