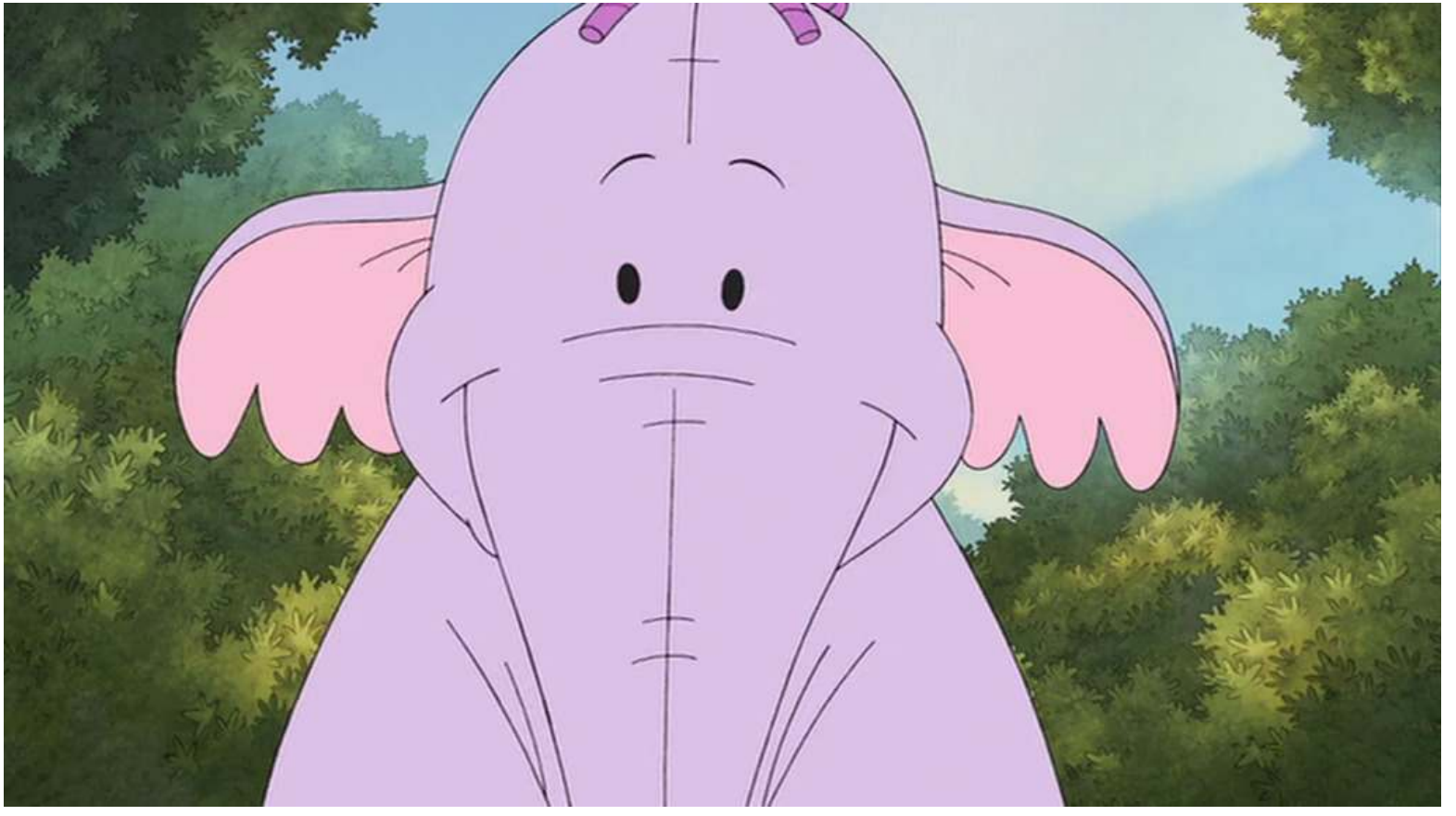
» SNUFFLEUPAGUS

A ghostly elephant,
in your php stack,
killing bug classes,
defeating attacks,
and virtual-patching
what is remaining.

» Bonjour



» I'm sorry

» Good evening

- We're glad 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 super-duper ***creative*** people, 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 (grsecurity ♥)
- We have some custom IDS
- We have a (cool) 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.

» 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

> 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

» THE SADNESS

Fig 5. We're dealing with this every day at work

» 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;
}
```

» STORY TIME!

» Your security team

Fig 1. The security team trying to improve the security of your website

» Your blog

Fig 2. You know, the one based on a decade-old Drupal that cries for updates

» Your management

Fig 3. But the management is telling you that you're not allowed to update anything

» Hackers

Fig 4. Your database is harvested on a monthly basis by hackers

» Protections

Fig 5. "We have a fancy and expensive WAF, everything is ok"

» Kiddies

Fig 6. A new Drupal vulnerability is published, kiddies are pwning everything

» Your website is pwned

Fig 7. Your website is defaced with a big goatse

» Customers are going away

Fig 8. The management is screaming around

» Fixing the website

Fig 9. The security team spends the week-end removing webshells

» So, 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 :'(

» So we thought about it, …

Fig 10. Us thinking about it.

» And wrote our own module!

Fig 11. Snuffleupagus

» Snuffleupagus?!

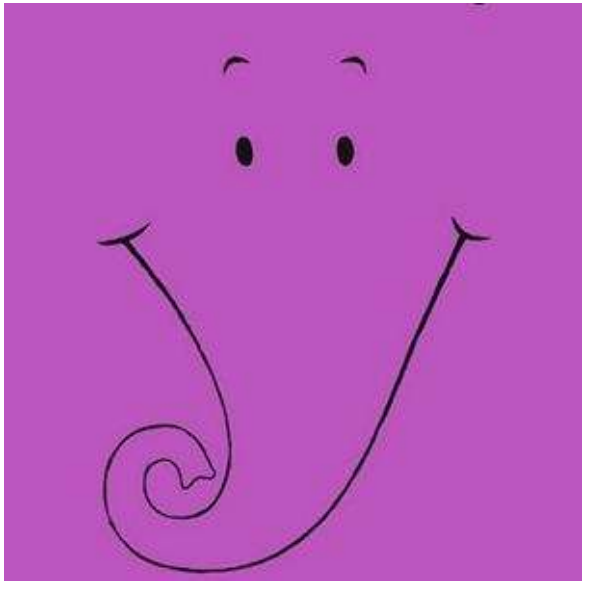
> Aloysius Snuffleupagus, more commonly known as Mr. Snuffleupagus, Snuffleupagus or Snuffy for short, is one of the characters on ***Sesame Street***.
>
> He was created as a woolly mammoth, without tusks or (visible) ears, and has a long thick pointed tail, similar in shape to that of a dinosaur or other reptile.
>
> — wikipedia

» Why the Heffalump then?

> Your slides are fun, and it's an elephant, so it's php related enough.
>
> — a coworker of ours

And there was a picture of Winnie the Pooh in the talk about Turla!

» PHP-LEVEL VIRTUAL PATCHING



» The issue

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

» THE SADNESS

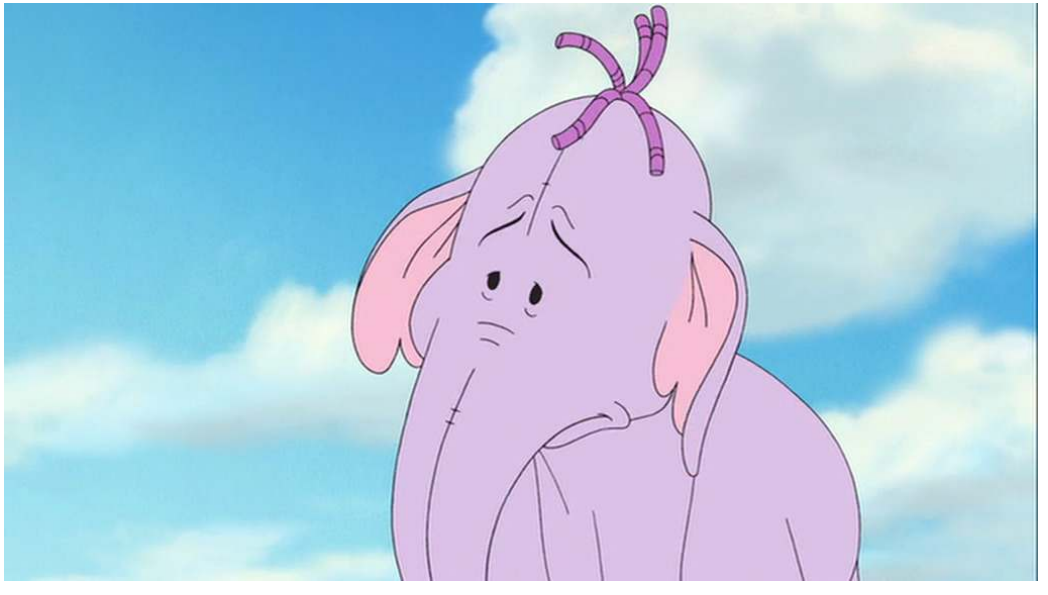


Fig 12. A sad website awaiting its security updates

» How we're helping

- Disable `system` globally:

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

- Allows `system` calls in a specific file

```
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:

```
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 designed[1] 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

```
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
```

» WRITING RULES

Fig 13. When the security team realises that it needs to write a lot of rules.

» AIN'T NOBODY HAS TIME TO WRITE RULES

» Dead bug classes

Fig 14. The security team dancing over dead bugs

» 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 ***environnment 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`
- You can set the `samesite` option on cookies, to ***kill CSRF***[1]

[1] Kudos to my intern, Sylvain Lefevre, for implementing this ♥.

» 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").enable();
```

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 `random_int`.

» XXE

» What the documentation is saying

Not a single warning ;)

» Nothing about XXE

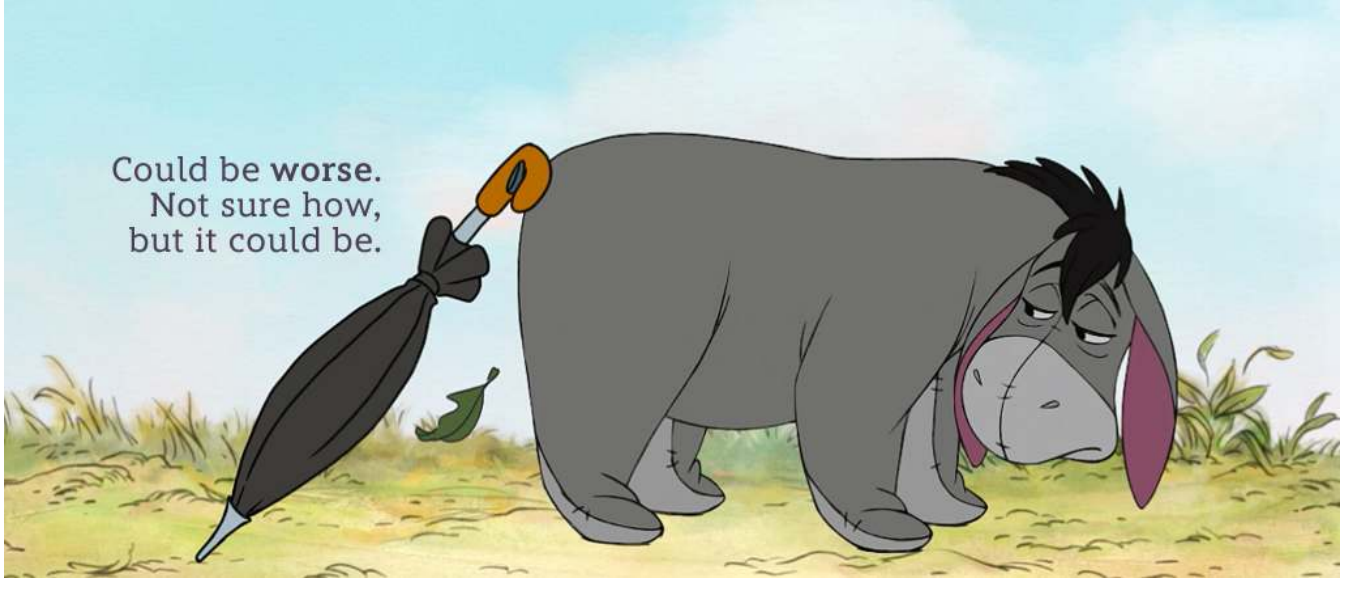


Fig 4. At least it's not enabled by default[1]

[1] For certain XML parsers, not all of them.

» 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

» A couple of months ago, in a bar, ...

> — I'm equally bored and drunk..
>
> — Let's audit something
>
> — What about LibreNMS?
>
> — Sure, whatever

» UNAUTHENTICATED RCE

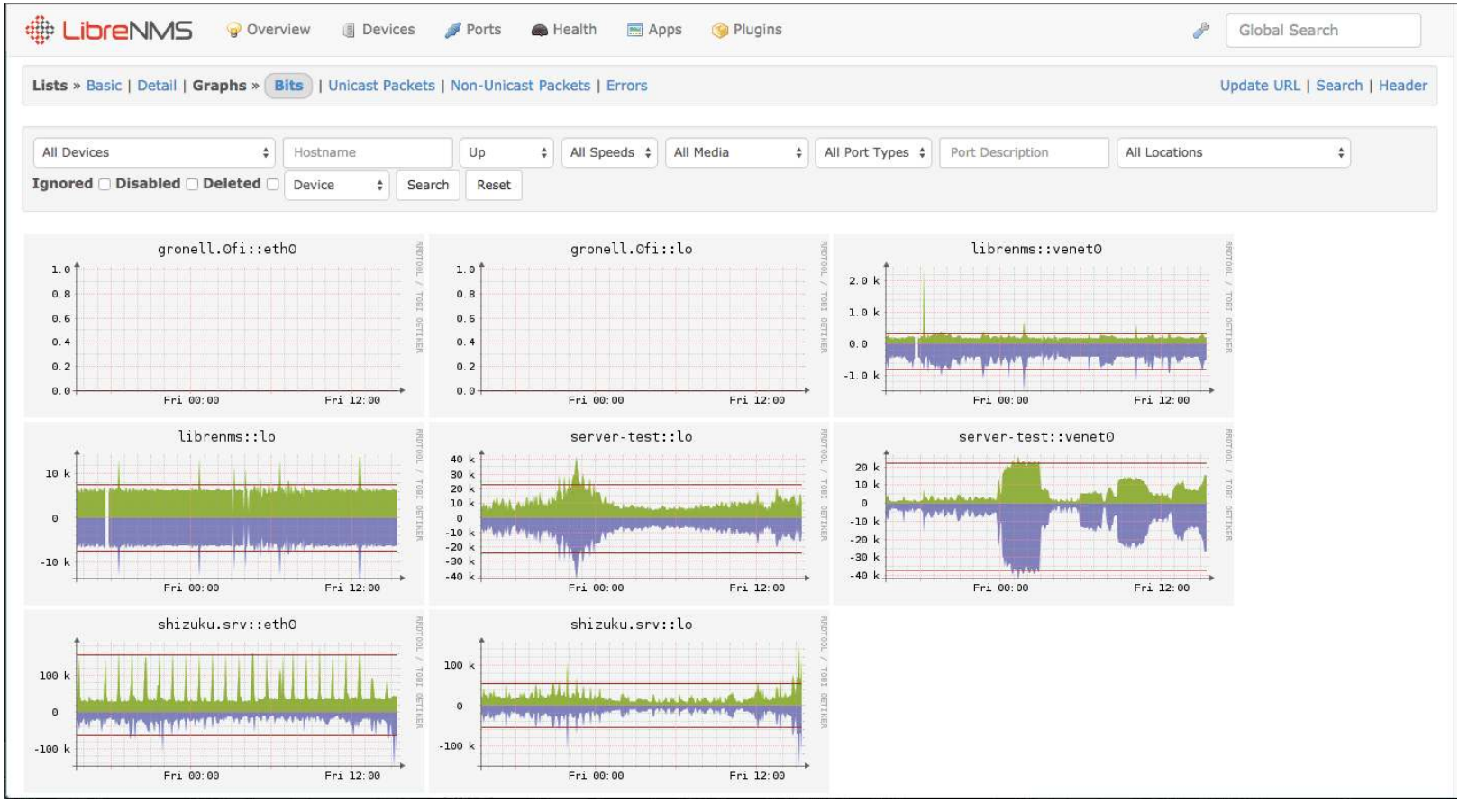
Fig 15. LibreNMS users in the room.

» What is librenms?

Welcome to LibreNMS, a fully featured network monitoring system that provides a wealth of features and device support.

More than 650 stars on github, and according to Twitter, has a lot of users.

» LIBRENMS SCREENSHOT



» RESPONSIBLE DISCLOSURE

- We burned those vulns at a private conference
- We showed them in a public one
- We're now publishing a metasploit module[1] for them

[1] https://github.com/rapid7/metasploit-framework/pull/9213

» Authentication bypass

install.php

```
if (empty($_POST) && !empty($_SESSION) && !isset($_REQUEST['stage'])) {
 $_POST = $_SESSION;
} else {
  $_SESSION = $_POST;
}
```

- `curl -i http://.../install.php -d "authenticated=1&userlevel=10&user_id=1"`
- You are now `admin`, congratulations.

» Virtual patching it

```
sp.filename("install.php").var("_POST[authenticated]").value_r(".*").drop();
sp.filename("install.php").var("_POST[userlevel]").value_r(".*").drop();
sp.filename("install.php").var("_POST[user_id]").value_r(".*").drop();
```

» RCE

netcmd.php

```
$host = clean($_GET['query']);
    if (filter_var($host, FILTER_VALIDATE_IP, FILTER_FLAG_IPV6) ||
        filter_var($host, FILTER_VALIDATE_IP, FILTER_FLAG_IPV4) ||
        filter_var('http://'.$host, FILTER_VALIDATE_URL)) {
...
swtch($_GET['cmd']):
...
  case 'nmap':
      ...
      $cmd = $config['nmap']." $host";
...
  $output = `$cmd`;
```

- `http://...//netcmd.php?cmd=nmap&query=http://xxxx;echo${IFS}$(id)`

```
Starting Nmap 7.40 ( https://nmap.org ) at 2017-05-25 19:35 CEST
Nmap done: 0 IP addresses (0 hosts up) scanned in 0.03 seconds
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

» Virtual patching it

```
sp.filename("system").param("command").value_r("[^a-z0-9:/.]").drop();
```

» 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_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("#");
```

» Free vulnerabilities

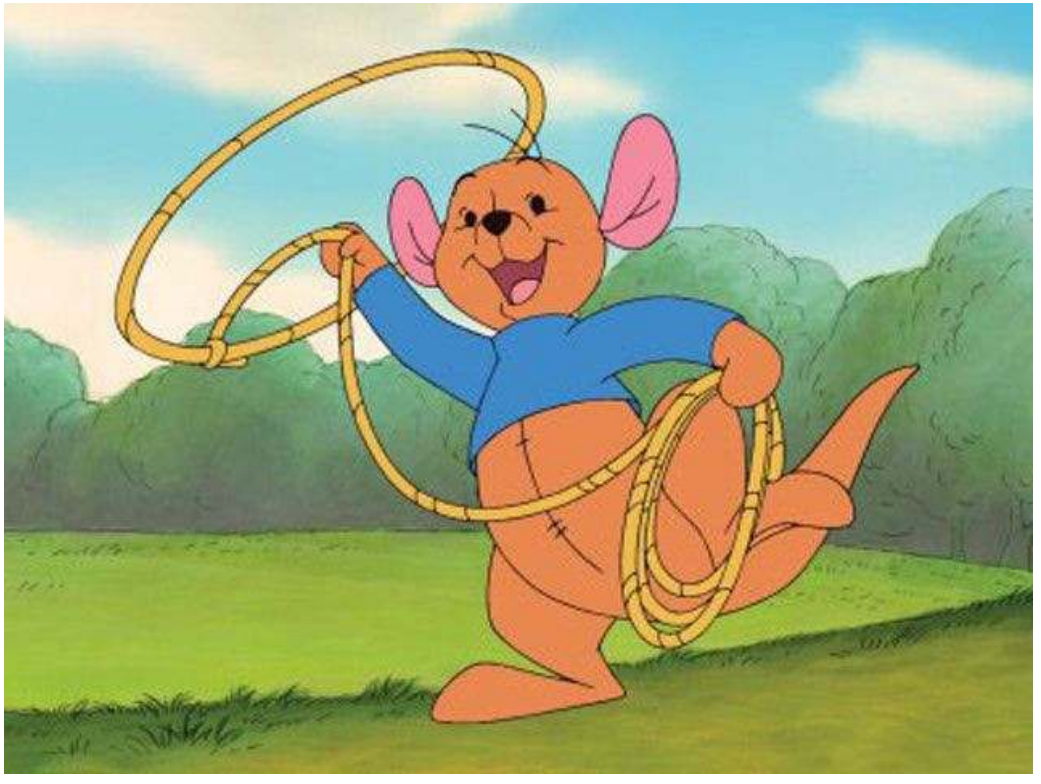


Fig 16. The security team getting ready to capture new bugs

» 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("[;`&|\n]").dump().allow();
```

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

» PERFORMANCE IMPACT?

» Performance impact

Fig 17. The management arguing with the security team about how security slows down the website

» Performance impact

Fig 17. A typical PHP application running at full speed

» WHAT'S LEFT TO DO

- Improving our dereferencing system, so you can hook anything
- Finding and fixing bugs
- Killing more bug-classes, like sloppy-comparisons and SQLI[1]
- Party party party

[1] We're working on it ;)

» WHERE CAN YOU GET THIS WONDER?

- 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

» CUSTOMERS

Fig 18. Customers lining up on the website without noticing that it's a PHP trashfire.

» SECURITY TEAM

Fig 19. The security team is now friend with your PHP application.

» DEVELOPERS

Fig 20. Developers are writing code as usual, but it's ok.

» 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*`

» 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 more than ***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
- Folks that ~~called us names~~ gave us constructive feedback

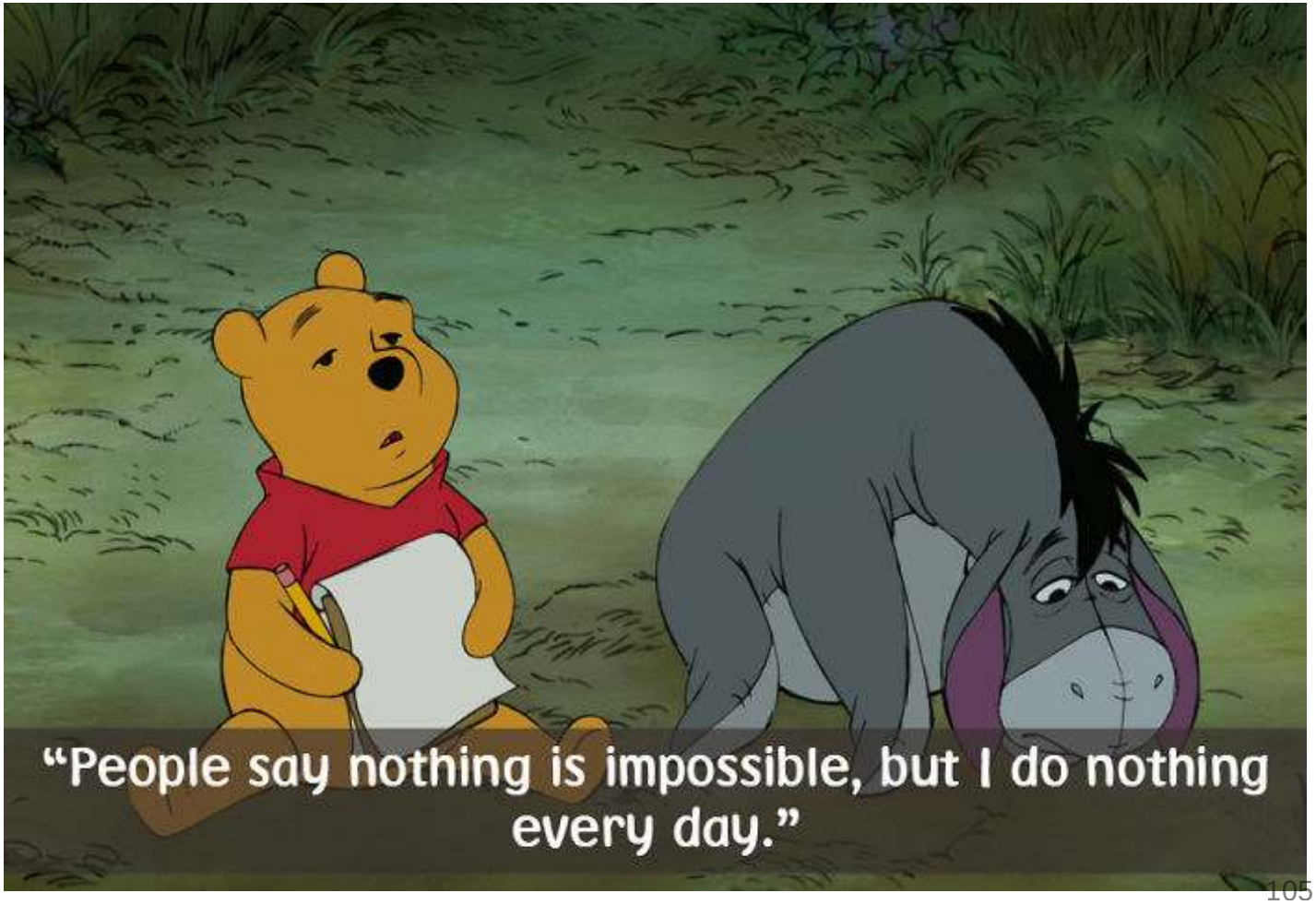
» GET IT!

- https://github.com/nbs-system/snuffleupagus
- https://snuffleupagus.readthedocs.io
- Come talk to us!

» BONUS ONE

» BONUS TWO

» BONUS THREE



» BONUS FOUR