» Snuffleupagus

Killing bug classes,
virtual-patching the rest!

» Who are we?

- We're working at the same (French[1]) company
- In the security team.
- It's called ***NBS System***
- And it's a hosting company, you know, for websites.

[1] Hence our lovely accent.

» What are we trying to solve?

We're hosting a lot of various [redacted] php applications using CMS [redacted] [redacted] [redacted] [redacted] [redacted], and we'd like to prevent our customers from being pwned.

» What we were doing so far

- We have a lot of os-level hardening
- We have some custom IDS
- We have a (cool) WAF called naxsi

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

» Some words about php

> Its syntax draws upon C, Java, and Perl, and is easy to learn. The main goal of the language is to allow web developers to write dynamically generated web pages quickly, but you can do much more with PHP.
>
> — the php documentation

» Still words about php

> Well, there were other factors in play there. htmlspecialchars was a very early function. Back when PHP had less than 100 functions and ***the function hashing mechanism was strlen()***. In order to get a nice hash distribution of function names across the various ***function name lengths names were picked specifically to make them fit into a specific length bucket***.
>
> — Rasmus Lerdorf, creator of PHP

» Words about php, again

> I don’t know how to stop it, ***there was never any intent to write a programming language*** […] I have absolutely no idea how to write a programming language, I just kept adding the next logical step on the way.
>
> — Rasmus Lerdorf, creator of PHP

» By the way…

The php way to kill bug classes is to (sometimes) add a warning to its documentation, like this, about `rand`:

> This function does not generate cryptographically secure values, and ***should not be used for cryptographic purposes***. If you need a cryptographically secure value, consider using `random_int()`, `random_bytes()`, or `openssl_random_pseudo_bytes()` instead.

» Fortunately...

Did we mention that ***anyone*** is able to add comments to the official PHP documentation?

> If you are looking for generate a random expression, like password with alphanumeric or any other character, use this function:

```
function GeraHash($qtd){
$Caracteres = 'ABCDEFGHIJKLMOPQRSTUVXWYZ0123456789';
$QuantidadeCaracteres = strlen($Caracteres);
$QuantidadeCaracteres--;

$Hash=NULL;
    for($x=1;$x<=$qtd;$x++){
         $Posicao = rand(0,$QuantidadeCaracteres);
        $Hash .= substr($Caracteres,$Posicao,1);
    }

return $Hash;
}
```

» What about hardening php itself?

- Suhosin did it, and it worked great, but we're in 2017 and
  - It has some useless features
  - It lacks some useful features
  - It's not very industrializable
- Suhosin7 is not production-ready anyway :'(
- Lets write our own PHP hardening patch!
- Behold the ***snuffleupagus***!

» Snuffleupagus?!

> 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

» An elephant as *majestic* as php itself

» PROVIDING FINE-GRAINED CONTROL

» What we need

`disable_functions` lacks granularity, making it hard to use it in ***production***, however, it's a good way to make backdooring ***a lot harder***.

» How we're helping

- Disable `system` globally:

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

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

```
sp.disable_functions.function("system").filename("update.php").hash("d2..a");
```

- Allow `system` to be called in a specific file:

```
sp.disable_functions.function("system").filename("update.php");
```

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 designed[1] the rules syntax 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 as fuck by the way.

» `system()` injections

» What the documentation is saying

> 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***.

» What people are doing

```
<?php
$ip_addr = system("dig +short " . $_GET["address"]);
echo "The ip adress 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

```
sp.disable_function.function("system").param("command").value_r("[$|;&`]");
```

» `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

```
// 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

```
sp.disable_function.function("mail").param("additional_parameters").value_r("\-");
```

» Killing bug classes

Because no one will bother writing virtual-patching rules.

» Session-cookie stealing via XSS

Like suhosin, we're encrypting[1] cookies with a secret key tied to the ***IP*** and ***user-agent*** of the user.

[1] And authenticating ;)

» 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

```
$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:

```
sp.upload_validation.script("tests/upload_validation.sh")
sp.upload_validation.simulation(0)
```

One trick is to rely on `vld`[1] to ensure file doesn't contain php code:

```
$ 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']);
```

» What we're getting

- `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
- `CVE-2015-8562`: Unauth RCE in Joomla
- `CVE-2015-7808`: Unauth RCE in vBulletin
- `CVE-2014-1691`: Unauth RCE in Horde
- `CVE-2012-5692`: unauth RCE in IP.Board

» 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`.

» `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

```
$password_reset_token = rand(1,9) . rand(1,9) . [...] . rand(1, 9);
```

» What we're getting

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

» How we're killing it

We're simply replacing every call to `rand` and `mt_rand` with `random_int`.

» XXE

» What the documentation is saying

Not a single warning ;)

» What people are doing

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

» What we're getting

- `CVE-2015-5161`: Unauth arbitrary file reading on Magento
- `CVE-2014-8790`: Unauth RCE in GetSimple CMS
- `CVE-2011-4107`: Authen LFI in PHPMyAdmin

» How we're killing it

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

» Practical example

» Redacted slide

On this slide, we burned several ***0days***, and showed how to patch them with snuffleupagus.

» Unrelated misc things

```
# 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(1);

# Ghetto sqli detection
sp.disable_functions.function_r("mysqli?_query").ret("FALSE");
sp.disable_functions.function_r("PDO::query").ret("FALSE");

# Ghetto sqli hardening
sp.disable_functions.function_r("mysqli?_query").param("query").value_r("/\*");
sp.disable_functions.function_r("mysqli?_query").param("query").value_r("--");
sp.disable_functions.function_r("mysqli?_query").param("query").value_r("#");
sp.disable_functions.function("PDO::query").param("query").value_r("/\*");
sp.disable_functions.function("PDO::query").param("query").value_r("--");
sp.disable_functions.function("PDO::query").param("query").value_r("#");
```

» HARVESTING 0DAYS

If you've got something like this

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

You can do something like that

```
sp.disable_function.function("system").var("var").regexp("[;`&|]").dump().log();
```

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

» Ongoing implementations

- `include/require` support in ***partial disable functions***.
- `eval`-specific rules
- `generic` function hooking

» What's left to do

- Dealing with `include`, `require` at al.
- Playing nice with `eval`
- Finding and fixing bugs
- Killing more bug-classes, like sloppy-comparisons and SQLI[1]

[1] We're working on it ;)

» What we're going to do

- Finding beta testers[1]
- Releasing it as open-sauce
- Keep maintaining it for years

Send us an email at `snuffleupagus@nbs-system.com` if you want to join the party!

» SPEAKING OF PHP, DID YOU KNOW THAT...

» PHP SUPPORTS EMOJI

```
<?php
function 💩 ($♥) {
    echo $♥;
}
$👩‍⚕ = 1;
echo $👩‍⚕;
💩(42);
```

» PHP7 IS NOW USING `zend_string`

- `Z_STRVAL` to get the `char*` value from a `zval*`
- `ZSTR_VAL` to get the `char*` value from a `zend_string*`
- `Z_STR` to get the `zend_string*` from a `zval*`
- `ZVAL_STRING` to create a `zval` from a `char*`
- `ZVAL_STR` to create a `zval` from a `zend_string*`
- `ZSTR_ALLOCA_ALLOC` to allocate a `zend_string*`
- `STR_ALLOCA_ALLOC` does the same thing.
- `ZSTR_ALLOCA_INIT` to allocate **and** init a `zend_string` from a `char*`
- `ZVAL_NEW_STR` assign a `zval*` from a `zend_string*`

» If you want to walk the calltrace back

You have to:

1. Overwrite the current context with the previous one
2. Rebuild the symbols with `zend_rebuild_symbol_table`
3. Do your business
4. Goto *1*
5. Restore the first one back.

» MANDATORY FINAL QUOTE

> There are only two kinds of languages: the ones people complain about and the ones nobody uses.
>
> — Bjarne Stroustrup

I guess this is why php is used a lot.

» Cheers

- The RIPS people for their awesome scanner
- SectionEins for Suhosin and inspiration
- websec.fr for showcasting our most convoluted exploits

» QUESTIONS ?

EVERY
DAY I'M
SNUFFLIN'