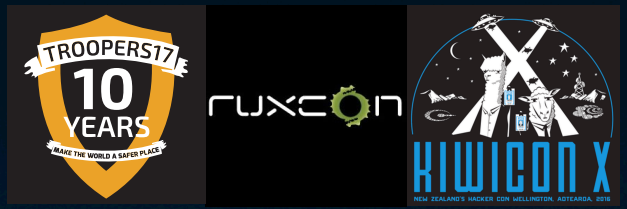
PHP Internals: Exploit Dev Edition

EMMANUEL LAW



aura
INFORMATION SECURITY

POWERED
BY KORDIA

Background

- Principal Security Consultant @ AURA, NZ
- Pentesting for living
- @libnex
- Found some PHP bugs

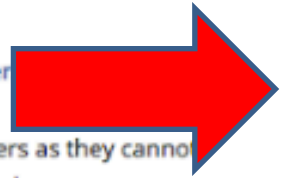
Minimum *Demonstrate the presence of a security bug with probable remote exploitation potential.*	$500

The project maintainers have final decision on which issues constitute security vulnerabilities. Only issues that are tagged as `Type: Security` by a project maintainer will be consider for bounty eligibility. The Panel will respect their decision, and we ask that you do as well.

It's important to keep in mind that not all submissions will qualify for a bounty, and that the decision to award a bounty is entirely at the discretion of the Panel.

Submission Process

- Disclose a previously unknown security vulnerability **directly to the** project maintainer
- Follow the disclosure process established by the project maintainers.
- Clearly demonstrate the security vulnerability. Respect the time of the project volunteers as they cannot invest significant effort into incomplete reports. Low-quality reports may be disqualified.
- Once a public security advisory has been issued, please submit a report here. You **must not** send us the details of the vulnerability until it has been validated, accepted, and publicly disclosed by the project maintainers.



Hackers thanked (32)

ryat
Reputation: 346

fms
Reputation: 208

l4w
Reputation: 156


libnex
Reputation: 141


haquaman
Reputation: 119

All Hackers

Agenda

- FUZZING PHP
  - Attacking the Engine
  - Strategies
  - Results
- EXPLOITING PHP INTERNALS
  - Heap Management
  - Exploit
  - Demo

What are we fuzzing?

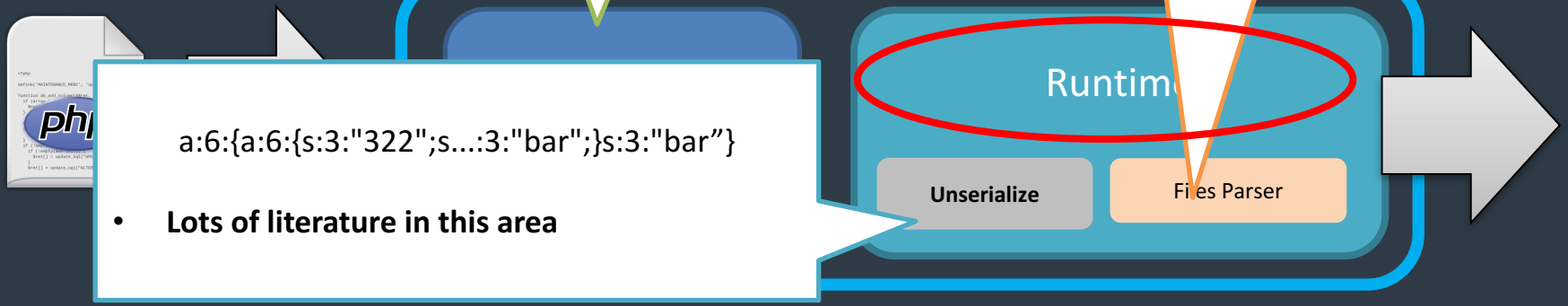
- Attack Surfac

<?php $a[]/=$a=a? >

<?php [[][]] ?>

Generic File Fuzzers:

- AFL
- HongFuzz
- Etc etc



a:6:{a:6:{s:3:"322";s...:3:"bar";}s:3:"bar”}

- **Lots of literature in this area**

Fuzzing Runtime

- Targeting runtime Functions, Classes, Methods, method etc.....
- Produces a mix of both local and remote vulnerabilities
- **time_sleep_until ( )** vs **mysql_escape_string( )**

Why not AFL it?

ARMY OF ONE.

- 2009 Desktop
- Core i7-960, 3.2GHz
- 8GB

Why not just AFL ?

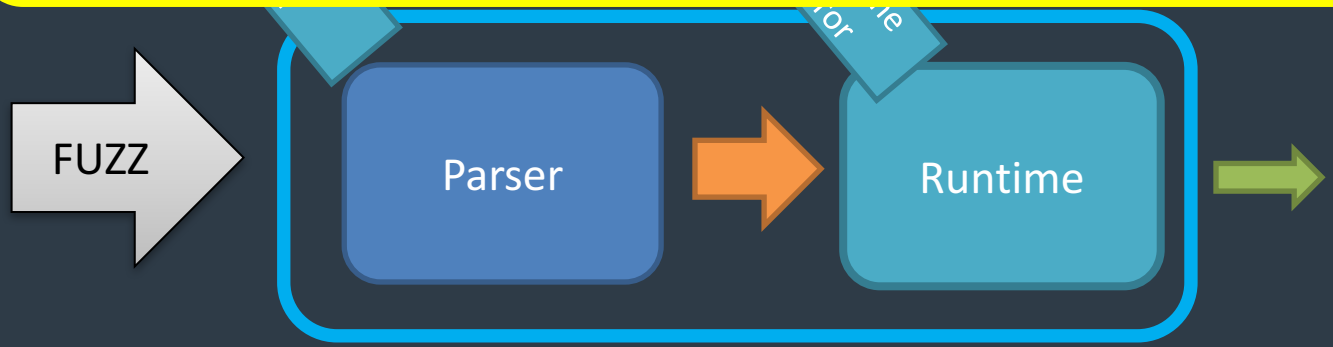
- Army of One
- Different fuzzers find different bugs
- AFL is not very good for language interpreters

PHP is Finicky

- Strict Syntax
- Strict Arguments (Relatively)
  - Number of arguments

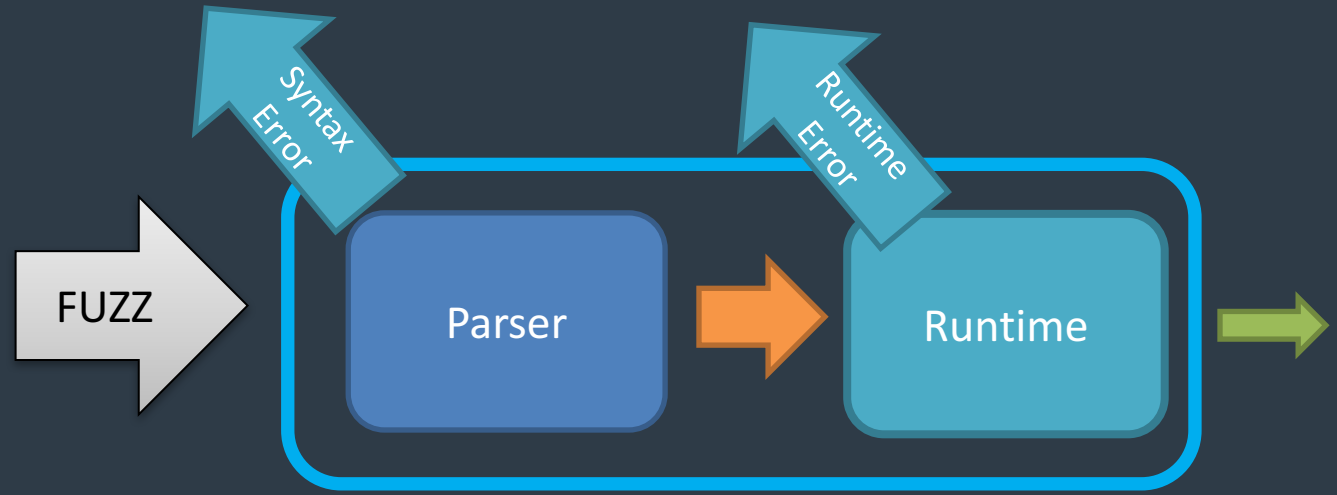
```
> php -r strcmp('a');

Warning: strcmp() expects exactly 2 parameters, 1 given in
```



PHP is Finicky

- Syntax Checks
- Relatively Strict Arguments
  - Number of arguments
  - Types



```
bool imagecopymerge ( resource $dst_im , resource $src_im
 , int $dst_x , int $dst_y , int $src_x, int $src_y , int
 $src_w , int $src_h , int $pct )
```

PHP Primitive Types: String, Integer, Float, Boolean, Array, Object ,NULL, Resource

Success rate: 1 in 134,217,728

Existing PHP Fuzzers

- Minerva (2005) & Phuzzy
  - Very basic fuction fuzzer
- **LangFuzz (2012)**
  - Good Ideas
  - Close source
- Malamute (2014) (Unit Test + Radamsa)
  - Not syntax aware

Introducing Phzzer

Phzzer

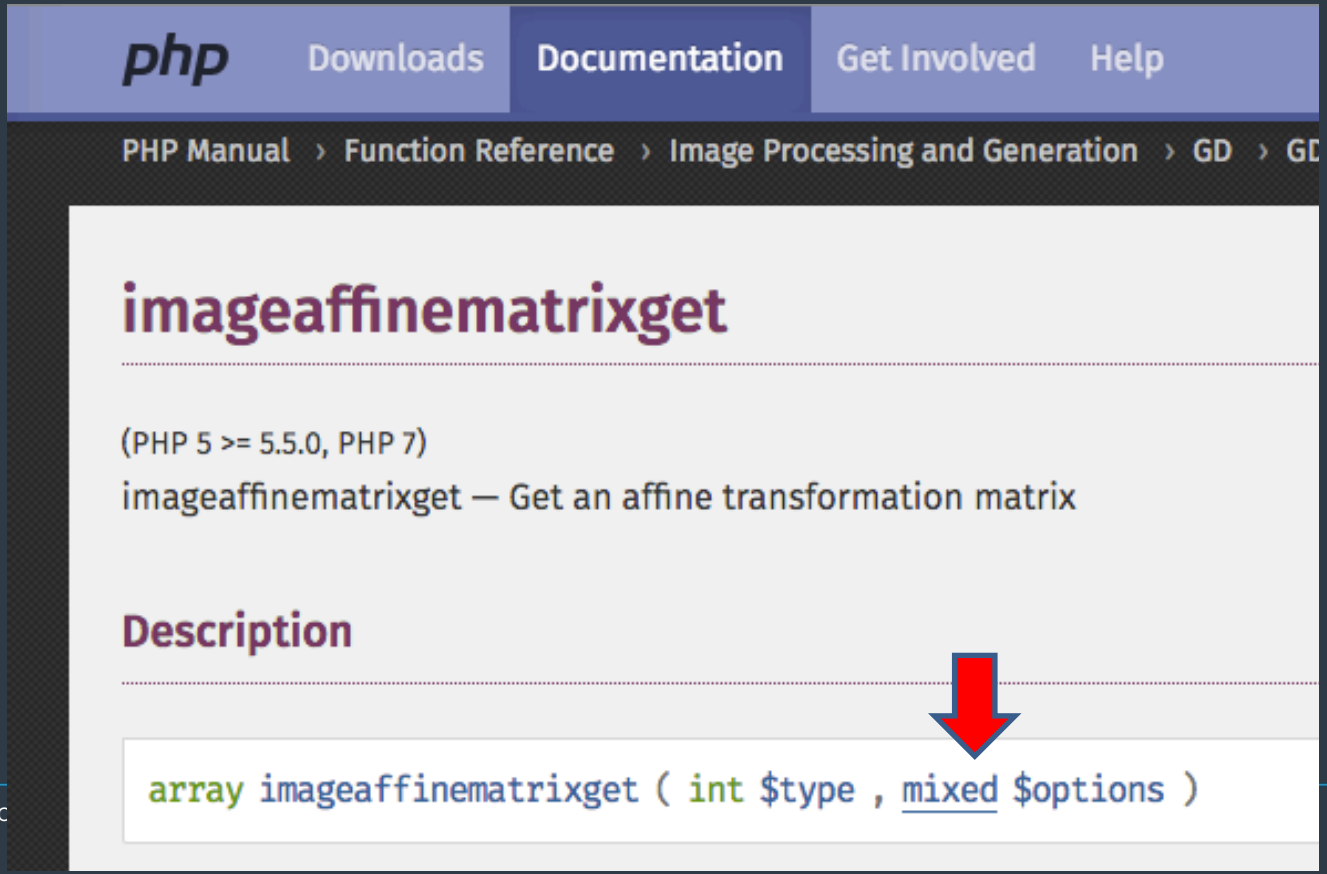
- Grammar aware
  - Generates scripts that are syntactically correct
- Contextually aware
  - Arguments, Variable Instantiation etc.
- “Drop & Fuzz” philosophy

Phzzer: Grammar

- Loose interpretation of PHP grammar
- Not perfect but good enough
- Augmented by regression test cases

Contextual Awareness::
Requires Runtime Knowledge

- Solution 1: PHP Online Doc
  - http://php.net/manual/



Building a Knowledge DB

- Solution 1: PHP Online Doc
  - http://php.net/manual/

- Cons:
  - Webscraper? Really?
  - Standard functions/classess + a selection of curated Extensions
    - What if my php is not compiled with those extensions?
    - What if I want to fuzz an obscure/custom extension?
  - “Mixed Types”

Building a Knowledge DB

- Solution 2: Reflection
  - get_defined_functions()
  - Run `php --rf <function>’

```
> php --rf substr_count
Function [ <internal:standard> function substr_count ] {

  - Parameters [4] {
    Parameter #0 [ <required> $haystack ]
    Parameter #1 [ <required> $needle ]
    Parameter #2 [ <optional> $offset ]
    Parameter #3 [ <optional> $length ]
  }
}
```


```
php --rc datetime
Class [ <internal:date> class DateTime implements DateTimeInterface ] {

  - Constants [11] {

    Constant [ string COOKIE ] { l, d-M-Y H:i:s T }
    Constant [ string ISO8601 ] { Y-m-d\TH:i:sO }
.....
  - Static methods [3] {
    Method [ <internal:date> static public method __set_state ] {
    }

    Method [ <internal:date> static public method createFromFormat ] {

      - Parameters [3] {
        Parameter #0 [ <required> $format ]
```


Building a Knowledge DB

- Solution 2: Reflection @ Runtime
- Pros:
  - Only functions/classes which are present would be fuzzed
  - Able to handle Custom/uncommon extensions
- Cons:
  - Doesn’t tell you the argument type

Building a Knowledge Base

- Solution 3: Runtime Enumeration
  - Feed basic types into functions
  - Parse error message if any

- Pros:
  - Find out Argument types, including “Mixed”.

```
php -r "substr_count ( [ ],'a');”
Warning: substr_count() expects parameter 1 to be string, array given in ……
```


Reflection + Enumeration

- Works Surprisingly well
- Number of arguments + Types
- Object Fuzzing:
  - Enumerate Class Constructor
  - Instantiate Object
  - Enumerate Methods
- Dynamically generated at runtime => Only functions/classes that exist are fuzzed

Building a Knowledge Base

- Solution 4: Runtime Instrumentation
  - PHP internals: zend_parse_parameters() + sister functions
  - Frida Framework
  - Write Hooks in Javascript

Description

```
bool imagewbmp ( resource $image [, mixed $to [, int $foreground ]] )
```

Accepts 3 Param

```
zend_parse_parameters(ZEND_NUM_ARGS(), "r|z/!ll", &imgind, &to_zval, &quality, &basefilter) == FAILURE)
```

Resource

Optional

Mixed

Int

Int

4th undocumented Param

Procedural style

```
IntlCalendar intlcal_from_date_time ( mixed $dateTime ), Locale
```

Building a Knowledge Base

- Solution 4: Run time Instrumentation

- Pros:
  - More accurate arg types
  - Discover Hidden Parameters

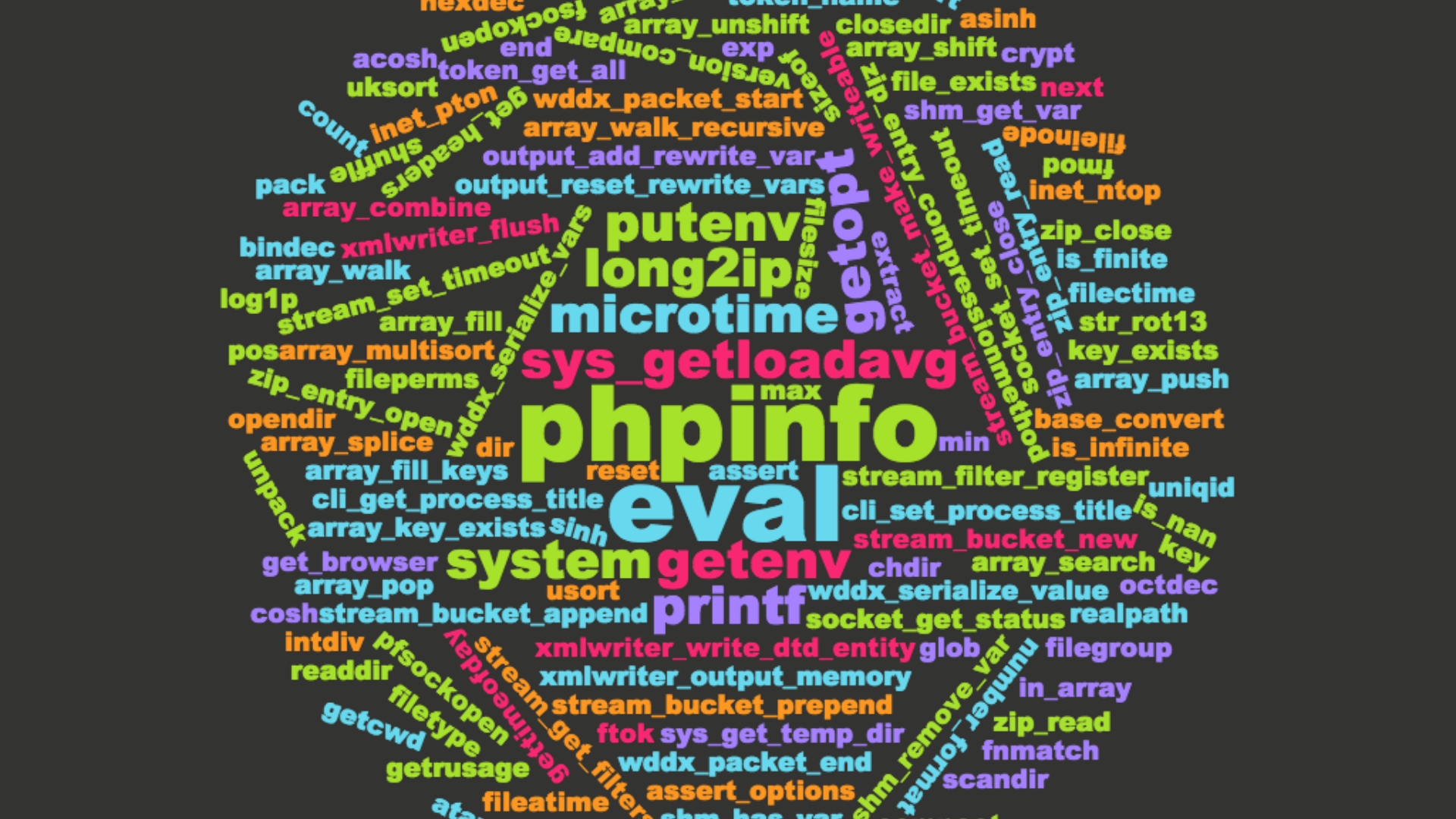
New in PHP 7

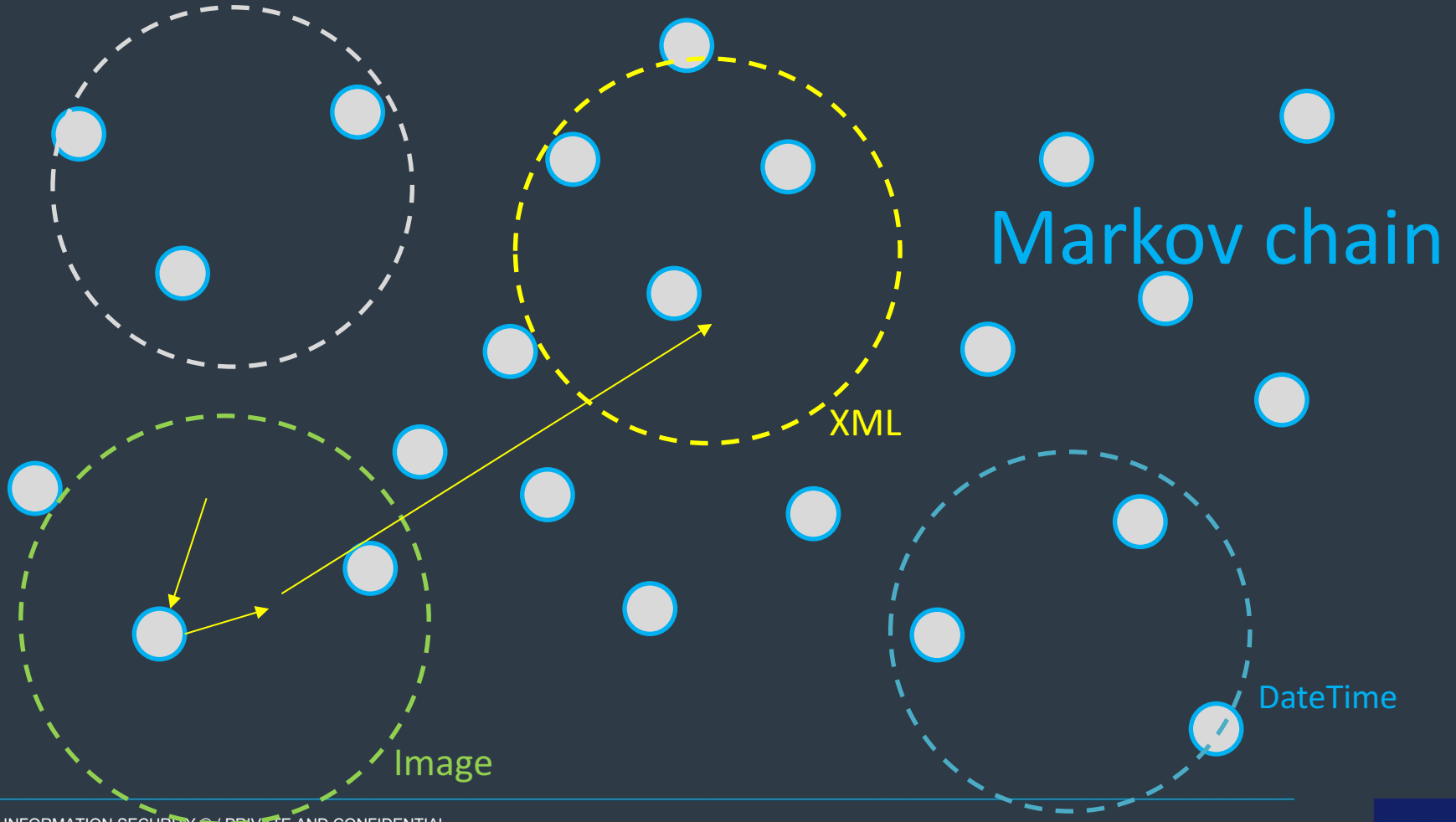
- Zend Fast Parameter Parsing:
  - Implemented as macros: FAST_ZPP
  - Affects a small subset of core functions
  - Disable @ Compile time*
    - Not possible PHP >7.0.11

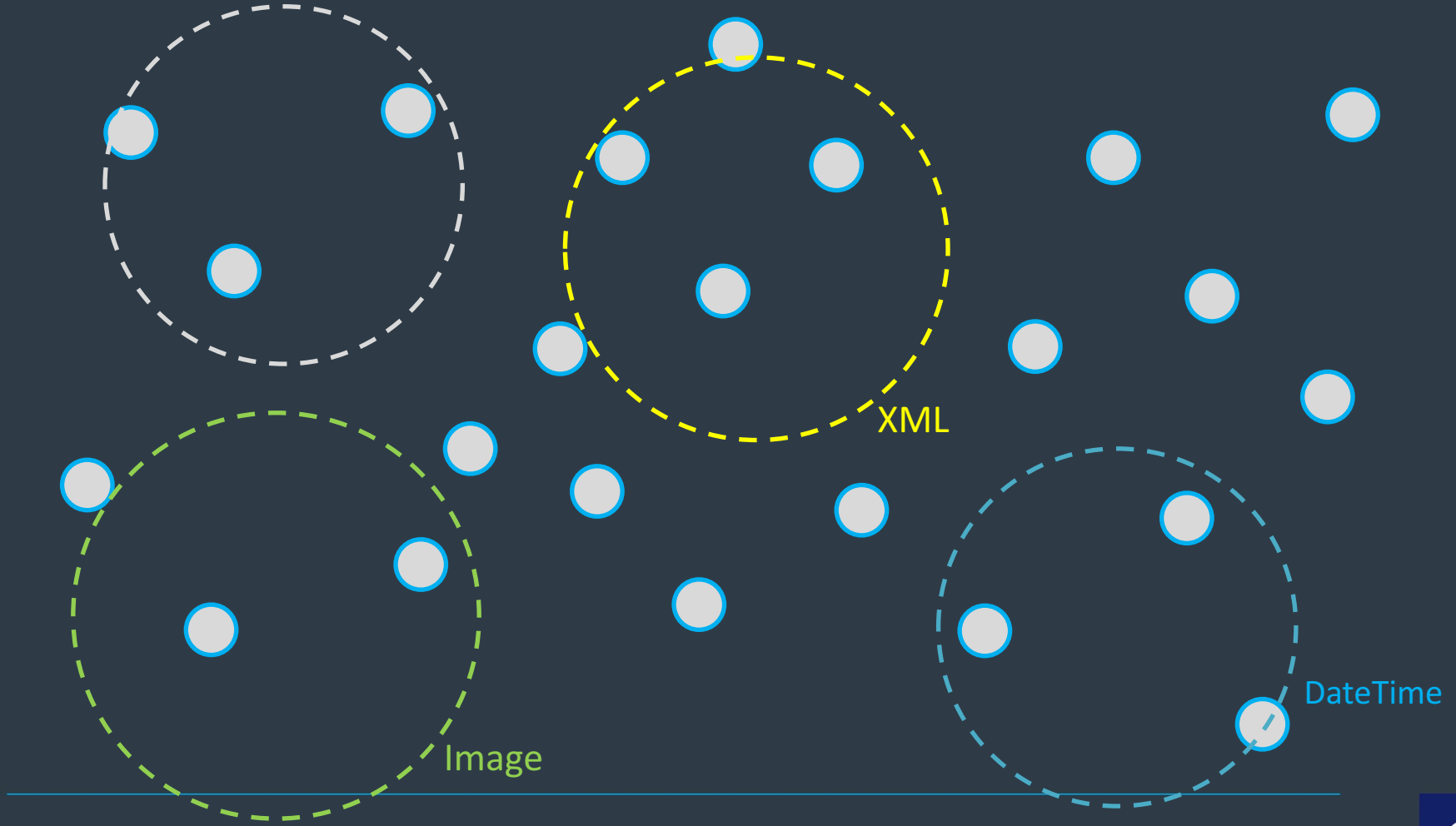
*Patch https://gist.github.com/libnex/af84816a3b9632a474f8f6e263b9d711

Some Fuzzing Strategies….

Function Clustering


phpinfo
eval
putenv
long2ip
microtime
getopt
sys_getloadavg
system
getenv
printf
array_unshift
closedir
asinh
array_shift
crypt
file_exists
next
shm_get_var
filetype
fileinode
fmod
inet_ntop
zip_close
is_finite
filectime
str_rot13
key_exists
array_push
base_convert
is_infinite
uniqid
is_nan
key
octdec
realpath
filegroup
in_array
zip_read
fnmatch
scandir
assert_options
fileatime
wddx_packet_end
sys_get_temp_dir
ftok
stream_bucket_prepend
xmlwriter_output_memory
xmlwriter_write_dtd_entity
glob
wddx_serialize_value
socket_get_status
chdir
array_search
stream_bucket_new
cli_set_process_title
stream_filter_register
min
max
reset
assert
dir
array_splice
opendir
zip_entry_open
unpack
array_fill_keys
cli_get_process_title
array_key_exists
sinh
get_browser
array_pop
usort
coshstream_bucket_append
intdiv
readdir
pfsockopen
getcwd
getrusage
stream_get_filters
gettimeofday
array_walk_recursive
output_add_rewrite_var
output_reset_rewrite_vars
wddx_packet_start
version_compare
exp
end
acosh
token_get_all
uksort
inet_pton
get_headers
shuffle
count
pack
array_combine
bindec
xmlwriter_flush
array_walk
stream_set_timeout
log1p
array_fill
posarray_multisort
fileperms
wddx_serialize_vars
filesize
extract
stream_bucket_make_writeable
zip_entry_compressionmethod
stream_socket_set_timeout
zip_entry_close
zip_entry_read
zip_entry_open
shm_remove_var
number_format


Markov chain
XML
Image
DateTime

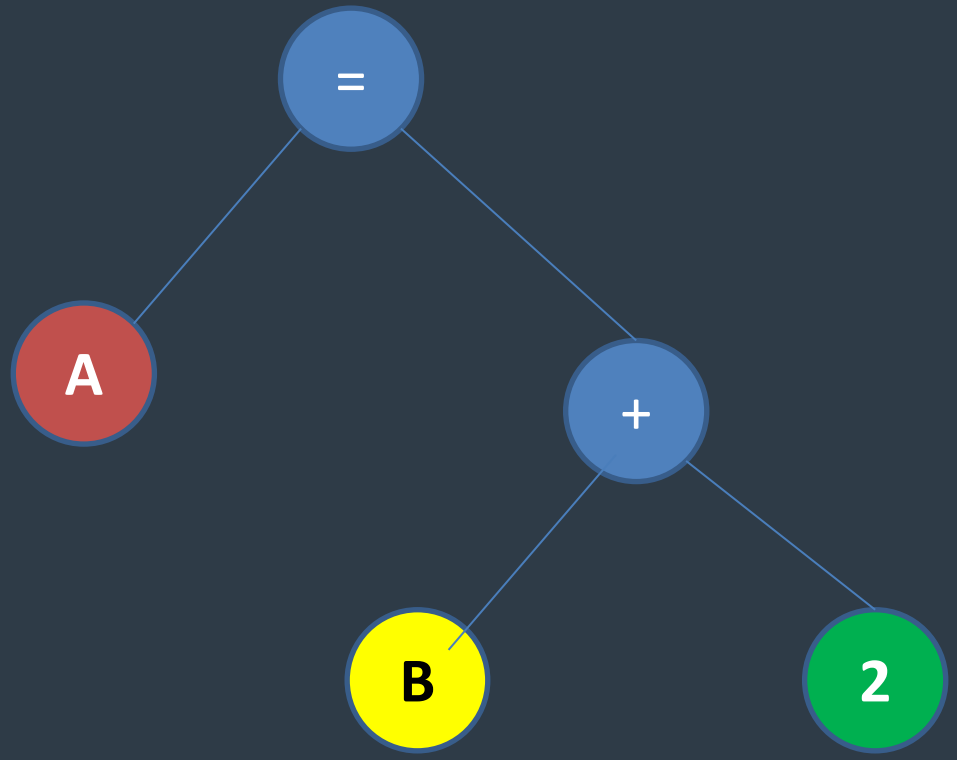

XML
DateTime
Image

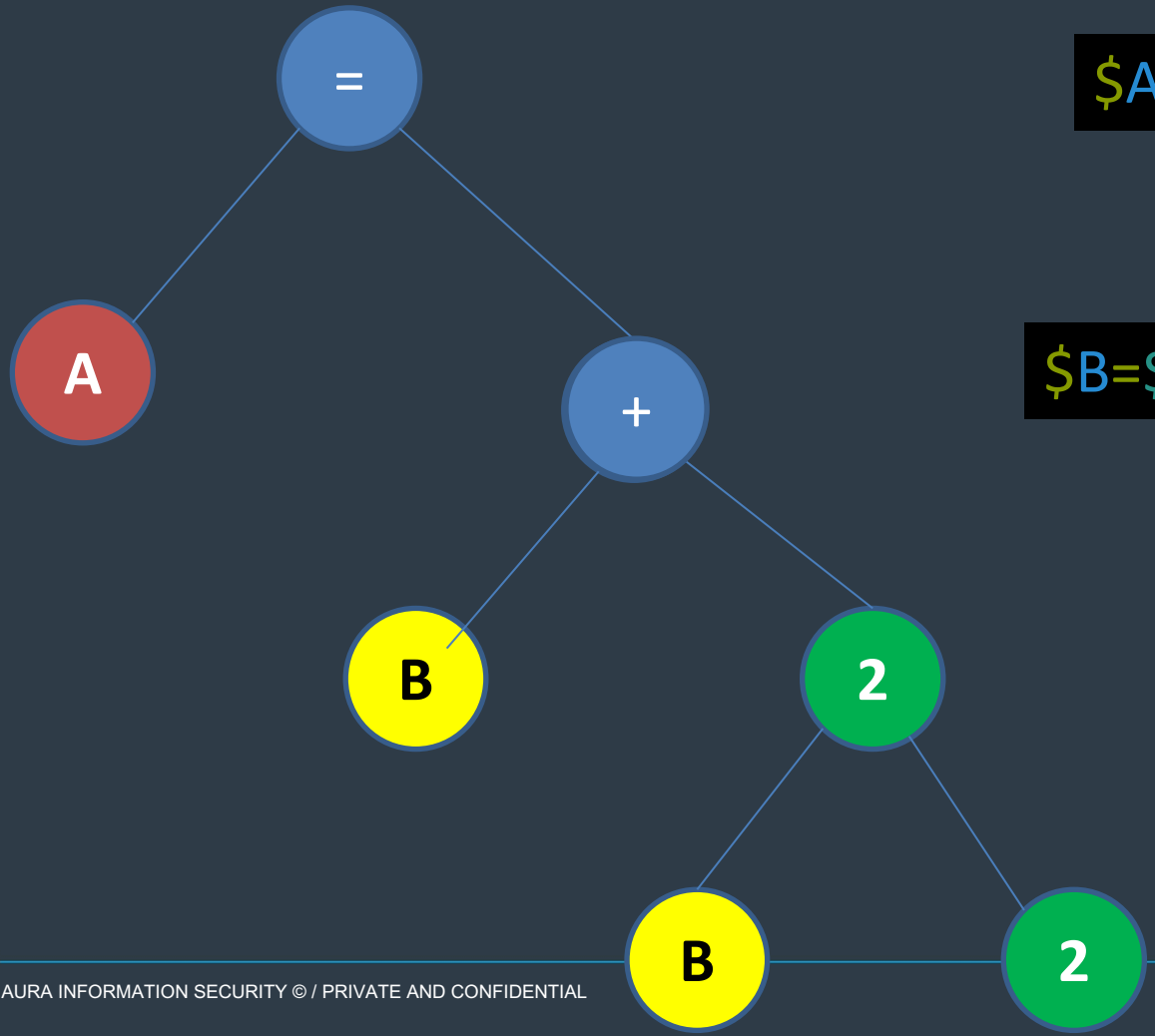
```php
<?php

date_timezone_get(date_sub(date_sunrise(ctype_print( $var ))));
```


Node Mutation

```
$A=$B+2;
```

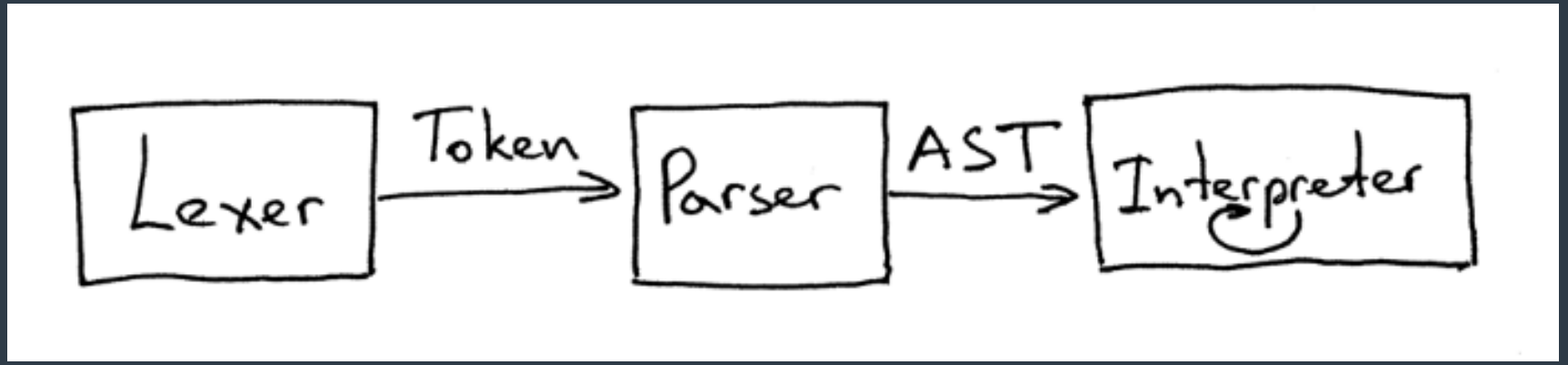





```
$A=$B+2;
```

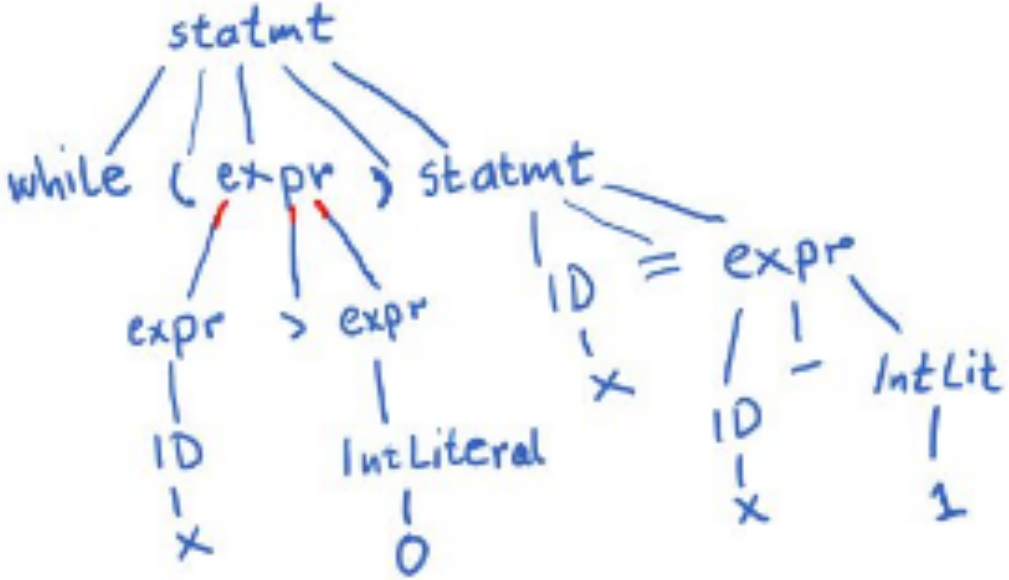
```
$B=$A+($B+2);
```

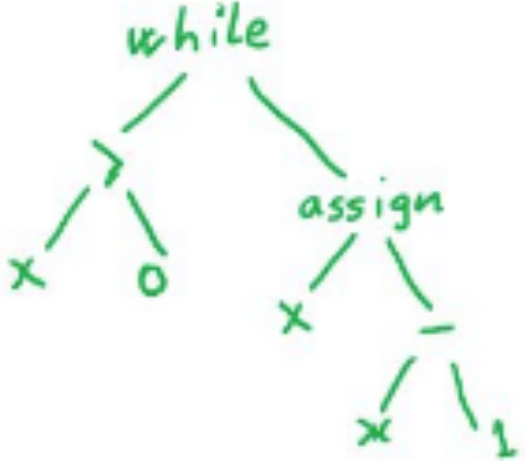
- LangFuzz does something similar:
  - Operates on tokens using a Lexer
  - Phzzer operates on Abstract Syntax Tree (AST)



Parse Tree vs Abstract Syntax Tree (AST)

while (x > 0) x = x - 1





Node Mutation

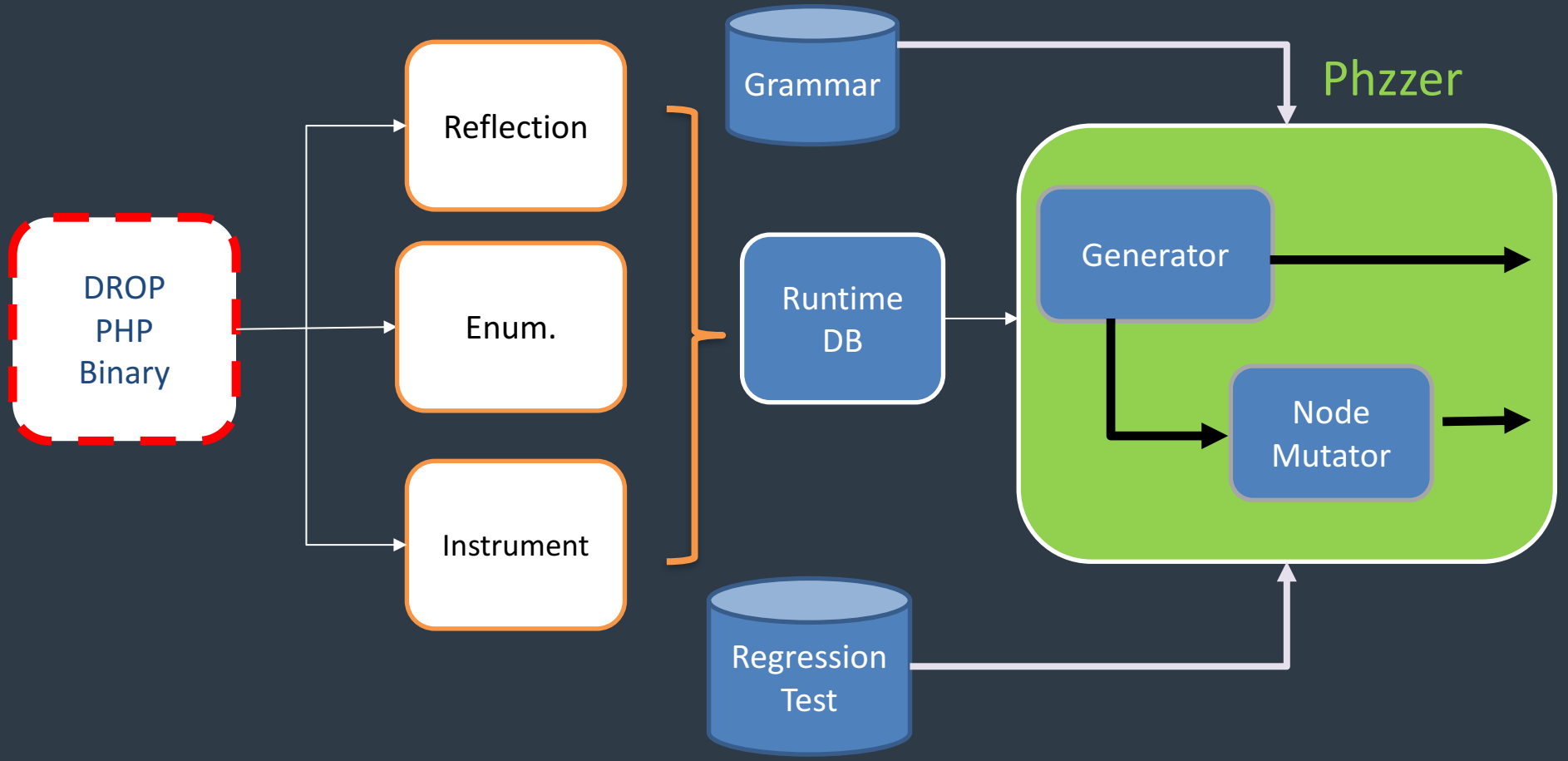
- What AST?!
- Newly implemented in PHP 7

Harnessing Regression Test

bug29944.phpt

```
--TEST--
Bug #29944 (function defined in switch crashes PHP)
--FILE--
<?PHP
$a = 1;
$b = "1";
switch ($a) {
    case 1:
        function foo($bar) {
            if (preg_match('/\d/', $bar)) return true;
            return false;
        }
        echo foo($b);
}
?>

===DONE===
--EXPECT--
1
===DONE===
```


Grammar
Phzzer
Reflection
DROP
PHP
Binary
Enum.
Runtime
DB
Generator
Node
Mutator
Instrument
Regression
Test

Other Fuzzing “Strategy”

- Disable Zend Memory Manager
  - USE_ZEND_ALLOC=0
  - Heap Buffer Overflow in php_escape_shell_arg( ) *
  - Falls back to malloc() rather than php’s emalloc()
  - Good results
- Choosing the right python
  - Phzzer is computationally intensive
  - Pypy FTW

*CVE-2016-1904

- PHP 7.0.0
  - ~120 Uniq crashes
- PHP 7.0.16
  - 59 Uniq Crashes
    - Stack BOF x 8
    - Heap BOF x 12
    - Use After Frees x 6
    - 6 x Misc. WildWrite, double-free, badfree etc
    - Unknown x 27

Demo

(Timecheck)

Exploitation: PHP Internals

PHP “User Land”

Zend Internal

Zend Engine

Disable System(), eval etc

Double Free Vulnerability

- Class **SplStack()***
- Local Exploitation
- Trigger via:

```php
<?php

$var_1=new SplStack();
$var_1->offsetSet(-1, stdObject() ); //stdObject will be double-freed
```

*CVE-2016-3132

Double Free Vulnerability

```
ZEND_FUNCTION(SplStack::Offset) {
……

            if (index < 0 || index >= intern->llist->count) {
             zval_ptr_dtor(value);
             zend_throw_exception( "Offset invalid or out of range", 0);
            …..
            }
…..

zend_vm_stack_free_args(call);

}
```


Double Free
Unlink
Double Link List
Write
What
Where
Profit !

ASLR ☹

Zend Memory Manager

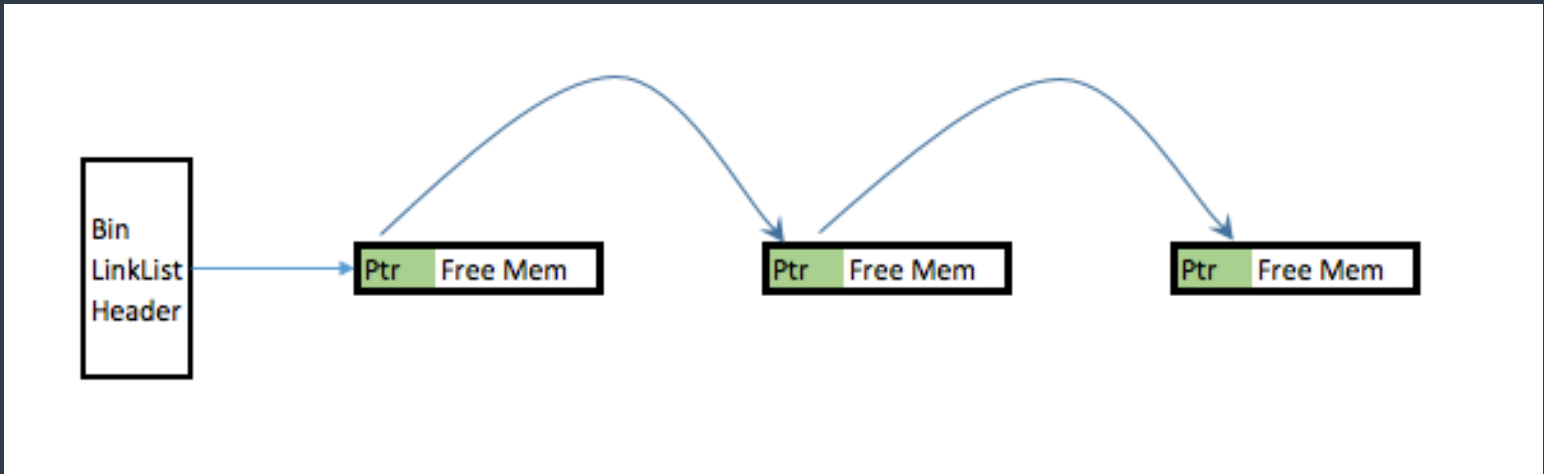
- Manage PHP’s memory allocation
  - emalloc, ,erealloc, efree etc..
- 3 Kinds of Allocator
  - Small Heap Allocator (<3072 bytes)
  - Large Heap Allocator (< 2 mb)
  - Huge Heap Allocator (> 2mb)

Small Heap Allocator

- Memory chunks categorized into “bins”:
  - Bin #1 : contains chunk sizes from 1 - 8 bytes
  - Bin #2: contains chunk sizes from 9 - 16 bytes
  - Bin #3: contains chunk sizes from 17 - 24 bytes
  - Bin #YouGetTheIdea....

Small Heap Allocator

- Freeing a memory chunk puts it back into the bin
- 8 bytes metadata containing Forward_pointer to next chunk
- Single link list



Step 0: Exploitation Feasibility

- Triggering the vulnerable **SplStack()** causes run time error

```
Fatal error: Uncaught OutOfRangeException: Offset invalid or out of range
```

- Try …catch wouldn’t work
- Final chance:

```
set_exception_handler ( Callback)
```


Step 1: Battle Plan

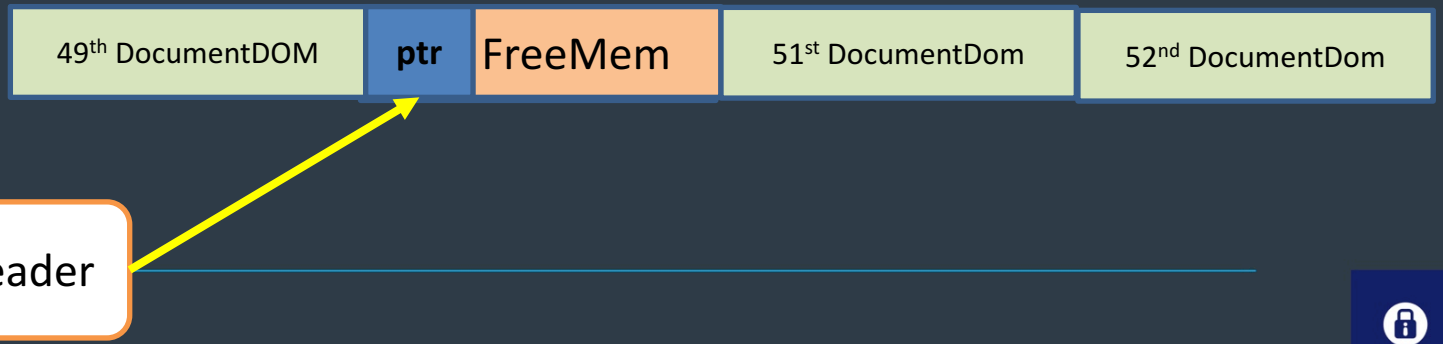
- What do we want to double free?
- DOMDocument Object
  - Size of struct is small enough
  - Fits a Bin # that is not commonly used within PHP internals
  - Has a struct member in a particular offset

Step 2: Massaging the Heap

- Allocate contiguous chunks + poke a gap

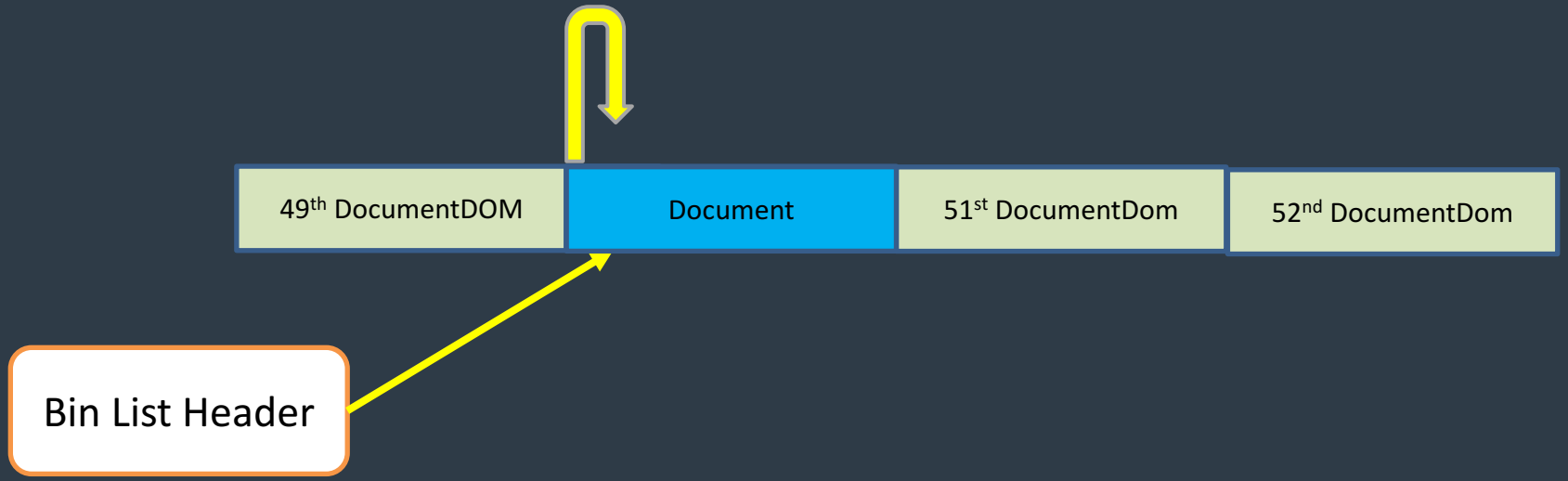
```
for ($x=0;$x<100;$x++){
            $z[$x]=new DocumentDom);
}

unset($z[50]);
```



Step 3:Trigger the Vul

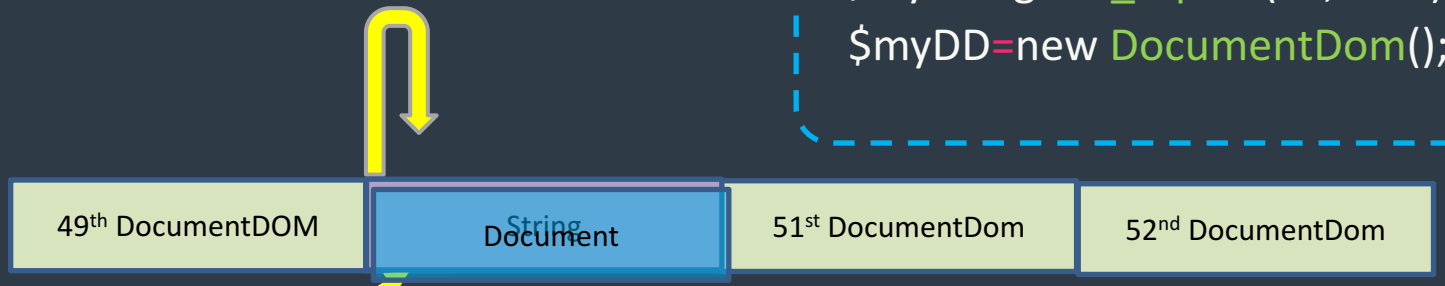
- new SplStack() -> offsetSet(-1,new DocumentDom);
- Triggers 1 Allocation
- 2 Deallocation in same memory spot



Step 4: Initial Exploitation

- Abuse Heap abnormally
- PHP thinks there 2 free chunk
- Allocating 2 chunks will now occupy same position

```
$mystring=str_repeat('C',0x20);
$myDD=new DocumentDom();
```



Bin List Header

Step 4: Initial Exploitation

- String allocated size is 0x20
- Len field overwritten

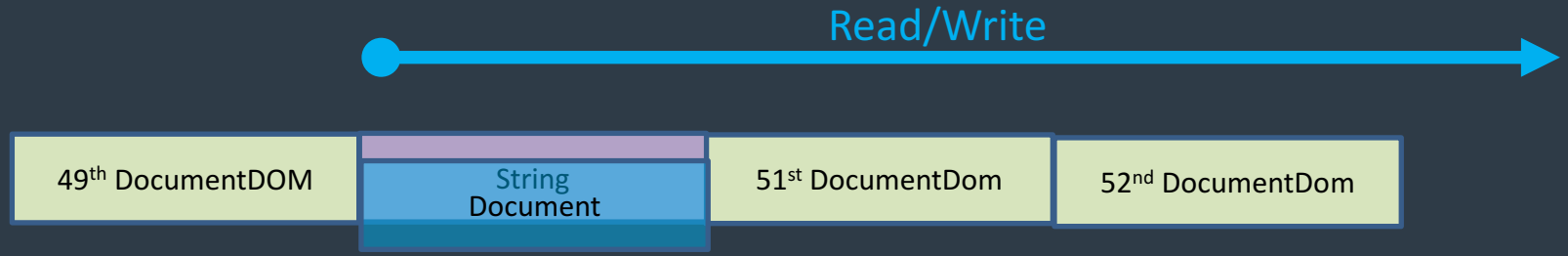
```
struct _zend_string {
        zend_refcounted_h gc;
        zend_ulong        h;
        size_t            len;
        char              val[1];
};
```

overwrite

```
typedef struct _dom_object {
        void *ptr;
        php_libxml_ref_obj *document;
        HashTable *prop_handler;
        zend_object std;
} dom_object;
```

Step 4: Initial Exploitation

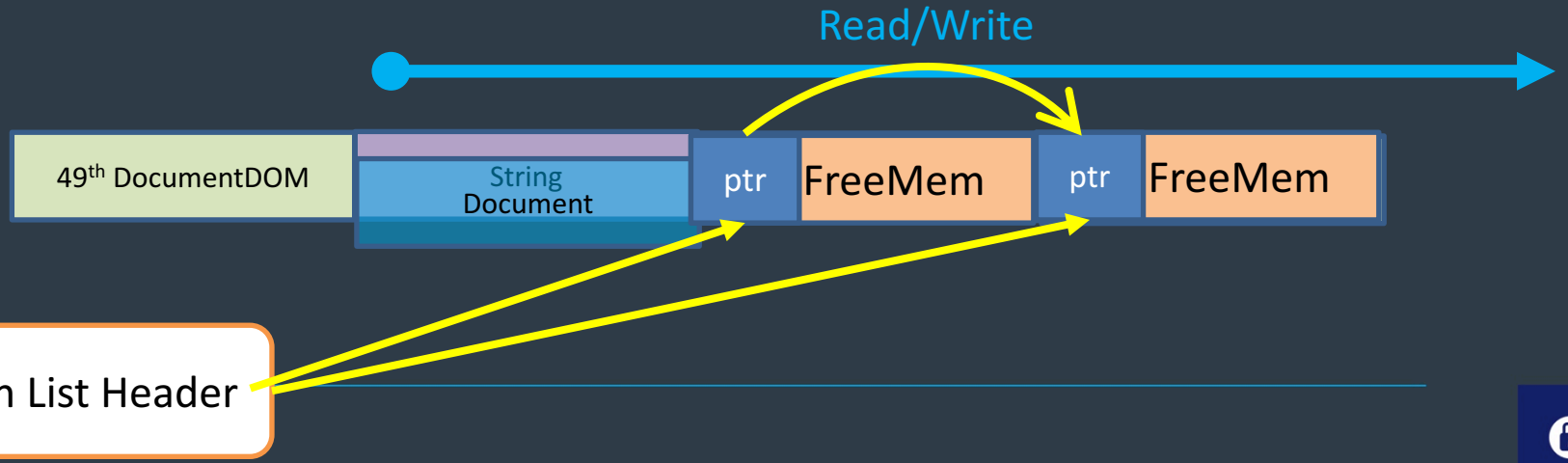
- String allocated size is 0x20
- len overwritten by DomDocument.prop_handler*
- len >> 0x20
- Ability to read very large amount of bytes!!



Step 5: Where Am I

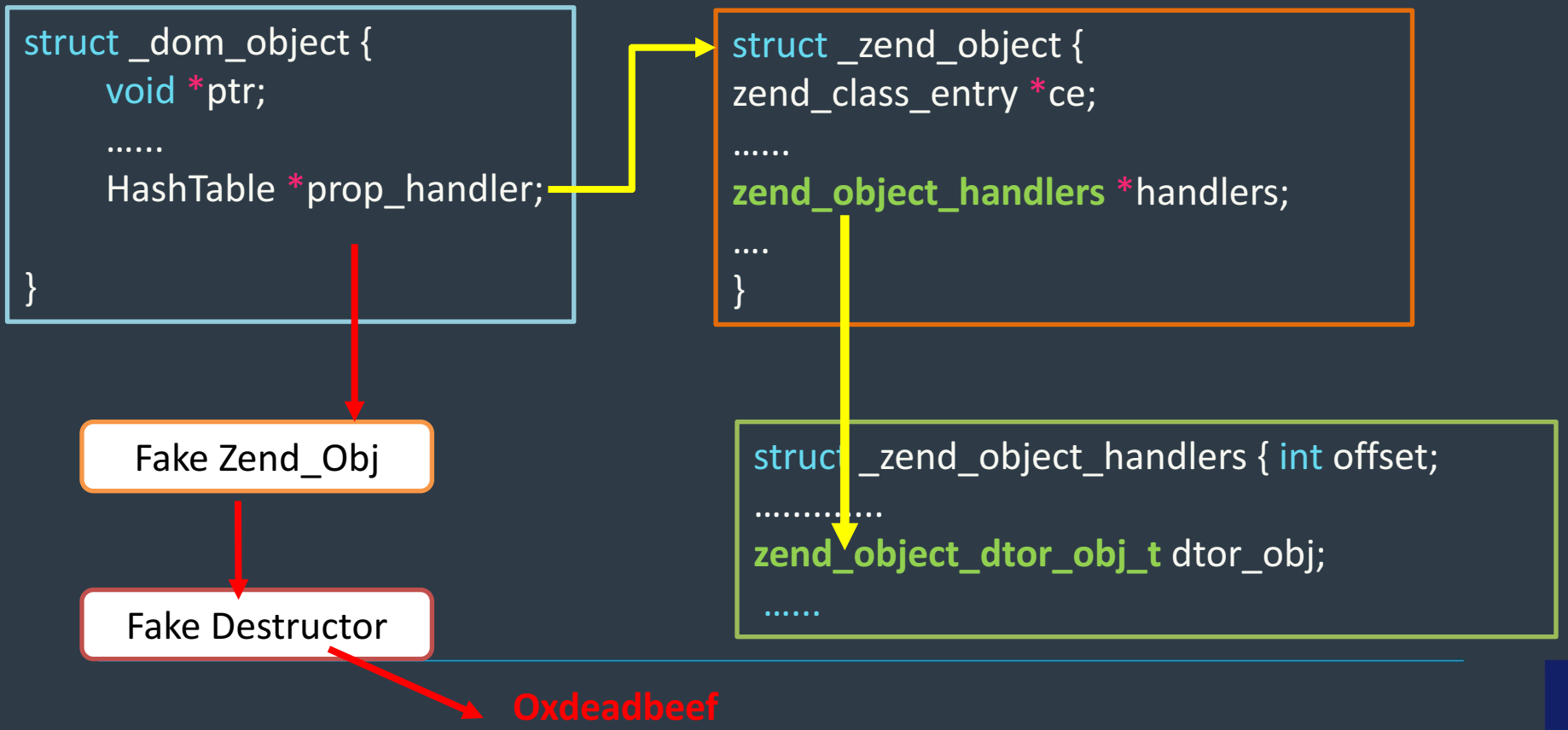
- Unknown Location in memory

```
Unset(DocumentDom[52]);
Unset(DocumentDom[51]);
```



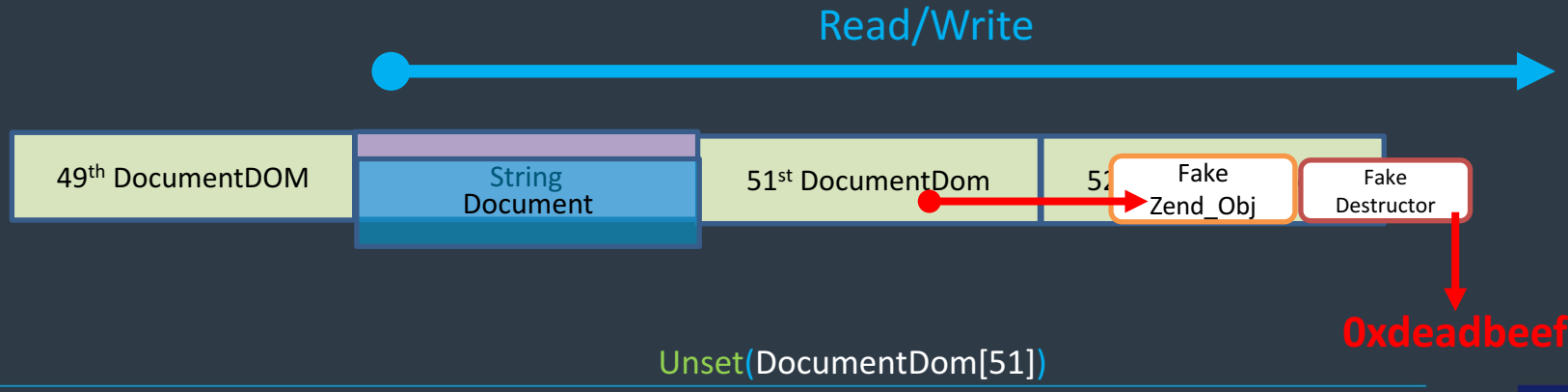
2 Methods of Exploitation

Method1: Destructor



Method1: Destructor

- Control $EIP – 0xdeadbeef



Method1: Destructor

ASLR bypass via leaking handlers & pointers

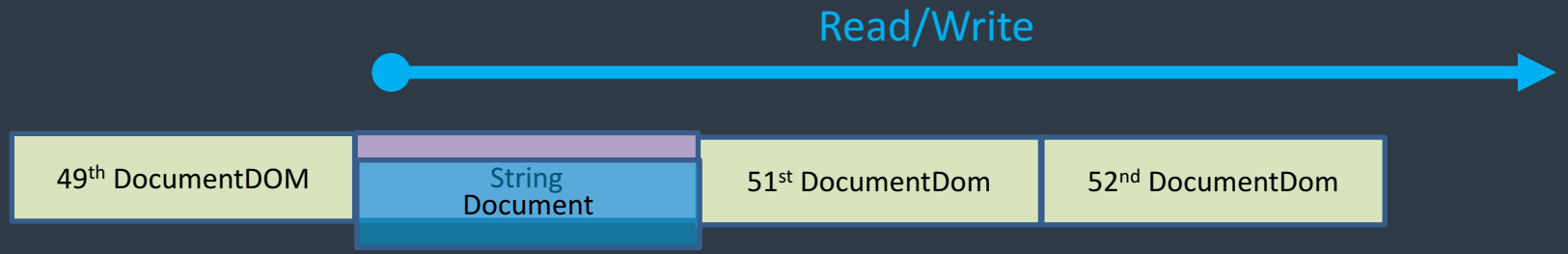
Not really portable: 101 PHP distros ☹

Method2: Re-enable disable_functions

- Stefan Esser 2009 Paper*
- Still works in php7
- Re-enable systems() etc..
- No shellcode, rop and messy stuff..
- Need a Read/Write-what-where primitive

* https://www.blackhat.com/presentations/bh-usa-09/ESSER/BHUSA09-Esser-PostExploitationPHP-PAPER.pdf

Current Situation



String Php5 vs PHp7

PHP5

```
union _zvalue_value {
..
        char *val;
        int len;
} str;
```

PHP7

```
struct _zend_string {
        .......
        size_t len;
        char val[1];
}
```

PHP 7:

- No more char* ptr
- String value stored as part of struc
- Struct Hack: Flexible array members*

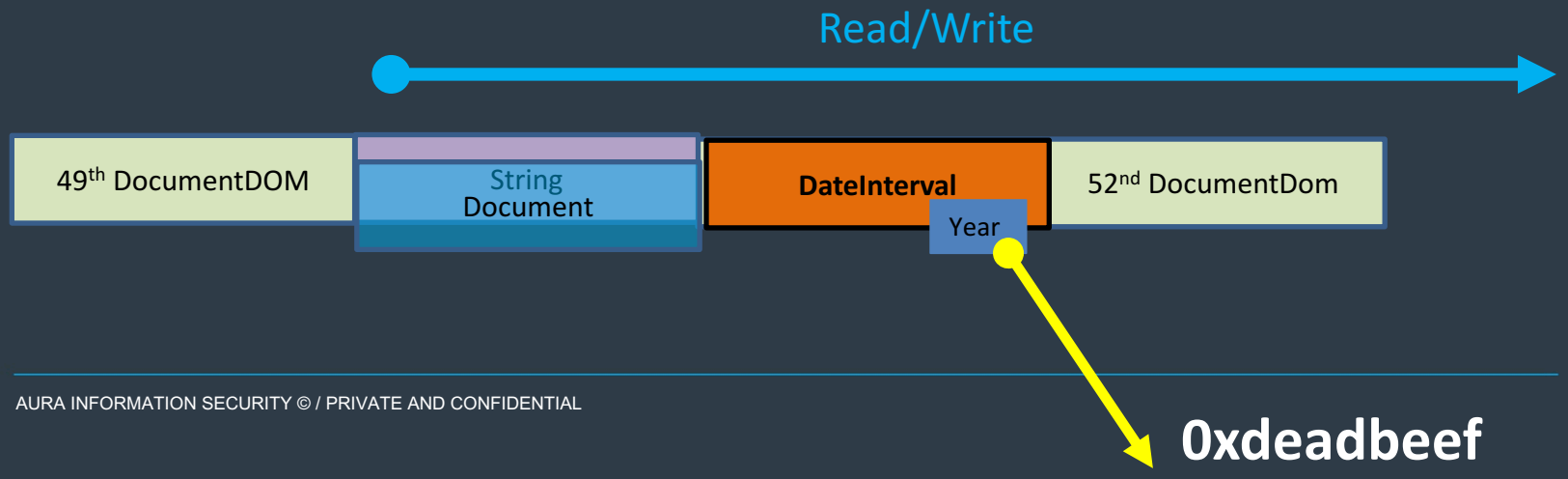
*https://nikic.github.io/2015/06/19/Internal-value-representation-in-PHP-7-part-2.html

```
struct _date_interval_obj {
...
relative_time *diff;
....
 }
```

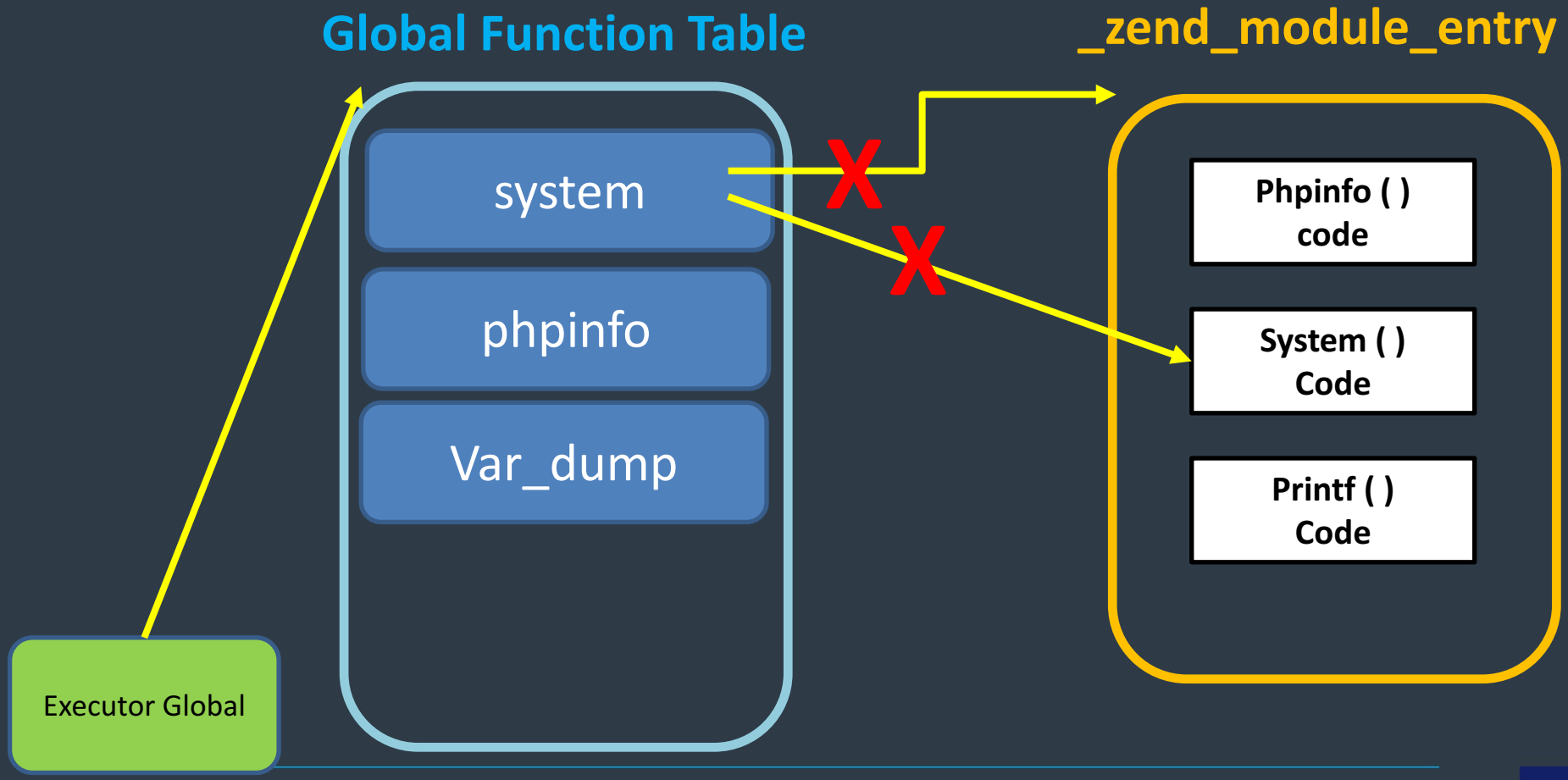
```
struct timelib_relative_time {
long long year;
long long month;
long long day;
 long long hour;
....
}
```

Simple DataTypes ☺

```
Unset(DocumentDom[51]);
$primitive=New DateInterval () ;
$mystring[0x42]=0xdeadbeef
echo $primitive->year;
```



How PHP Disable_Function works

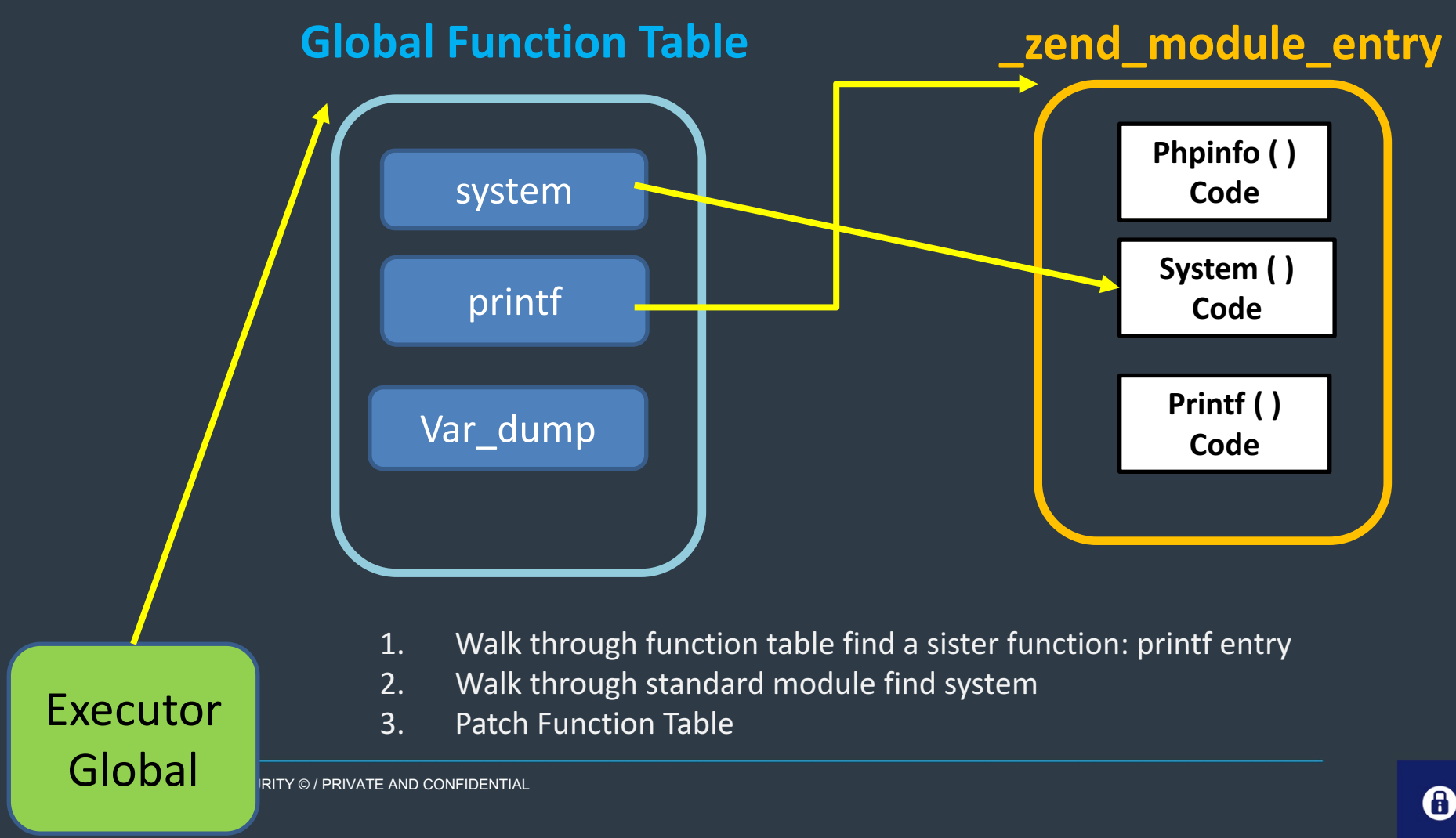

Global Function Table
_zend_module_entry
system
phpinfo
Var_dump
X
X
Phpinfo ( )
code
System ( )
Code
Printf ( )
Code
Executor Global

Finding Executor Global

```
struct _zend_executor_globals {
.....
int error_reporting;
.....
 HashTable *function_table;
}
```

Fixed offset

1. error_reporting(0x11223344)
2. Scan .BSS for 0x11223344
3. Get function_table address



1. Walk through function table find a sister function: printf entry
2. Walk through standard module find system
3. Patch Function Table

Demo

Questions?

@libnex