Snuffleupagus

A elephant with some salt,
in your php stack,
killing bug classes,
and virtual-patching,
what is remaining.
Backlog

We gave subsets of this presentation at various conferences, using various themes.
» At an invite-only conference

❤ BerlinSides ❤
At a small conference in Switzerland

♥ Black Alps ♥
At a big conference in Luxembourg

♥ Hacklu ♥
Can you guess our current theme?
» Hint
Hellooooooooooo
» Good evening

- We're super thrilled to be here
- We're working together at the same (French¹) company
- In the security team.
- It's called *NBS System*
- And it's a hosting company, you know, for websites.
- Also known as *the cloud*.

¹ Hence why we have the same lovely accent than everyone here.
» Story time!
Your security team

Fig 1. They are kick-ass and super-cool.
There is a new $customer website

Fig 1. The marketing is so happy about it, it's so shiny
» Using a fixed version wordpress

Fig 1. Your security team reaction
» The web agency

Fig 1. Artistic's depiction of your web agency
The agency was convincing

*Fig 1.* They told the management that they take security seriously
» Management

Fig 1. Management says that everything will be fine.
Your security team isn't convinced

Fig 1. This isn't going to end well
Hackers on the internet

Fig 1. Wow, look at this old-school wordpress, noice
» Surprise disclosure of a wordpress' RCE on FD

Fig 1. Your security team is "busy" at a conference: they aren't reachable
» Kiddies are pwning your website

Fig 1. Kiddies, launching exploits
Your DB is encrypted by a lame ransomware

Fig 1. "Wait, what backups are you talking about", replied your admin
» Public image

*Fig 1.* Your company is looking like a bunch of idiots.
» Fixing the website

**Fig 1.** Your security team spent their week-end removing webshellz
What problem are we trying to solve?

1. We're hosting several thousands of websites, most of them are written in PHP.

2. PHP is an *old-school trigger-happy footgun language*, with massively creative users.

How do we prevent our customers from being pwned on a daily basis?
What we were doing so far

- We have a dedicated security team
- We have cool OS-level hardening (grsecurity ♥)
- We have custom IDS
- We have a fancy WAF called *naxsi*

But not everything is patchable with those and we can *not*² touch the PHP code.

¹ And to be honest, we don’t want to.
» Can't we harden PHP itself?

- **Suhosin** did it, and it worked great, but we're in 2018 and:
  - It has some useless features
  - It lacks some useful features
  - It is not very industrializable
  - It doesn't fly on PHP7
So we wrote our own hardening module, in C!

Fig 1. Snuffleupagus
» Snuffleupagus?

cbrocas commented 5 days ago

Hi Ju and friends!
As a conference organizer you are going to come to speak about this project, I had to deal with this f**ing name far more than I would ever wanted to!
Please be kind with your users, just drop this Sn{uflepags} horror name and choose something short, pronounceable and user convenient :) 
Thanks for preserving the infosec community health :D
Cheers, Christophe
Snuffleupagus?

jvoisin commented 5 days ago • edited

I’ll be happy to:

1. buy **you** a beer at *pass the salt*;
2. explain on stage in great details why we chose this specific name;
3. Update and close this issue with the slides after the talk

Is this an acceptable solution for you?

cbrocas commented 4 days ago

Great answer! Particularly awaiting the point 2) ;-) 
This solution is totally fine from my point of view!
Aloysius Snuffleupagus, more commonly known as Mr. Snuffleupagus, Snuffleupagus or Snuffy for short, is one of the characters on *Sesame Street*.

He was created as a woolly mammoth, without tusks or (visible) ears, and has a long thick pointed tail, similar in shape to that of a dinosaur or other reptile.

— wikipedia
» Where does it live

Apache
- mod_cgi
- mod_auth
- mod_heartmonitor

PHP
- pdo.so
- snuffleupagus.so
- sodium.so

Filesystem
- index.php
- admin.php
- backdoor.php
» PHP-level virtual patching
The issue

- `disable_function` can globally forbid usage of arbitrary functions
- Your CMS is using `system` for its update mechanism
- Either forbid `system` or keep your website up to date
- This is why we can't have nice things.
» How we're helping

- Disable `system` globally:

  ```javascript
  sp.disable_functions.function("system").drop();
  ```

- Allows `system` calls in a specific file

  ```javascript
  sp.disable_functions.function("system").filename("up.php").allow();
  sp.disable_functions.function("system").drop();
  ```

- Allow `system` calls in a file, with a matching sha256:

  ```javascript
  sp.disable_functions.function("system").filename("up.php").hash("13..a").allow();
  sp.disable_functions.function("system").drop();
  ```

We even provide a **user-friendly** script to generate a configuration file, freezing dangerous functions usage.
What can we do with php-level virtual-patching?
About the syntax

We designed¹ the rules syntax like this:

- 24 different filters
- Documentation for everything
- Lots of examples

to be able to easily patch:

- every *wordpress* CVE since 2010
- the *RIPS advent calendar*
- a lot of *high-profile* web exploits
- our own 0dayz ;)

¹ Designing configuration formats is awful, if you’re wondering.
Examples

```php
sp.disable_function("PHPThingy::MyClass::method_one>internal_func").drop();
sp.disable_function("admin_cron_thingy").cidr("127.0.0.1/32").allow();
sp.disable_function("admin_cron_thingy").drop();
sp.disable_function.function("render_tab3").var("_REQUEST[tab]").value_r("\" ").drop();
sp.disable_function.function("system").pos("0").value_r("[^a-z]").drop();
```
Regarding this morning

```javascript
sp.disable_function.filename("change.php").param("confirmpassword").param_type("array").drop();
sp.disable_function.filename("change.php").param("newpassword").param_type("array").drop();
sp.disable_function.filename("change.php").param("oldpassword").param_type("array").drop();
sp.disable_function.filename("change.php").param("login").param_type("array").drop();

# Will this work?
sp.disable_function.function("ldap_bind").ret("false").drop();
```
What can we do with this?
system() injections
When allowing user-supplied data to be passed to this function, use `escapeshellarg()` or `escapeshellcmd()` to ensure that users cannot trick the system into executing arbitrary commands.

```php
<?php
    $ip_addr = system("dig +short " . $_GET['address']);
    echo "The ip address of \$_GET['address'] is $ip_addr";
?>
```
What we're getting

- CVE-2017-7692: Authen RCE on SquirrelMail
- CVE-2016-9565: Unauth RCE on Nagios Core
- CVE-2014-1610: Unauth RCE on DokuWiki
- *Every single* shitty modem/router/switch/IoT.

How we're (kinda) killing it

```python
sp.disable_function.function("system").param("command").value_r("[$;\n\`"]) .drop();
```
mail related RCE
What the documentation is saying

The `additional_parameters` parameter can be used to pass *additional flags* as command line options to the program configured to be used when sending mail.

Known since 2011, popularized by RIPS.

What people are doing

```php
// Olol, sending some emails
mail(..., $_GET['a']);
```
» What we're getting

- CVE-2017-7692: Authen RCE in SquirrelMail
- CVE-2016-10074: RCE in SwiftMailer
- CVE-2016-10033: RCE in PHPMailer
- CVE-2016-9920: Unauth RCE in Roundcube
- RCE in a lot of webmails

» How we're ( kinda) killing it

```javascript
sp.disable_function.function("mail").param("additional_parameters").value_r("\-").drop();
```
» Writing rules

*Fig 1.* When the security team realises that it needs to write a lot of rules.
Nobody has time to write rules

So lets kill some bug classes!
» Session-cookie stealing via XSS

Like suhosin, we're encrypting cookies with a secret key tied to:

- The *user-agent* of the client
- A *static key*
- And *environment variable* tat you can set to:
  - The *ip address*¹
  - The *TLS extended master key*
  - ...  

¹ Not the best idea ever: in 2017, people are roaming *a lot.*
» Misc cookies things

- If you're coming over https, your cookies get the `secure` flag
- If cookies are encrypted, they are `httpOnly`
- Support for `SameSite` to kill CSRF
RCE via file-upload
What the documentation is saying

Not validating which file you operate on may mean that users can access sensitive information in other directories.

What people are doing

```php
$uploaddir = '/var/www/uploads/';
$uploadfile = $uploaddir . basename($_FILES['userfile']['name']);
move_uploaded_file($_FILES['userfile']['tmp_name'], $uploadfile)
```
What we're getting

- CVE-2001-1032: RCE in PHP-Nuke via file-upload
- ...
- 15 years later
- ...
- CVE-2016-9187: RCE in Moodle via file-upload

There are 850 CVE entries that match your search
— cve.mitre.org
How we're killing it

Suhosin style:

```php
sp.upload_validation.script("tests/upload_validation.sh").enable();
```

One trick is to rely on `vld¹` to ensure file doesn't contain php code:

```bash
$ php -d vld.execute=0 -d vld.active=1 -d extension=vld.so $file
```

¹ Vulcan Logic Disassembler. (yes)
Unserialize
» What the documentation is saying

*Do not* pass untrusted user input to `unserialize()` [...]. Unserialization can result in code being loaded and executed [...].

» What people are doing

$my_object = unserialize($_GET['o']);
» Small aparté about **unserialize**

**Fig 1.** The security team reading PHP's mailing list
» What we're getting

- **CVE-2012-5692**: unauth RCE in IP.Board
- **CVE-2014-1691**: Unauth RCE in Horde
- **CVE-2015-7808**: Unauth RCE in vBulletin
- **CVE-2015-8562**: Unauth RCE in Joomla
- **CVE-2016-????**: Unauth RCE in Observium (leading to remote root)
- **CVE-2016-5726**: Unauth RCE in Simple Machines Forums
- **CVE-2016-4010**: Unauth RCE in Magento
- **CVE-2017-2641**: Unauth RCE in Moodle

» How we're killing it

Php will discard any garbage found at the end of a serialized object: we're simply appending a **hmac** at the end of strings generated by **serialize**.

It looks like this:

```
$s = "a";650609b417904d0d9bfb1fc44a975d13ecdf6b02b715c1a06271fb3b673f25b1
```
rand and its friends
What the documentation is saying

This function does not generate cryptographically secure values, and should not be used for cryptographic purposes.

What people are doing

```php
$password_reset_token = rand(1,9) . rand(1,9) . [...] . rand(1, 9);
```
» What we're getting

- CVE-2008-4102: Auth bypass in Joomla
- ...
- CVE-2015-5267: Auth bypass in Moodle
- Various captcha bypasses

» How we're killing it

We're simply replacing every call to `rand` and `mt_rand` with `random_int`.
» What the documentation is saying

Not a single warning ;)

» What people are doing

```php
$xmlfile = file_get_contents('php://input');
$dom = new DOMDocument();
$dom->loadXML($xmlfile);
$data = simplexml_import_dom($dom);
```
» What we're getting

- CVE-2011-4107: Authen LFI in PHPMyAdmin
- ...
- CVE-2015-5161: Unauth arbitrary file reading on Magento

» How we're killing it

We're calling `libxml_disable_entity_loader(true)` at startup, and *nop'ing* its call.
Unrelated misc things

```php
# chmod hardening
sp.disable_function.function("chmod").param("mode").value_r("7$");
sp.disable_function.function("chmod").param("mode").value_r("o\+w");

# backdoors detection
sp.disable_function.function("ini_get").param("var_name").value("open_basedir");
sp.disable_function.function("is_callable").param("var").value("system");

# prevent execution of writeable files
sp.readonly_exec.enable();

# Ghetto sqli detection
sp.disable_functions.function_r("mysqli?_query").ret("FALSE").dump().allow();
sp.disable_functions.function_r("PDO::query").ret("FALSE").dump().allow();
```
Harvesting 0days

If you've got something like this

```bash
$line = system("grep $var dict.txt");
```

You can do something like that

```javascript
sp.disable_function.function("system").var("var").regexp("[;`&|\n]").dump().allow();
```

And wait until someone finds a vuln to collect a working exploit.
Performance impact

- Currently deployed on (at least) one Alexa1 top 1k website.
- We're using it on some customers
- No performance impact noticed
- We're (kinda) only hooking the functions that you specify
- Filter-matching is written with performances in mind
What's left to do

- Killing more bug-classes, like sloppy-comparisons and SQLI¹
- Improve the virtual patching capabilities
- Party party party

¹ We're working on it ;)

How to get this wonder?

- [https://github.com/nbs-system/snuffleupagus](https://github.com/nbs-system/snuffleupagus) for the sauce code
- [https://snuffleupagus.rtfd.io](https://snuffleupagus.rtfd.io) for the (amazing) documentation
- Come talk to us, we're friendly!
There are only two kinds of languages: the ones people complain about and the ones nobody uses.

— Bjarne Stroustrup

Did you know that more than \( \frac{3}{4} \) of the web is using PHP?
Cheers

- The *RIPS* people for their awesome scanner
- *SectionEins* for Suhosin and inspiration
- The *HardenedPHP* project for leading the way
- *websec.fr* for showcasing our most convoluted exploits
- Our guinea pigs *friends* who alpha-tested everything
- Folks that *called us names* gave us constructive feedback
- Pass the Salt for accepting our talk ❤