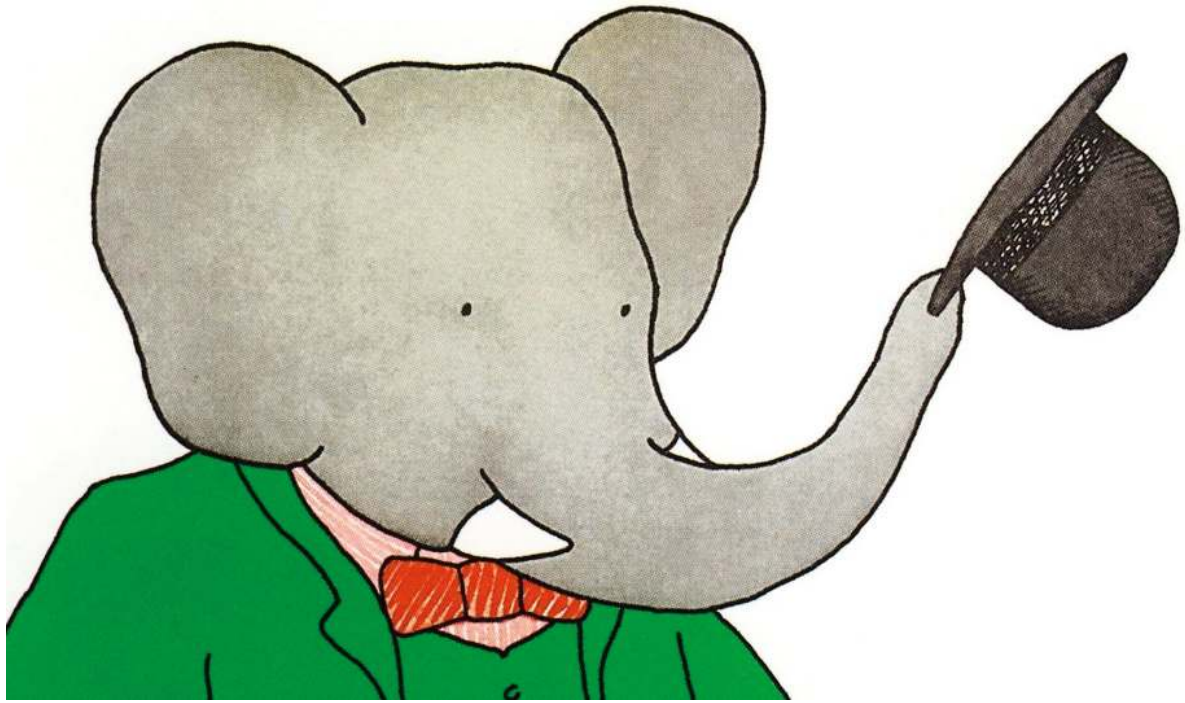
» SNUFFLEUPAGUS

A pretty elephant,
killing bug classes,
putting sparadrap on the rest.

» Bonjour



» Good morning

- We're happy to be here
- 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 php applications,
using CMS written by many different super-creative people around the world,
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***[1] touch the PHP code.

[1] Nor do we want to.

» Some stories about PHP

Fig 1. The security team learning about the development processes

» 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

» Words about php, again and again

> I was really, really bad at writing parsers. ***I still am really bad at writing parsers***.
>
> — Rasmus Lerdorf, creator of PHP

» Words about php, again and again and again and again

> We have things like protected properties. We have abstract methods. We have all this stuff that your computer science teacher told you you should be using. ***I don't care about this crap at all***.
>
> — 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:

```php
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;
}
```

» Developers

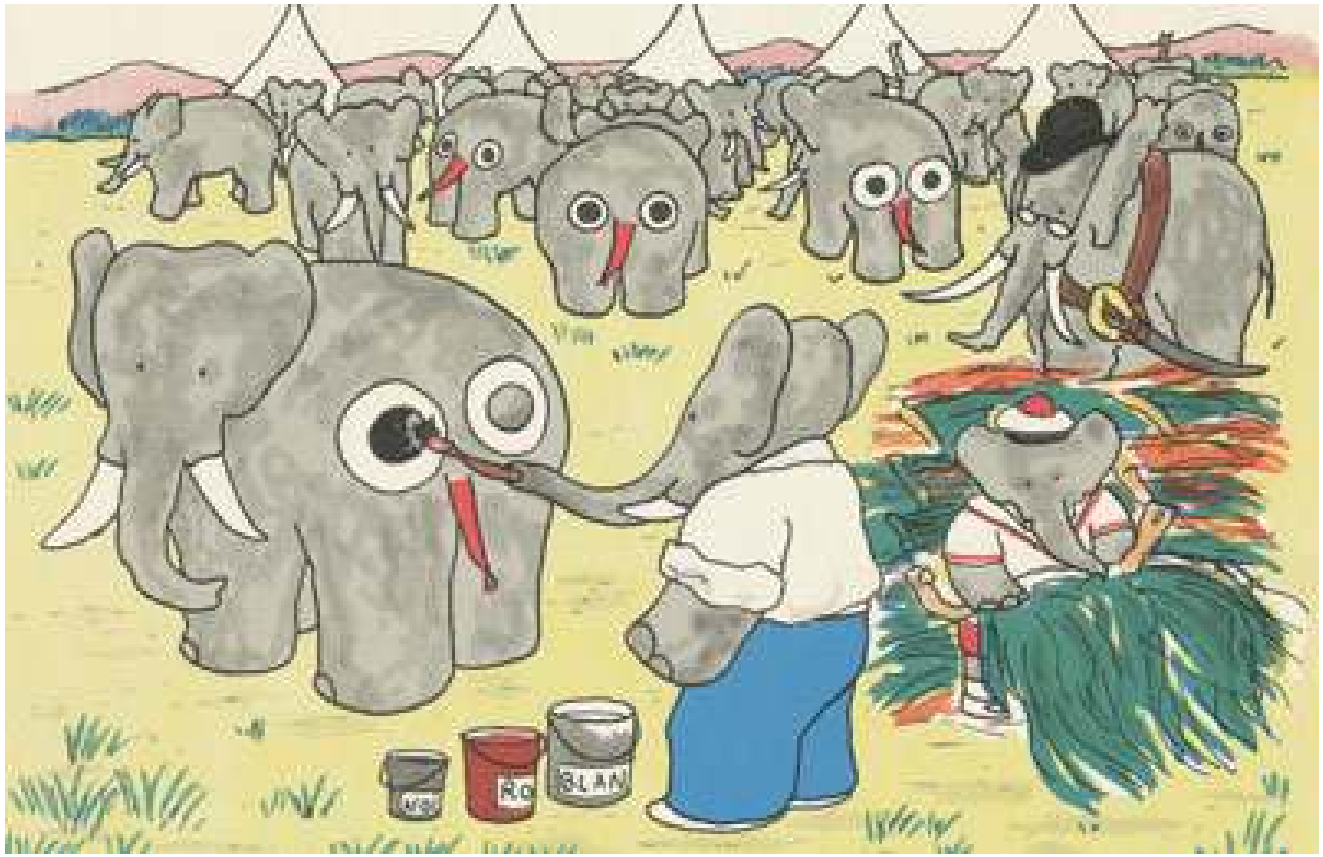


Fig 2. Picture of a developer copy-pasting vulnerable code between projects

» Hackers

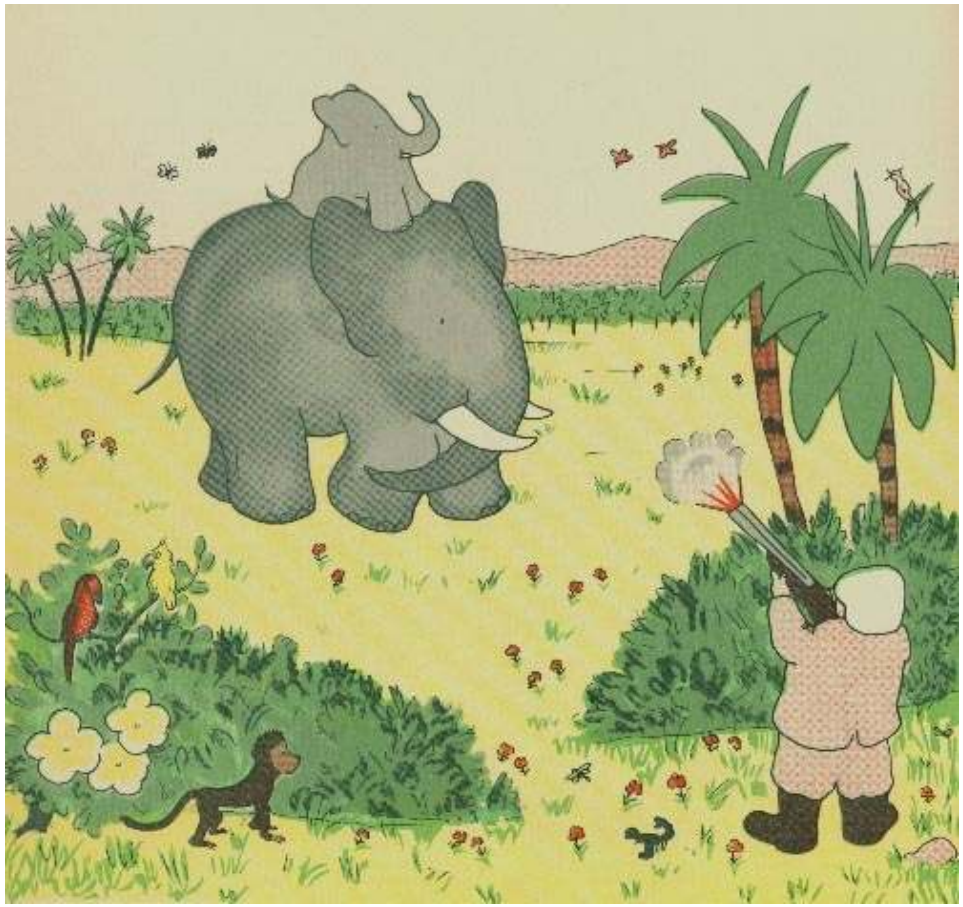
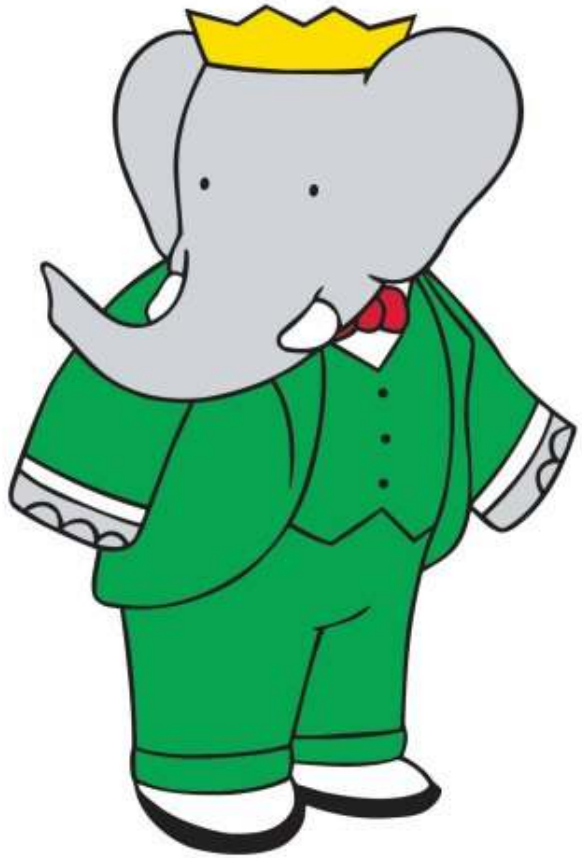


Fig 3. Rare footage of a hacker remotely pwning a PHP application

» What about hardening php itself?

- Suhosin did it, it worked great,
  but we're in 2017:
  - 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***!



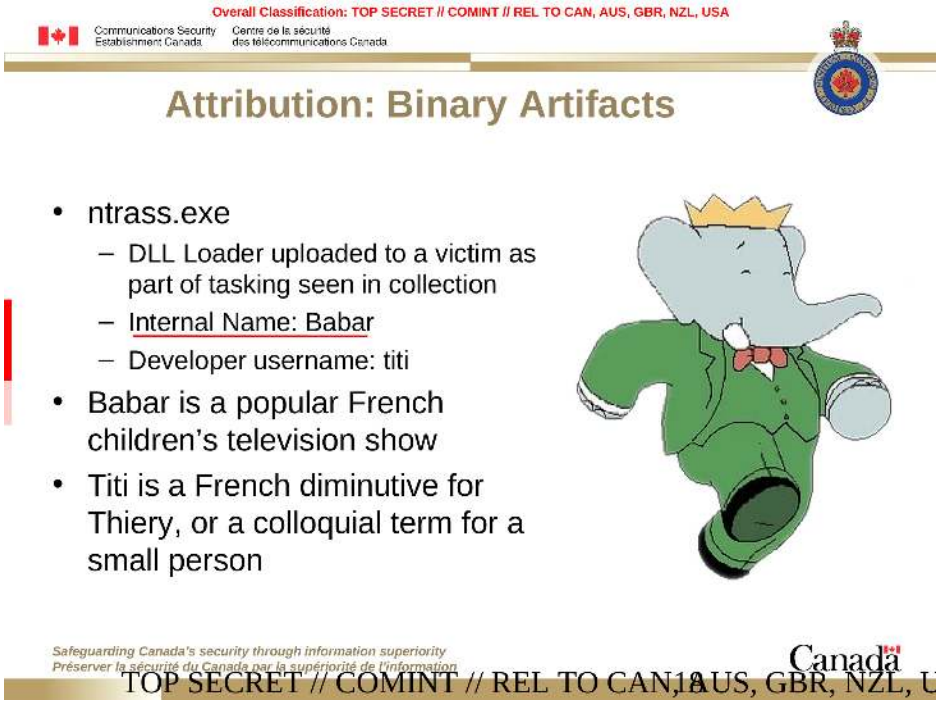
» Snuffleupagus?!

> 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

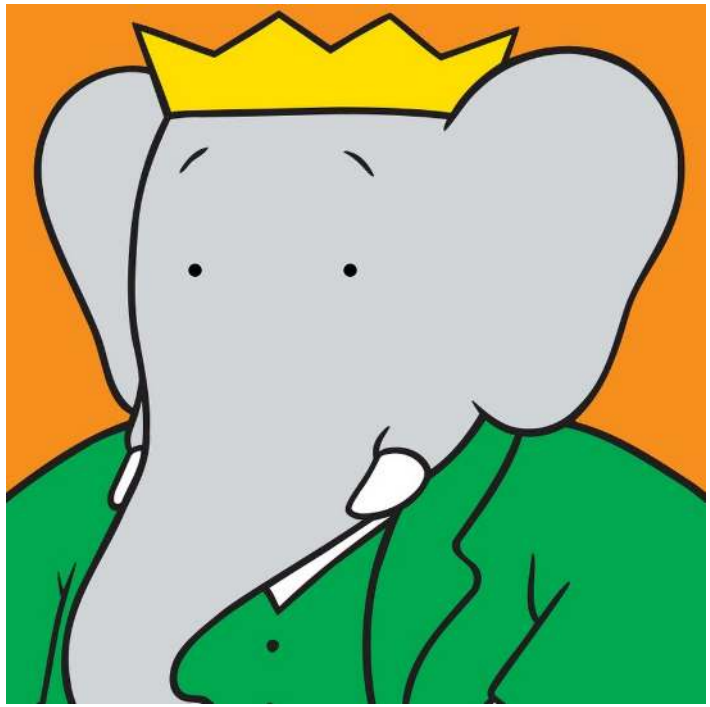
» The Snuffleupagus!

» Why Babar then?

» Totally Spies! talk from hacklu 2015!



» PHP LEVEL VIRTUAL-PATCHING



» The issue

- `disable_function`, that can globally forbid the usage of arbitrary functions
- Your CMS update system is using `system`
- Either forbid `system`, or keep your website out of date.
- That's why we can't have nice things.

» How we're helping

- Disable `system` globally:

```
sp.disable_function.function("system").drop();
```

- Allow `system` calls in a specific file:

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

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

```
sp.disable_function.function("system").filename("up.php").hash("d2..a").allow();
sp.disable_function.function("system").drop();
```

We even provide a **user-friendly** script to generate a configuration file, freezing dangerous functions usage.

» About the syntax

We "designed"[1] the rules syntax like this:

- 24 different filters
- Everything is documented
- Lots of examples

to be able to patch:

- every ***wordpress*** CVE since 2010
- the ***RIPS advent calendar***
- a lot of ***high-profile*** web exploits
- our own 0dayz, more on this in a few slides…

[1] Designing configuration formats is ***super-duper-awful as fuck*** in case you're wondering.

» Rules examples

```
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();
```

» WHAT CAN WE DO WITH THIS?

» `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
<?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("[$|;&\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

```
// 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("\-").drop();
```

» MANAGING RULES

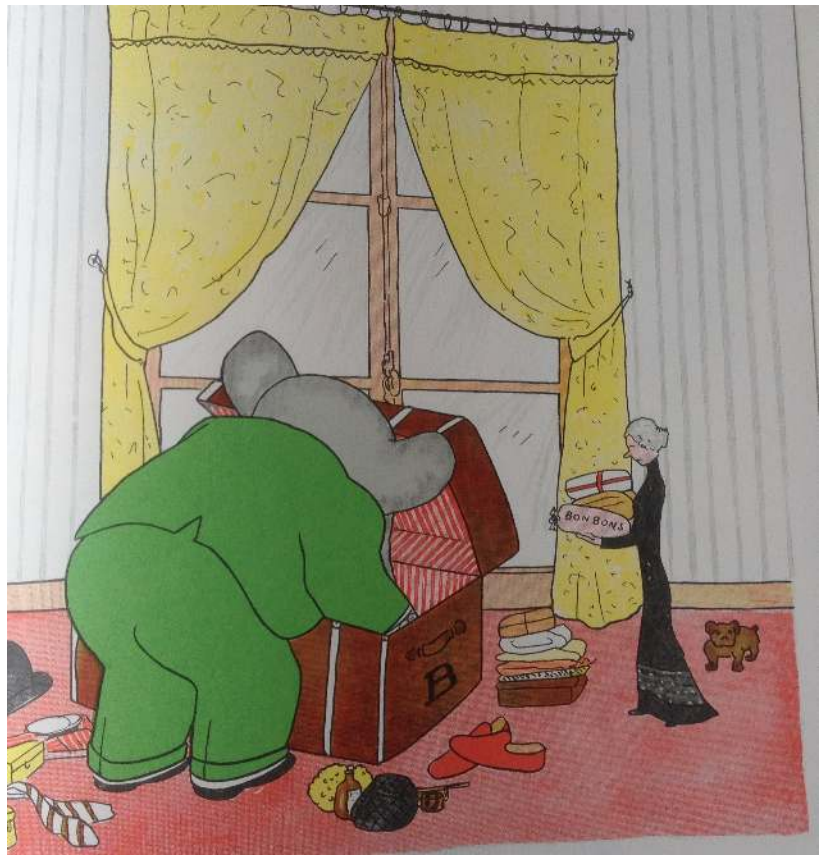


Fig 4. The security team bringing new rules to the sysadmin

» AIN'T NOBODY HAS TIME TO WRITE RULES

» DEAD BUG CLASSES

Fig 5. PHP CMS free from various security bug classes

» Session-cookie stealing via XSS

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

- The ***user-agent*** of the user
- A ***static key***
- An ***environnement variable*** that you can set to:
  - The ***IP addresses***[1]
  - The ***TLS extended master key***
  - …

[1] No the best idea ever: in 2017, people are roaming ***a lot***.

» Misc cookie things

- If you're coming over https, your cookie gets the `secure flag`
- If cookies are encrypted, they are `httpOnly`

» 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`.

It looks like this:

```
s:1:"a";650609b417904d0d9bbf1fc44a975d13ecdf6b02b715c1a06271fb3b673f25b1
```

» `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 the secure PRNG `random_int`.

» XXE

» What the documentation is saying

Not a single word about this ;)

» 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.

» THE SECURITY TEAM IS FINALLY LESS GRUMPY

Fig 6. Photo of the security team admiring a protected PHP stack

» Time for a practical example

» Earlier yesterday evening

> — What about auditing something to find vulns to burn?
>
> — What about burning old ones instead so we can get wasted at the lobby?
>
> — Even better!

» REDACTED

On this slide, we burned a RCE in a well-known monitoring software, and shown how to patch it with snuffleupagus.

» MISC COOL STUFF

» 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();

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

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

» COLLECTING

Fig 8. The security team welcoming new vulnerabilities

» 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().drop();
```

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

» PERFORMANCE IMPACT

- Snuffleupagus is currently deployed on an Alexa[1] top 8k website.
- We're using it on some customers
- No performance impact noted
- We're only hooking the functions that you specify
- Filter-matching is written with performances in mind

[1] Totally not ***toolslib.net*** and some ***Malwarebytes*** backend

» CLOSE-TO-NO-PERF-IMPACT

Fig 9. A Snuffleupagus-hardened stack running at full speed

» WHAT'S LEFT TO DO

- Fixing known bugs
- Finding and fixing new bugs
- Killing more bug-classes, like CSRF, sloppy-comparisons and SQLI[1]
- Party hard[2]

[1] We're working on it ;)
[2] We're working on it too.

» How can you get this?

- https://github.com/nbs-system/snuffleupagus for the sauce code
- https://snuffleupagus.rtfd.io for the (amazing) documentation
- Come talk to us, we're friendly!

» RESULTS

Fig 10. Customers using the website without noticing that it's a PHP trashfire

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

» PHP SUPPORTS EMOJI

```php
<?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*`

» CRITICAL RECEPTION

> So basically this "security module" is like an elephant in a china store, walking around and clumsily breaking things with the best of intentions. What a nightmare...
>
> — Someone on /r/php

.

> Over all, lots of good features pretty cleverly and unobtrusively implemented.
>
> — Soneone on /r/netsec

» MANDATORY FINAL QUOTE

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

Did you know that ¾ 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 showcasting our most convoluted exploits
- Our ~~guinea pigs~~ friends who alpha-tested everything
- People that ~~called us names~~ gave us constructive feedback