VB2018 paper: From Hacking Team to hacked team to...?

Filip Kafka

ESET, Slovakia

Table of contents

Abstract

Hacking Team first came under the spotlight of the security industry following its damaging data breach in July 2015. The leaked data revealed several zero-day exploits being used and sold to governments, and confirmed suspicions that *Hacking Team* had been doing business with oppressive regimes. But what happened to *Hacking Team* after one of the most famous hacks of recent years?

Hacking Team's flagship product, the Remote Control System (RCS), was detected in the wild at the beginning of 2018 in 14 different countries, including some of those that had previously criticized the company's practices. In this paper, we will present the evidence that convinced us that the new, post-hack *Hacking Team* samples can be traced back to a single group – not just any group, but *Hacking Team*'s developers themselves.

Furthermore, we will share previously undisclosed insights into *Hacking Team*'s post-leak operations, including the targeting of diplomats in Africa, uncover the digital certificates used to sign the malware, and share details of the distribution vectors used to target the victims. We will compare the functionality of the post-leak samples with that of

the leaked source code. To help other security researchers, we'll provide tips on how to efficiently extract details from these newer *VMProtect*-packed RCS samples. Finally, we will show how *Hacking Team* sets up companies and purchases certificates for them.

Introduction

Since being founded in 2003, the Italian spyware vendor *Hacking Team* has gained notoriety for selling surveillance tools to governments and their agencies across the world. The capabilities of its flagship product, the Remote Control System (RCS), include extracting files from a targeted device, intercepting emails and instant messaging, as well as remotely activating a device's webcam and microphone. The company has been criticized for selling these capabilities to authoritarian governments [1] – an allegation it has consistently denied.

When *Hacking Team* itself suffered a damaging hack in July 2015 [2], the reported use of RCS by oppressive regimes was confirmed [3].

The 400GB of leaked internal data included:

- the spyware source code
- the spyware for different platforms
- five zero-day exploits: 1x *Windows LPE*, 3x *Adobe Flash*, 1x *Adobe Reader*
- a UEFI rootkit
- evidence of selling an injection proxy for performing various MitM attacks
- the once-secret list of customers
- the pricelist
- internal communications

Due to the severity of the leak, *Hacking Team* was forced to ask its customers to suspend all use of RCS, and was left facing an uncertain future.

The security community has been keeping a close eye on the company's efforts to get back on its feet. With both the source code and a ready-to-use builder leaked, it came as no surprise when cybercriminals started reusing the spyware. This was the case in January 2016, when Callisto Group reused the source code in one of their campaigns [4]. Recent reports have revealed that in June 2016, *Hacking Team* received funding from a mysterious investor with ties to Saudi Arabia [5].

Discovery of post-leak samples

In the early stages of our investigation, the *Citizen Lab* provided us with RCS samples used in 2016 and 2017, which led to the discovery of a version of the spyware currently being used in the wild and signed with a previously unseen valid digital certificate.

Our further research uncovered several more samples of *Hacking Team*'s spyware created after the 2015 hack, all being slightly modified compared to variants released before the source code leak.

The samples were compiled between September 2015 and October 2017. We have deemed these compilation dates to be authentic, based on *ESET* telemetry data indicating the appearance of the samples in the wild within a few days of those dates.

Unpacking the samples

All the samples are packed with *VMProtect*, a commercial anti-piracy protector, which was also the case with pre-leak samples. We notified *VMProtect*'s developers and asked them to blacklist the licence used to pack spyware, but no action was taken.

In this section, we will explain how we unpacked the samples of modified *Hacking Team* spyware.

Extracting details from the VMProtect-packed samples

There are two approaches – the first is intended to extract only some (valuable) details from the sample, such as the C&C; the second approach is to fully unpack the sample, including rebuilding the IAT (Import Address Table). The first approach is obviously easier and quicker, but to be able to fully analyse the samples, the second approach is needed.

First approach

For the first approach, the sample is run and after some time, when the sample unpacks itself, the process is dumped. Then the C&C or other information can be searched for in the dump. Indeed, this is not a sophisticated method, but it is important to mention this easy but still effective approach. For dumping you can use:

1. *Hacking Team*'s *VMProtect* dumper – a simple tool developed by *Hacking Team*'s developers, which runs the *VMProtect*-packed sample and dumps the process memory a few times after the sample unpacks itself.
2. Any of your favourite memory-dumping tools.

Second approach

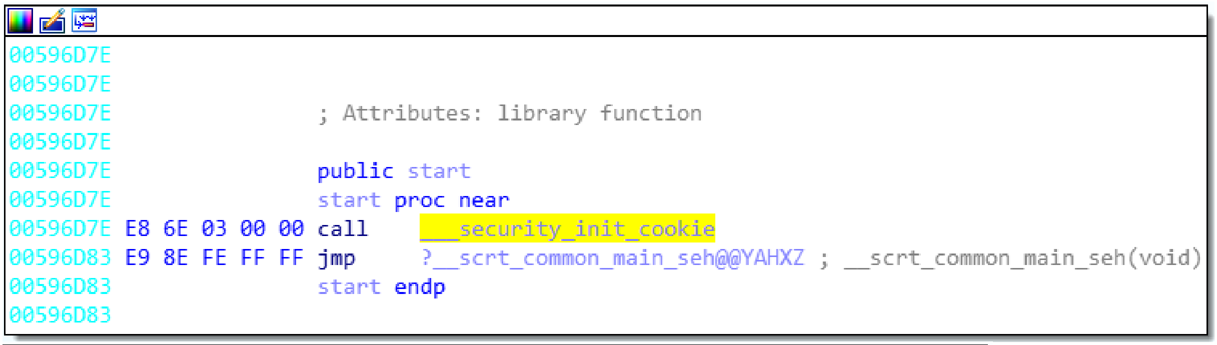
The second approach represents typical unpacking, and results in a fully working unpacked PE file. We unpacked the sample dynamically (i.e. by executing it).

This approach includes the following steps:

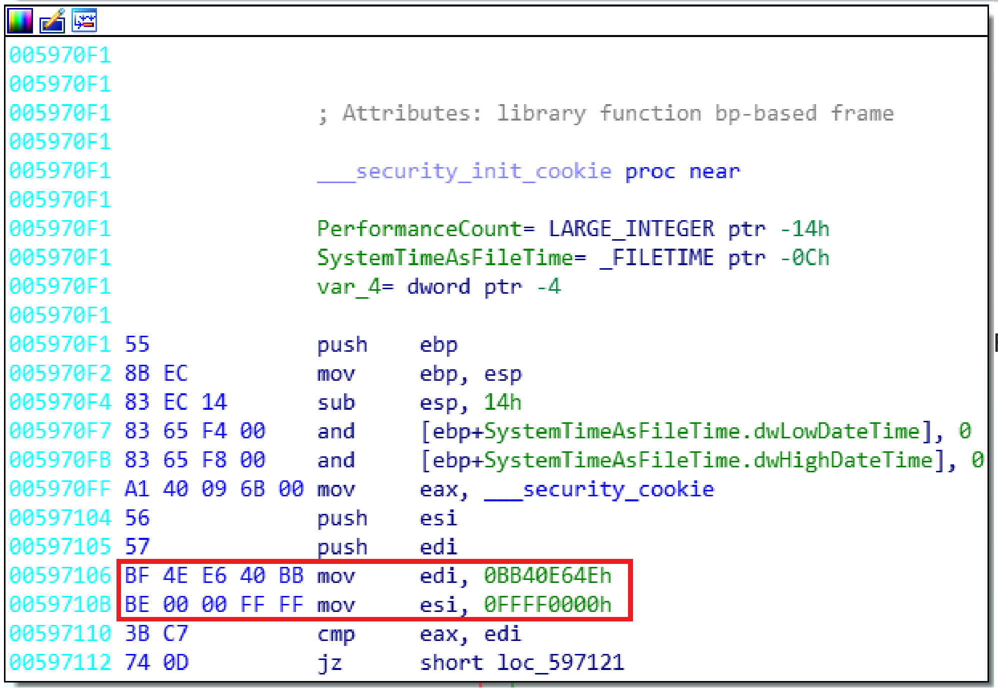
1. Run the sample and find the OEP (Original Entry Point).
2. If imports (calls to API) are wrapped, figure out the real API function and rewrite the wrapped call.
3. Dump and rebuild imports with typical tools.

The steps explained in detail:

1. As *VMProtect* users can choose various protection settings, including detection of virtual machine or debugger, the difficulty of this part depends on what protection settings are used. We will first explain how to do the unpacking when the protection is not set to detect a debugger or a virtual machine, then we will extend our approach to bypass those detections.
    1. Run the sample inside the debugger. After some time, pause it. Then search in the memory for:
        1. Typical MSVC entry point. Programs compiled with *Microsoft* Visual Studio have very common code on startup, so looking for 'magic bytes' in the initialization of the security cookie (Figure 1) is enough to identify the OEP. Once the OEP has been identified, place a hardware breakpoint on the address and run the packed sample once again until it pauses at the OEP.
        2. If it wasn't compiled with *Microsoft* Visual Studio, or it is still somehow masquerading, it is necessary to search for the code belonging to the original application.
    2. If the protection is set to detect a debugger, it is necessary to use dumping tools as explained before, because there are no plug-ins available to successfully hide a debugger from *VMProtect*.
2. Once the OEP is found, there might still be a problem with API functions – usually, *VMProtect* puts a 'wrapper' on them. This means that it won't call APIs directly, but instead it calls code in the .vmp section which computes the address of the API, pushes it on the stack, and finally returns to it. This wrapper makes work more difficult for typical IAT rebuilders. In order to get rid of the wrapper, it is necessary to solve the address of the API and then rewrite the wrapped call to the real address of the API. Solving the address of the API can be done either by emulation or by execution until it returns from the .vmp section to the section with imports (usually the address starts with 0x7f....).
3. When the OEP is found and imports are in the original form without the wrapper, the process can be dumped and the IAT rebuilt using typical tools like *OllyDump*, *Scylla* or *ImpRec*.



```
00596D7E
00596D7E
00596D7E                    ; Attributes: library function
00596D7E
00596D7E                    public start
00596D7E                    start proc near
00596D7E E8 6E 03 00 00 call    ___security_init_cookie
00596D83 E9 8E FE FF FF jmp     ?__scrt_common_main_seh@@YAHXZ ; __scrt_common_main_seh(void)
00596D83                    start endp
00596D83
```



```
005970F1
005970F1
005970F1                    ; Attributes: library function bp-based frame
005970F1
005970F1                    ___security_init_cookie proc near
005970F1
005970F1                    PerformanceCount= LARGE_INTEGER ptr -14h
005970F1                    SystemTimeAsFileTime= _FILETIME ptr -0Ch
005970F1                    var_4= dword ptr -4
005970F1
005970F1 55                 push    ebp
005970F2 8B EC              mov     ebp, esp
005970F4 83 EC 14           sub     esp, 14h
005970F7 83 65 F4 00        and     [ebp+SystemTimeAsFileTime.dwLowDateTime], 0
005970FB 83 65 F8 00        and     [ebp+SystemTimeAsFileTime.dwHighDateTime], 0
005970FF A1 40 09 6B 00 mov     eax, ___security_cookie
00597104 56                 push    esi
00597105 57                 push    edi
00597106 BF 4E E6 40 BB mov     edi, 0BB40E64Eh
0059710B BE 00 00 FF FF mov     esi, 0FFFF0000h
00597110 3B C7              cmp     eax, edi
00597112 74 0D              jz      short loc_597121
```

Figure 1: Typical bytes on entry point in programs compiled with Microsoft Visual Studio.

Analysis of the post-leak samples

Distribution and targets

As for the distribution vector of the post-leak samples we analysed, in at least two cases, we detected the spyware in an executable file disguised as a PDF document. The names of the files suggest the malware was spread via spear-phising emails sent to high-profile targets such as diplomats:

- 'Requirement for Diplomatic Passport Service.pdf.t.exe'
- 'Note Verbale No 00023AM-ADD2017 du 17 janvier 2017 .exe'
- 'Petition 2017 rasdt...t.exe'
- 'rawshi nawaxoy harim kurdstan.exe'

Our systems have detected these new *Hacking Team* spyware samples in 14 countries. We choose not to name the countries in order to prevent potentially incorrect attributions based on these detections, since the geo-location of the detections doesn't necessarily reveal anything about the origin of the attack.

Architecture and functionality

The malware has two stages – Scout (first stage) and Soldier or Elite (second stage; regular and premium version). The second stage, the actual payload, is deployed after a few initial checks carried out by Scout.

In the post-leak samples we analysed, Scout and Soldier had the following functionality:

Scout (version 28):

- Installs itself, checks if other instances are already running
- Performs AV-bypassing tricks
- Collects basic information about the computer
- Checks for possible upgrades of itself / Soldier / Elite

Soldier (version 1025):

- Collected data is packed, encrypted and stored in the registry and later sent to the C&C server
- Proper memory management, error handling
- Capabilities: steal data from clipboard, steal data from social networks, steal passwords and other data from browsers, take screenshots, activate camera, determine geolocation based on Wi-Fi networks, record *Skype* calls, record keystrokes, monitor mouse clicks, schedule uninstallation of itself
- Example of a configuration embedded in the file:

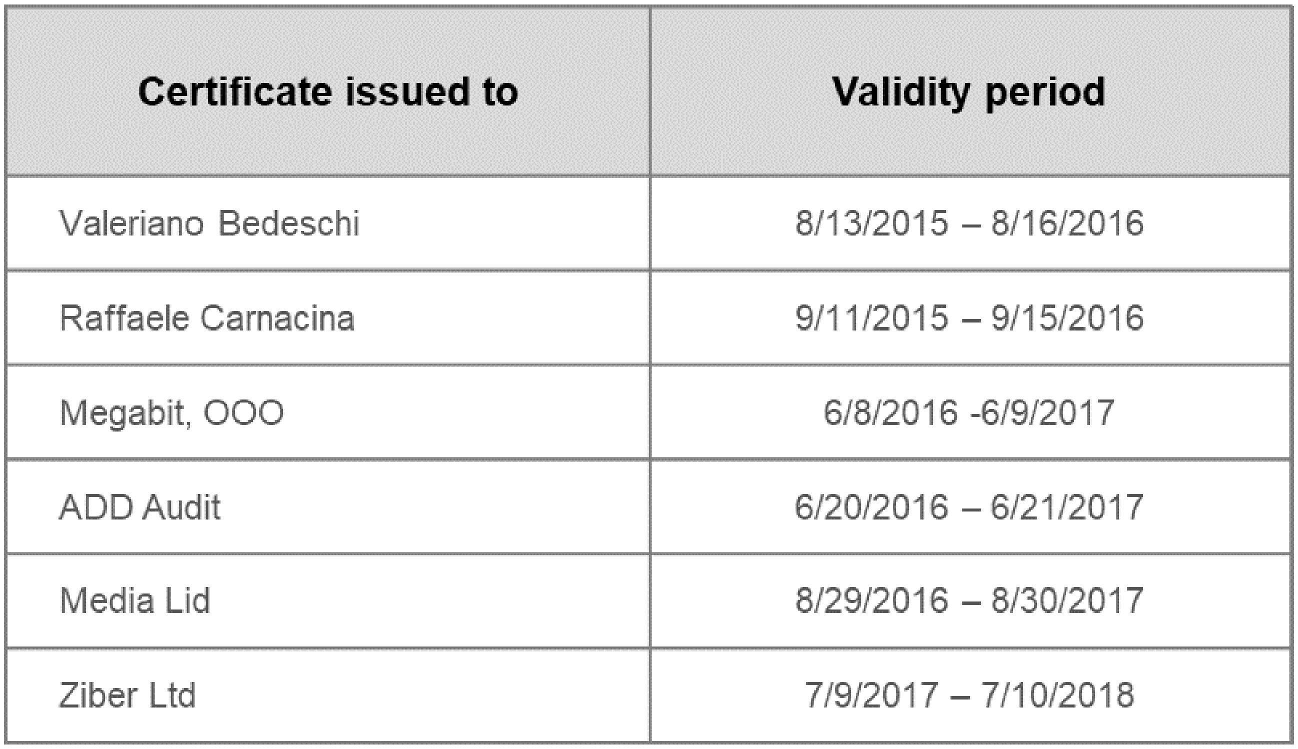
```
{"camera":{"enabled":false,"repeat":0,"iter":0},"position":{"enabled":false,"repeat":0},"screenshot":
{"enabled":true,"repeat":120},"photo":{"enabled":false},"file":{"enabled":false},"addressbook":{"enabled"
:false},"chat":{"enabled":false},"clipboard":{"enabled":false},"device":{"enabled":true},"call":{"enabled"
:false},"messages":{"enabled":false},"password":{"enabled":false},"keylog":{"enabled":false},"mouse":
{"enabled":false},"url":{"enabled":false},"sync":{"host":"149.154.153.223","repeat":120},"uninstall":
{"date":null,"enabled":false}}
```

Attribution

Further analysis led us to conclude that all the analysed post-leak samples can be traced back to a single group, rather than being isolated instances of diverse actors building their own versions from the leaked *Hacking Team* source code.

Certificates

One indicator supporting this is the sequence of digital certificates used to sign the samples – we found six different certificates issued in succession. Four of the certificates were issued by *Thawte* to four different companies, and two are personal certificates issued to Valeriano Bedeschi (*Hacking Team* co-founder) and someone named Raffaele Carnacina.



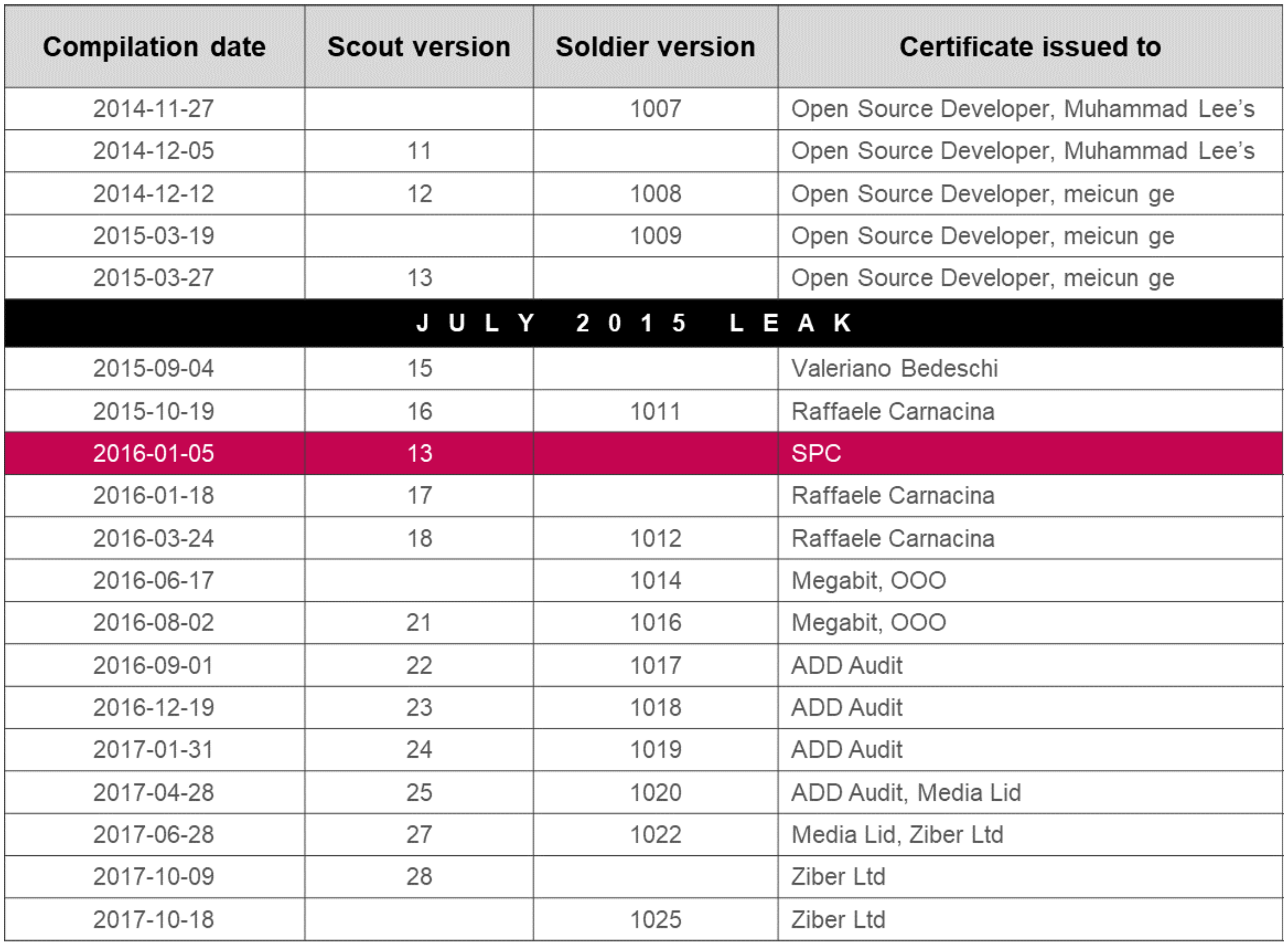
Certificate issued to	Validity period
Valeriano Bedeschi	8/13/2015 – 8/16/2016
Raffaele Carnacina	9/11/2015 – 9/15/2016
Megabit, OOO	6/8/2016 -6/9/2017
ADD Audit	6/20/2016 – 6/21/2017
Media Lid	8/29/2016 – 8/30/2017
Ziber Ltd	7/9/2017 – 7/10/2018

Figure 2: The sequence of digital certificates used to sign the post-leak samples. Samples signed by Valeriano Bedeschi were most likely used for testing purposes, as the C&C was set to an internal network (172.16.1.206).

Versioning

The versioning observed in the analysed samples continues where *Hacking Team* left off before the breach, and follows the same patterns. *Hacking Team*'s habit of compiling Scout and Soldier consecutively, and often on the same day, can also be seen across the newer samples.

Figure 3 shows the compilation dates, versioning and certificate authorities of *Hacking Team Windows* spyware samples seen between 2014 and 2017. Post-leak samples with renewed versioning begin with the sample signed by Valeriano Bedeschi. Reuse of the leaked source code by Callisto Group is marked in red.



Compilation date	Scout version	Soldier version	Certificate issued to
2014-11-27		1007	Open Source Developer, Muhammad Lee's
2014-12-05	11		Open Source Developer, Muhammad Lee's
2014-12-12	12	1008	Open Source Developer, meicun ge
2015-03-19		1009	Open Source Developer, meicun ge
2015-03-27	13		Open Source Developer, meicun ge
JULY 2015 LEAK			
2015-09-04	15		Valeriano Bedeschi
2015-10-19	16	1011	Raffaele Carnacina
2016-01-05	13		SPC
2016-01-18	17		Raffaele Carnacina
2016-03-24	18	1012	Raffaele Carnacina
2016-06-17		1014	Megabit, OOO
2016-08-02	21	1016	Megabit, OOO
2016-09-01	22	1017	ADD Audit
2016-12-19	23	1018	ADD Audit
2017-01-31	24	1019	ADD Audit
2017-04-28	25	1020	ADD Audit, Media Lid
2017-06-28	27	1022	Media Lid, Ziber Ltd
2017-10-09	28		Ziber Ltd
2017-10-18		1025	Ziber Ltd

Figure 3: Hacking Team Windows spyware samples seen between 2014 and 2017.

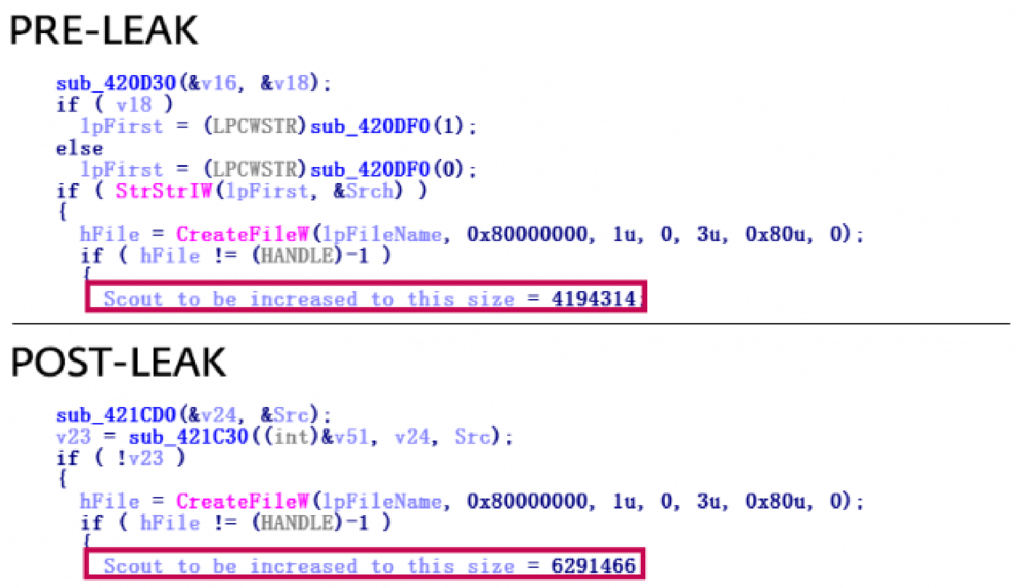
Modifications in line with Hacking Team development habits

Furthermore, our research has confirmed that the changes introduced in the post-leak updates were made in line with *Hacking Team*'s own coding style. The changes are often found in places that indicate a deep familiarity with the code, and follow *Hacking Team*'s previously established development patterns (visible in the leaked source code).

Other than these specific changes, the majority of the code is without any modifications. It is highly improbable that some other actor – that is, other than the original *Hacking Team* developer(s) – would make changes in exactly these places when creating new versions from the leaked *Hacking Team* source code.

1. Difference in Scout Startup file size

As shown in Figure 4, one of the subtle differences between the pre-leak and post-leak samples is a difference in Startup file size. When Scout installs on the system, it copies itself into the Windows Startup folder and appends random data to the end of the binary. This trick was used as an evasive technique against one anti-virus product. Before the leak, the copied file was padded to occupy 4MB. In the post-leak samples, this file copy operation is padded to 6MB.



```
sub_420D30(&v16, &v18);
if ( v18 )
  lpFirst = (LPCWSTR)sub_420DF0(1);
else
  lpFirst = (LPCWSTR)sub_420DF0(0);
if ( StrStrIW(lpFirst, &Srch) )
{
  hFile = CreateFileW(lpFileName, 0x80000000, 1u, 0, 3u, 0x80u, 0);
  if ( hFile != (HANDLE)-1 )
  {
    Scout to be increased to this size = 4194314
```

```
sub_421CD0(&v24, &Src);
v23 = sub_421C30((int)&v51, v24, Src);
if ( !v23 )
{
  hFile = CreateFileW(lpFileName, 0x80000000, 1u, 0, 3u, 0x80u, 0);
  if ( hFile != (HANDLE)-1 )
  {
    Scout to be increased to this size = 6291466
```

Figure 4: Startup file size copy changed from 4MB pre-leak to 6MB post-leak.

2. Improved random number generation

Scout samples from before the leak used the GetTickCount and Rand functions to generate random numbers for increasing their size by appending the numbers at the end of the binary. In the post-leak samples, the number generation has been improved by using the CryptGenRandom API function. Only if this function fails are the previous functions used.

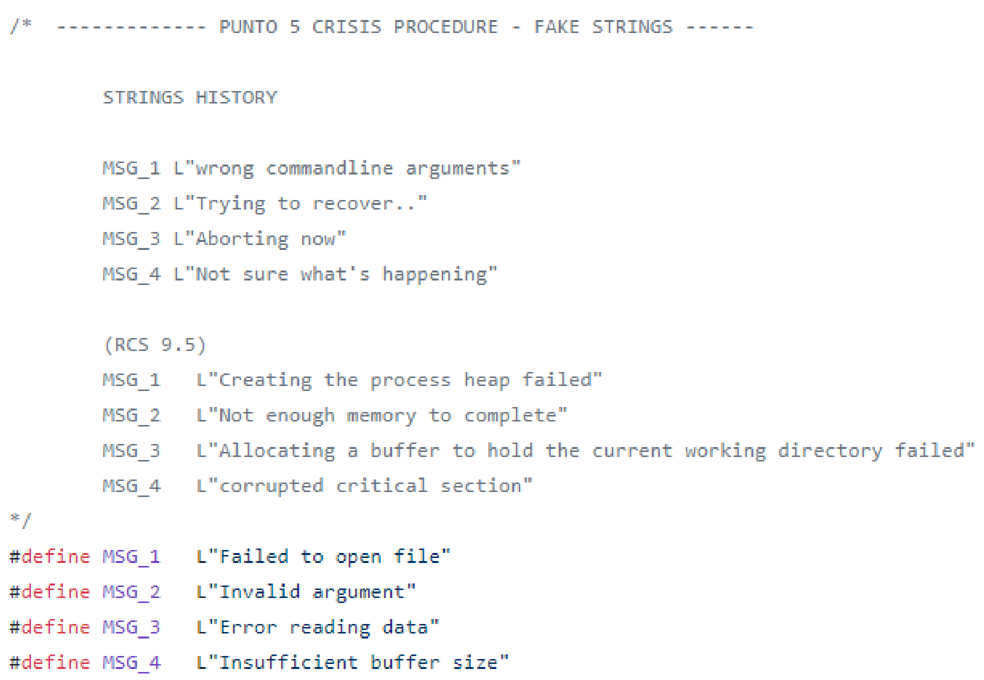
3. Changes in MySleep function

Hacking Team developers used their own MySleep function – in the samples from before the leak, it was implemented for bypassing sleep patches in many sandboxes. It consisted of the GetCurrentThread and WaitForSingleObject *Windows* API functions. In the post-leak samples, the MySleep function is still present, but now comprises the CreateEvent, WaitForSingleObject and CloseHandle *Windows* API functions.

4. Change in fake error message strings

The pre-leak samples contain fake error message strings to be used in case the spyware is run with a system process ID (i.e. 0 or 4). A process executed by a regular user would never have these ID values, so this trick might be aimed at sandboxes or emulators.

The content of the fake error messages had been changed regularly before the leak; the history of these changes was revealed in the leaked source code (Figure 5).



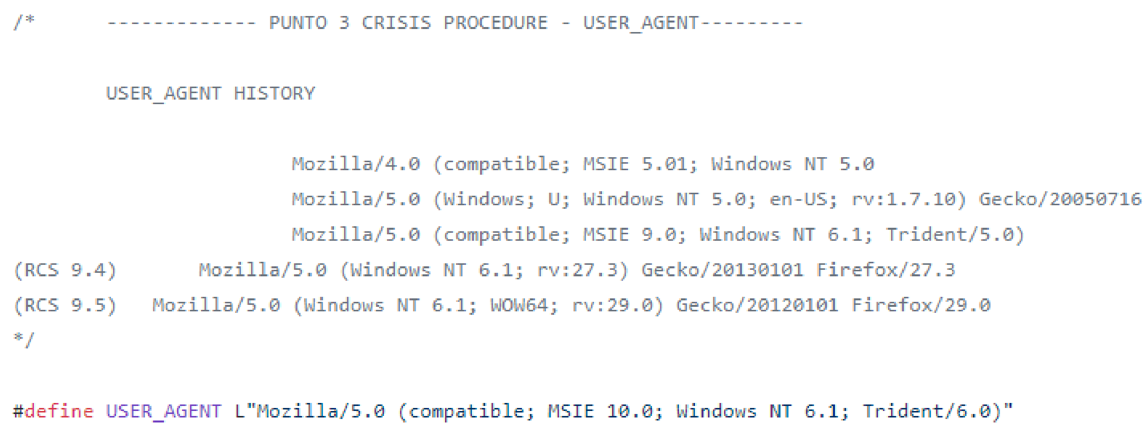
```
/*  ------------- PUNTO 5 CRISIS PROCEDURE - FAKE STRINGS ------

        STRINGS HISTORY

        MSG_1 L"wrong commandline arguments"
        MSG_2 L"Trying to recover.."
        MSG_3 L"Aborting now"
        MSG_4 L"Not sure what's happening"

        (RCS 9.5)
        MSG_1   L"Creating the process heap failed"
        MSG_2   L"Not enough memory to complete"
        MSG_3   L"Allocating a buffer to hold the current working directory failed"
        MSG_4   L"corrupted critical section"
*/
#define MSG_1   L"Failed to open file"
#define MSG_2   L"Invalid argument"
#define MSG_3   L"Error reading data"
#define MSG_4   L"Insufficient buffer size"
```

Figure 5: Fake strings history in the Scout leaked source code. RCS 9.5 includes Scout versions 11 and 12.

In the post-leak samples, the strings have been changed again, this time to the message 'Aborting now'.

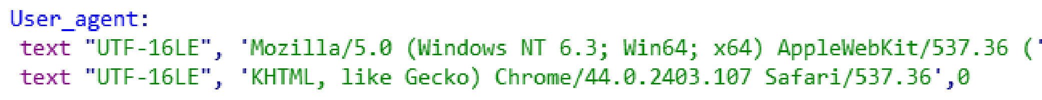
5. Change in user agents

Compared to pre-leak samples, some of the parameters of the user-agent string used for HTTP protocol when communicating with the C&C server are different in the newer samples. (See Figures 6 to 10.)



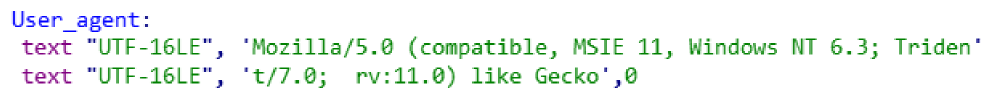
```
/*      ------------- PUNTO 3 CRISIS PROCEDURE - USER_AGENT---------

        USER_AGENT HISTORY

                        Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0
                        Mozilla/5.0 (Windows; U; Windows NT 5.0; en-US; rv:1.7.10) Gecko/20050716
                        Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0)
(RCS 9.4)       Mozilla/5.0 (Windows NT 6.1; rv:27.3) Gecko/20130101 Firefox/27.3
(RCS 9.5)   Mozilla/5.0 (Windows NT 6.1; WOW64; rv:29.0) Gecko/20120101 Firefox/29.0
*/

#define USER_AGENT L"Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; Trident/6.0)"
```

Figure 6: User-agent string history in the Scout leaked source code. RCS 9.4 includes Scout version unknown; RCS 9.5 includes Scout versions 11 and 12.



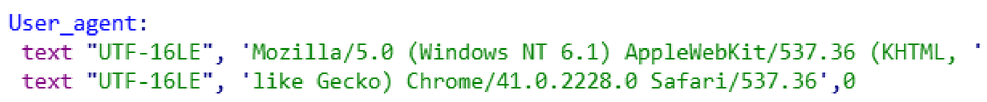
```
User_agent:
 text "UTF-16LE", 'Mozilla/5.0 (Windows NT 6.3; Win64; x64) AppleWebKit/537.36 ('
 text "UTF-16LE", 'KHTML, like Gecko) Chrome/44.0.2403.107 Safari/537.36',0
```

Figure 7: Altered user-agent string in Scout version 15.



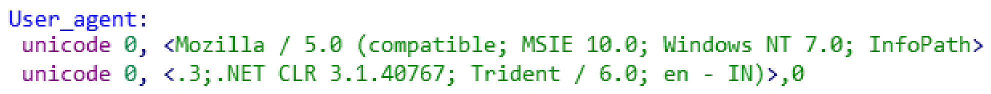
```
User_agent:
 text "UTF-16LE", 'Mozilla/5.0 (compatible, MSIE 11, Windows NT 6.3; Triden'
 text "UTF-16LE", 't/7.0;  rv:11.0) like Gecko',0
```

Figure 8: Altered user-agent string in Scout version 17.



```
User_agent:
 text "UTF-16LE", 'Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, '
 text "UTF-16LE", 'like Gecko) Chrome/41.0.2228.0 Safari/537.36',0
```

Figure 9: Altered user-agent string in Scout version 22.

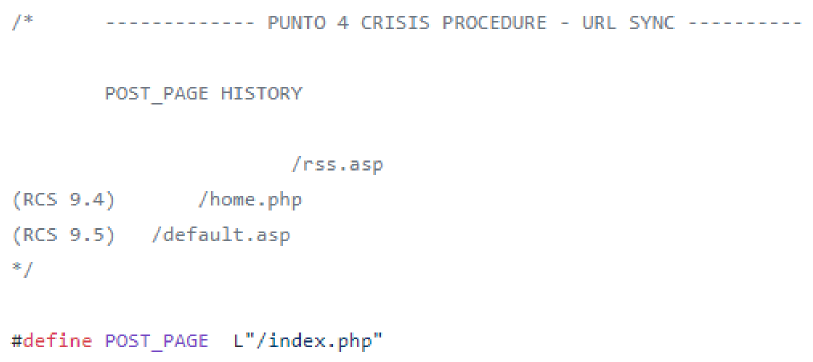


```
User_agent:
 unicode 0, <Mozilla / 5.0 (compatible; MSIE 10.0; Windows NT 7.0; InfoPath>
 unicode 0, <.3;.NET CLR 3.1.40767; Trident / 6.0; en - IN)>,0
```

Figure 10: Altered user-agent string in Scout version 28.

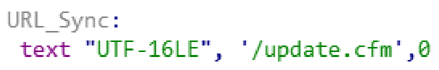
6. Changes in strings used for C&C sync

The URL path used for HTTP protocol when communicating with the C&C server is another part of the code that was frequently changed in the pre-leak samples. In the post-leak samples, these paths also vary from sample to sample.



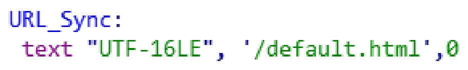
```
/*      ------------- PUNTO 4 CRISIS PROCEDURE - URL SYNC ----------


        POST_PAGE HISTORY


                        /rss.asp
(RCS 9.4)       /home.php
(RCS 9.5)   /default.asp
*/


#define POST_PAGE  L"/index.php"
```

Figure 11: Strings used from C&C sync in the Scout leaked source code. RCS 9.4 includes Scout version unknown.; RCS 9.5 includes Scout versions 11 and 12.



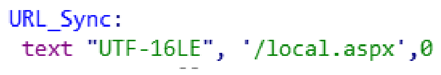
```
URL_Sync:
 text "UTF-16LE", '/update.cfm',0
```

Figure 12: Alter URL Sync string in Scout version 15.



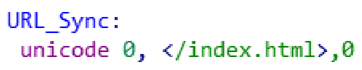
```
URL_Sync:
 text "UTF-16LE", '/default.html',0
```

Figure 13: Alter URL Sync string in Scout version 17.



```
URL_Sync:
 text "UTF-16LE", '/local.aspx',0
```

Figure 14: Alter URL Sync string in Scout version 18.



```
URL_Sync:
 unicode 0, </index.html>,0
```

Figure 15: Alter URL Sync string in Scout version 28.

Conclusion

None of these indicators, by themselves, represents conclusive evidence of *Hacking Team*'s renewed activity. However, viewing them as a whole lets us claim with high confidence that, with one obvious exception, the post-leak samples we've analysed are indeed the work of the *Hacking Team* developers, and not the result of source code reuse by unrelated actors, such as in the case of Callisto Group in 2016.

References

[1] The Citizen Lab. Mapping Hacking Team's "Untraceable" Spyware. February 2014. https://citizenlab.ca/2014/02/mapping-hacking-teams-untraceable-spyware/.

[2] Franceschi-Bicchierai, L. Spy Tech Company 'Hacking Team' Gets Hacked. Motherboard. July 2015. https://motherboard.vice.com/en_us/article/gvye3m/spy-tech-company-hacking-team-gets-hacked.

[3] Hern, A. Hacking Team hacked: firm sold spying tools to repressive regimes, documents claim. The Guardian. July 2015. https://www.theguardian.com/technology/2015/jul/06/hacking-team-hacked-firm-sold-spying-tools-to-repressive-regimes-documents-claim.

[4] F-Secure Labs. Callisto Group. April 2017. https://www.f-secure.com/documents/996508/1030745/callisto-group.

[5] Franceschi-Bicchierai, L. Hacking Team Is Still Alive Thanks to a Mysterious Investor From Saudi Arabia. Motherboard. January 2018. https://motherboard.vice.com/en_us/article/8xvzyp/hacking-team-investor-saudi-arabia.

Download PDF

Latest articles:

Cryptojacking on the fly: TeamTNT using NVIDIA drivers to mine cryptocurrency

TeamTNT is known for attacking insecure and vulnerable Kubernetes deployments in order to infiltrate organizations' dedicated environments and transform them into attack launchpads. In this article Aditya Sood presents a new module introduced by…

Collector-stealer: a Russian origin credential and information extractor

Collector-stealer, a piece of malware of Russian origin, is heavily used on the Internet to exfiltrate sensitive data from end-user systems and store it in its C&C panels. In this article, researchers Aditya K Sood and Rohit Chaturvedi present a 360…

Fighting Fire with Fire

In 1989, Joe Wells encountered his first virus: Jerusalem. He disassembled the virus, and from that moment onward, was intrigued by the properties of these small pieces of self-replicating code. Joe Wells was an expert on computer viruses, was partly…

Run your malicious VBA macros anywhere!

Kurt Natvig wanted to understand whether it's possible to recompile VBA macros to another language, which could then easily be 'run' on any gateway, thus revealing a sample's true nature in a safe manner. In this article he explains how he recompiled…

Dissecting the design and vulnerabilities in AZORult C&C panels

Aditya K Sood looks at the command-and-control (C&C) design of the AZORult malware, discussing his team's findings related to the C&C design and some security issues they identified during the research.

Bulletin Archive