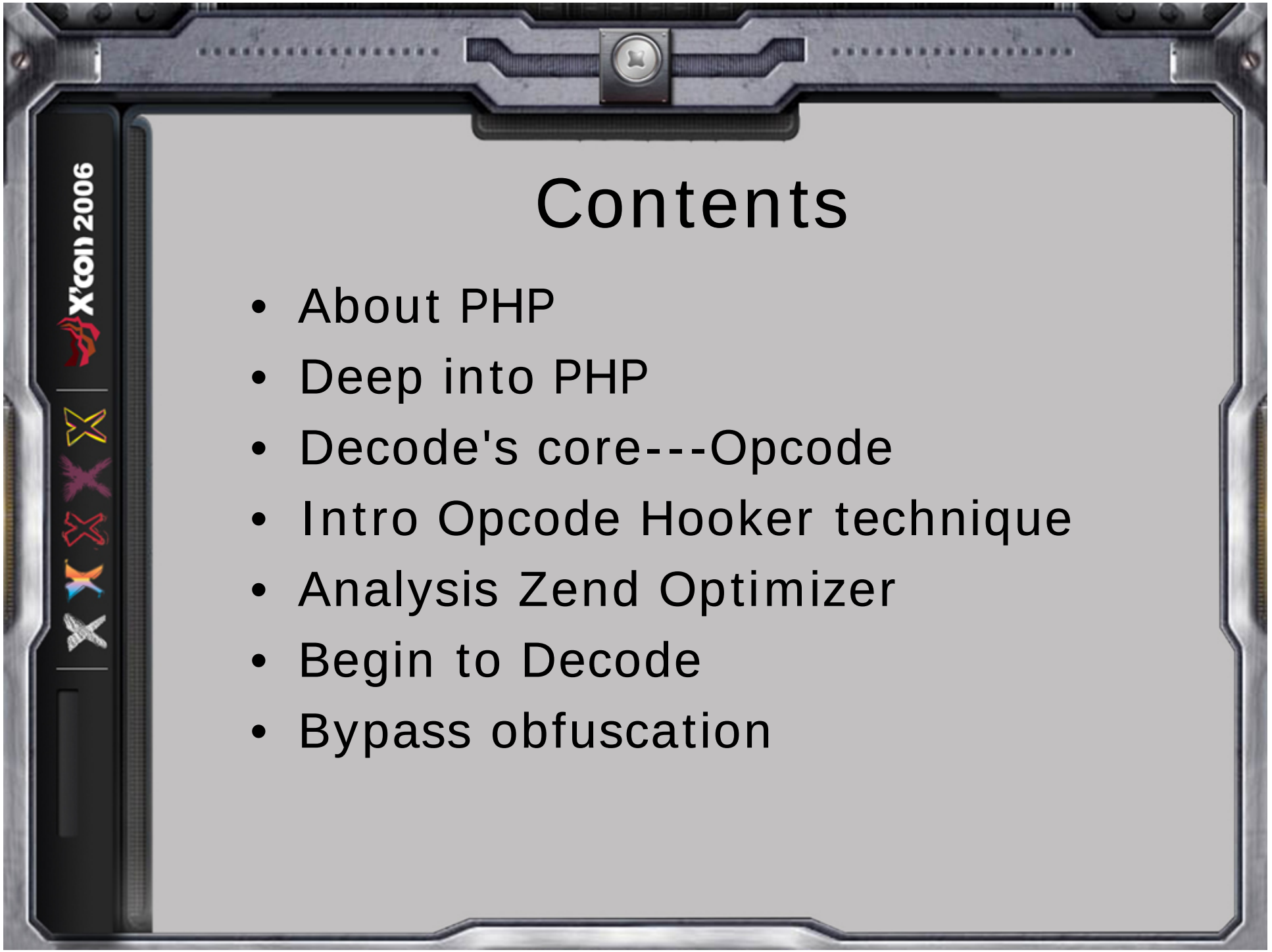

Decode Zend
Darkness / Airsupply
X'Con2006



Contents

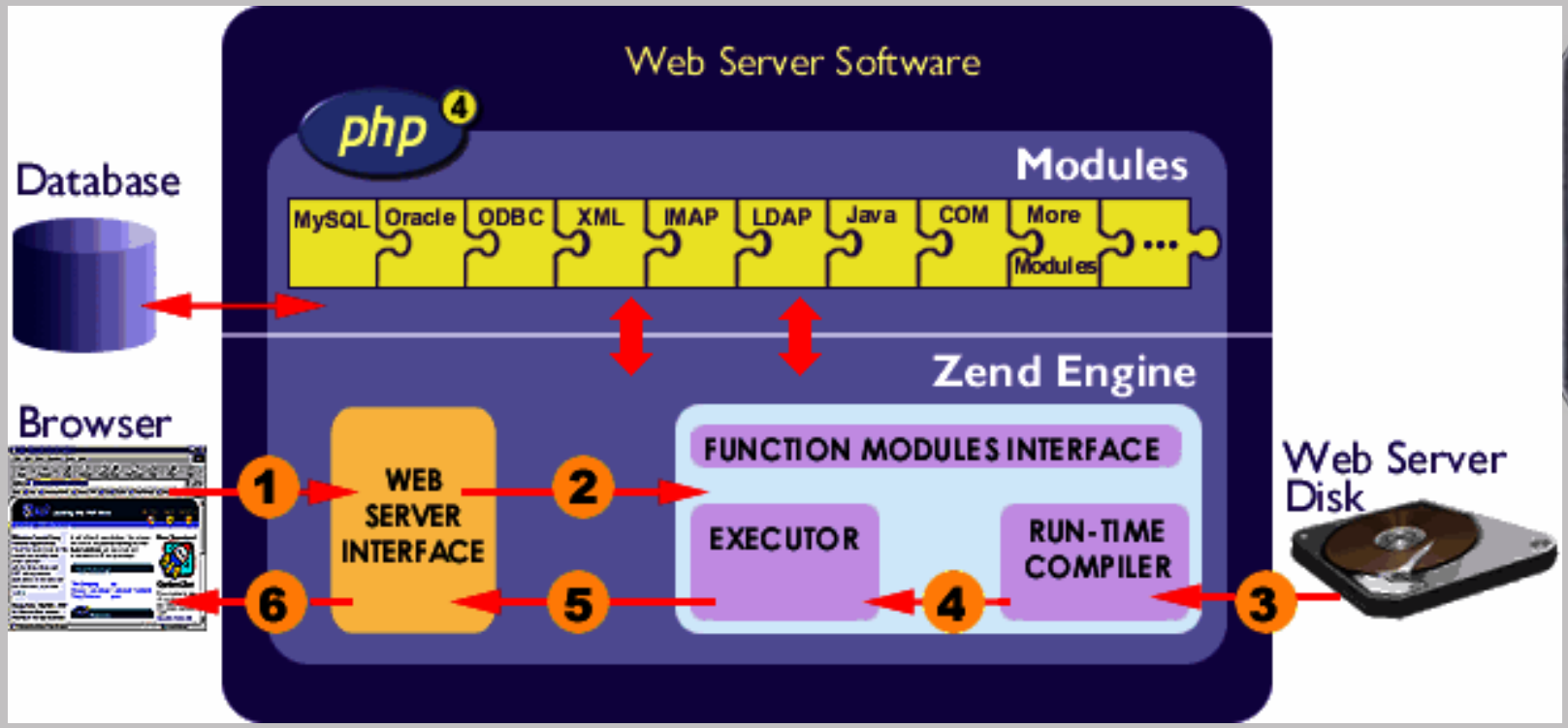
- About PHP
- Deep into PHP
- Decode's core---Opcode
- Intro Opcode Hooker technique
- Analysis Zend Optimizer
- Begin to Decode
- Bypass obfuscation

About PHP

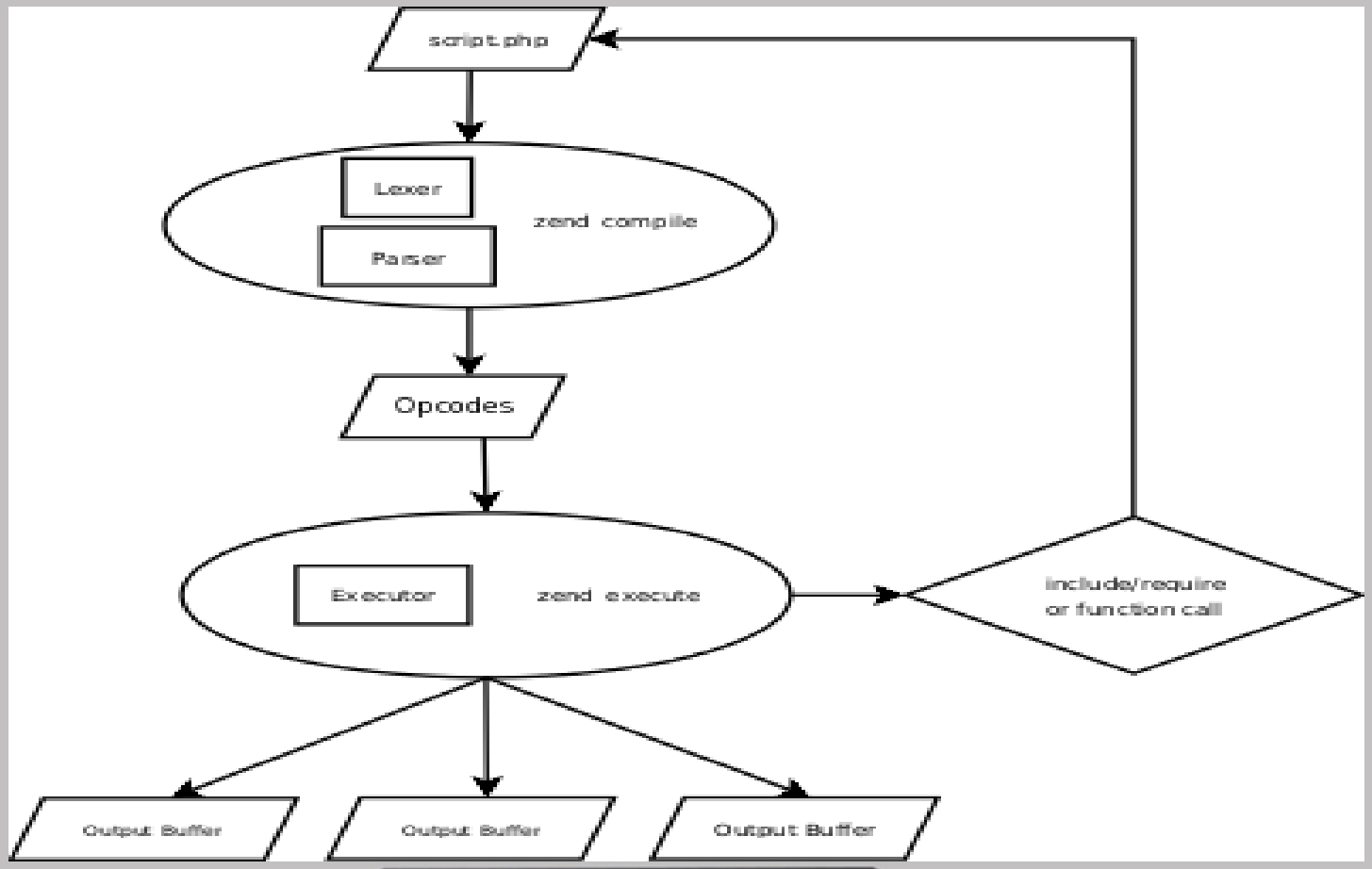
- concise but not simple
- multi-platform
- widely used

Inside PHP

- Php core and ZendCore



1、generates intermediate code (op-codes)from the actual PHP code. grouped into op-arrays--zend_compile
2、execute intermediate code on a virtual machine.--zend_execute

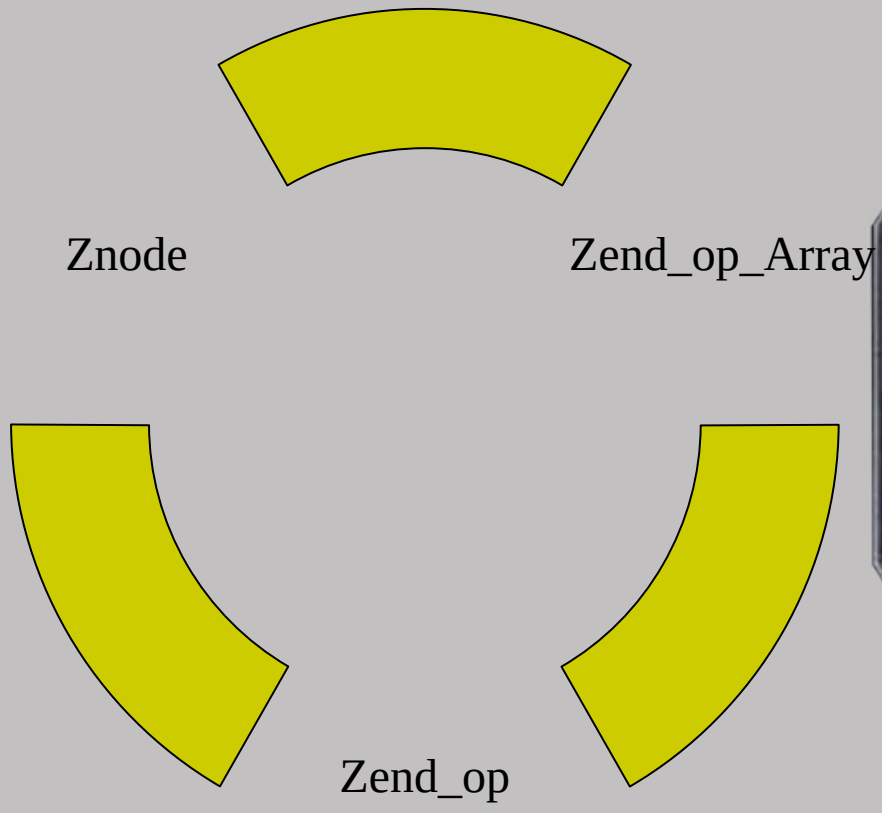


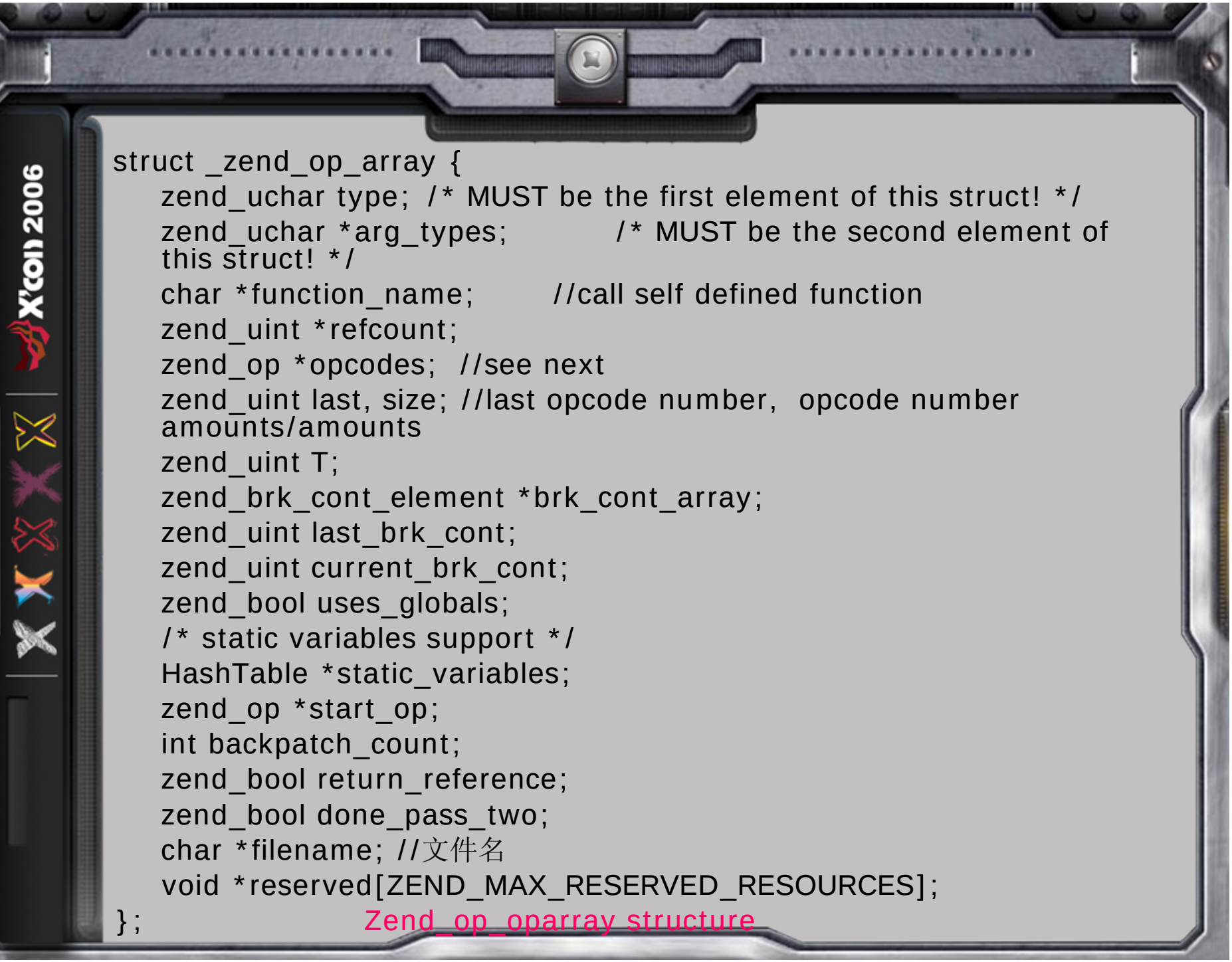
Opcode's structure

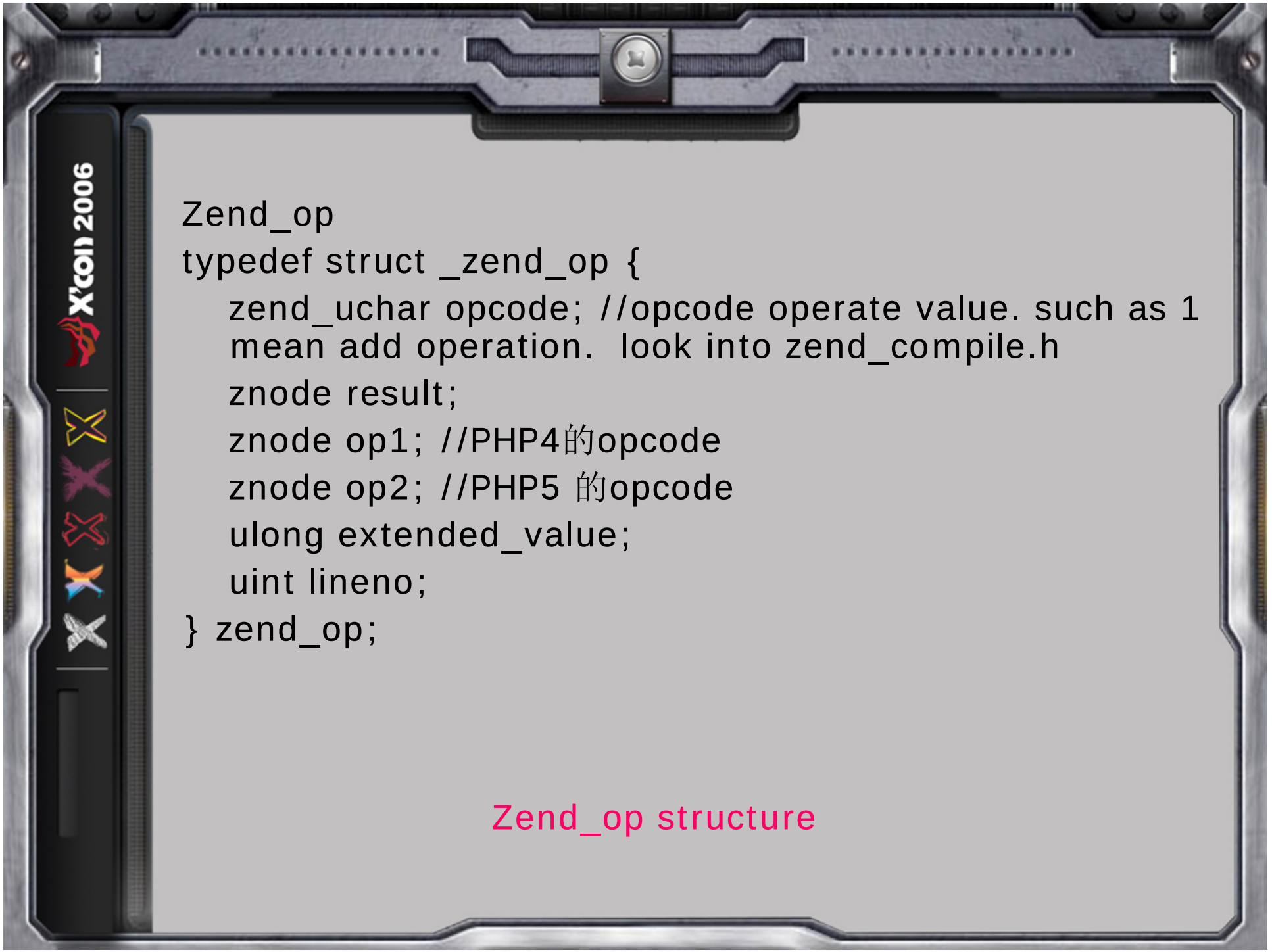
Opcode include:

- Zend_op_array
- Znode,
- Zend_op3

Zend_op_array was compiled by Zend engine and sent to Zend executing module



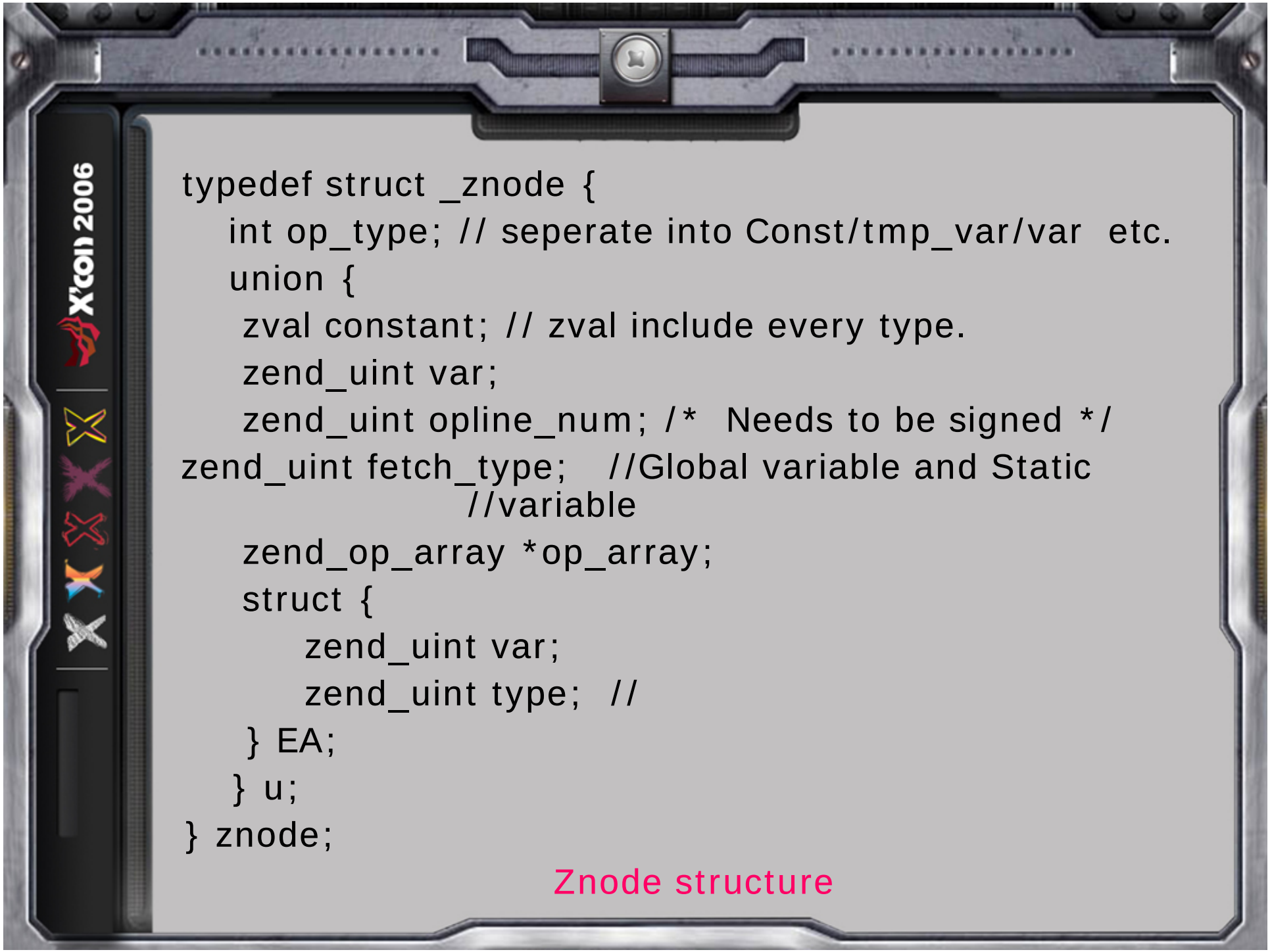

X'con 2006
struct _zend_op_array {
zend_uchar type; /* MUST be the first element of this struct! */
zend_uchar *arg_types; /* MUST be the second element of this struct! */
char *function_name; //call self defined function
zend_uint *refcount;
zend_op *opcodes; //see next
zend_uint last, size; //last opcode number, opcode number amounts/amounts
zend_uint T;
zend_brk_cont_element *brk_cont_array;
zend_uint last_brk_cont;
zend_uint current_brk_cont;
zend_bool uses_globals;
/* static variables support */
HashTable *static_variables;
zend_op *start_op;
int backpatch_count;
zend_bool return_reference;
zend_bool done_pass_two;
char *filename; //文件名
void *reserved[ZEND_MAX_RESERVED_RESOURCES];
};
Zend_op_oparray structure

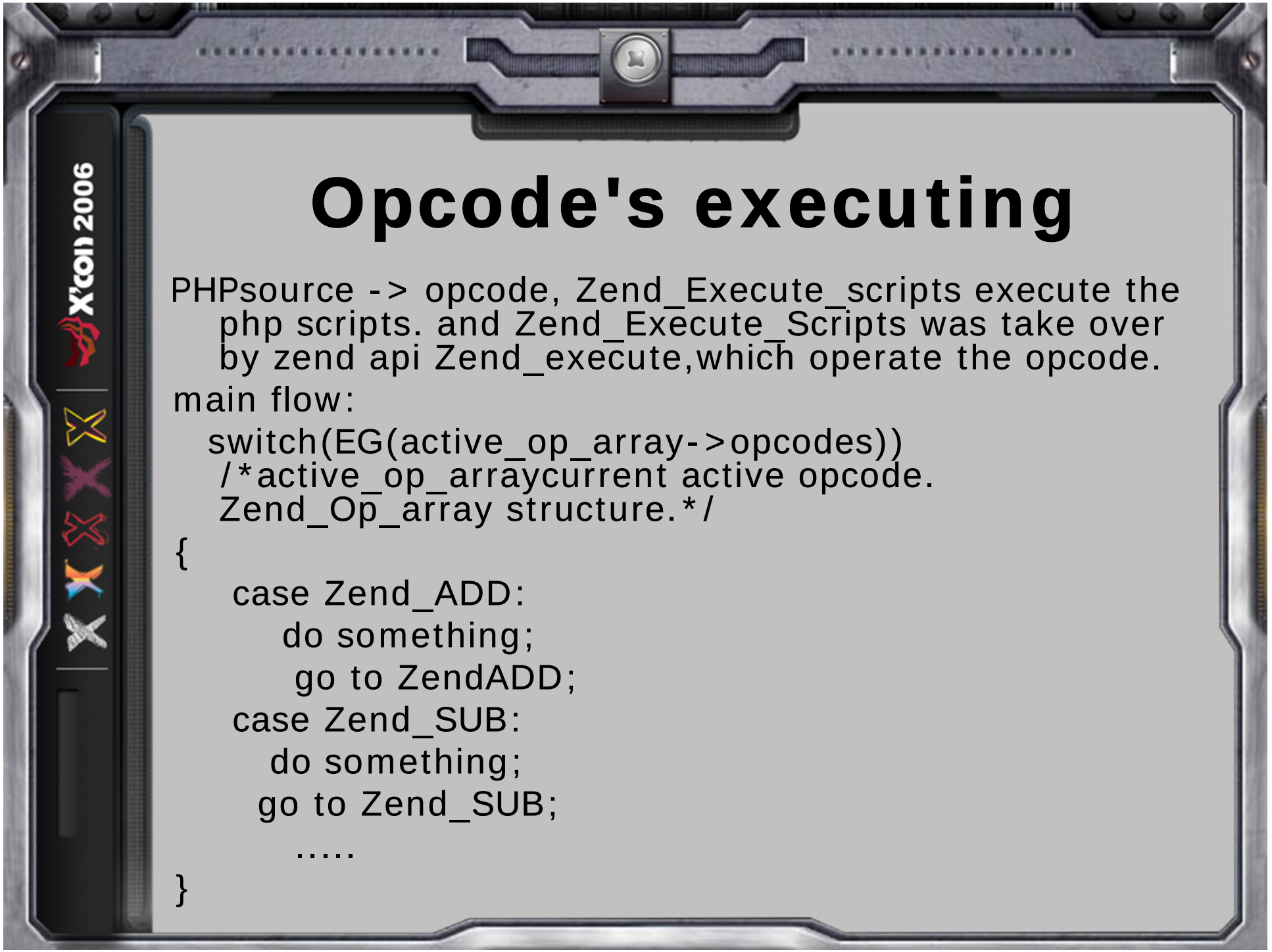


Zend_op

```
typedef struct _zend_op {
    zend_uchar opcode; //opcode operate value. such as 1
    mean add operation.  look into zend_compile.h
    znode result;
    znode op1; //PHP4的opcode
    znode op2; //PHP5 的opcode
    ulong extended_value;
    uint lineno;
} zend_op;
```

Zend_op structure


X'con 2006
typedef struct _znode {
int op_type; // seperate into Const/tmp_var/var etc.
union {
zval constant; // zval include every type.
zend_uint var;
zend_uint opline_num; /* Needs to be signed */
zend_uint fetch_type; //Global variable and Static
//variable
zend_op_array *op_array;
struct {
zend_uint var;
zend_uint type; //
} EA;
} u;
} znode;
Znode structure

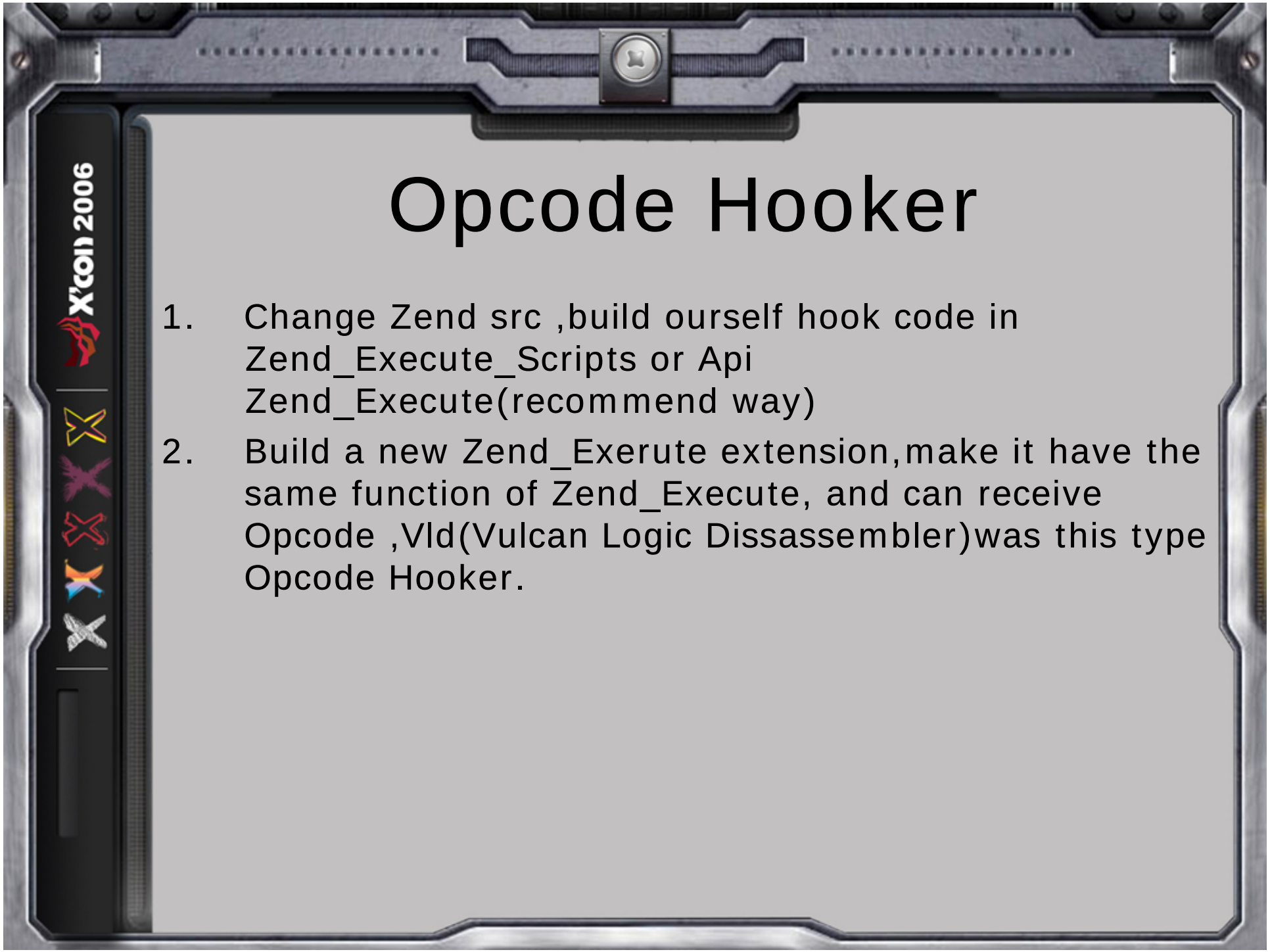


Opcode's executing

PHPsource -> opcode, Zend_Execute_scripts execute the php scripts. and Zend_Execute_Scripts was take over by zend api Zend_execute,which operate the opcode.

main flow:

```
switch(EG(active_op_array->opcodes))
 /*active_op_arraycurrent active opcode.
 Zend_Op_array structure.*/
{
    case Zend_ADD:
        do something;
         go to ZendADD;
    case Zend_SUB:
       do something;
      go to Zend_SUB;
         .....
}
```



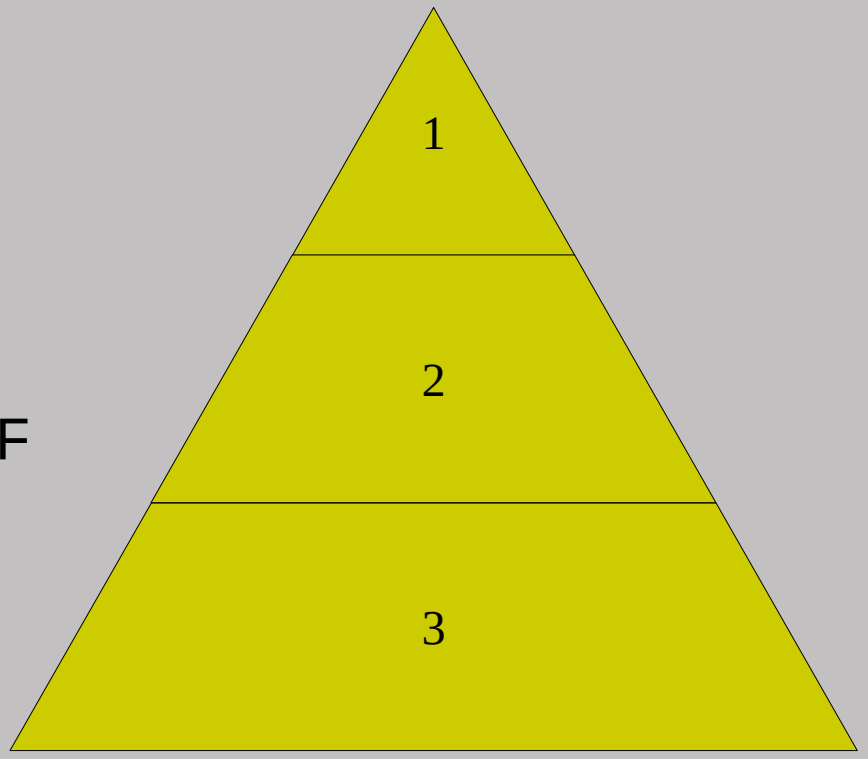
Opcode Hooker

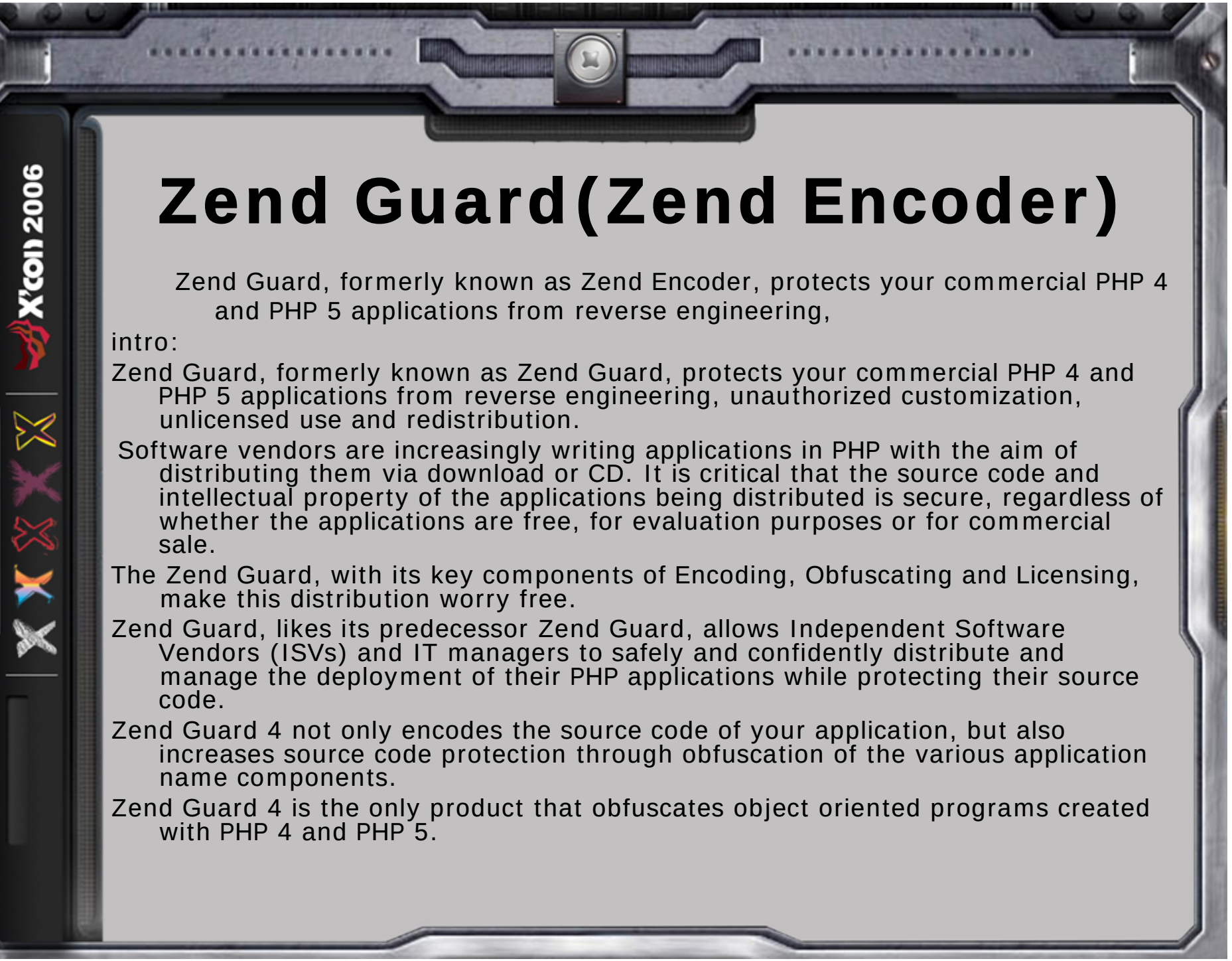
1. Change Zend src ,build ourself hook code in Zend_Execute_Scripts or Api Zend_Execute(recommend way)
2. Build a new Zend_Exerute extension,make it have the same function of Zend_Execute, and can receive Opcode ,Vld(Vulcan Logic Dissassembler)was this type Opcode Hooker.

extended Opcode Hooker mechanics

PHP extension run level at startup:

1. PHP_MINIT_FUNCTION
2. PHP_RINIT_FUNCTION
3. PHP_*SHUTDOWN_FUNCTION





Zend Guard(Zend Encoder)

Zend Guard, formerly known as Zend Encoder, protects your commercial PHP 4 and PHP 5 applications from reverse engineering,

intro:

Zend Guard, formerly known as Zend Guard, protects your commercial PHP 4 and PHP 5 applications from reverse engineering, unauthorized customization, unlicensed use and redistribution.

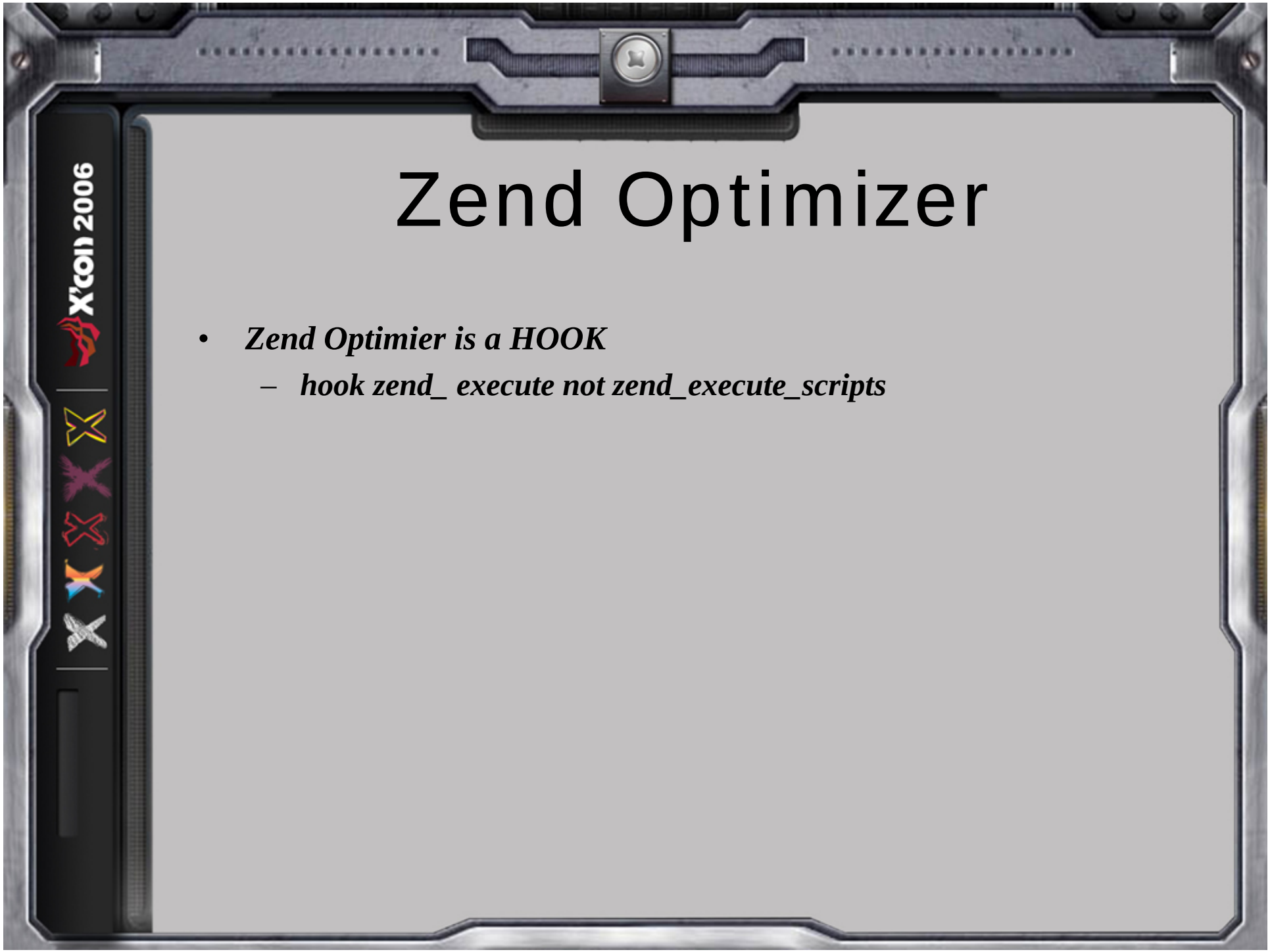
Software vendors are increasingly writing applications in PHP with the aim of distributing them via download or CD. It is critical that the source code and intellectual property of the applications being distributed is secure, regardless of whether the applications are free, for evaluation purposes or for commercial sale.

The Zend Guard, with its key components of Encoding, Obfuscating and Licensing, make this distribution worry free.

Zend Guard, likes its predecessor Zend Guard, allows Independent Software Vendors (ISVs) and IT managers to safely and confidently distribute and manage the deployment of their PHP applications while protecting their source code.

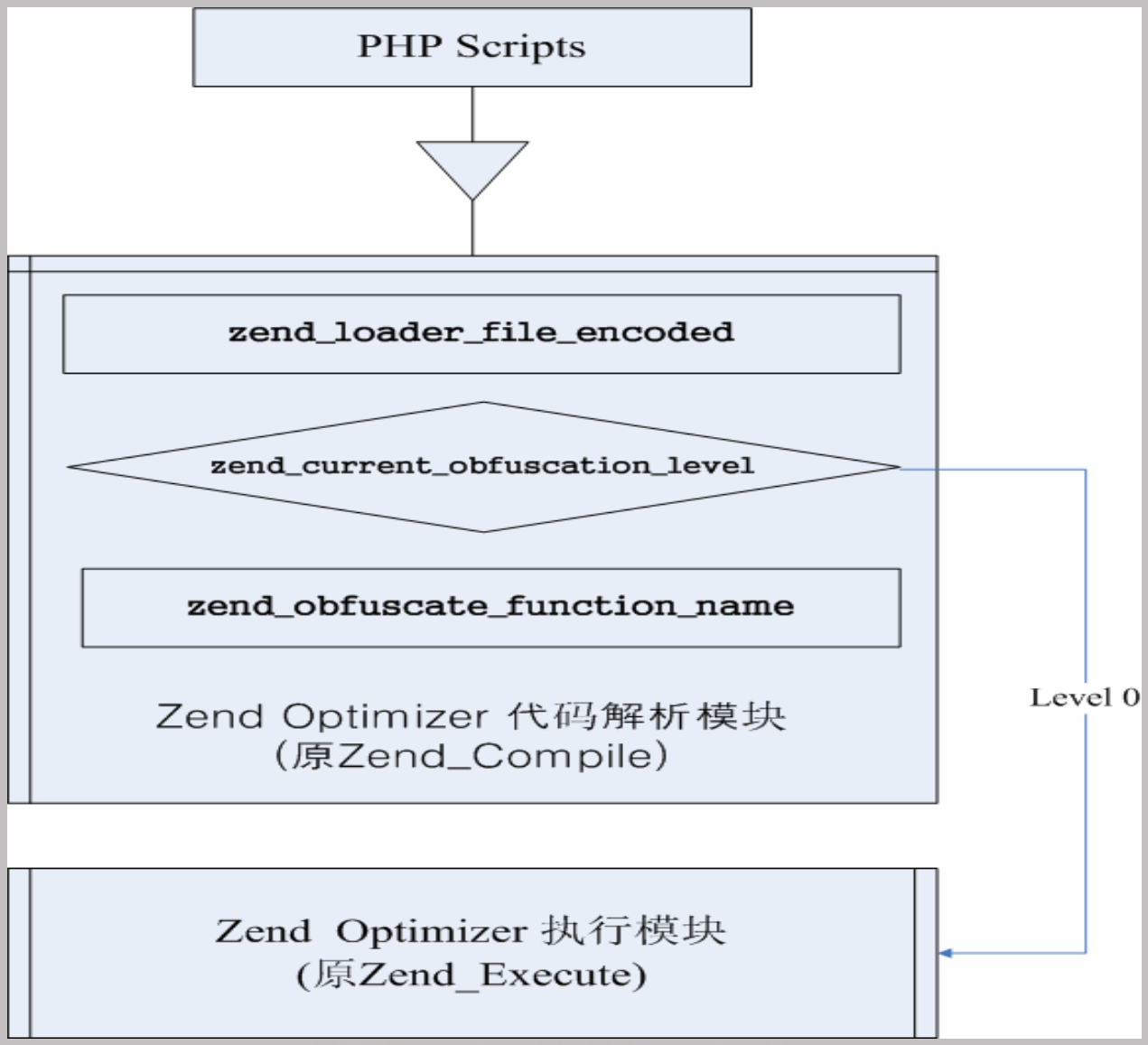
Zend Guard 4 not only encodes the source code of your application, but also increases source code protection through obfuscation of the various application name components.

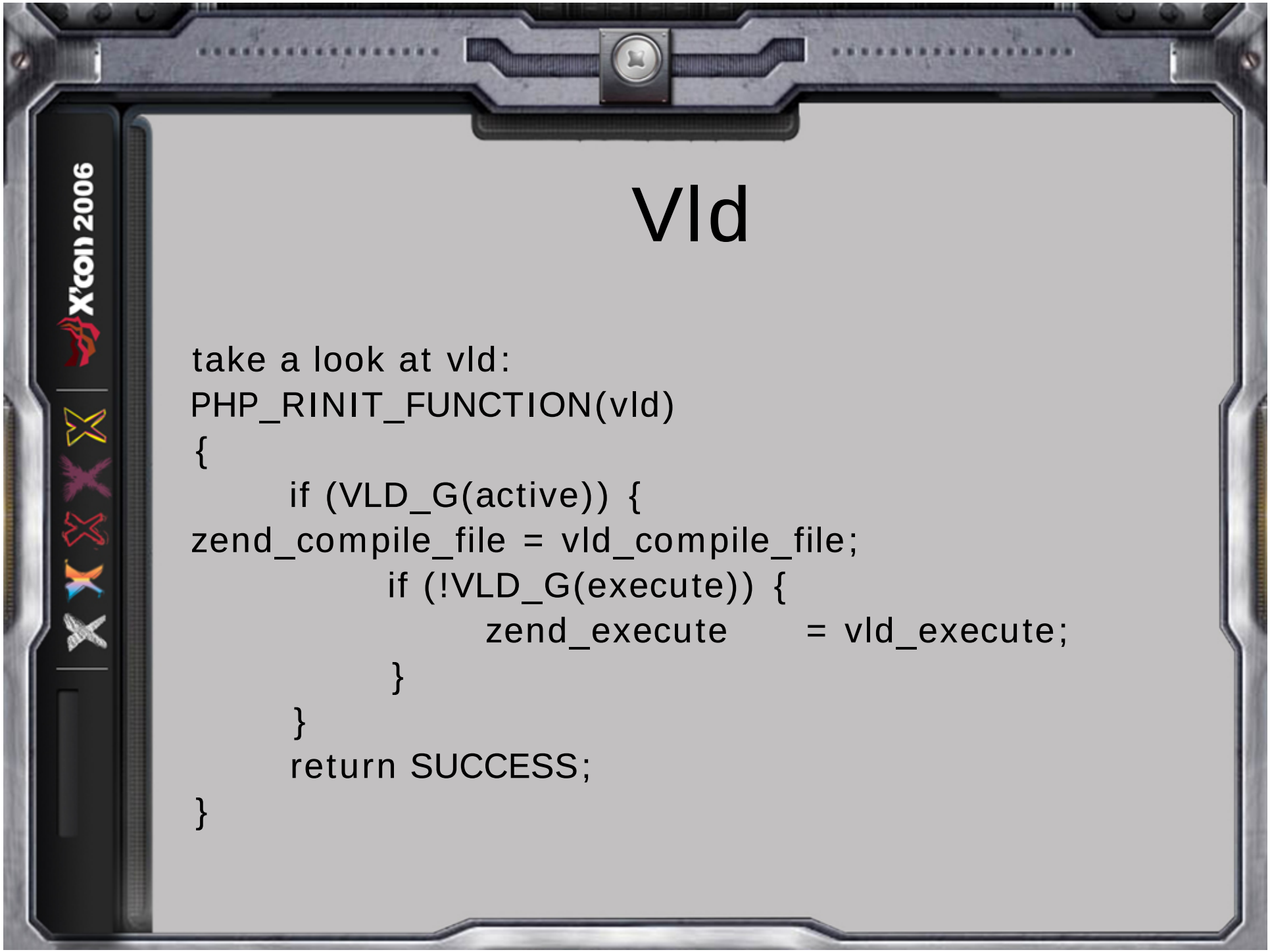
Zend Guard 4 is the only product that obfuscates object oriented programs created with PHP 4 and PHP 5.



Zend Optimizer

- ***Zend Optimier is a HOOK***
  - ***hook zend_ execute not zend_execute_scripts***

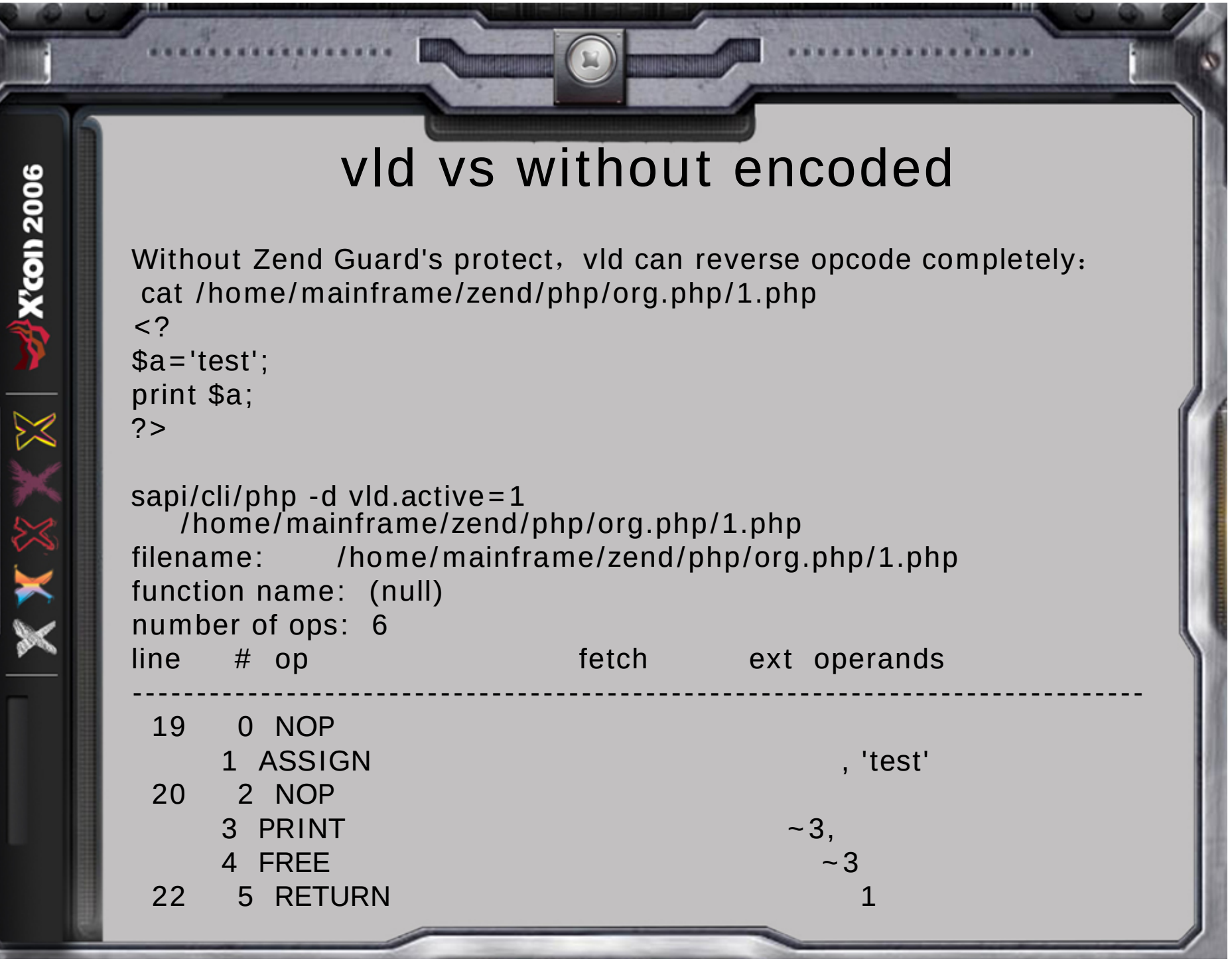

PHP Scripts
zend_loader_file_encoded
zend_current_obfuscation_level
zend_obfuscate_function_name
Zend Optimizer 代码解析模块
(原Zend_Compile)
Level 0
Zend Optimizer 执行模块
(原Zend_Execute)



Vld

take a look at vld:

```
PHP_RINIT_FUNCTION(vld)
{
        if (VLD_G(active)) {
zend_compile_file = vld_compile_file;
                if (!VLD_G(execute)) {
                        zend_execute      = vld_execute;
                }
        }
        return SUCCESS;
}
```

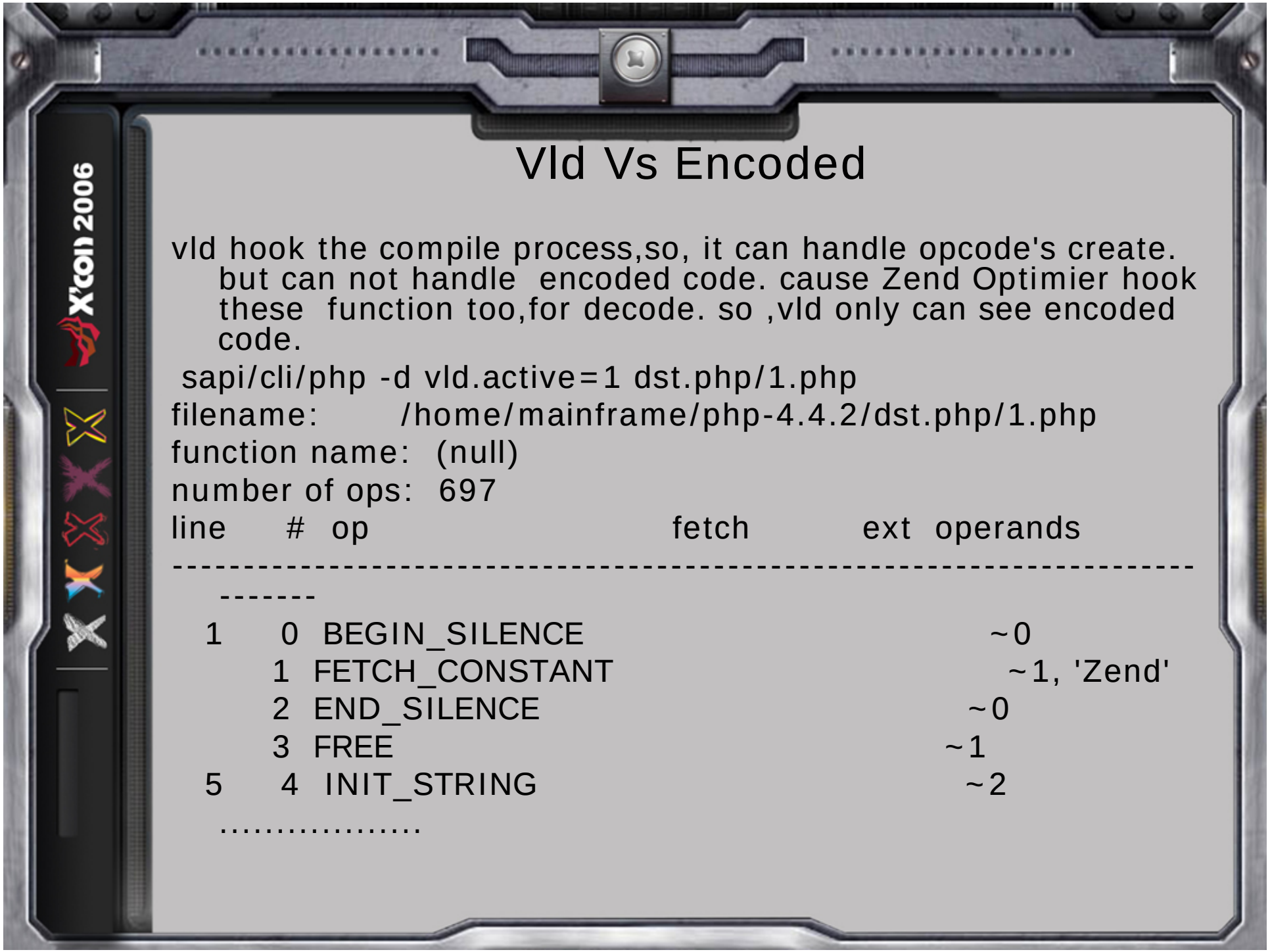


vld vs without encoded

Without Zend Guard's protect，vld can reverse opcode completely：

```
 cat /home/mainframe/zend/php/org.php/1.php
<?
$a='test';
print $a;
?>

sapi/cli/php -d vld.active=1
    /home/mainframe/zend/php/org.php/1.php
filename:       /home/mainframe/zend/php/org.php/1.php
function name:  (null)
number of ops:  6
line     #  op                           fetch          ext  operands
---------------------------------------------------------------------------
  19     0  NOP
         1  ASSIGN                                           , 'test'
  20     2  NOP
         3  PRINT                                         ~3,
         4  FREE                                            ~3
  22     5  RETURN                                            1
```



Vld Vs Encoded

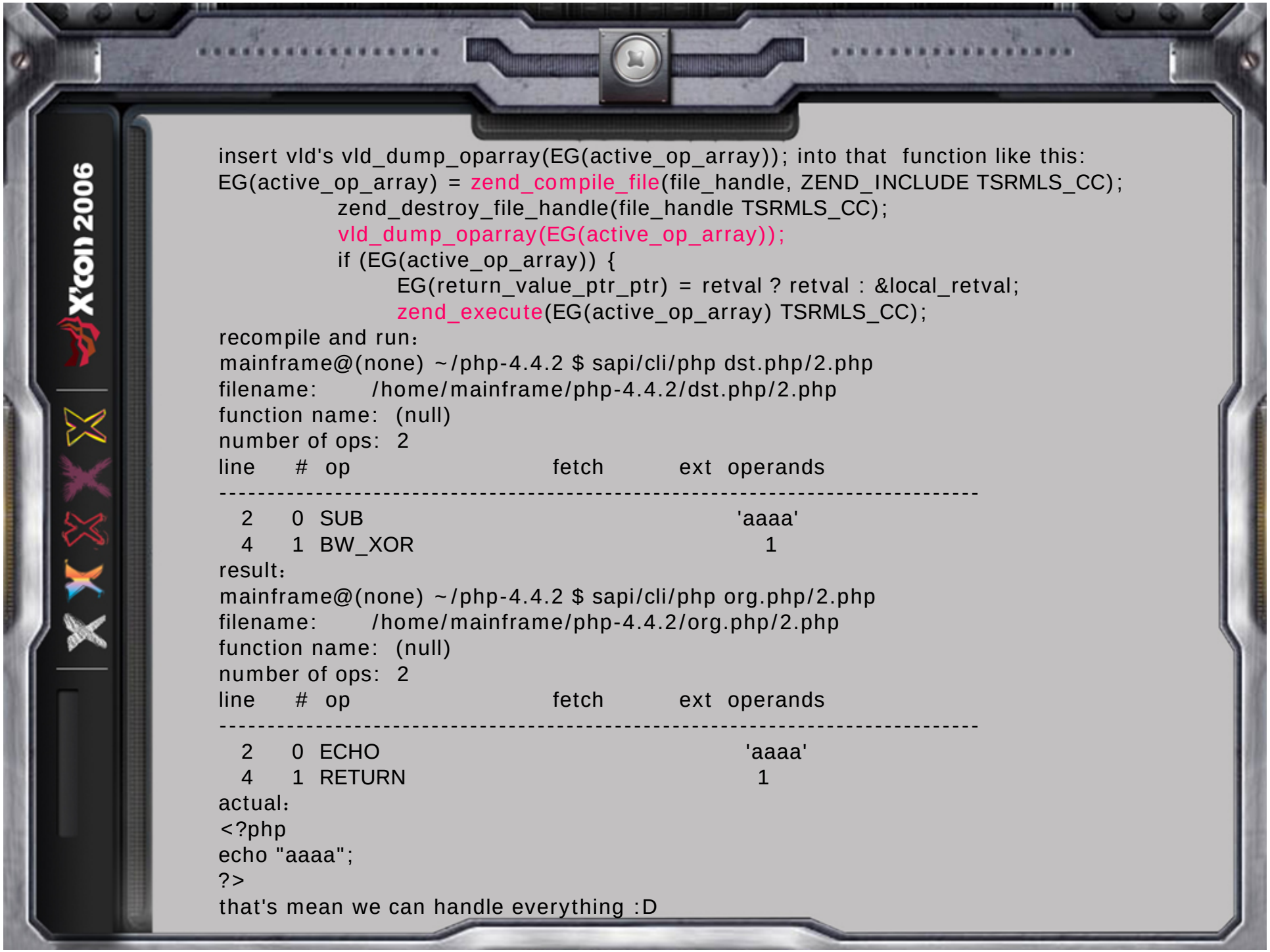
vld hook the compile process,so, it can handle opcode's create. but can not handle encoded code. cause Zend Optimier hook these function too,for decode. so ,vld only can see encoded code.

```
 sapi/cli/php -d vld.active=1 dst.php/1.php
filename:       /home/mainframe/php-4.4.2/dst.php/1.php
function name:  (null)
number of ops:  697
line     #  op                           fetch          ext  operands
-------------------------------------------------------------------------
   -------
  1     0  BEGIN_SILENCE                                    ~0
        1  FETCH_CONSTANT                                     ~1, 'Zend'
        2  END_SILENCE                                      ~0
        3  FREE                                           ~1
  5     4  INIT_STRING                                      ~2
   ..................
```

Api Zend_Execute_Scripts

```
ZEND_API int zend_execute_scripts(int type TSRMLS_DC, zval **retval,
    int file_count, ...)
    {
.....
      for (i=0; i<file_count; i++) {
            .......
            }
            EG(active_op_array) = zend_compile_file(file_handle,
    ZEND_INCLUDE TSRMLS_CC);
            zend_destroy_file_handle(file_handle TSRMLS_CC);

            if (EG(active_op_array)) {
                  EG(return_value_ptr_ptr) = retval ? retval :
    &local_retval;
                  zend_execute(EG(active_op_array) TSRMLS_CC);
                  ......
      }
      va_end(files);
      EG(active_op_array) = orig_op_array;
      return SUCCESS;
}
```



insert vld's vld_dump_oparray(EG(active_op_array)); into that function like this:

```
EG(active_op_array) = zend_compile_file(file_handle, ZEND_INCLUDE TSRMLS_CC);
            zend_destroy_file_handle(file_handle TSRMLS_CC);
            vld_dump_oparray(EG(active_op_array));
            if (EG(active_op_array)) {
                  EG(return_value_ptr_ptr) = retval ? retval : &local_retval;
                  zend_execute(EG(active_op_array) TSRMLS_CC);
```

recompile and run:

```
mainframe@(none) ~/php-4.4.2 $ sapi/cli/php dst.php/2.php
filename:       /home/mainframe/php-4.4.2/dst.php/2.php
function name:  (null)
number of ops:  2
line     #  op                   fetch        ext  operands
-------------------------------------------------------------------------------
   2     0  SUB                                     'aaaa'
   4     1  BW_XOR                                    1
```

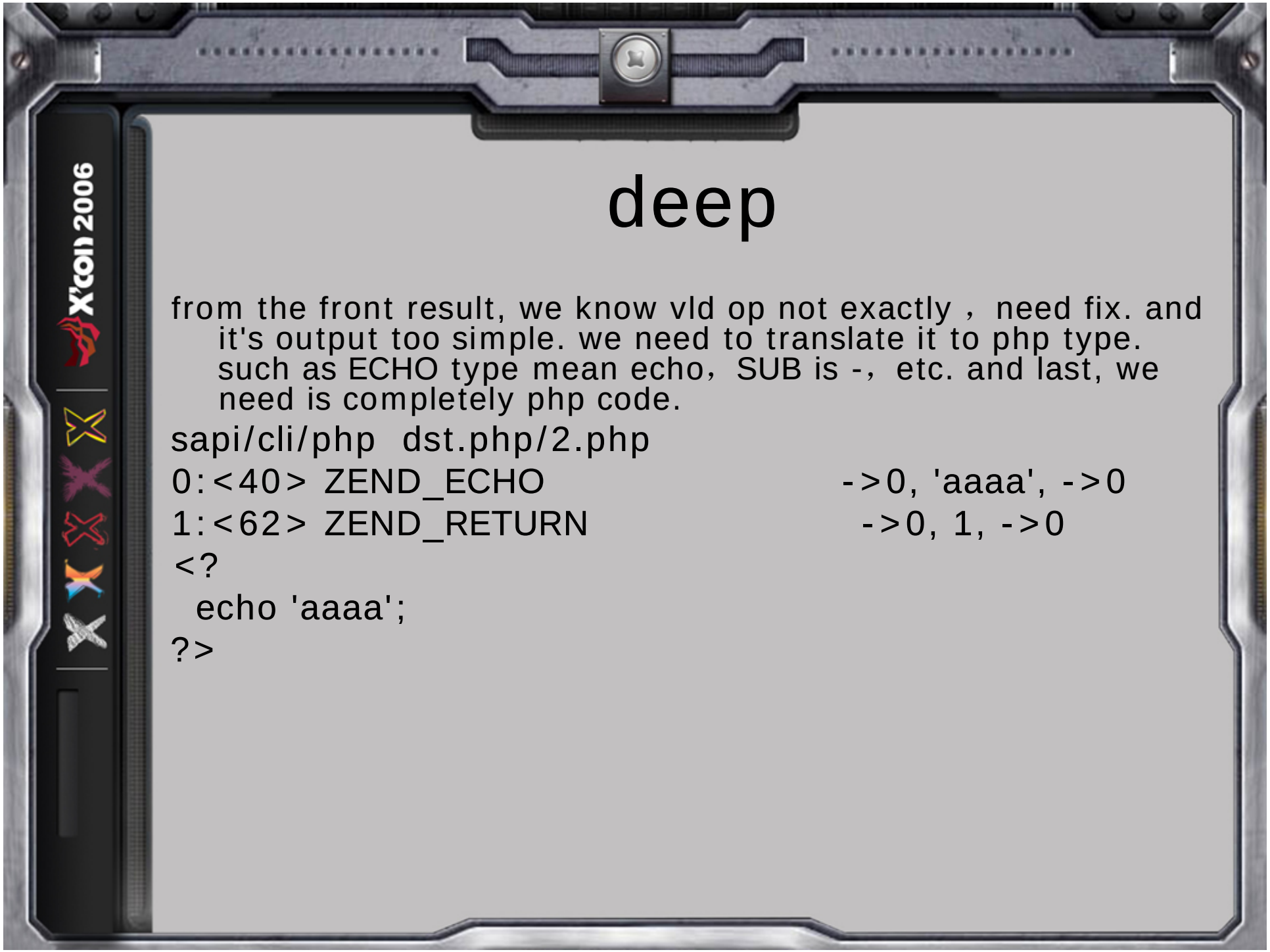
result:

```
mainframe@(none) ~/php-4.4.2 $ sapi/cli/php org.php/2.php
filename:       /home/mainframe/php-4.4.2/org.php/2.php
function name:  (null)
number of ops:  2
line     #  op                   fetch        ext  operands
-------------------------------------------------------------------------------
   2     0  ECHO                                    'aaaa'
   4     1  RETURN                                    1
```

actual:

```
<?php
echo "aaaa";
?>
```

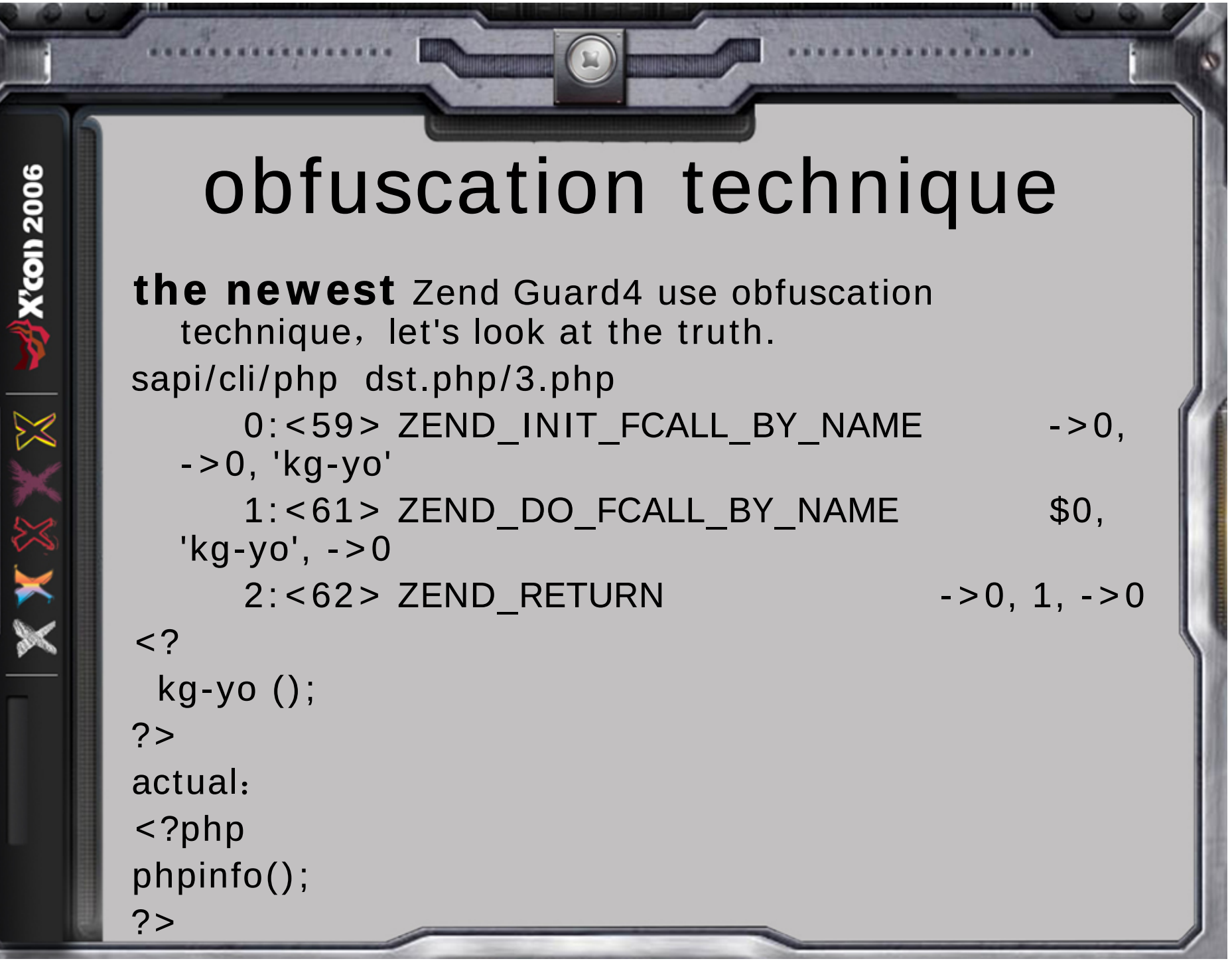
that's mean we can handle everything :D



deep

from the front result, we know vld op not exactly ， need fix. and it's output too simple. we need to translate it to php type. such as ECHO type mean echo， SUB is -， etc. and last, we need is completely php code.

```
sapi/cli/php  dst.php/2.php
0:<40> ZEND_ECHO                    ->0, 'aaaa', ->0
1:<62> ZEND_RETURN                   ->0, 1, ->0
<?
  echo 'aaaa';
?>
```



obfuscation technique

the newest Zend Guard4 use obfuscation technique，let's look at the truth.

sapi/cli/php dst.php/3.php

```
        0:<59> ZEND_INIT_FCALL_BY_NAME          ->0,
   ->0, 'kg-yo'
        1:<61> ZEND_DO_FCALL_BY_NAME            $0,
   'kg-yo', ->0
        2:<62> ZEND_RETURN                   ->0, 1, ->0
```

```
<?
  kg-yo ();
?>
```

actual：

```
<?php
phpinfo();
?>
```

bypass obfuscating

- reverse obfuscation(i am not xiaoyun wang)
- use zend_obfuscate_function_name make a function name table

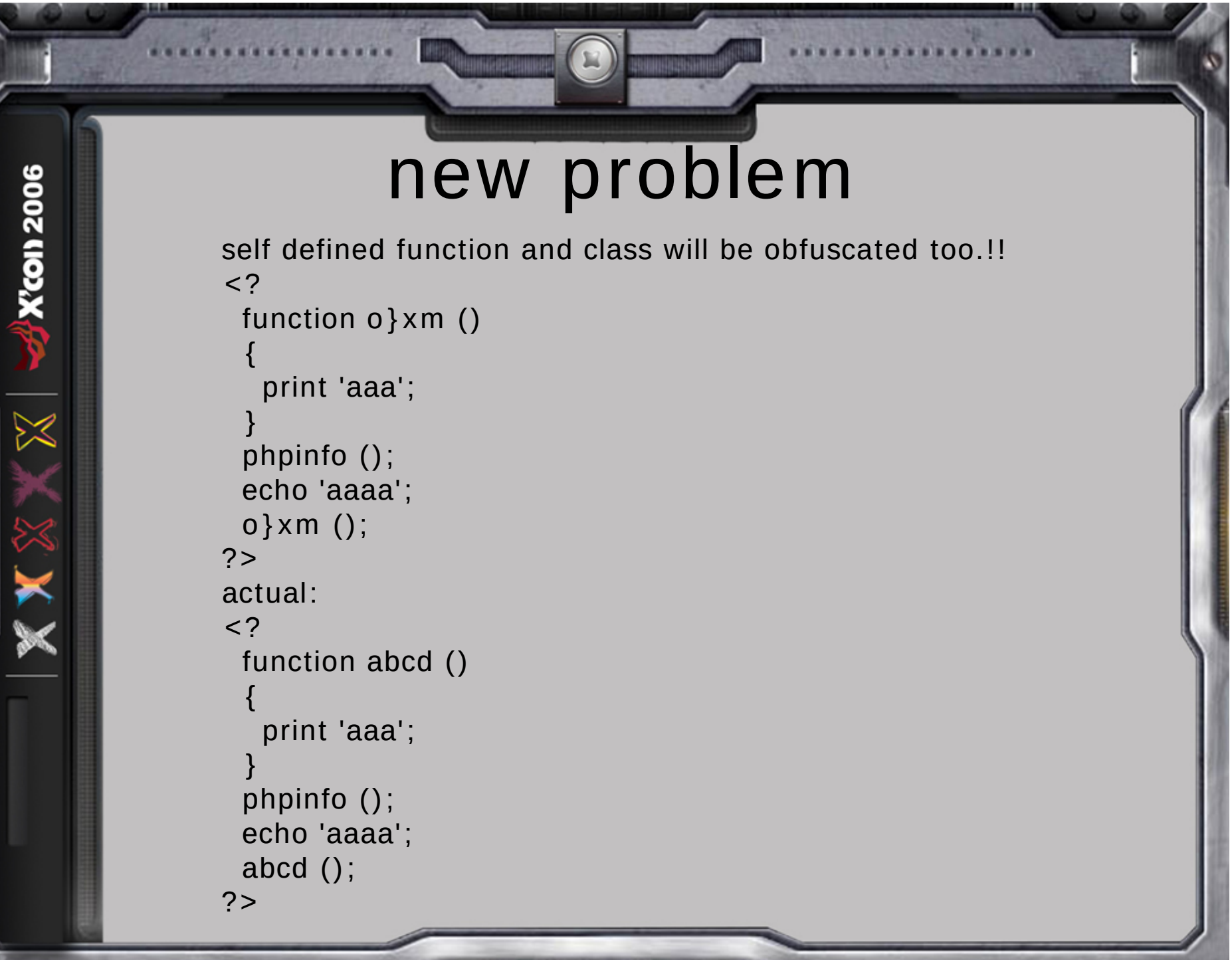
zend_obfuscate_function_name:

```
<?

$module = 'standard';
$functions = get_extension_funcs("standard");

foreach($functions as $func) {
   printf( "%s();\r\n",
zend_obfuscate_function_name($func));

}
?>
```



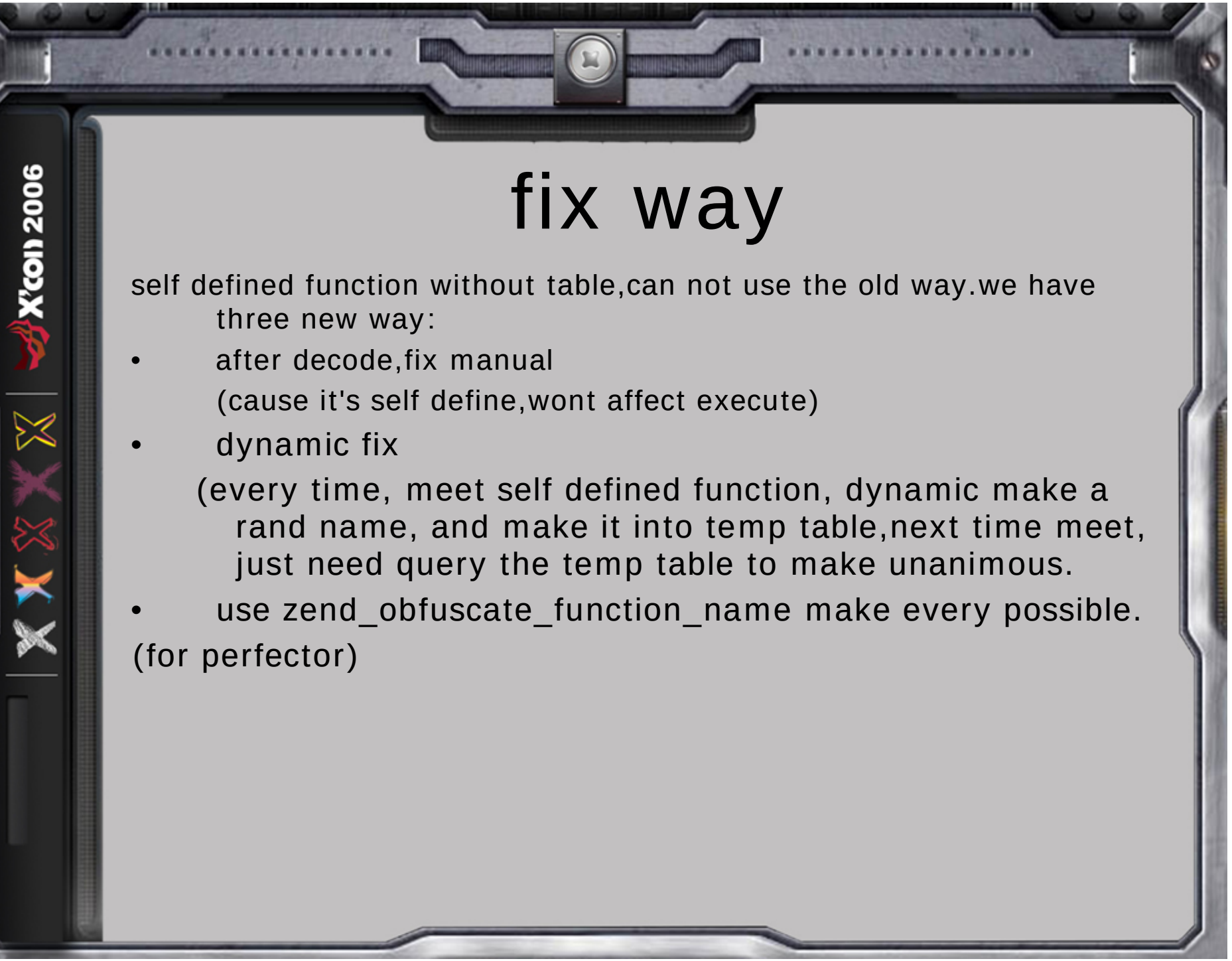
new problem

self defined function and class will be obfuscated too.!!

```
<?
  function o}xm ()
  {
    print 'aaa';
  }
  phpinfo ();
  echo 'aaaa';
  o}xm ();
?>
```

actual:

```
<?
  function abcd ()
  {
    print 'aaa';
  }
  phpinfo ();
  echo 'aaaa';
  abcd ();
?>
```

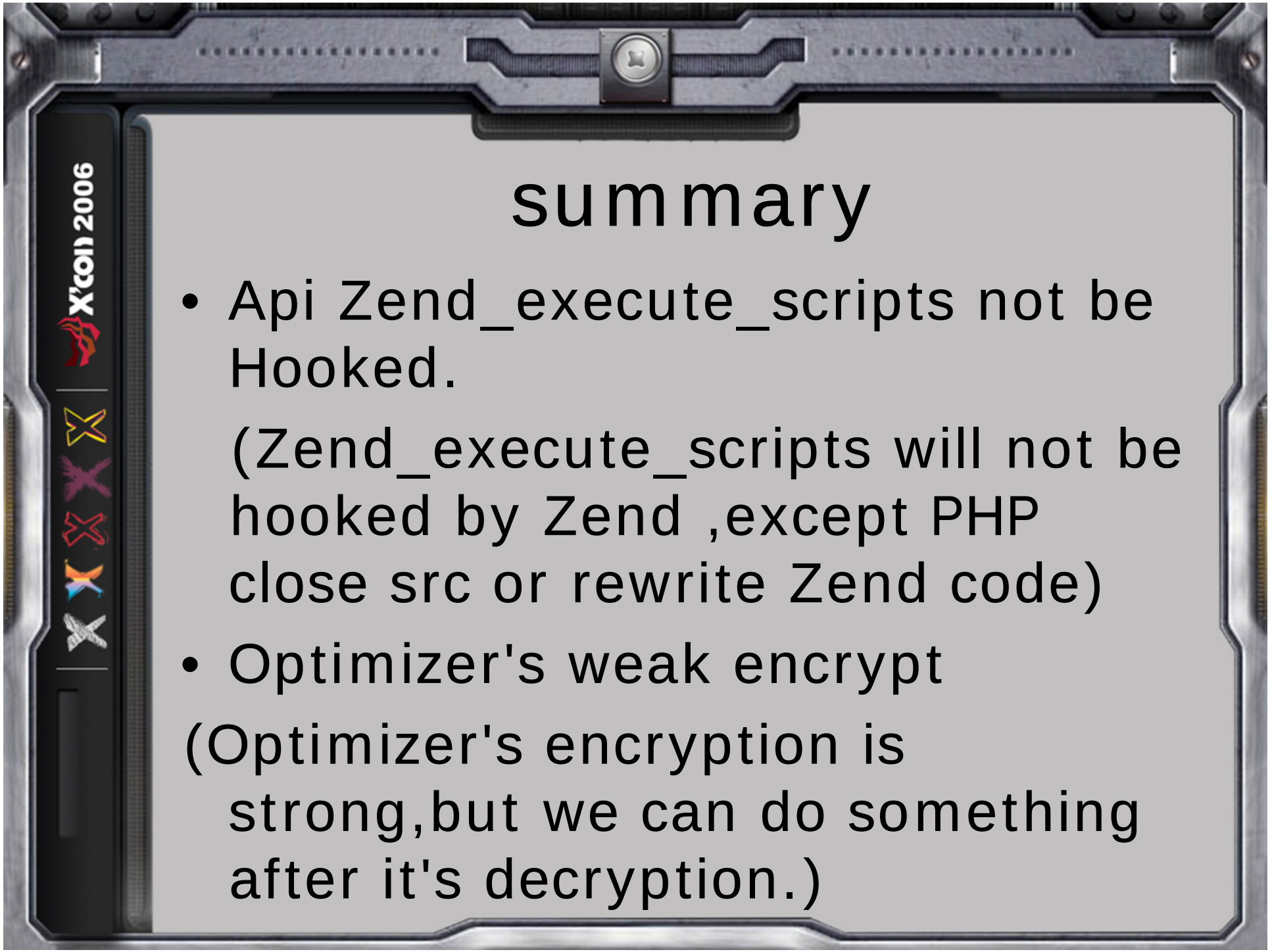


fix way

self defined function without table,can not use the old way.we have three new way:

- after decode,fix manual
  (cause it's self define,wont affect execute)
- dynamic fix
  (every time, meet self defined function, dynamic make a rand name, and make it into temp table,next time meet, just need query the temp table to make unanimous.
- use zend_obfuscate_function_name make every possible.

(for perfector)

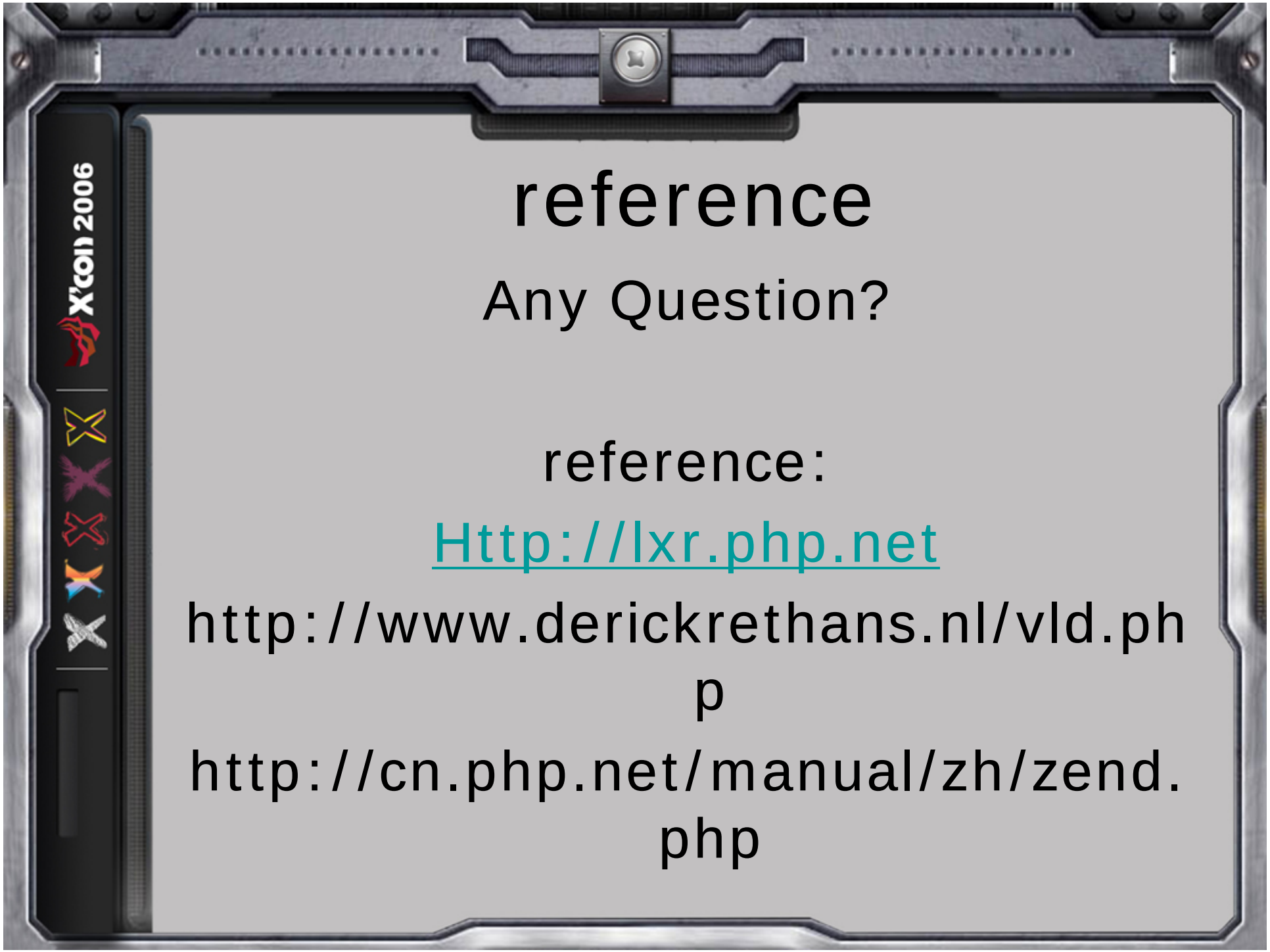


summary

- Api Zend_execute_scripts not be Hooked.

  (Zend_execute_scripts will not be hooked by Zend ,except PHP close src or rewrite Zend code)
- Optimizer's weak encrypt

(Optimizer's encryption is strong,but we can do something after it's decryption.)



reference

Any Question?

reference:

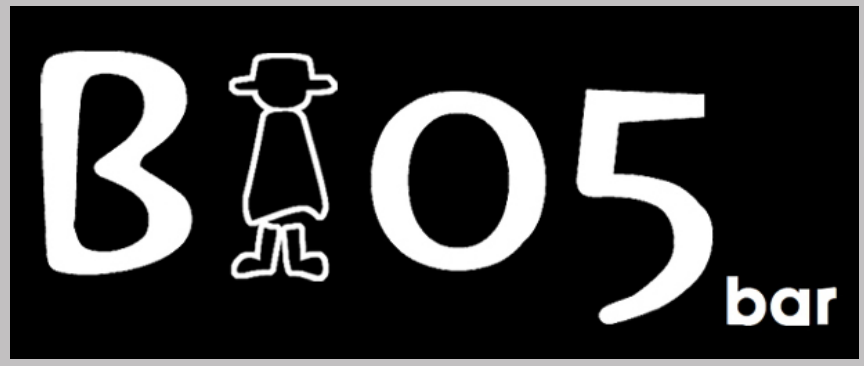
Http://lxr.php.net

http://www.derickrethans.nl/vld.php

http://cn.php.net/manual/zh/zend.php

Thanks

Without B105 bar's beer, we can not finish this topic.



welcome to:

irc.0x557.net: 9940 (SSL)

Channel: #segfault