

PHPIDS

Monitoring attack surface activity

A presentation by Mario Heiderich
For OWASP AppSec Europe 2008

Who?



Mario Heiderich

- CSO for ormigo.com in Cologne, Germany
- Lead developer / co-founder PHPIDS
- GNUCITIZEN core member



What?



Webapp Security – say what!

The developers' dilemma

Webapp alarm devices

Regex black-mæjick

Blacklisting 2.0

ormigo 



GNUCITIZEN
Cutting-edge Think tank | Ethical Hacker Outfit

Tough love



```
<img/  
/onerror="[$y=('al')]&[$z=$y+'ert']  
[a=(1?/ev/:0)[-1]+$y]($z)(1)"  
src=x>
```

What does *this* code do? Anyone?



The dilemma



Usability vs. Security

Insecurity 2.0

One in a million

Unaware malignity?

ormigo 



GNUCITIZEN
Cutting-edge Think tank | Ethical Hacker Outfit

Who knows?



Developers and time pressure

Complexity – do you *really* know HTML?

JS, SQL, PHP, LDAP, XML, **OMG...**

It's full of ... vectors

I don't see it - *thus* it doesn't exist

ormigo 



GNUCITIZEN
Cutting-edge Think tank | Ethical Hacker Outfit

Do what now?



Install a WAF Appliance?
Strip what's looking weird?
Employ a logfile monkey?
Fallback to static HTML?



Maybe no!



PHPIDS *detects* badness

Pricing: 0€

LGPL

Slim, fast and...

**... tested by security experts all over the world
over months**

ormigo 



GNUCITIZEN
Cutting-edge Think tank | Ethical Hacker Outfit

What does it do?



Not much, really!



Receiving



First of all:
The developer defines what to scan.



Converting



The input is being *analysed*, *converted* and ***normalized*** to a certain level before hitting the regular expressions.

And the mysterious PHPIDS Centrifuge.



Matching



A XML/JSON ruleset covering various attack detection patterns

About 70 tagged regex rules

XSS, SQLI, RCE, LFI, DT, LDAPInjections, DoS...



Blacklisting*magic*



Generic attack detection – we will talk about that
in some minutes...

Meaning the PHPIDS Centrifuge

ormigo 



GNUCITIZEN
Cutting-edge Think tank | Ethical Hacker Outfit

Reporting



As slim as possible

An attack was detected...

... a result object is filled with the necessary data

ormigo 



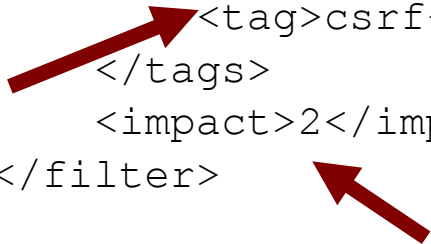
GNUCITIZEN
Cutting-edge Think tank | Ethical Hacker Outfit

Measuring



Any rule carries a numerical impact value.
Attack + Matching rules = Overall Impact.

```
<filter>
  <rule><![CDATA[(?:^>[\w\s]*<\/?\w{2,}>)]></rule>
  <description>finds unquoted attribute breaking in...</description>
  <tags>
    <tag>xss</tag>
    <tag>csrf</tag>
  </tags>
  <impact>2</impact>
</filter>
```



Reacting













Developers can define reactions based on the impact. Or the tags. Or the matching of one or several certain rules...



Logging

Use the integrated loggers – create backend tools like this:

 Eingegangene Angriffe

Impact	Seite	Feld	Vektor	IP	UserID	Datum	Test
64	/_eval, __unescape, ...	IDS_query_string	url=_eval, __unesca ...	87.79.54.64	---	01.05.2008 11:17:58	 PHIDS Test  POC  Löschen
22	/login/	data.User.email	' OR 1=1--#	87.79.54.64	---	01.05.2008 11:17:35	 PHIDS Test  POC  Löschen
34	/login/	data.User.email	"><script>alert(1)<..	87.79.54.64	---	01.05.2008 11:16:41	 PHIDS Test  POC  Löschen

Seite: /login/

Vektor: "><script>alert(1)</script>

But...

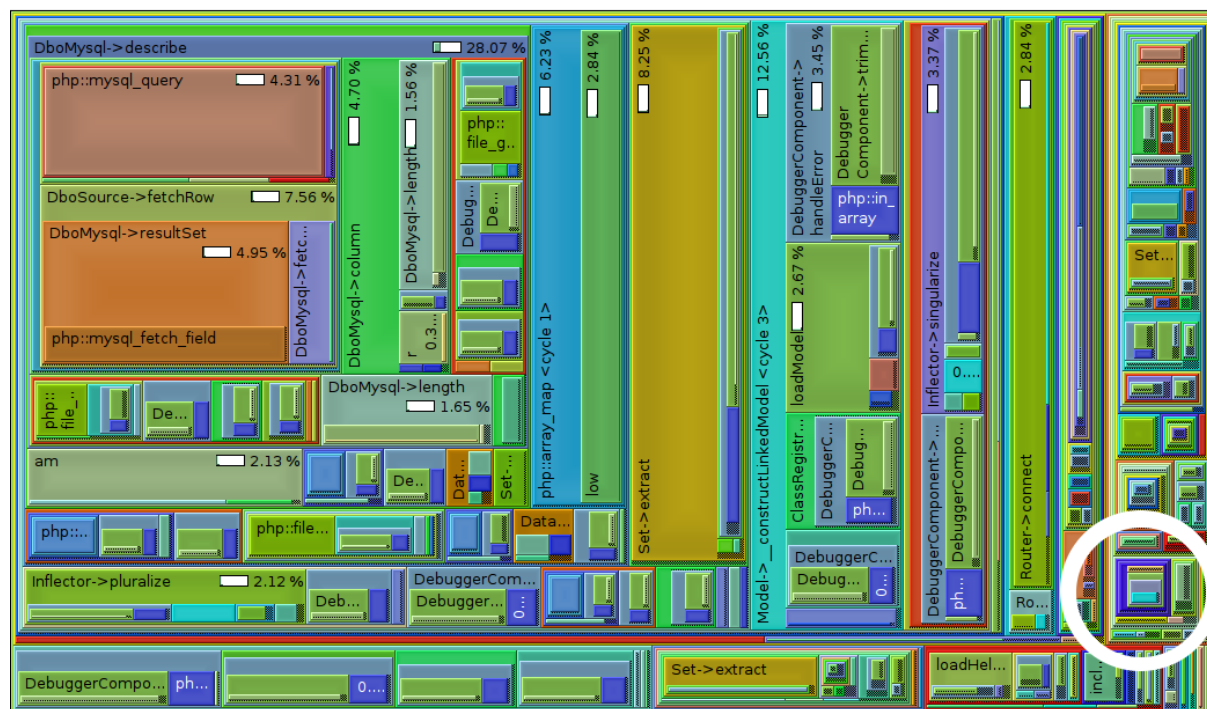


Isn't it *super slow* to pump user input through **70** regular expressions including a massive conversion process – again about **30-40** regular expressions?



Nup

Not when dealing with full-stack frameworks like CakePHP, Symfony, ZF or even WordPress



Choosing wisely



Nup? Nup! That's due to the caching mechanisms and a pre-selection.

95% of the user input won't even hit the rules and pass as harmless.

ormigo 



GNUCITIZEN
Cutting-edge Think tank | Ethical Hacker Outfit

But²...



What about **false alerts**?

Yes – depending on the application they exist. So the PHIDS sometimes needs some days to learn...

ormigo⁷



GNUCITIZEN
Cutting-edge Think tank | Ethical Hacker Outfit

Candy Time!



so - what about the PHPIDS Centrifuge?



The Centrifuge



Blacklisting alone is **useless**
Say thanks to SQL and JavaScript

```
ale&#x200d;rt(1)
```

```
aa'^+-(0)+-(0)='0
```

Unlimited ways of obfuscating payload



Know your foe



So what characterizes an attack?

Special chars! Loads of them!



Let's see..



```
if (strlen($value) > 25) {  
    // Check for the attack char ratio  
    $stripped_length = strlen(  
        preg_replace('/[\w\s\p{L}.,\/*]/ms', null, $value));  
    $overall_length = strlen(  
        preg_replace('/\w{3,}/', '123',  
        preg_replace('/\s{2,}/ms', null, $value)));  
  
    if($stripped_length != 0 && $overall_length/$stripped_length <= 3.5) {  
        $value .= "\n$[!!!]";  
    }  
}
```

ormigo 



GNUCITIZEN
Cutting-edge Think tank | Ethical Hacker Outfit

There's more...



```
if (strlen($value) > 40) {
    // Replace all non-special chars
    $converted = preg_replace('/[\\w\\s\\p{L}]/', null, $value);

    // Split string into an array, unify and sort
    $array = str_split($converted);
    $array = array_unique($array);
    asort($array);

    // Normalize certain tokens
    $schemes = array(
        '~' => '+', '^' => '+', '|' => '+', '*' => '+', '%' => '+',
        '&' => '+', '/' => '+' );
}
```



... and done!



```
$converted = implode($array);
$converted = str_replace(array_keys($schemes),
array_values($schemes), $converted);
$converted = preg_replace('/[+-]\s*\d+/', '+', $converted);
$converted = preg_replace('/[() [\] {}]/', '(', $converted);
$converted = preg_replace('/[!?,.:=]\/', ':', $converted);
$converted = preg_replace('/[^: (+]\/', null,
stripslashes($converted));

// Sort again and implode
$array = str_split($converted);
asort($array);
$converted = implode($array);

if (preg_match('/(?:\({2,}\+\{2,}:\{2,}\)|(?:\({2,}\+\{2,}:+)|' .
'(\{3,}\+\+\{2,}\)\/', $converted)) {
    return $value . "\n" . $converted;
}
}
```



The tests tell us...



...that almost all real world attacks, JS worms, SQL Injection exploits and other stuff are detected by the PHPIDS Centrifuge.

Those who *weren't* detected got caught by the rules.



Btw.. the tests!



PHPIDS is unit tested, regression tested and community driven.

Please don't have a look the test files!

ormigo 



GNUCITIZEN
Cutting-edge Think tank | Ethical Hacker Outfit

Back to our friend...



impact: 7

rule: `(?:=\s*\d*\.\d*\?\d*\.\d*)|(?:[|&]{2,}\s*)|(?!\d+\.\d*\?)|(?:\V:[\w.]+,)|(?=.\+[\d\W]*\s*`
rule-description: Detects common XSS concatenation patterns 2/2
impact: 4

rule: `(?:[^\w\s=]on(?:g\>)\w+[^=_.+]*=[^$]+(?:\W|\>)?)`
rule-description: Detects possible event handlers
impact: 4

rule: `(?:\<[\V]?(?:[i]?frame|script|input|button|textarea|style|base|body|meta|link|object|embed|`
rule-description: Detects possibly malicious html elements including some attributes
impact: 4

rule: `(?:\{2,\}\+\{2,\}|(?:\{2,\}\+\{2,\};+)|(?:\{3,\}\+\+\{2,\})|(?!\$[!@])`
rule-description: Detects unknown attack vectors based on PHPIDS Centrifuge detection
impact: 7

Overall impact: 49

```
<img/  
/onerror="[$y=('a')]&[$z=$y+'ert']  
[a=(1?/ev/:0)[-1]+$y]($z)(1)"  
src=x>
```

Send

So...



The PHPIDS detects attacks.

Developers can choose on how to react.

The PHPIDS knows them weird encodings and charsets.

It's free and OSS.

It's community driven

60 Members, ~1000 Posts in the various testing threads

ormigo 



GNUCITIZEN
Cutting-edge Think tank | Ethical Hacker Outfit

Plus



It's in use on dozens of *real* hightraffic sites.

neu.de, shoppero.com, astalavista.com, ormigo.com,
doccheck.com, sevenload.de...



10x guys!



The PHPIDS core members,
Gareth Heyes, David Lindsay, Eduardo Vela,
Kishor, Giorgio Maone, Reiners, Ronald, tx,
kuza55, the guys from schokoeks.org and so
many others!

ormigo 



GNUCITIZEN
Cutting-edge Think tank | Ethical Hacker Outfit

Questions?



Now's the time to ask!

Else you would have to check the [whitepaper](#) for yourself
– or drop me a line or post to the group or the forum or check
sla.ckers.org.

ormigo 



GNUCITIZEN
Cutting-edge Think tank | Ethical Hacker Outfit

Thanks a lot for listening!