
PHP Trick Trip
php
php
php
php
php

Who am I

- **문시우** aka **munsiwoo**
- 선린인터넷고등학교 3학년
- CodeRed, Layer7 (Leader)
- //munsiwoo.kr
- //fb.com/munsiwoo
- @pretty_mun

前 Index

- PHP 소개 - PHP와 젠드 엔진이란 무엇인가요?
- PHP 트릭 소개 - 트릭이란 무엇인가요?
- PHP Type cast & Comparison 트릭 분석
- PHP Filesystem functions 트릭 소개 및 분석
- PHP 5.3 path truncate 이슈 소개 및 분석
- PHP preg_match 함수 이슈 소개 및 분석
- PHP parse_url 함수 이슈 소개 및 분석
- Reference & 질문과 답변

Index 

- PHP 소개 - PHP와 젠드 엔진이란 무엇인가요?
- PHP 트릭 소개 - 트릭이란 무엇인가요?
- PHP Type cast & Comparison 트릭 분석
- PHP Filesystem functions 트릭 소개 및 분석
- Reference & 질문과 답변

What is PHP?

- 범용성을 지닌 널리 사용되는 오픈 소스 스크립트 언어
- 주로 웹 애플리케이션을 작성할 때 쓰임
- 다양한 이유로 인해 호불호가 심하게 갈리는 언어
- 프레임워크는 대표적으로 Laravel, CodeIgniter, CakePHP

- 여러모로 개발자 뒷목잡게 만드는 언어 (이유는 뒤에서 설명)

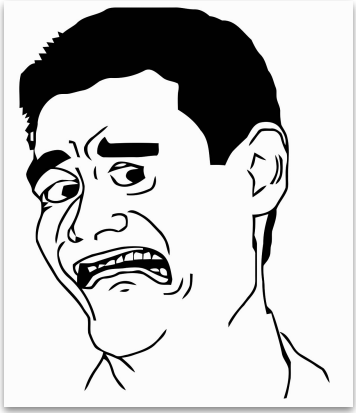
Advantages of PHP

- 오픈 소스 (github.com/php/php-src)
- 태그만 있으면 실행 (HTML, PHP 콜라보)
- 실행 속도가 비교적 빠르다. (PHP7 기준)
- 문법이 간결해서 배우기 쉽다.

```php
<?php
    $var = "Hello, PHP!";
    echo $var;
?>
```

Disadvantages of PHP

- PHP 함수 네이밍 (일관성이 없음)
  - base64_encode, urlencode
- HTML, PHP 콜라보 (유지보수 문제)
- 수 많은 이슈, 트릭 (개발자 뒷목)

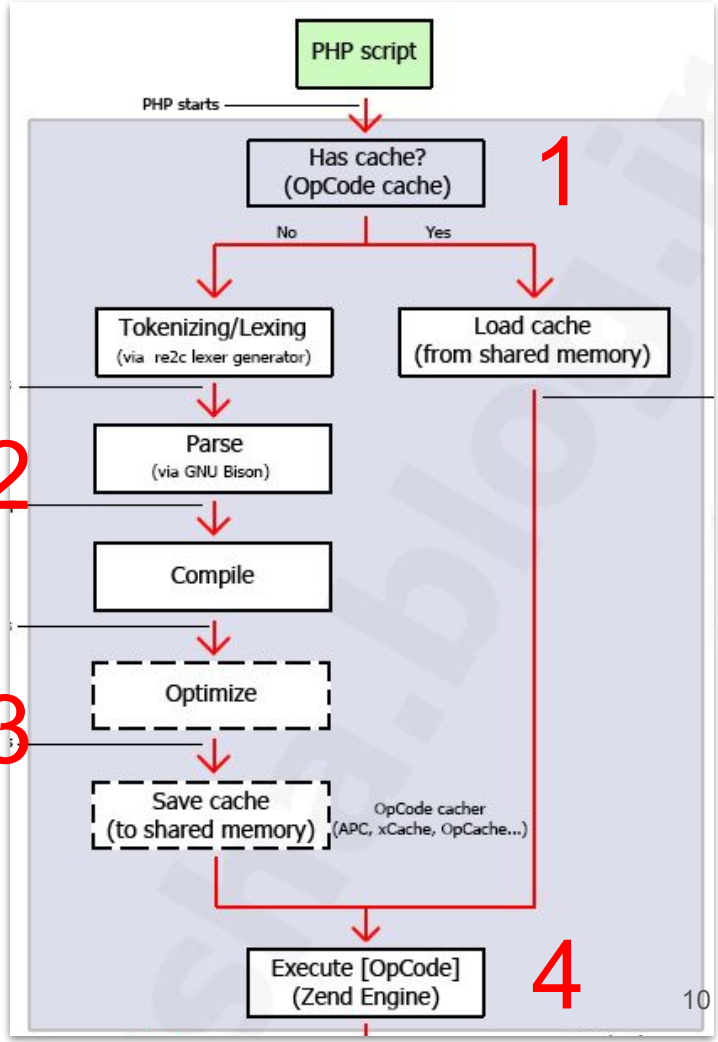


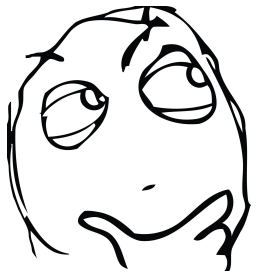
What is Zend Engine?

- 1999년 PHP 4를 최초로 등장
- PHP의 핵심 구현체라고 할 수 있는 PHP 인터프리터 역할
  - PHP 스크립트 분석, 바이트코드 변환, 최적화, 실행 담당
  - 핵심 부분을 형성함과 함께 매우 기초적 단계에서 PHP를 구현
- 개발자 이름이 Zeev와 Andi 라서 Zend 라고한다.

How does PHP(Zend Engine) work?

1. 이전에 캐싱한 옵캐시가 있는지 확인
2. 없다면 파싱 작업을 거치고 옵코드로 변환
3. 옵코드(바이트코드) 최적화 및 저장(캐싱)
4. 최적화한 옵코드를 차례대로 실행
5. 실행 결과를 반환





What is PHP Trick?

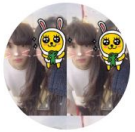
Q. 당신에게 PHP 트릭이란 어떤 의미인가요?

PHP 트릭은 대단합니다 선생님

그거슨 웹해커들의 지성을 엿볼 수 있는 훌륭한 수단입니다.

php 트릭은 발암입니다

간단하게 얘기하자면



트릭은 예술 입니다

걍 쓰레기임을 증명해주는 버그

PHP 트릭은 탈모를 부릅니다

대회 때마다 머리털 다 빠져요 제발 그만내주세요 ㅜㅜ

아니 php 느낌 살려서 ㅓ

php trick === php bug

What is PHP Trick?

- 트릭 - 사람의 눈을 현혹시키는 속임수
- PHP 트릭 - PHP 에서 사람의 눈을 현혹시키는 속임수
- 공식 문서에 나와있지 않은 숨겨진 기능 / 오류
- 언어 구조에서 발생하는 버그나 함수에서 발생하는 버그
- 한마디로 PHP 버그!

PHP Type Cast & Comparison Trick

What is Type casting? (Type Juggling)

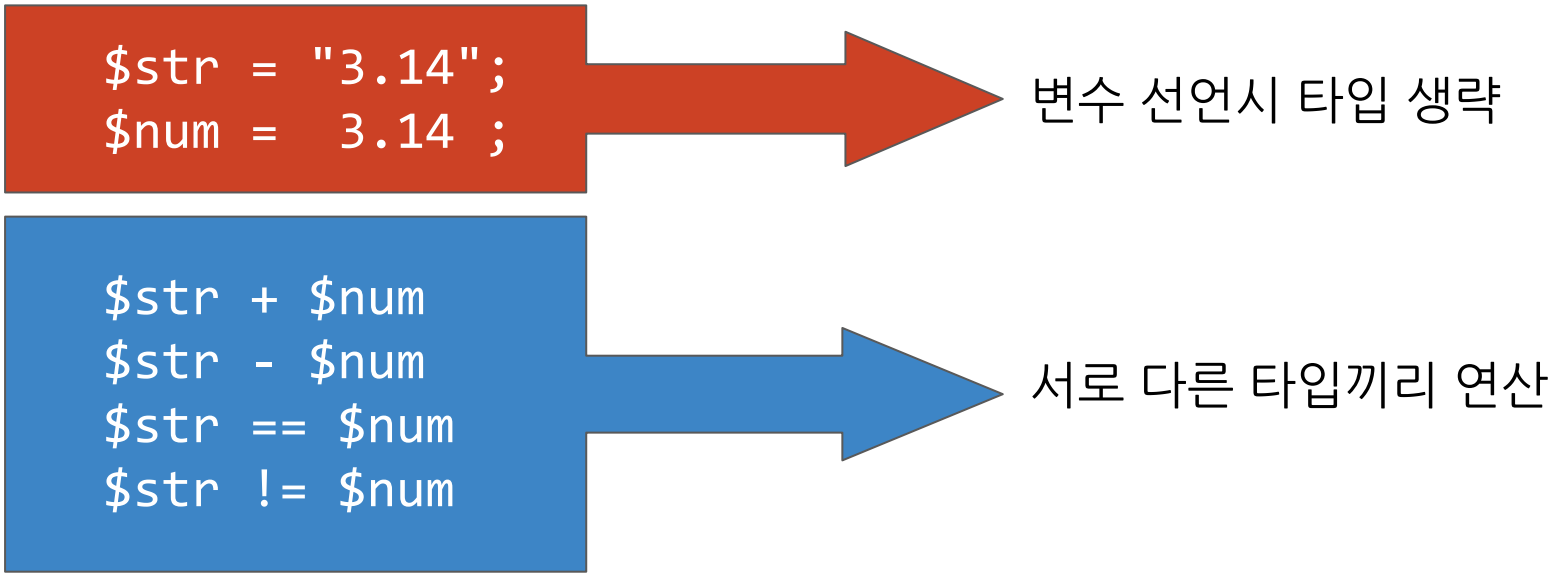
프로그래밍에서 타입 캐스팅이란 무엇인가요?

```
settype($str, "integer");
settype($num, "string");

(int)$string

(object)$array

(array)$object
```

데이터 타입을 변환하는 것.

PHP is support Auto type casting (Type Juggling)

PHP에서는 타입 변환을 자동으로 해준다. 아래는 오토 타입 캐스팅의 예시다.



PHP Auto type casting Examples

```
var_dump('1' + 2);
var_dump('1' + '2');
var_dump('55' + 45.5);
```

(True)

```
var_dump('5abc' == 5);
var_dump('0.10' == '0.10000000000000001');
var_dump('0e12345' == '0e67890');
```

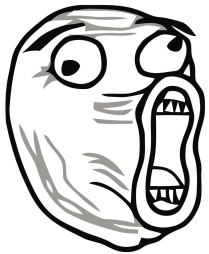
PHP Auto type casting Example

'1' + '2'	---------->	int(3)
'55' + 45.5	---------->	float(100.5)
'5abc' == 5	---------->	bool(true)

PHP Auto type casting Example

"0e1234" == "0e5678"

bool(true) - It's called Magic Hash !



PHP Magic Hash?

Why md5('240610708') is equal to md5('QNKCDZO')? - Stack Overflow

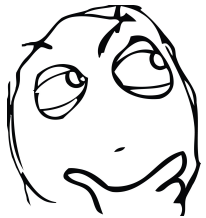
https://stackoverflow.com/.../why-md5240610708-is-equal-to-md... ▾ 이 페이지 번역하기

답변 2개

2015. 9. 2. - md5('QNKCDZO') 's result is 0e830400451993494058024219903391 . They are both float number format strings (numerical strings), and if you use == in **php**, ...

```
md5('240610708') == md5('QNKCDZO') ----> True!

md5('240610708') → 0e462097431906509019562988736854
md5('QNKCDZO')   → 0e830400451993494058024219903391
```

Why?

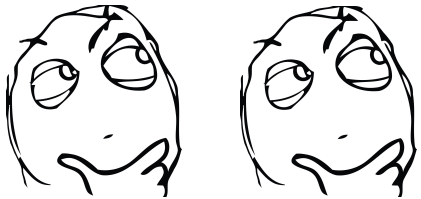


(number) e (number) 꼴은 지수 표현으로 인식하여

문자열 '0e1234' 은 0e1234로 변환 된다, 따라서 0이 되고
문자열 '0e5678' 또한 0e5678로 변환되어, 최종적으로 0이 된다.

'0e1234' == '0e5678' (True) -----> 0 == 0 (True)

So why?



PHP에서 == 연산자는 느슨한 비교 연산 (Loose comparison)

즉, 타입은 비교하지 않고 타입 캐스팅 후 값만 비교한다.

Loose comparisons with ==

	TRUE	FALSE	1	0	-1	"1"	"0"	"-1"	NULL	array()	"php"	""
TRUE	TRUE	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE
FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE	TRUE
1	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
0	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	TRUE	TRUE
-1	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE
"1"	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
"0"	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE
"-1"	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE
NULL	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	TRUE
array()	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	FALSE
"php"	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE
""	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE

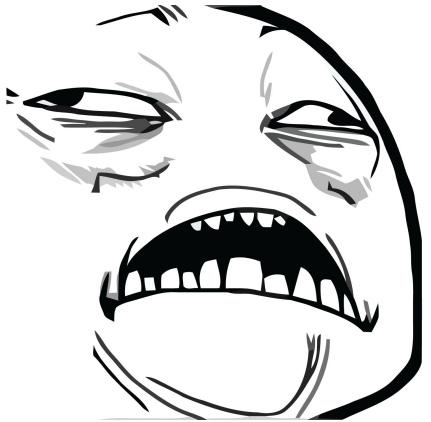
"33e1" == "3300E-1" (True)

"15e0005" == "1500000" (True)

'1000e-1' == '100' (True)

"99e-2" == "0.99" (True)

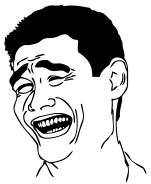
'99e+2' == '9900' (True)



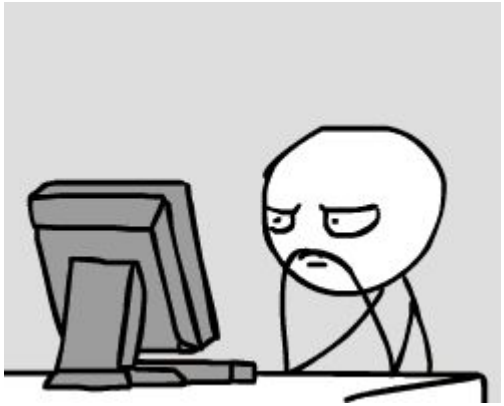
?

엄격한 비교 연산자, ===

Hmmteresting ...



대소비교 연산자는 어떨까?



```
<?php
var_dump("1000e-1" < "1"); // bool(false)
var_dump("1000e-4" < "1"); // bool(true)
```

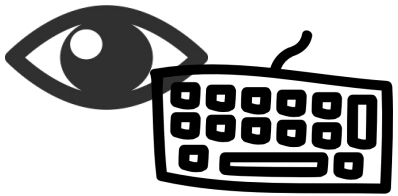

php
php
php
php

U MAD?

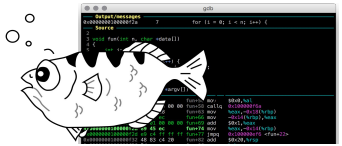
How to debug PHP

PHP 소스코드 다운 → 소스코드 수정 후 컴파일 **(static debugging)**
불편함을 느끼고 GDB를 활용 → 좀 더 편하게 분석 **(dynamic debugging)**

불편함의 예시) 특정 로직에서 내부적으로 어떤 함수가 호출되는지 모를 때



+



You can download PHP source code

https://museum.php.net

I downloaded **7.0.30** version!

```
$ wget https://museum.php.net/php7/php-7.0.30.tar.gz
$ tar -xvf php-7.0.30.tar.gz

(you have to install gcc, g++, libxml2, libxml2-dev)

$ ./configure --disable-all --enable-debug
$ make
$ make install
```

모든 문제의 시작은 ..

문자형 타입의 변수를 숫자형 타입으로 변환하고 비교하면서 문제가 발생

'0.10' == '0.1' (True)

'15e0005' == '1500000' (True)

'1000e-1' == '100' (True)

'99e-2' == '0.99' (True)

'99e+2' == '9900' (True)

string to number (integer, double ..)

타입 변환은 어디에서 어떻게 이루어질까?

PHP Variable Structure

PHP 변수는 내부에서 zval 이라는 구조체로 처리된다.

```
typedef struct _zval_struct {
    zvalue_value value;             /* variable value */
    zend_uchar type;                /* value type */
    zend_uint refcount__gc;         /* reference counter */
    zend_uchar is_ref__gc;          /* reference flag */
} zval;
```

PHP Operators

```
+  -  *  /  %  .

==  !=  ===  !==  <  >  <=  >=

|  &  ^  <<  >>
```

```
int add_function(zval *result, zval *op1, zval *op2 TSRMLS_DC);                 /*  +  */
int sub_function(zval *result, zval *op1, zval *op2 TSRMLS_DC);                 /*  -  */
int mul_function(zval *result, zval *op1, zval *op2 TSRMLS_DC);                 /*  *  */
int div_function(zval *result, zval *op1, zval *op2 TSRMLS_DC);                 /*  /  */
int mod_function(zval *result, zval *op1, zval *op2 TSRMLS_DC);                 /*  %  */
int concat_function(zval *result, zval *op1, zval *op2 TSRMLS_DC);              /*  .  */
int bitwise_or_function(zval *result, zval *op1, zval *op2 TSRMLS_DC);          /*  |  */
int bitwise_and_function(zval *result, zval *op1, zval *op2 TSRMLS_DC);         /*  &  */
int bitwise_xor_function(zval *result, zval *op1, zval *op2 TSRMLS_DC);         /*  ^  */
int shift_left_function(zval *result, zval *op1, zval *op2 TSRMLS_DC);          /*  << */
int shift_right_function(zval *result, zval *op1, zval *op2 TSRMLS_DC);         /*  >> */
int boolean_xor_function(zval *result, zval *op1, zval *op2 TSRMLS_DC);         /* xor */

int is_equal_function(zval *result, zval *op1, zval *op2 TSRMLS_DC);            /*  == */
int is_not_equal_function(zval *result, zval *op1, zval *op2 TSRMLS_DC);        /*  != */
int is_identical_function(zval *result, zval *op1, zval *op2 TSRMLS_DC);        /* === */
int is_not_identical_function(zval *result, zval *op1, zval *op2 TSRMLS_DC);    /* !== */
int is_smaller_function(zval *result, zval *op1, zval *op2 TSRMLS_DC);          /*  <  */
int is_smaller_or_equal_function(zval *result, zval *op1, zval *op2 TSRMLS_DC); /*  <= */
```

compare_function

php-src/Zend/zend_operators.c

==, !=, <, >, <=, >= 와 같은 비교 연산자를 사용하면
두 값을 비교해주는 compare_function 함수를 호출한다.

```
ZEND_API int ZEND_FASTCALL compare_function(zval *result, zval *op1, zval *op2)
{
    int ret;
    int converted = 0;
    zval op1_copy, op2_copy;
    zval *op_free, tmp_free;
```

compare_function

php-src/Zend/zend_operators.c

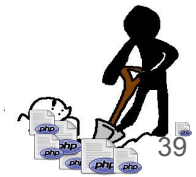
switch/case 로 아래와 같은 조건을 거치며
비교하는 타입마다 다르게 처리하고 있다. (case 부분 코드는 생략)

```
switch (TYPE_PAIR(Z_TYPE_P(op1), Z_TYPE_P(op2))) {
    case TYPE_PAIR(IS_LONG, IS_LONG):
    case TYPE_PAIR(IS_DOUBLE, IS_LONG):
    case TYPE_PAIR(IS_LONG, IS_DOUBLE):
    case TYPE_PAIR(IS_DOUBLE, IS_DOUBLE):
    case TYPE_PAIR(IS_ARRAY, IS_ARRAY):
    case TYPE_PAIR(IS_NULL, IS_TRUE):
    case TYPE_PAIR(IS_STRING, IS_STRING):
```

compare_function → zendi_smart_strcmp

비교하는 대상이 둘 다 문자열이고 서로 다르다면 **zendi_smart_strcmp** 함수를 호출한다.

```
case TYPE_PAIR(IS_STRING, IS_STRING):
    if (Z_STR_P(op1) == Z_STR_P(op2)) {
        ZVAL_LONG(result, 0);
        return SUCCESS;
    }
    ZVAL_LONG(result, zendi_smart_strcmp(Z_STR_P(op1), Z_STR_P(op2)));
    return SUCCESS;
```



zendi_smart_strcmp → is_numeric_string_ex

zendi_smart_strcmp 함수에선 문자열이 숫자로 이루어져 있는지 체크하기 위해서
is_numeric_string_ex 라는 함수로 사용자가 입력한 문자열 2개를 넘겨줍니다.

```
ZEND_API zend_long ZEND_FASTCALL
zendi_smart_strcmp(zend_string *s1, zend_string *s2)
{
    int ret1, ret2;
    int oflow1, oflow2;
    zend_long lval1 = 0, lval2 = 0;
    double dval1 = 0.0, dval2 = 0.0;

 if ((ret1 = is_numeric_string_ex(s1->val, s1->len, &lval1, &dval1, 0, &oflow1))
 && (ret2 = is_numeric_string_ex(s2->val, s2->len, &lval2, &dval2, 0, &oflow2))) {
```

is_numeric_string_ex → _is_numeric_string_ex

```
static zend_always_inline zend_uchar
is_numeric_string_ex(const char *str, size_t length,  zend_long *lval,
double *dval, int allow_errors, int *oflow_info)
{
     if (*str > '9') {
          return 0;
     }
     return _is_numeric_string_ex(str, length, lval, dval, allow_errors, oflow_info);
}
```

_is_numeric_string_ex function

문자열이 숫자로 이루어져 있는지 확인 후
변수의 타입과 값을 숫자형 타입에 맞게 변환해주는 작업을 한다.

```
ZEND_API zend_uchar ZEND_FASTCALL
_is_numeric_string_ex(const char *str, size_t length, zend_long *lval,
double *dval, int allow_errors, int *oflow_info) /* {{{ */
{
    const char *ptr;
    int digits = 0, dp_or_e = 0;
    double local_dval = 0.0;
    zend_uchar type;
```

_is_numeric_string_ex (goto check_digits)

```
for(type = IS_LONG; !(digits >= MAX_LENGTH_OF_LONG && (dval || allow_errors == 1)); digits++, ptr++){
check_digits:
      if (ZEND_IS_DIGIT(*ptr)) {
            tmp_lval = tmp_lval * 10 + (*ptr) - '0';
            continue;
      } else if (*ptr == '.' && dp_or_e < 1) {
            goto process_double;
      } else if ((*ptr == 'e' || *ptr == 'E') && dp_or_e < 2) {
            const char *e = ptr + 1;

            if (*e == '-' || *e == '+') {
                  ptr = e++;
            }
            if (ZEND_IS_DIGIT(*e)) {
                  goto process_double;
            }
      }

      break;
}
```

_is_numeric_string_ex (goto check_digits)

check_digits 부분에서는 아래와 같이 지수 표현인지 검사를 거친 뒤
double로 변환하는 작업을 위해 process_double 부분으로 넘어간다.

```
else if ((*ptr == 'e' || *ptr == 'E') && dp_or_e < 2) {
    const char *e = ptr + 1;

    if (*e == '-' || *e == '+') {
        ptr = e++;
    }
    if (ZEND_IS_DIGIT(*e)) {
        goto process_double;
    }
}
```

123e+45

123e+45

123e+45

_is_numeric_string_ex (goto process_double)

process_double에서는 변수의 타입을 double형으로 바꿔주고
해당 변수의 값 또한 double로 변환하기 위해서 zend_strtod 함수를 호출한다.

```
process_double:
    type = IS_DOUBLE;

    if (dval) {
        local_dval = zend_strtod(str, &ptr);
    } else if (allow_errors != 1 && dp_or_e != -1) {
```

zend_strtod function

변수의 값을 string에서 double로 변환해주는 함수

- "3.14" (string) → 3.14 (double)
- "12345" → 12345.0

zend_strtod is similar to strtod.

- strtod 함수도 실수 형태 문자열을 double형 실수로 변환해주는 함수다.
- 아래의 결과도 True가 나오니 zend_strtod 함수는 잘못이 없어 보인다.

```
main() {
	char *s1 = "99e-2";
	char *s2 = "0.99";
	char *ptr;

	if(strtod(s1, &ptr) == strtod(s2, &ptr))
		printf("True");
	else
		printf("False");

	return 0;
}
```

"99e-2" == "0.99" (True)

그렇다면 문제점은 무엇일까?

1. 문자끼리 논리 비교 연산을 할 때 타입을 숫자형 타입으로 변환한다는 점

2. 〈, 〉, 〈=, 〉= 와 같이 크기를 비교하는 연산자는 아직 compare_function 함수를 호출하므로 타입 변환을 거치고 비교하는 느슨한 비교 연산의 특성을 그대로 가지고 있다.

타입캐스팅에 대한 개발자의 자세

1. ==, != 연산이 꼭 필요한 상황이 아니라면 === 혹은 !== 을 사용한다.
2. 항상 타입 캐스팅을 생각하고, 예의주시하며 코딩한다.
3. PHP는 주기적으로 업데이트하여 최신 버전을 유지한다.

믿었던 엄격한 비교 연산자의 배신

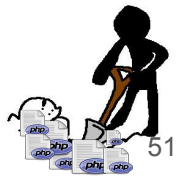
Strict comparison operator Trick

(Codegate 2016 Hack Nojam - Web prob)

```
[4294967296 => 'admin'] === [0 => 'admin']
```

5.4.0 - 5.4.43, 5.5.0 - 5.5.26, 5.6.0 - 5.6.10

TRUE



First trip is end!

PHP Filesystem functions Trick

What is Filesystem function?

rename() ⟶ 파일 이동 및 이름 바꾸기

unlink() ⟶ 파일을 삭제해주는 함수

fopen() ⟶ 파일 입출력 처리를 할 때 파일을 선택해주는 함수

⟶ file, file_get_contents, file_put_contents ..

etc functions (mkdir, rmdir, touch ..)

rename, unlink

(php-src/ext/file.c)

```
PHP_FUNCTION(rename)
{
    char *old_name, *new_name;
    size_t old_name_len, new_name_len;
    zval *zcontext = NULL;
    php_stream_wrapper *wrapper;
```

```
PHP_FUNCTION(unlink)
{
    char *filename;
    size_t filename_len;
    php_stream_wrapper *wrapper;
```

rename, unlink → php_error_docref

```
if (!wrapper->wops->rename) {
    php_error_docref(NULL, E_WARNING, "%s wrapper does not support renaming",
    wrapper->wops->label ? wrapper->wops->label : "Source");
    RETURN_FALSE;
}
```

```
if (!wrapper->wops->unlink) {
    php_error_docref(NULL, E_WARNING, "%s does not allow unlinking",
    wrapper->wops->label ? wrapper->wops->label : "Wrapper");
    RETURN_FALSE;
}
```

wrapper = php_stream_locate_url_wrapper

```
if (zend_parse_parameters(ZEND_NUM_ARGS(), "p|r", &filename, &filename_len, &zcontext) == FAILURE) {
      RETURN_FALSE;
}

context = php_stream_context_from_zval(zcontext, 0);
wrapper = php_stream_locate_url_wrapper(filename, NULL, 0);    <-- 여기 (here)
```


```
if (zend_parse_parameters(ZEND_NUM_ARGS(), "p|r", &filename, &filename_len, &zcontext) == FAILURE) {
      RETURN_FALSE;
}

context = php_stream_context_from_zval(zcontext, 0);
wrapper = php_stream_locate_url_wrapper(filename, NULL, 0);  <-- 여기 (here)
```

php_stream_locate_url_wrapper

php-src/main/stream/streams.c

php_stream_locate_url_wrapper 함수는 처리하는 path의 **wrapper**를
검사하고 각 **wrapper** 에 맞게 처리하기 위한 함수다.

```
PHPAPI php_stream_wrapper *php_stream_locate_url_wrapper
(const char *path, const char **path_for_open, int options)
{
      HashTable *wrapper_hash = (FG(stream_wrappers) ? FG(stream_wrappers) : &url_stream_wrappers_hash);
      php_stream_wrapper *wrapper = NULL;
      const char *p, *protocol = NULL;
      int n = 0;

      if (path_for_open) {
            *path_for_open = (char*)path;
      }
```

data wrapper는 **php_stream_rfc2397_wrapper** 라는 구조로 처리가 되는데
php_stream_rfc2397_wrapper → wops 안에있는 **unlink, rename** 에는 **0**이 들어있다.

```
php_register_url_stream_wrapper("php", &php_stream_php_wrapper);
php_register_url_stream_wrapper("file", &php_plain_files_wrapper);
php_register_url_stream_wrapper("glob", &php_glob_stream_wrapper);
php_register_url_stream_wrapper("data", &php_stream_rfc2397_wrapper);
php_register_url_stream_wrapper("http", &php_stream_http_wrapper);
php_register_url_stream_wrapper("ftp", &php_stream_ftp_wrapper);
```

```
gdb-peda$ print wrapper
$5 = (php_stream_wrapper *) 0xed8f70 <php_stream_rfc2397_wrapper>

gdb-peda$ print wrapper->wops
$6 = (php_stream_wrapper_ops *) 0xed8fa0 <php_stream_rfc2397_wops>

gdb-peda$ print wrapper->wops->unlink
$7 = (int (*)(php_stream_wrapper *, const char *, int, php_stream_context *)) 0x0
```

php_stream_rfc2397_wrapper

```
{
  wops = 0xed8fa0 <php_stream_rfc2397_wops>,
  abstract = 0x0,
  is_url = 0x1
}
```

php_stream_rfc2397_wops

```
{
  stream_opener = 0x6c2a40 <php_stream_url_wrap_rfc2397>,
  stream_closer = 0x0,
  stream_stat = 0x0,
  url_stat = 0x0,
  dir_opener = 0x0,
  label = 0xbb4dc1 "RFC2397",
  unlink = 0x0,
  rename = 0x0,
  stream_mkdir = 0x0,
  stream_rmdir = 0x0,
  stream_metadata = 0x0
}
```

wrapper->wops->unlink == 0

wrapper->wops->rename == 0

data wrapper (RFC 2397)

http://php.net/manual/en/wrappers.data.php

그렇다, 애초에 unlink와 rename 함수는
data wrapper를 지원하지 않는다고 공식 문서에 써있었다..

Supports unlink()	No
Supports rename()	No
Supports mkdir()	No
Supports rmdir()	No

unlink, rename is not supported data wrapper

정보

리눅스에서는 "**data:a**" 라는 이름의 파일을 생성할 수 있다. (윈도우 X)

unlink, rename does not work.

unlink('data:a');

unlink(): **RFC2397** does not allow unlinking in php shellcode on line 1

rename('data:a', 'abc');

rename(): **RFC2397** wrapper does not support renaming in php shell code on line 1

```php
function move_to_backup($original_path, $ext) {
    $filename = random_str().'.'.$ext;
    $backup_path = 'backup/'.$filename;

    rename($original_path, $backup_path);
    if(file_exists($backup_path)) {
        return true;
    }
    return false;
}
```

Example probs

https://github.com/munsiwoo/PHP-Trick-Trip/

```php
<?php
error_reporting(0);

if($_GET['file']) {
    $file = basename($_GET['file']);
    if(file_exists($file)) die('bye');
    include $file;
}

show_source(__FILE__);
```

fopen function trick (detail)

http://blog.munsiwoo.kr/

PHP Trick 은 어떻게 찾을까요?

- Zend Engine Auditing / Fuzzing
- PHP Fuzzing
- Feel 100%

References

https://phpinternals.net
http://www.phpinternalsbook.com
http://php.net/manual/en/internals2.php
https://www.slideshare.net/ax330d/dphpwas
https://museum.php.net
https://github.com/bowu678/php_bugs
http://yousha.blog.ir/

Thank you!

Kafuuchin0, howdays, SHGroup, NextLine

QnA

mun.xiwoo@gmail.com
blog.munsiwoo.kr

Thank you!