Analyzing an Off-the-Shelf Surveillance Software: Hacking Team Case Study

Stanislav Špaček, Pavel Čeleda, Martin Drašar, Martin Vizváry

{spaceks, celeda, drasar, vizvary}@ics.muni.cz

Institute of Computer Science
Masaryk University
Brno, Czech Republic

Abstract

In July 2015, a major distributor and developer of covert surveillance tools, Italian company Hacking Team, has been hacked. Due to the attack, nearly 400 GB of internal data leaked on sharing networks. The data contained the latest version of the surveillance software named Galileo, including full technical and user documentation. We use this opportunity to examine key features of surveillance software that was designed for governmental agencies and its specification was kept secret. In this paper, we deploy the system in an isolated virtual environment and test its behavior during a surveillance operation. We use collected information to classify the advancement level of Galileo among similar mass-spread malware and the advanced persistent threats tools. With the hindsight of nearly two years, it is also possible to evaluate the impact the data leak had.

Keywords:

Hacking Team, Remote Control System, KYPO Cyber Range, Advanced Persistent Threat, malware builder kit

1 Introduction

End-to-end encryption is now a de-facto standard for online communication and transferred messages are increasingly difficult to eavesdrop. This not only serves for better protection of private data, but it may be misused to organize criminal activities in relative secret. Law enforcement agencies with sufficient resources and proficient personnel can try to overcome this issue and put together a specialized team to develop their own tools for cyber espionage. But if resources are limited, cheaper and simpler solution is preferred. Several versions of off-the-shelf surveillance software are available for purchase directly from developers.

One of these developers is an Italian company Hacking Team. Its surveillance software is called Remote Control System Galileo and according to Hacking Team it should be able to circumvent end-to-end encryption and access encrypted files. More specific information concerning the software is not publicly available and even the list of Hacking Team's customers is kept secret. Because other surveillance software developers keep similar secrecy, it is difficult to determine what is such software really capable of and creates room for speculations of potential misuse.

In July 2015, Hacking Team's internal network was hacked by an unknown hacker and a large amount of data was downloaded and made public. It included Remote Control System (RCS) Galileo 9.6, the latest version of Hacking Team's RCS at the time, together with full technical documentation and user manuals. This presented a unique opportunity to examine the system in detail, check if it uses any novel approaches to defeating cyber security measures and learn how to adapt to them to better defend against attackers in the future. Furthermore, the analysis of the RCS could provide an insight into how the more complex malware might evolve. For these reasons, we analyzed the RCS both in a static environment and during a simulated operation and the gathered information was used to compare its capabilities to other similar malware groups – publicly available malware builder kits, botnets and previously studied APT cases.

The remainder of this paper is organized into five sections. Section 2 describes related articles that report on the Hacking Team's leaked data. Section 3 shows the RCS architecture. Section 4 presents testing of the RCS functions during a simulated operation in a fully isolated environment of the KYPO Cyber Range of Masaryk University [1]. Section 5 sums up the properties and functions of the RCS to compare it with similar groups of malware. Section 6 concludes the paper with lessons learned.

2 Related Work

Nearly 400 GB of leaked data that allowed analysis of the Hacking Team's RCS were shared by an unknown individual on P2P networks on the 5th of July 2015. Most of the archives including the P2P source were already abandoned but some parts may still be found. At the time of writing, the full data is still accessible [2]. Technical analysis of the data was made by Greenwood shortly after the data leak [3]. Greenwood examined the RCS in a series of blog posts, where he focused on code analysis of several modules like a virtualization detection or an integrated remote "kill switch".

The Hacking Team's RCS was a target of research for Citizen Lab of the University of Toronto. Citizen Lab had been publishing a series of articles between 2012 to 2015, predating the July 2015 data leak, aimed to reveal the perpetrators of cyber-attacks on individuals considered dissidents and members of press. Usage of the RCS Galileo was confirmed by Citizen Lab in all these attacks and involvement of several different governments, e.g., Sudan and Ethiopia was suspected [4]. The Hacking Team's business data contained in the data leak later confirmed these suspicions.

This research was later used by Marczak in his doctoral dissertation [5]. His work was focused on the issue of defending individuals against targeted cyber-attacks initiated by nation-states. During his fieldwork among potential targets, all attempted attacks were analyzed and categorized. Besides the RCS Galileo, similar surveillance software, FinFisher – developed by company Gamma [6], was detected in several attack attempts. Based on his research, Marczak then proposes a defensive approach that should detect and defeat similar attacks.

A survey to identify vulnerabilities to social engineering infection vectors among potential victims of targeted cyber-attack was made by Marczak and Paxson [7]. This family of infection vectors is particularly important while defending against RCS, since most of its vectors use these principles to infiltrate target device. The survey concludes that despite free availability of malware detection tools, they were not widely used among the surveyed population.

A case study regarding Hacking Team and its operations on 0-day exploit market was conducted by Tsyrklevitch [8]. Tsyrklevitch used leaked emails between Hacking Team and several 0-day exploit providers to reconstruct processes Hacking Team set up to be supplied by 0-day exploits as reliably as possible. Providing RCS with 0-day exploits was a difficult effort, because they become virtually useless as soon as they are made public. At the time of the data leak, Hacking Team was using six exclusive 0-day exploits for various devices.

Burkart and McCourt wrote a case study of Hacking Team studying the political economy for a surveillance software product [9]. They focused on legal status of online surveillance technologies designed for lawful interception. This term refers to gathering personal information or other infringement of personal privacy rights by governmental agencies, but only after such an action was approved by legal institution, e.g., court.

3 Architecture of Remote Control System

Most of the knowledge presented in this section originates from technical specification and user guides for RCS Galileo. These guides were intended as a customer knowledge base and contain enough information for any computer system administrator to set up the RCS and manage its operation. The documentation was considered private by Hacking Team and became public after the data leak.

To describe the RCS, we use the same terminology Hacking Team used in their documentation. The term *target* denotes the individual that should be surveyed. This individual generally owns one or more *target devices*. The *operation* groups target and its corresponding target devices that are under surveillance. An *investigation* is generally composed of several operations and it is administered by dedicated team of investigators. The operation is started by compiling and installing an *agent* into target device. The agent is a tailored spyware, created by operators through *compilation* process and designed for use with the specific target device it is compiled for.

3.1 Components Structure of Remote Control System

The RCS is composed of five nodes where each of them manages a different functionality. Their connection schema is described on Figure 1.

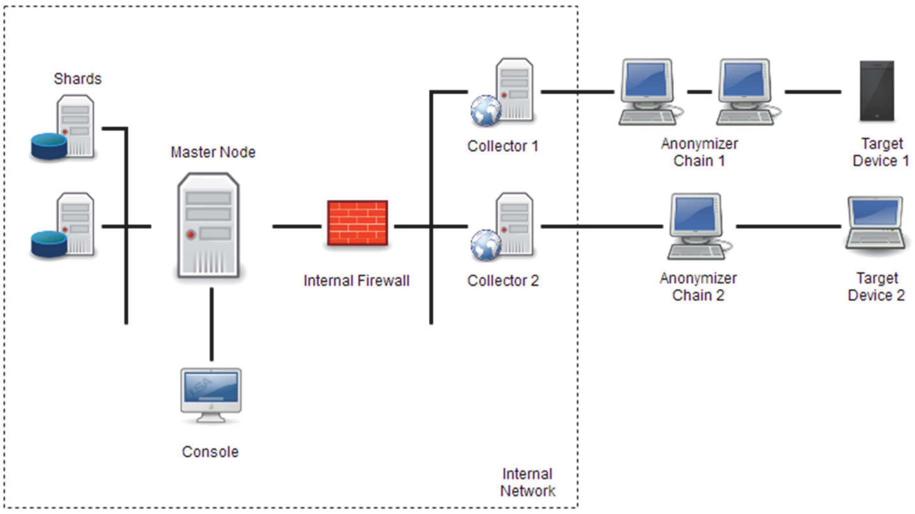

Figure 1: The connection between nodes of RCS Galileo.

The core server is the *Master Node*. It runs an instance of the RCS and allows remote access to it through administrative graphical interface. All operations carried out in the RCS are conducted through this interface on the Master Node. The data extracted from target devices in various investigations flow through this node to be processed. The Master Node collects the data, presents it to its operators through the graphical interface and allows it then to be stored in the RCS internal database.

The *Collector* acts as an edge element for the RCS internal network. It communicates on the Internet side with an Anonymizer chain and on the internal network with the

Master Node. Its function is twofold: first, it protects the Master Node by communicating with it through an internal firewall, second it alleviates load on the Master node by tending to assigned Anonymizer chains.

The *Anonymizer* is a proxy that serves to obfuscate the identity of the collector it is assigned to, should someone try to trace the agent back to its source. Without such proxy, the operator could be identified by the Collector's IP address through simple network traffic analysis. To ensure a greater level of secrecy, Anonymizers are recommended to organize into chains. The Anonymizers may be provided by one or more external suppliers.

The remaining two system components are the *Shard* and the *Console*. The Shard is a "clone" of the Master Node that should be able to act as a backup if the Master Node fails, but it is an optional module. If one or more Shards are present in a system configuration, the internal database is distributed between them to share the load. The Console node may be represented by any desktop or laptop with the RCS console client installed. The client connects to the Master Node's administrative interface and a console user can access RCS functions according to their assigned system role.

The data extraction and synchronization of configuration between the Master Node and an agent is always initiated by the agent as an outgoing connection from the target device. The connection passes through a chain of Anonymizers and its corresponding Collector. The Master Node assigns extracted data to the relevant operations and alerts the investigators. In reply to the agent, the Master Node may send new instructions or activate new functions on the target device, if those were planned earlier by the operator. The RCS behavior in this regard is similar to a botnet client-server communication.

3.2 Node Hardware Requirements

Hacking Team specified minimal hardware requirements in their documentation. These requirements are rather high, but are aimed for medium or large installations. The system must have significant performance surplus to cope with many parallel investigations. During our testing, we found that for a small scale operation the requirements are much lower. The recommended specification for each node can be found in the Table 1. Minimal tested requirements are specified in brackets, if the node was part of the configuration we tested.

Node	RAM (GB)	HDD System (GB)	HDD Data (GB)	Operating System
Master Node	96 (32)	300 (40)	2400 (-)	WS*
Shard	96 (-)	300 (-)	2400 (-)	WS*

Collector	16 (2)	600 (20)	-	WS*
Anonymizer	0,256 (1)	10 (10)	-	CentOS 6
Console	8 (-)	320 (-)	-	Windows/OS X

Minimal tested requirements are in brackets.
*Windows Server 2008 R2 SP1 64bit Enterprise Edition

Table 1: Hardware requirements of RCS nodes.

3.3 Administrative Interface

The RCS administrative interface is accessible from anywhere in the internal network through the console. The console runs in Adobe Air runtime environment and can be installed on any personal computer with Windows or OS X operating system. The whole interface is point-and-click and for the more complex functions the system offers system wizards that guide the user through the entire process.

The interface is designed for concurrent work on a large amount of independent investigations. Their number is theoretically only limited by available memory and data storage. To allow such parallelism, the RCS has strictly defined user authorization. Besides administrators with unrestricted access to all functions and collected data, there are a number of other roles. Each user registered to the RCS is assigned one or more roles and is a member of one or more investigation teams. The teams may share relevant information with others or restrict access to data specific for their investigation. Even inside the team, its members do not have access to all collected evidence or to all functions, e.g., the only role that can compile and apply agents to target devices is the technician. To assist in investigations, the RCS uses its own modelling language to map relations between system entities. These entities comprise targets, target devices and even other investigations.

3.4 Malware Component of RCS Galileo

While the main part of the RCS is situated in an internal network, it would be useless without agents, essentially a targeted spyware, that act as its eyes and ears. The agent infiltrates the target device and collects data by activating device's various functions. The data is then synchronized with the Master Node. When all the desired data is extracted, the agent is silently purged from the target device. It goes through four phases during its life cycle.

The first phase is a compilation, where the operator specifies what the infection vector and surveillance functions should be. At the beginning, the operator chooses the target device and the system runs the compilation wizard modified to contain only functions and infection vectors available for this device's configuration. The selection is based mainly on the type of device (desktop, tablet etc.) and its operating system. The compilation may be executed in two ways. In the basic mode, the operator chooses from simple preset surveillance functions like periodic recording from devices camera, downloading messages, e-mails or other data. In the advanced mode, the functions may be customized for more specific actions by provided modelling language. The operator may specify a function based on entities Event, Action and Module. Generic objects like Timer are also available. Such an advanced function might look like this:

> *ON EVENT the target device is located on specific GPS coordinates* (Event)
> *ACTIVATE microphone* (Action and Module)
> *FOR 10 minutes* (Timer)

The next phase is the infiltration phase. The agent is installed into the target device through the chosen infection vector. The RCS collection totaled nine distinct infection vectors. All common and most of unusual operating systems for desktop and mobile devices were supported at the time of the data leak. Social engineering can be considered the most prominent one. The system offers a number of ways to propagate agent through e-mails, SMS messages and web links, but the operator has to devise a backstory to make the target cooperate. An easier way may be through so-called melted application, where the agent is installed along a legitimate application. An agent may be compiled even for offline devices, but infiltration and data extraction requires physical access to the target device. Two of the infection vectors require special actions to be undertaken. The network injector is a man-in-the-middle device provided by Hacking Team that can push the agent into the web traffic for example as a fake flash plugin. The injector has to be installed on the ISP level and thus requires its cooperation. As a last resort, a 0-day exploit could be used. Hacking Team granted access to them only per request and only in special cases. If the request was granted, the exploit was accessible only for a limited time. These precautions were aimed on extending the exploit's lifespan.

Infection vectors in the leaked RCS version:

- Installation package – a simple binary file to be executed on a target device.
- USB installation – a hidden file executed automatically after a USB drive is connected.
- Melted application – the agent is injected into a harmless application installer.
- Network injection – the agent is distributed by Hacking Team's network injector.
- Offline installation – a simple on-site installation. Requires physical access.

- QR code – the agent is downloading into a target device by scanning a special QR code.
- Silent installer – a binary file, which after execution installs agent silently.
- WAP push – a target device is sent a message with link to the agent.
- Exploit – depends on actual available 0-day exploits. Has to be consulted with Hacking Team support.

The third phase is the data collection phase. When the agent is installed in the target device, it begins to silently collect data as per its configuration. The agent can access files, calendar appointments, browser history; it can read stored passwords or key-log new ones. In most operating systems, it gives the RCS operator full control over the infected device. During the data collection phase, the agent periodically, at the specified time, sends its archive to the Master Node. With this synchronization, it may also receive other instructions.

The last phase is the extraction phase. When the data collection is complete, the operator closes the operation and the system sends all associated agents a kill command through the synchronization channel. When an agent receives this command, it terminates all activity and removes itself from a device.

The agent uses stealth techniques to avoid detection in the target device during data collection phase. The emphasis on stealth is apparent already during the infiltration phase. The agent has three stages – *scout*, *soldier* and *elite*. The scout has very limited functionality, the soldier can access basic functions and the elite can use all surveillance functions accessible on the device. The initial infection is done by the scout, it collects basic information about the device, checks for antivirus and decides if it allows promoting the agent to higher stage. The promotion is then done during the synchronization. If the scout finds an unsupported antivirus or detects a virtualized environment, the promotion is blocked and the operator cannot override it. In such case, the surveillance has to be done by other means.

The agent also includes a so called crisis module that detects potentially dangerous changes in the target device. In its default configuration, it may remove itself if an antivirus is installed or temporarily halt synchronization if network traffic analysis tools are detected. Its actions may be tuned in the advanced mode of agent compilation.

4 Surveillance Operation in Isolated Environment

To test behavior of the RCS during investigation, we decided to deploy it in a virtual environment. The virtual environment was created in the KYPO Cyber Range [1]. For the purpose of our experiment, the RCS operated on machines below recommended specifications as shown in Table 1. The RCS was set up on minimal amount of machines: one of each Master Node, Collector, Anonymizer and Console. A target device was added into the virtualized environment on a separate network. The device emulated a common smartphone running an on older version of Android operating system (4.4.2). The network topology is described on Figure 2.

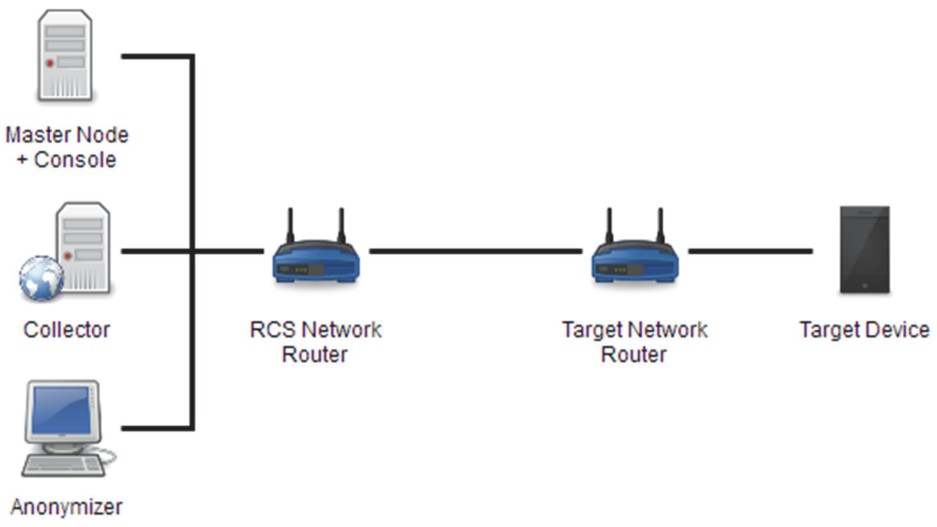

Figure 2: Network schema for simulated surveillance operation.

4.1 Setting Up the System

The setup process consisted of four phases: separate installation on all three nodes and initial configuration. The difficult part was the first phase, installation of the Master Node, but the process and all prerequisites were already described elsewhere [10]. To minimize the number of machines, the Master Node served simultaneously as the Console. In the second and third phase, the Collector and the Anonymizer were coupled with the Master Node. This process was also already explored in another guide [11]. The fourth phase was carried out by the system automatically, the operator should just check final inter-component connectivity through the administrative interface.

4.2 Agent Compilation

After setting the RCS up, we created some system roles to form a team and assigned ourselves a new investigation. The goal was to infiltrate the target Android device and explore what data we can extract. We created an entity for the device and entered a configuration wizard for agent compilation. In the first step, we chose what the surveyed data should contain – SMS messages, stored documents and a screenshot every 120 seconds. The synchronization with the Master Node was set to once per 10 minutes. In the second step, the wizard proposed infiltration vectors suitable for Android OS. We chose infiltration through a melted application and supplied a harmless apk file. Right before compilation, the RCS displayed a warning against uploading any of the files to virustotal [12] or sending them through public network unencrypted.

4.3 Infiltration of Target Device

To trick our target into downloading our malicious file, we conducted a spear phishing attack that contained a link to the infected application. During installation, the target was prompted for application permissions, so it is important to choose an application that corresponds with agent functions, so the requested permissions do not raise any suspicion. The chosen application should also sink into the spear phishing backstory. Once installed, the agent remains active in the target device even after the application it was associated to was uninstalled.

4.4 Data Extraction

After target device infiltration, the agent started collecting the defined data and contacted the assigned collector through its anonymizer chain at set intervals. When new data was received, the Master Node displayed an alert on dashboards of associated users, so they could review it. After review, the collected data was either archived in the provided operation data structure, or discarded as irrelevant.

The data collection process proved to be rather noisy when we used the Wireshark to observe the impact on the network traffic [13], but we consider this a problem only in a simulated environment. In a real device, the RCS synchronization would probably get lost among other auto-updating functions and would not immediately raise suspicion.

4.5 Exfiltration and Evaluation

When we received all the desired data, we decided to close the operation. The RCS automatically terminates all agents associated to closed investigations at the nearest synchronization. After the operation closure, no change was apparent in the target device. The agent stopped responding and no further network traffic from target device was detected.

The RCS interface proved efficient for individual cyber espionage operations. We did not encounter any serious problems during the simple surveillance operation in our testing environment. The system was convenient for work in a small team, but would probably scale well even for larger operations.

We consider the tested hardware requirements as minimal even for such a simple scenario. We did not experience any significant slowdowns, but the interface sometimes felt unresponsive and actions took longer time to complete. If more consecutive operations are run, the RCS Master Node in particular should be assigned more resources.

5 Comparison

At the time of the data leak, it was unknown how advanced the system might be in comparison with the already established malware. This section aims to compare RCS Galileo with similar kinds of malware already in use in July 2015 and to identify new approaches it brought. With the hindsight of nearly two years it is even possible to conclude if either these approaches were incorporated in recent malware, or if the data leak had modest effect on malware development.

When classified among other malware groups, the RCS behaves partly as spyware and partly as a multilayered botnet. Furthermore, the style of RCS operation displays similarities with APT. As Table 2 shows, the RCS agent's life cycle fits the life cycle of an APT attack, as described by Mandiant in its analysis of Chinese APT1 espionage report [14]. The RCS interface for agent compilation is also equivalent to malware builder kits, which can be bought on specialized online marketplaces.

For our comparison, we chose four categories to cover all important malware properties: infection vectors, provided functions on target device, stealth techniques and backend services.

5.1 Infection Vectors

The total number of nine distinct infection vectors provided by RCS can be considered high, even though not all were available for any operating system. Several vectors for special cases were included, namely the network injector, the offline installation for devices that reside only on internal networks, and somewhat limited access to 0-day exploits. The rest of the vectors required some level of cooperation of the target and thus relied on social engineering. It is apparent that despite the amount of RCS infection vectors, there is not much in the way of new techniques of device infiltration, possibly except the network injector.

Phase	Advanced Persistent Threat Life Cycle	Remote Control System Galileo Life Cycle
1.	Initial reconnaissance	Initial reconnaissance
2.	Weak spot identification	Agent compilation
3.	Establishing foothold	Scout level agent infiltration
4.	Privilege escalation	Agent promotion
5.	Internal reconnaissance	-
6.	Maintaining presence	Target surveillance
7.	Complete mission	Agent deactivation

Table 2: Similarities between life cycles of APT and RCS.

Social engineering techniques were already widely used by mass spread malware and targeted spear-phishing often acted as a prelude to a full scale APT attack. Attacks through 0-day exploits were also sometimes encountered in both APT and mass spread malware [15]. Offline infection vectors, while practically useless for mass spread malware, already proved their conditional usefulness in APT. During the APT attack – Stuxnet – that disrupted the Iranian nuclear program, an offline vector through a USB flash drive was used to infiltrate even isolated parts of the network [16].

On the other hand, the network injector provided a reliable infection vector that did not need any target input. Its ability to inject the agent into network traffic was especially useful if the target was using low security wi-fi network. For this purpose, Hacking Team provided the injector with WEP and WPA security breaking tools, but even then, the injector still depended on a weak or a misconfigured security and required physical presence in the same location as the target.

5.2 Surveillance Functions

The collection of surveillance functions provided by RCS was, similarly to infection vectors, rather large. In most cases, the RCS operator gained the administrator privileges for the infected system, so they could collect any data on device and access any of its modules.

Basic functions accessible on the device were comparable to any mass spread spyware, but the real strength of RCS was in the ability to combine basic functions like taking a photo with camera or activating microphone during advanced agent compilation. The operator could specify exact conditions that would trigger the desired function, chain functions and schedule them into an exact timetable. This activity is supported by interface wizards so designing complex functions is not difficult even for in-experienced operator. It is the most significant advantage over malware builder kits and from otherwise highly adaptable APT attacks, where the operator needs to code his own malware for specific purpose.

5.3 Stealth Capabilities

Stealth was apparently one of the main goals during RCS development and use. Several stealth functions are implemented and the whole surveillance process is subordinate to keeping the infiltration secret. The RCS stealth techniques and motivation to stay undetected are more similar to an APT than to mass spread malware. For mass spread malware, discovery means just a minor setback, for RCS and APT though, the discovery of an agent could ruin the whole investigation and bring unwanted publicity to the operator or directly to Hacking Team.

One of the more novel approaches was the staged infection – scout/soldier/elite – that RCS used to avoid detection by antivirus software. It is usually too slow to be used for mass spread malware that needs to be distributed as fast as possible during its short life cycle, but it still might fit into some special cases. The APT attacks use similar process during initial infiltration, but not as streamlined as the three stages of RCS. Even the crisis module in RCS agent has an equivalent. It could be observed in mass spread malware – while the agent's strategy is "run if threatened", the malware often tries to attack, circumvent or disable the antivirus software.

The most obvious disadvantage the RCS has when compared to APT is the impossibility to change the hardcoded system behavior. During an investigation, it sometimes may be worth to take a risk and infiltrate protected target. The off-the-shelf surveillance software won't allow this, since modifying its core behavior by operators is not allowed.

5.4 Management Interface

The management interface is probably the most advanced part of the RCS. It provides system role administration, data processing and target and agent management. Because the RCS is intended for governmental agencies that can't provision their own cyber espionage APT-like groups, the interface contains guides for less experienced operators – manuals, tooltips and wizards to walk them through more complex processes. Its worst disadvantage when compared with APT is the inability of customization for special operations. The interface cannot be changed in any manner.

If the included guides were insufficient or if any problems arose, Hacking Team provided customer support that would help to resolve them. The RCS was also getting automatic updates whenever a new feature was added. These services were provided for additional fee charged periodically. Such a model could be adapted to fit mass spread malware. The rentals of botnets for DDoS attacks or bitcoin mining might serve as an example.

A business model similar to RCS appeared recently – ransomware-as-a-service called Satan [17]. This malware allows the potential attackers to register a free account and then access a simple configuration interface. They set their own bitcoin account there for payments and choose their desired infection vectors. While the attackers handle distribution, the payments and further development and updates are maintained by the author, who takes a cut of collected ransom. This model could have been inspired by RCS distribution and support.

The RCS support came with a caveat – the customer had to countersign an end user license agreement, which aimed to prevent misuse. Had any unlawful surveillance been reported to Hacking Team, it reserved the right to immediately withdraw any support and revoke the RCS license. As the leaked data proved, Hacking Team was very benevolent and only decided to revoke the license when company's reputation was at stake. However, the data also confirmed the speculations of a "kill switch". Hacking Team could remotely disable any instance of their software. Such a hole in a cyber espionage agency's equipment may pose a serious risk for their users.

6 Conclusion

The vulnerabilities used by Hacking Team's 0-day exploits were fixed rapidly in all concerned application. Fingerprints of agents were added to antimalware filters and no large attack by a repurposed RCS was reported. The direct impact of the RCS Galileo leak on other malware was not significant, predominantly due to swift response. It heralded the end of Adobe Flash, since three out of five discovered 0-day exploits targeted it. Adobe Flash was already considered unsafe, and is now slowly being abandoned.

We found only two use cases where Galileo could affect malware development. The most noticeable advancement that could be implemented in other types of malware is Galileo's administrative interface. It is designed for simple use even by inexperienced users. The first use case of the interface is in conjunction with proper APT malware collection that could be modified on the system level. It is ability that RCS misses, but that would make APT attacks more accessible for teams with only a few cyber espionage experts. Such lower requirements might make APT attacks more common in the future. The second use case is for mass spread malware. A new type of ransomware, under the name Satan, was detected recently and features an administrative interface in a model that reminds the RCS Galileo. In this model, the malware author rents their malware for free and is rewarded by a percentage of his customers' gain. By being directly interested in customers' success, the next logical step would be providing them with product support in a similar fashion as Hacking Team supported its customers.

Our study has shown that when used properly, the RCS can provide access to most of the private data stored within the device under surveillance. We evaluated the processes of infiltration and data extraction and found out that while the RCS agent could infect most of the operating systems in use, it relied on techniques already well known from mass spread malware and APT attacks. Only a handful of novelty approaches was identified, mainly in the field of targeted surveillance. It resulted from our comparison, that while RCS poses threat if operated properly, the inventive approaches often seen in APT attacks are missing.

Acknowledgement

This research was supported by the Security Research Programme of the Czech Republic 2015 – 2020 (BV III/1 VS) granted by the Ministry of the Interior of the Czech Republic under no. VI20162019014 Simulation, detection, and mitigation of cyber threats endangering critical infrastructure.

References

[1] KYPO Cyber Range [online]. [Accessed on 8 April 2017]. Retrieved from https://www.kypo.cz/

[2] Transparency Toolkit, HT Index [online]. [Accessed on 10 April 2017]. Retrieved from https://ht.transparencytoolkit.org/

[3] Greenwood, J. Hacking Team RCS Analysis – (or is that 'Hacked Team'?) [online], 2015. [Accessed on 10 April 2017]. Retrieved from https://www.4armed.com/blog/hacking-team-rcs-analysis-hacked-team/

[4] Marczak, W. R et al. Mapping Hacking Team's "Untraceable" Software [online], 2014. [Accessed on 2 April 2017]. Retrieved from https://citizenlab.org/wp-content/uploads/2015/03/Mapping-Hacking-Team%E2%80%99s-_Untraceable_-Spyware.pdf

[5] Marczak, W. R. Defending Dissidents from Targeted Digital Surveillance (Doctoral dissertation), 2016.

[6] FinFisher [online]. [Accessed on 23 March 2017]. Retrieved from http://www.finfisher.com/FinFisher/index.html

[7] Marczak, W. R., and Paxson, V. Social Engineering Attacks on Government Opponents: Target Perspectives In *Proceedings on Privacy Enhancing Technologies*, pp. 152–164, 2017.

[8] Tsyrklevich, V. Hacking Team: A Zero-Day Market Case Study [online], 2015. [Accessed on 2 April 2017]. Retrieved from https://tsyrklevich.net/2015/07/22/hacking-team-0day-market/

[9] Burkart, P., and McCourt, T. The International Political Economy of the Hack: A Closer Look at Markets for Cybersecurity Software In *Popular Communications*, pp. 37–54, 2017.

[10] LookaPW. Installing the RCS Master Node, Console and Network Injector [online], 2015. [Accessed on 15 March 2017]. Retrieved from https://gist.github.com/LookaPW/2d2e7adbe131873dcc23

[11] Gr3ym0nk77. RCS Galileo Full Installation [online forum comment], 2015. [Accessed on 15 March 2017]. Message posted to https://forum.exploit.in/pda/index.php/t95775.html

[12] Virustotal [online]. [Accessed on 10 April 2017]. Retrieved from https://www.virustotal.com/

[13] Wireshark [online]. [Accessed on 10 April 2017]. Retrieved from https://www.wireshark.org/

[14] Mandiant. APT1: Exposing One of China's Cyber Espionage Units [online], 2013. [Accessed on 8 April 2017]. Retrieved from https://www.fireeye.com/content/dam/fireeye-www/services/pdfs/mandiant-apt1-report.pdf

[15] Tankard, C. Advanced Persistent Threats and How to Monitor and Deter Them In *Network Security*, pp. 16-19, 2011.

[16] Langner, R. To Kill a Centrifuge: A Technical Analysis of What Stuxnet's Creators Tried to Achieve [online], 2013. [Accessed on 2 April 2017]. Retrieved from: http://www.langner.com/en/wp-content/uploads/2013/11/To-kill-a-centrifuge.pdf

[17] Abrams, L. New Satan Ransomware Available Through a Ransomware as a Service [online], 2017. [Accessed on 9 April 2017]. Retrieved from https://www.bleepingcomputer.com/news/security/new-satan-ransomware-available-through-a-ransomware-as-a-service-/