GANON SPY SYSTEM:
AN ENTIRE ESPIONAGE PLATFORM

Salleras Soler, Xavier
Curs 2015-2016

Directors: Vanesa Daza i Matteo Signorini
GRAU EN ENGINYERIA EN SISTEMES AUDIOVISUALS

Treball de Fi de Grau
GANON SPY SYSTEM: AN ENTIRE ESPIONAGE PLATFORM

Xavier Salleras Soler

FINAL PROJECT

AUDIOVISUAL SYSTEMS ENGINEERING
ESCOLA SUPERIOR POLITÈCNICA UPF
2016

PROJECT DIRECTORS
VANESA DAZA
MATTEO SIGNORINI
Acknowledgements

I would like to thank all the professors that have given lectures in my degree which made me to see the importance of cyber-security and also made me decide to do this project. Thanks to Vanesa Daza and Matteo Signorini too, for giving me ideas and advices in my daily work. I also want to say thank you to those people who started five years ago being my classmates but now, they are my friends. Moltes gràcies també als meus germans per ajudar-me sempre que ho he necessitat i, sobretot, gràcies als meus pares per donar-me la oportunitat d'estudiar i fer que tot això hagi estat possible.
Abstract

In July of 2015, 400 GB of information about an Italian company called Hacking Team were filtered. Inside this amount of information we can find emails, conversations, passwords of different accounts… But also the source code of Remote Control System Galileo (RCS Galileo), the main program developed by the company. This program is able to spy different kind of devices, and was used by different governmental institutions to spy citizens around the world. The aim of this project is to show how this espionage system works, and after that, develop a new spy system based on the basics of RCS Galileo, in order to understand how a cyber-criminal could attack us, and to be able to develop new tools which could defend us from them.

Resumen

En Julio de 2015, se filtraron 400 GB de información sobre la compañía italiana llamada Hacking Team. Dentro de esta cantidad de información se podían encontrar emails, conversaciones, contraseñas de distintas cuentas… Pero también el código fuente del Remote Control System Galileo (RCS Galileo), el principal programa desarrollado por la compañía. Este programa es capaz de espiar diferentes tipos de dispositivos, y fue utilizado por diferentes instituciones gubernamentales para espiar ciudadanos de todo el mundo. El objetivo de este proyecto es demostrar como funciona este programa de espionaje y, después, desarrollar un nuevo sistema de espionaje basado en las principales características de RCS Galileo, con el fin de entender como un ciber criminal podría atacarnos, y de ser capaces de desarrollar nuevas herramientas con las cuáles poder defendernos de ellos.
Prologue

Since I was a child, I have had interest in knowing how technology works. Starting a degree in Audiovisual Systems Engineering made me to discover a lot of things about computers, networks, software… a lot of questions came to my mind when I started to learn about these topics: could an encrypted communication be intercepted? Could someone discover my WiFi’s password in less than ten seconds? Is it possible that while I am using my computer, someone is watching what I am doing?

Maybe when you get the knowledge to perform some of these actions, you get afraid of opening a computer because you realize that someone with bad intentions could be spying you, using similar technics. But then, you think a little bit about it and realize that you could learn more than them, in order to try to improve the systems that people use.

Usually, medias talk about hackers as cyber-criminals who use their skills to steal information, bank accounts, extort people… It is actually a description of a cracker. A hacker, as it was described before to be related with crackers, is a person who likes technology and has passion for discover how a system works, in order to improve its security. And that is why I am writing this. I would like to learn more about cyber-security, try to discover how espionage platforms work, and try to find ways to protect us from cyber-criminals.
# SUMMARY

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acknowledgements</td>
<td>iii</td>
</tr>
<tr>
<td>Abstract</td>
<td>v</td>
</tr>
<tr>
<td>Prologue</td>
<td>vii</td>
</tr>
<tr>
<td>Summary</td>
<td>ix</td>
</tr>
<tr>
<td><strong>1. INTRODUCTION</strong></td>
<td></td>
</tr>
<tr>
<td><strong>2. REMOTE CONTROL SYSTEM GALILEO</strong></td>
<td></td>
</tr>
<tr>
<td>2.1 What is RCS Galileo?</td>
<td>3</td>
</tr>
<tr>
<td>2.2 Galileo structure</td>
<td>3</td>
</tr>
<tr>
<td>2.3 How is an attack performed?</td>
<td>4</td>
</tr>
<tr>
<td>2.4 More espionage systems</td>
<td>5</td>
</tr>
<tr>
<td>2.5 What do we learn from here?</td>
<td>8</td>
</tr>
<tr>
<td><strong>3. CYBER-SECURITY METHODOLOGIES</strong></td>
<td>9</td>
</tr>
<tr>
<td>3.1 Server-client methodologies</td>
<td>9</td>
</tr>
<tr>
<td>3.2 Cryptography</td>
<td>9</td>
</tr>
<tr>
<td>3.2.1 Cryptography methodologies</td>
<td>9</td>
</tr>
<tr>
<td>3.2.2 Communication protocol</td>
<td>11</td>
</tr>
<tr>
<td>3.3 Usable security</td>
<td>13</td>
</tr>
<tr>
<td>3.3.1 Error messages</td>
<td>13</td>
</tr>
<tr>
<td>3.3.2 Passwords complexity</td>
<td>15</td>
</tr>
<tr>
<td>3.3.3 Two-factor authentication</td>
<td>15</td>
</tr>
<tr>
<td>3.4 Local Area Network attacks</td>
<td>16</td>
</tr>
<tr>
<td>3.4.1 Accessing the WiFi</td>
<td>16</td>
</tr>
<tr>
<td>3.4.2 Hacking the LAN</td>
<td>19</td>
</tr>
<tr>
<td>3.5 Security in Android OS</td>
<td>20</td>
</tr>
<tr>
<td><strong>4. DESIGN</strong></td>
<td>23</td>
</tr>
<tr>
<td>4.1 Installing the exploit</td>
<td>23</td>
</tr>
<tr>
<td>4.2 Exploit</td>
<td>24</td>
</tr>
<tr>
<td>4.3 Client console</td>
<td>25</td>
</tr>
<tr>
<td>4.4 Server</td>
<td>27</td>
</tr>
<tr>
<td><strong>5. RESULTS AND IMPROVEMENTS</strong></td>
<td>29</td>
</tr>
<tr>
<td>Bibliography</td>
<td>31</td>
</tr>
<tr>
<td>Appendix: user manual</td>
<td>33</td>
</tr>
</tbody>
</table>

ix
1. INTRODUCTION

Nowadays, cyber-security has become a field which people and companies care much more about than few years ago. Concepts as privacy or governmental espionages are in the news day after day since June of 2013, when Edward Snowden revealed loads of NSA secret documents, which demonstrated that USA spied many governs around the world.

It was July of 2015 when another big amount of information was filtered. This time were 400GB of internal information about Hacking Team¹ (from now HT), an Italian company dedicated to create espionage software, sold to many governmental institutions around the world and it is demonstrated that they used it in order to spy a lot of citizens, mainly in order to find terrorists and different kinds of criminals. From here we can enter into a controversial debate about ethical hacking, for example, is fair to spy someone with no proof of his innocence? Is justified that a governmental institution could possess this kind of tools? I will not enter in this discussion but I will try to answer another question which comes from the last one: is fine to teach the population how an espionage software works? Maybe it could help cyber-criminals to perform their evil acts. But if many more people is tough how to defense themselves against criminal cyber-attacks I think that we win a lot. So I think so, it is fine to educate people in this field, and it is the purpose of this project.

Inside the 400GB of information from HT we can find emails, contacts, passwords… but also the source code of Remote Control System Galileo (RCS Galileo), the main espionage program created by this company. What I do in this project is to analyze the source code of this software, explain how it works, and to create my own espionage platform which I named Ganon Spy System (GSS), based on the basics of RCS Galileo).

GSS is able to infect an Android phone and to retrieve data from it, without no knowledge of the user, and send this data to an anonymous server which stores it, to be retrieved from a client console whenever the attacker wants. We will understand the importance of maintaining your devices secure, by seeing how much information can be retrieved from your smartphone, not just the photos or videos you store in your device memory, but information such as your location or the state of your device as well. We will talk with detail about that in the next pages.

¹ Hacking Team: more information can be found at www.hackingteam.com
2. REMOTE CONTROL SYSTEM GALILEO

2.1 What is RCS Galileo?

Remote Control System Galileo is an espionage software developed by the Italian company Hacking Team. By reading the information revealed, we can determine that it is a program not used to perform non-directed attacks but attacks having knowledge of who the target is. This software has a basic structure which can be adapted to the customer’s needs. For example, if the customer needs to infect an iPhone, the exploit which the company prepares for the customer will be different from the case in which the target is an Android phone. An interesting point here is that these exploits are one-time shoot, which means that each compilation can be used just once because they are installed themselves in the device which executes them, deleting any trail they could leave.

2.2 Galileo structure

RCS Galileo has an structure as the one shown in Fig. 1. We have these elements, explained in detail:

- **Master node**: This is the most important element, a server which stores all the information retrieved from the targets and manages all the data used by the system.

- **Collectors**: they are previous servers which collects data from targets around the world. Then, they send it to the master node.

- **Console**: a desktop client used to establish connection with the master node and manage target attacks, retrieve data, etc.

- **Anonymizers**: they are complementary servers used to maintain the master node hidden from everyone. The information is sent through them in order the leave no trail of where the master node is.
• **Network injector:** it is a tool used to install the exploit into the desired device, and performs the attack through a shared Internet connection such as a bar or an airport WiFi were the target is connected to.

![RCS Galileo structure](image)

**Fig. 1:** RCS Galileo structure

### 2.3 How an attack is performed?

Once the target is located, the first thing to do is to install the exploit in their devices. The user could try to send an email or similar to the target with a link which would infect their devices, but in some cases it would not work.
HT developed a tool called **Network Injector** which does this commit. This tool is supposed to work in local networks such as a bar or an airport WiFi, and it is basically a huge *Man In The Middle* (from now MITM) software. A MITH is a process which puts your computer between two nodes, commonly another computer and a router, and this allows you to sniff all the traffic between these two points. This tool has many features to perform the injection, for example:

- **Change a website’s content:** if we change the link available in a site, the victim could download the exploit instead of the file he was looking for.

- **Inject directly the exploit into a download process:** when the user is downloading a file, it is possible to inject the exploit into the packets.

An optional but useful way to perform the attack, is by using **anonymizers**. They can be used to make the connection anonymous once the device is infected, but can be used to infect the target as well. They are basically HTTP proxies where to inject the packets if you get the target to be connected to.

Once the target has installed the exploit, there is no way to detect it, as it does not have a high impact in the battery life and neither in the device performance. The processes it executes are hidden for the task manager as well.

Now, the target is completely infected, and the data is sent to the anonymizers (in case they were used), proxy servers which makes the connection to the assigned collector untraceable. The collector is allocated in a zone close to the target, and stores the information to be sent to the master node. The master node is connected to the console executed by the user in a remote computer, and he is able to get all the files allocated into the victim’s device, as well as take pictures, take screenshots, record audio with the microphone, check if the display is on or off, get the victim’s GPS location, etc.

### 2.4 More espionage systems

There are many espionage softwares we can find. They are similar to RCS Galileo in some points, but different in others.
One of them is **Mobile Spy**\(^2\), a commercial software used to monitor different kinds of smartphones. The company describes this software as a software used for parents to spy on their children, or bosses to spy on their employees. The difference with Galileo is that this last one is used with no target's device physical access, but Mobile Spy users need to own the device which is supposed to be infected. They have to install the software into the device and then get it back to their targets.

The methodology of the espionage is similar to Galileo: it logs different kinds of information about the device and sends it to a master server. Then, the user can have access to this information.

Another well-known espionage software is **FlexiSPY**\(^3\). This is the same case than the last one, and moreover, the first step to install the software is to root or jailbreak the device where the user wants to install it, so again he needs to own the target’s device.

---

\(^2\) Mobile Spy: more information can be found at: [www.mobile-spy.com](http://www.mobile-spy.com)

\(^3\) FlexiSPY: more information can be found at: [www.flexispy.com](http://www.flexispy.com)
Analyzing that, we can achieve the conclusion that these companies have good developers who have worked on a very handy softwares. They bring the option to the user to spy any device, but the problem is that is the same functionality that softwares like Teamviewer have: they can get all the information from a device in real-time, with the difference with Teamviewer that they do not show that there is a current connection established.

These softwares are handy in some cases, nevertheless, when a real attack to a target has to be performed they lose sense. Where cyber-security skills have more importance is when the attacker has to install the software without the target having knowledge of it, and these companies do no provide this feature. Another point we can see here, is that they use their own servers in order to get the information, and it is a big mistake: anyone with a minimum knowledge on cyber-security will be able to discover where the information is sent, and uninstall the software. As we mentioned in previous chapters, it is important to have an anonymous structure which leaves no trails about the attacker. The final conclusion is that these softwares will not be useful to investigate different kinds of criminals, so it is important to look for companies focused on developing professional espionage systems.

---

Teamviewer: more information can be found at: www.teamviewer.com
2.5 What do we learn from here?

Once we have seen what Galileo is able to do, we know that anyone with bad intentions could spy our personal life if we do not take care when we use Internet. Now we are going to implement a new tool able to spy Internet connected Android phones, just like Galileo does. By doing this, we will understand better how espionage platforms work, and we will be able to find new ways to defend ourselves from them.
3. CYBER-SECURITY METHODOLOGIES

In order to develop a complex system as the one we want to do, the first thing to be done is to investigate a lot about cyber-security, we must analyze similar server-client applications, learn about cryptography, usable security, local network attacks and security in Android OS.

3.1 Server-client methodologies

In order to establish communications between the nodes of our platform, the main element we need is a server. As we want to implement a desktop based application, directly connected to the server, the best idea is to design it to work through a socket. So, we are going to have a program in our server always listening to determinate ports through the mentioned socket. This server is now supposed to have communication with a client application, and authenticate it in order to manage different operations.

In the client side, what the program is going to do is to start a connection with the server, by providing a determinate IP and a port number. By doing that, we will have and stream of information traveling between these two nodes, and what each of these nodes do is to put different packets of information into the stream, to be caught for the receiving node.

The same operation is going to be performed in the exploit side, but changing the port in order to have two well-differentiated streams.

3.2 Cryptography

3.2.1 Cryptography methodologies

One of the most important things into this kind of applications, as well as in many other programs, is to have an extremely secure infrastructure. This comes by making all the connections completely secret, we are talking about cryptography.
Cryptography is the idea of converting a set of characters into another one in order to hide the original message when, for example, is being sent through a determinate channel to another person. This is done by using a key, which through and algorithm is replacing the characters of the original message one by one. In the simplest case, with the same key, another person is able to decode the message.

An important factor here is the complexity of the algorithm. By doing some probabilistic calculations, we can determinate the possibilities that a key and an algorithm have to be decoded. A basic example of cryptography is as follows:

We have a 32-bits key. Each single character is 1 byte, which is 8 bits, so we have a key of 4 characters. Then we have:

Message to encrypt: *Hacking*
Encryption key: *root*

In order to make the example simpler, we are going to encrypt just letters using a key which contains just letters as well. In this example we use the american alphabet:

| LETTER | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| NUMBER | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |

Now, the idea is to shift all the characters of the message as many positions in the alphabet as the matching letter of the key position’s number is. When the message is longer than the key, the key is repeated. Let’s see it in detail:
So our encrypted message for the message HACKING is YOQDZBU. Now, to decrypt it, we just have to do the inverse process.

3.2.2 Communication protocol

Next factor to consider, is our communication protocol between nodes. We can have a private-key algorithm or a public-key algorithm:

- **Private-key**: In this case, the key has to be completely secret, which means that it cannot be shared with anyone. If we suppose, for example, a chat, the person who starts an encrypted conversation with anyone else, has to send first of all the shared key, in order to make the receiver able to decrypt the messages. By doing it, you are exposing the key to be stolen, and then the conversation is not going to be secret anymore. It would be just one way to do that, and it would be by creating the same key at the same time in both devices, but without sharing it through a network.

- **Public-key**: In this case, everything is more complex but more usable and secure. The public-key algorithm uses a method which is able to create a key to encrypt a message, and this message cannot be decrypted using the same key, it has to use another key generated by the algorithm. The communication process is as follows (detailed in Fig. 4):
1 - Node A wants to send a message to Node B.
2 - Node A asks Node B for an encrypting key.
3 - Node B receives the request, and generates an encrypting key E1 and a decrypting key D1.
4 - Node B sends E1 to Node A.
5 - Node A receives E1 and encrypts a message M1 using that key.
6 - Node A sends M1 to Node B.
7 - Node B receives M1 and decrypts it using D1.

8 - Node B wants to reply Node A.
9 - Node B asks Node A for an encrypting key.
10 - Node A receives the request, and generates an encrypting key E2 and a decrypting key D2.
11 - Node A sends E2 to Node B.
12 - Node B receives E2 and encrypts a message M2 using that key.
13 - Node B sends M2 to Node A.
14 - Node A receives M2 and decrypts it using D2.

Fig. 4: Public-key communication between two nodes.

Now, both nodes have encrypting and decrypting keys, and they can have a completely secret conversation.
3.3 Usable security

When we are talking about the term usable security we are talking about ways to design a good interface which makes the user to take the safest decisions when he is using the application. It can also be defined basically as interface options which makes the system safer.

3.3.1 Error messages

A common example is the “This connection is untrusted” error message that some browsers shows when you are trying to access a site which has an invalid SSL certificate. You can see this error in Fig. 5, showed by Mozilla Firefox:

![Mozilla Firefox error message](image)

Fig. 5: Mozilla Firefox error message

We can see as the message is giving you information about the error. In many cases, when a user gets a prompt, he ignores it and accepts without reading the message. In this case, you have to options: click the “clear” button Get me out of here! in order to do not connect to the site, or drop down the option I Understand the Risks in order to access the site. If a user does not read the message and clicks the “clear” button, he will have taken the safest option. In order to access the site, he will have to read the message, understand it because it has to be written in a user-orientated syntax, and assume the risks.
3.3.2 Passwords complexity

Another example of usable security are the way people have to interact with passwords. A common attack to discover passwords are the dictionary-based attacks. These attacks tries to discover passwords by trying to authenticate into a system with loads of passwords from a huge list of passwords used by people around the world. These passwords can be as simple as password, 123456, football… And also the same words combined with numbers and capital letters.

In order to avoid these attacks, the passwords should be much more complex, combining letters and numbers with no sense. The problem of these passwords is that they are difficult to remember, and because of that the users decides to use easier passwords or write the complex passwords in a paper (which can be seen by anyone, for example in an office).

So, if the problem is to get a password easy to be remembered and secure at the same time, we should not use a combination of letters but a combination of concepts with no relation, which is not going to be in any dictionary list. An example of password following these rules could be: housechairhorse. By using this password, you just have to remember three concepts, and at the same time, it will be impossible to discover it by any dictionary-based attack.

The very long and complex passwords are also useful against another kind of attack: the brute-force attack. This attack tries to authenticate into a system by combining randomly different characters. It is used in a very restricted kind of passwords, for example a password which is known that just can contain four digits, eight letters or numbers… But if it is used with a very complex password, the probability to get it in less than a million of years, is almost zero.

But if we use a two or three concept password, and the algorithm is configured to start with the simplest combinations, there is still some chances to discover the password in a determinate amount of time. In order to avoid that, the system has to be configured to accept just one authentication each, for example, 5 seconds. And after three tries, the user has to wait, for example, one minute. By doing it, the probability to discover the password in less than a million of years, is zero again.
3.3.3 Two-factor authentication

We defined some rules used to set up a secure and usable password, but none of these rules is useful if our password is stolen in a network attack. A password is a shared secret between a user and a server, and when a secret is shared, is not secret anymore. So in order to make an authentication system secure, we have to use more factors than a single word, and these should be at least two of the three that follows:

- Something we know (a password)
- Something we are (for example our face, our retina, our digital print…)
- Something we have (for example our phone)

In the example of an smartphone, if we use a password to access the system, and then we are asked for our digital print, there will be no way to access the phone without your fingers. However, in the example of a server-client system, both factors could be stolen. If instead of this, we use a password and a randomly generated code sent to our phone which just can be used once, the authentication system becomes completely secure in authentication terms.

In the case that someone gets to fake a site where you put all the credentials, and an script tries to authenticate into the site immediately, the only thing you could do would be check the log of authentications of your site’s account. In order to avoid this case, the use of SSL certificates is extremely important. Although someone gets to stole your credentials directly from your browser, as the connection with the site is direct and the random code just can be used one time, he would have no time to authenticate in the system before you.

At this point, our main problem would be to avoid someone to get in into our device and access our site’s accounts through it, and that is exactly the kind of application we want to develop.
3.4 Local Area Network attacks

In the previous point, we talked about usable security and at the same time we saw an overview about network attacks. Now we are going to see them with more detail, dividing it in two points: accessing the WiFi and hacking the LAN.

3.4.1 Accessing the WiFi

If we are planning to attack some device with which we have a physic proximity, for example, our boss’s laptop in our office or our neighbors smartphone, the first thing we need is having access into the same Local Area Network (LAN). If we do not have physic access through an Ethernet wire to the router, and neither the WiFi password, we will have to find the way to know this password. In order to do this, we can find on internet a couple of tools that performs these actions, such as Aircrack-Ng. The first thing we have to do is putting the WiFi antenna of our computer in monitor mode, to sniff the WiFi signals on air, discovering in this way all the networks that are surrounding us. We put the monitor mode using this command: `airmon-ng start <your_wlan_interface>`. This may require you to kill some processes which interferes this operation:

![Aircrack in Kali Linux terminal, putting WiFi antenna in monitor mode.](image)

Aircrack-Ng: more information can be found at [www.aircrack-ng.org](http://www.aircrack-ng.org)
Next step is check all the WiFi signals that surround us, and take the BSSID and the channel of the router we are planning to attack:

![Table of WiFi signals](image)

**Fig 7:** Discovering a target in Kali Linux terminal.

Now we have what we needed:

**BSSID:** 90:F6:52:F0:48:07  
**CHANNEL:** 1

There are a couple of attacks we could do, one of them is a dictionary-based attack, which we are going to perform now against a WPA2-PSK authentication system (an information shown in Fig. 7 as well). Next step now is to sniff directly on the selected network, by using the parameters discovered, and write everything in different files. We use the command:

```
airodump-ng -c 1 --bssid 90:F6:52:F0:48:07 wlan0mon -w ganon
```

![Sniffing handshakes](image)

**Fig 8:** Sniffing handshakes in Kali Linux
We can see as a user is connected into the WiFi network we are planning to access. Now we will need a parameter called *handshake*, which can be got when a user authenticates into the router. So we have two options: wait until someone connects into the WiFi network, or disconnect everyone from the router.

We are not very patient, so let's go to disconnect everyone who is connected into the target router (disconnect just a determinate user would be possible as well):

We sent ten *deauthentication messages* to the router, and when our target connected again into the system, we captured the handshake.

Now we use the *aircrack-ng* command with a dictionary of words and the file *ganon-01.cap* that we generated in last step:
Done! We got the password. As we can see, it was a simple number, so it has been discovered using just 28 keys. Here we can see again the importance of complex passwords in our systems.

3.4.2 Hacking the LAN

One of the main attacks that someone can perform into a LAN is called Man In The Middle (MITM). It consists in putting your computer between two nodes, usually the router and another device, by using the ARP Tables, which define the communications between all the nodes in a LAN. There are many applications which perform this operation such as Ettercap\(^6\), Bettercap\(^7\), Evil Foca\(^8\)…

---

\(^6\) Ettercap: more information can be found at https://ettercap.github.io/ettercap/

\(^7\) Bettercap: more information can be found at www.bettercap.org

\(^8\) Evil Foca: more information can be found at www.elevenpaths.com/es/labstools/evil-focasp
Usually these attacks are performed using IPv4, but since some time ago the usage of IPv6 (which identifies all the nodes in the world directly by just a single IP) has increased a lot and these programs allow to attack devices under these conditions too.

Once we are located between two nodes, all the traffic between them goes through us and we can use some kind of softwares such as Wireshark in order to sniff the communications.
At this point, we could use Ettercap as well in order to change some contents which our target could have asked for to some server. For example, we could apply a filter like the one that follows:

```java
if (ip.proto == TCP && tcp.src == 80){
    replace("img src=", "img src="<some_image_URL>"");
}
```

This replaces all the images in a website by another images set up by the attacker, and this kind of attacks could be used for example for change a file someone is trying to download by another one with a malicious software installed in.

### 3.5 Security in Android OS

All the Operative Systems have their own security rules and many times a cyber-criminal’s success relies on the way the user uses them. For example, usually we could consider that a MAC OS X system is safer than a Windows system, but depending on the user’s usage it could not be true.

Another example are all the Android devices: they have an option which enables the device to install applications that do not come from Google Play. If the user knows what he is doing, should not be problem with that, but in many cases we can take advantage of that. We could, for example, change the content of a website which offers an open-source game, offering the user the same game but with a Trojan inside.
Another point where Android fails is that in versions older than 6.0, it asks for different kind of permissions (microphone access, camera access…) just when the user is downloading some application. It is very common for a user to accept everything without taking a look at every point, here we would be talking again about usable security.
4. DESIGN

We have seen that Galileo uses a server-client structure in order to perform its operations. We are going to use a similar structure, defined by the graphic that follows:

![Ganon Spy System structure](image)

It has a server which is the main part of the system. The server manages the exploits as well as the client console used by the attacker. This system has been designed to attack Android phones, which uses Java as their main programming language, so we developed the whole system using this language.

4.1 Installing the exploit

As we have seen, there are many kind of attacks to force a user to install a malware. In our case, we chose a direct-phishing attack. It consists simply in sending a message or
email to our victim with a link to our exploit. It is part of an open-source game, where we put the exploit in order to be executed always as a background process, from the moment that the victim installs it.

### 4.2 Exploit

Our exploit is an `.apk package` compiled using *Android Studio*. Once it is installed, our exploit acts independent from the game, in a different Thread executed periodically in background. All the processes are named with names that identifies the exploit with the game, so if the user has a good task manager, he never is going to think that it could be a Trojan.

The protocol that our exploit follows is first of all, collect different kind of data such as gallery’s images, pictures from the camera, audio records… for an amount of time and store everything in temporary files. Then, it tries to establish connection with the server by sending an authentication code defined previously by the attacker. If there is response and the code is correct, it sends everything to the server and then it deletes the temporary files, and starts the process again. If there is no response, the exploit deletes all the collected data.

This can be defined by the attacker: for example, the attacker can say the exploit to not delete the data, and try to send it again in the future. The attacker can also say our exploit to try to send the files every 30 second, 5 minutes, 2 hours… By doing an open-close connection makes the application more efficient, and does not impact so much in the battery life.

One of the most important points is to make the structure secure. In order to do that, all the communications exploit-server are encrypted using an *AES encryption*, with private keys predefined by the attacker (the attacker sets the keys before to install the exploit, so they do not have to be shared and are extremely secure) which are known only by the exploit and the server, the client console will use another key. Then, before to send all the files to the server, the exploit encrypts it with a different key which only the client console knows (which is commonly called *point-to-point encryption*). The server is going to store these
files encrypted, and the reason by we do it in that way is that if someone got to get in into the server, he will not be able to read anything about the target.

4.3 Client console

The aim our client console, programmed in Java, is to connect to our server in order to set the exploit attacking values (time between each send of files, delete or do not delete data if there is no server’s response…) and check the files out. When you execute the program, first of all it prompts an authentication screen:

![GSS authentication screen](image)

**Fig 12:** GSS authentication screen.

This screen asks the user for a name and a password predefined by the attacker and the server’s IP. By clicking “Log in”, the server receives the request and validates the credentials (validated directly into the code, with no usage of any database). If they are correct, it asks for a two-authentication code:
The attacker, previously will have set up a generator key into some kind of application such as Google Authenticator, and when he sends the generated code to the server and it is validated, the connection is established. This kind of authentications works as follows: there is a script in the server that when is executed, it checks the system data and time and generates a code based on them, but this code is just valid for thirty seconds directly related with the time, in other words, the script could check the time and generate a code that is just valid for two seconds that are remaining. At the same time, the process is being done in Google Authenticator, generating a thirty seconds valid code, which in case of being stolen, will not be useful anymore.

Once the client is authenticated, it enters into the main screen, which allows the user to configure parameters and retrieve files from the server. As we said when we were talking about the exploit, this connection is encrypted in a client-server way (just the client and the server know the encryption key). But when the client receives the encrypted files, it decrypts it by using the key that just it and the exploit know.
4.4 Server

The main idea was to have a desktop application which could be started, stopped or restarted in an easy way. We developed a server control application in Java with a graphic interface which can be installed in a server-orientated environments such as Windows Server. What this server does is having two different Threats listening to two different ports. The first one establishes connection with the exploit installed into a phone, and the second one with the client console. These connections follow the protocols mentioned when we talked about the exploit and the client. All the connections are logged in the server’s main screen, shown in Fig. 12.

![Fig. 14: GSS Server main screen.](image)

As the server is the device where the exploit connects to directly, it is extremely important to make it anonymous. There are many options: one of them is to use an inverse proxy. It consists in setting up a proxy which acts in an inverse way, which means that it is not the
client who connects to the proxy to establish connection with some site but is the server who requires the client to connect the proxy to see a copy of its contents. In this way, the exploit is connected with a fake, which could be offered by some company, and the real server’s location is unknown. The problem here is that is unknown only by the target, but it is not by the company who is offering the service, so they could demand the attackers.

In order to solve this, we have seen that RCS Galileo uses a method in which it uses so many nodes that would be impossible, or almost impossible, for someone to catch them. It would be the optimum method, but kind of expensive. However, we could still do another thing: to put the proxy server in a country where espionages were legal, in that way, no one could intercept the server and we would have a minimum anonymity.
5. RESULTS AND IMPROVEMENTS

We have seen the importance of cyber-security. Many people do not care about this topic and they never take a look at their browser’s bar in order to see if they are in a secure connection. Now, by seeing what someone with bad intentions is able to do, many people can realize about the importance of this field.

This project is open-source, so it can be improved in a lot of ways in order to make a good software able to test the security of our devices. For example, an improvement of the exploit able to reduce much more the impact on battery life, or able to retrieve much more data from the user in a more efficient way would help manufacture companies to improve the devices they are selling, and all of us take benefit of it.

We have seen that RCS Galileo uses a network of anonymizers in order to make the communications between the exploit and the master node anonymous. They have designed it in a way where they have to use their own servers, and it is expensive. Maybe our platform could be improved by using other open-source platforms such as TOR\(^9\), a client used to establish completely anonymous connections with different kind of servers. This is the project which many criminals use in order to access the darknets, which are the deeper networks of the deep-web. Into these darknets we can find loads of illegal sites where they sell weapons, drugs, etc. This project can be also used to extort people, steal information, etc. But if we use this technology in our project and understand how it works, we could find a way to discover some trails these connections could leave and to protect us from cyber-criminal attacks.

But again… is it fine to break down this kind of tools? Should we to break them down or should we to improve them? TOR’s aim is to bring privacy to our connections in order to avoid our data to be stolen by anyone, so maybe the point would be to improve all the tools we have as much as we can, in order to make our devices and connections completely secure against cyber-criminal attacks. Then, we will have to find other ways to avoid any illegal trade, so it could be another topic to be studied.

\(^9\) TOR: more information can be found at [www.torproject.org](http://www.torproject.org)
Bibliography

GALILEO RCS - RUNNING AN ESPIONAGE OPERATION. Joe Greenwood. www.4armed.com

Cómo funciona el software de espionaje de Hacking Team. Chema Alonso. www.elladodelmal.com

Hacking Team. WikiLeaks archive. www.wikileaks.org


Appendix: user manual

1 - Installing the server

We need a server side environment such as Windows Server 2008 R2 Standard or similar, and install there a Java environment. We also have to open ports 90 and 91. Then, execute the file GanonServer.jar and press Start service. Now our server is prepared for receiving connections.

2 - Installing the exploit

We need to open the GanonExploit Android Studio package and add into the main class the IP of the server the exploit will be connected with. Then, it is necessary to compile the project and get the .apk file. Finally, we have to perform an attack in order to install the application into some Android device. Once the exploit is installed, it will start to send files to the server.

3 - Installing the client

We need a desktop environment such as Windows or Mac OSX and install there a Java environment. Then, we have to execute the provided GanonClient.jar file and the application will start. By default, the credentials are:

user: admin
password: ganon

By default, in order to get the second factor authentication number, it is necessary to scan next picture with some application such as Google Authenticator:
This will set up an algorithm in your application, and every time you check it you are going to receive a new code you have to introduce in the prompted screen.

Now, you can retrieve files from the server as well as setting up some configurations.