University of Milan
Department of Scienze e Tecnologie

Master degree in Computer Security

Report of Richkware project

Framework for building Windows malware

Richkware
(*github.com/richkmeli/Richkware*)

Candidate:
Riccardo Melioli – 901664

Academic year 2016–2017

Contents

List of Figures

Chapter 1

Description

1.1 Project Structure

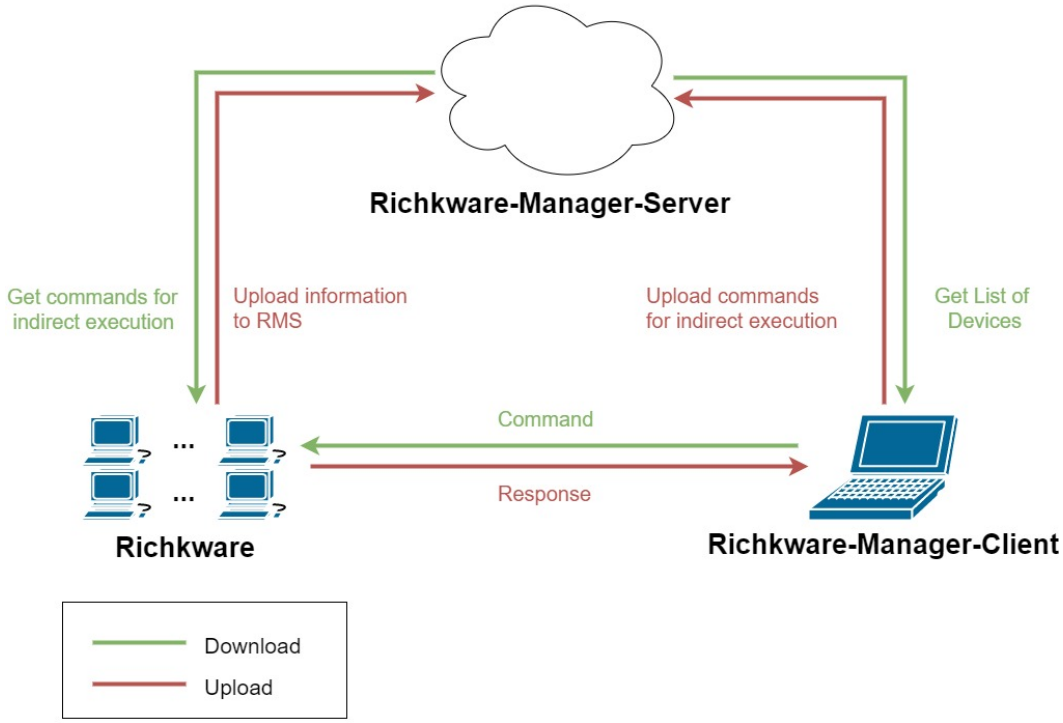


Figure 1.1: Project Structure

1.1.1 Richkware

Richkware is a library of network and OS functions, that you can use to create malware. The composition of these functions permits the application to assume behaviors referable to the following types of malware:

- **Virus**: it is a type of malicious software program that, when executed, replicates itself by modifying other computer programs and inserting its own

code. Infected computer programs can include, as well, data files, or the "boot" sector of the hard drive.

- **Worms**: is a standalone malware computer program that replicates itself in order to spread to other computers. Often, it uses a computer network to spread itself, relying on security failures on the target computer to access it
- **Bot**: is a computer connected to the Internet that has been compromised by a hacker, computer virus or trojan horse program and can be used to perform malicious tasks of one sort or another under remote direction.
- **Spyware**: is software that aims to gather information about a person or organization without their knowledge, that may send such information to another entity without the consumer's consent, or that asserts control over a device without the consumer's knowledge.
- **Keylogger**: is a type of surveillance software (considered to be either software or spyware) that has the capability to record every keystroke you make to a log file, usually encrypted. A keylogger recorder can record instant messages, e-mail, and any information you type at any time using your keyboard.
- **Scareware**: is a form of malware which uses social engineering to cause shock, anxiety, or the perception of a threat in order to manipulate users into buying unwanted software.

1.1.2 Richkware-Manager-Server

Service for management of hosts where is present a malware developed using Richkware framework.

It stores all malware informations in a SQL database:

- **Name**: name of device, where malware is present
- **IP**: malware IP address
- **Server Port**: TCP port opened by the malware, it allows the remote connection and the remote commands execution
- **Last Connection**: date and time of last malware connection
- **Encryption Key**: Server-side generated Encryption Key, is used from the malware to encrypt data.

1.1.3 Richkware-Manager-Client

Richkware-Manager-Server Client, gets the list of all hosts from the server and allows to **send commands** to run on the infected pc, by safe communication.

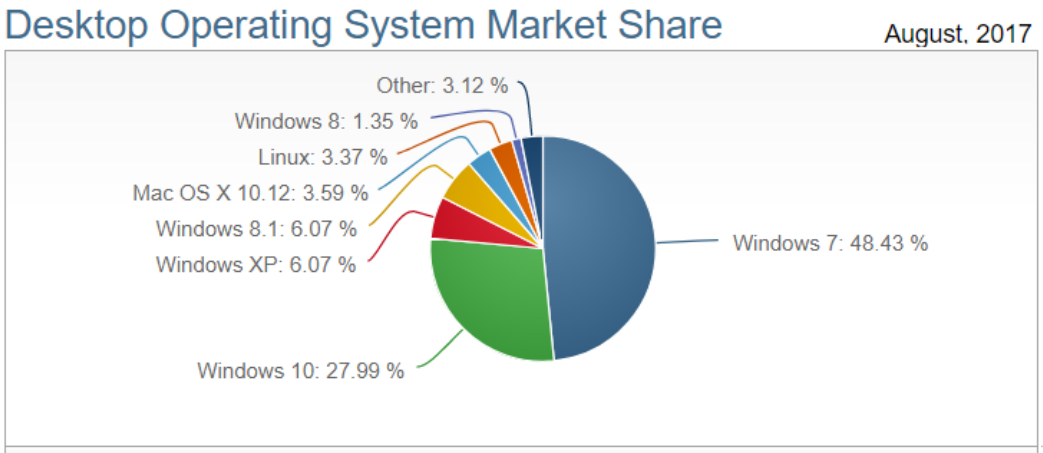
1.2 Goal

the goal of this project is to create a library that allows the development of any kind of malware in a simple way. Using this library makes it easier to create more complex and "intelligent" systems.

1.3 Target OS

The project is developed for the Microsoft Windows operating system as the target of attacks, because many **vulnerabilities** are discovered during the year, which could be exploited to gain more functionality.

Windows is the **most common** operating system, so there is more chance of infecting more computers



OPERATING SYSTEM	TOTAL MARKET SHARE
Windows 7	48.43%
Windows 10	27.99%
Windows XP	6.07%
Windows 8.1	6.07%
Mac OS X 10.12	3.59%
Linux	3.37%
Windows 8	1.35%
Mac OS X 10.11	1.09%
Mac OS X 10.10	0.72%
Windows Vista	0.46%
Windows NT	0.31%
Mac OS X 10.9	0.26%
Mac OS X 10.6	0.09%
Mac OS X 10.8	0.09%
Mac OS X 10.7	0.07%
Mac OS X 10.13	0.02%
Mac OS X 10.5	0.01%
Windows 2000	0.00%
Mac OS X 10.4	0.00%

Figure 1.2: OS share, souce: netmarketshare.com

Another aspect considered in choosing the target operating system is the possible propagation of malware, a typical scenario in Windows is installing crack to get paid services for free.

The system is developed for Windows and uses APIs that allow to interact with the system at low level, making detection of defense systems more difficult.

1.4 Programming language

The project was developed in C++ and Java languages, particular:

- **Java**: Richkware-Manager-Server e Richkware-Manager-Client
- **C++**: Richkware

C ++ was chosen for malware because it is a compiled language and executed directly by the operating system, not by an interpreter as in the case of interpreted languages. Many interpreters sometimes introduce additional levels of security and may not be present on a target machine, reducing malware propagation.

The C ++ libraries used for the project are:

- **C++ Standard Library**: stdio.h, stdlib.h, conio.h, ctime, string, ...
- **Microsoft**: windows.h, winable.h, winsock2.h, ws2tcpip.h, ...

Using Windows libraries has provided low-level, complete integration with the operating system, making it possible to develop functionality that makes malware more complete and adaptable to various applications.

1.5 Open-source software

The source code of the whole project is open-source, to provide a tool to system administrators or network operators who can have complete knowledge of the implementation and can use it in security system testing.

Another advantage is that open source software makes it possible for any external programmers to interact with software development, adding improvements, or reporting errors.

The repository links are:

- **Richkware**: https://github.com/richkmeli/Richkware
- **Richkware-Manager-Server**: https://github.com/richkmeli/Richkware-Manager-Server
- **Richkware-Manager-Client**: https://github.com/richkmeli/Richkware-Manager-Client

1.6 Building and execution

The malware can be built with MinGW from Windows or Linux environments by cross-compiling.

After creating the main, with the desired framework functions, using the command 1.1, you create the executable file.

There is the possibility to use compilation automation tool:

- **Make**
- **CMake**

Listing 1.1: Building library using make tool

```
make
```

In the Makefile there is the building procedure, so the steps of compiling and linking.

The command ref CompLib is a compilation example, then creating the object file, one of the files in the library.

The parameters used are:

- **-c**: Compile or assemble source files, but do not link them
- **-03**: Optimized code generation, "3" indicates the optimization value on a scale from 0 to 3. The optimized code makes it more difficult the reverse engineering
- **-o**: Puts the output in the file that follows the parameter.

Listing 1.2: Compiling Richkware file

```
g++ -O3 -c -o Richkware.o Richkware.cpp
```

The command 1.3 links the object files, and then creates the executable file. It's also included the object file of the "main". The parameters used are:

- **-lws2_32**: links the Windows Sockets API library (WSA), which defines the features for using Windows network services.
- **-static-libgcc -static-libstdc++**: it links statically the standard libraries, making the executable standalone
- **-o**: Puts the output in the file that follows the parameter.

Listing 1.3: Linking

```
g++ -static-libgcc -static-libstdc++ -o Richkware.exe
   Richkware.o ... [other file.o]... main.o -lws2_32
```

Malware runs only by running the .exe executable file.

Chapter 2

Richkware-Manager-Server e Richkware-Manager-Client

This chapter discusses the Richkware server and client and what their functionality is.

2.1 Project structure

The system consists of 3 parts:

- **Richkware instance**: malicious program that uses the Richkware library and has enabled services for exchanging information with RMS and RMC.
- **Richkware-Manager-Server**: Server that receives data from the Richkware instance
- **Richkware-Manager-Client**: Client that sends commands to Richkware instance.

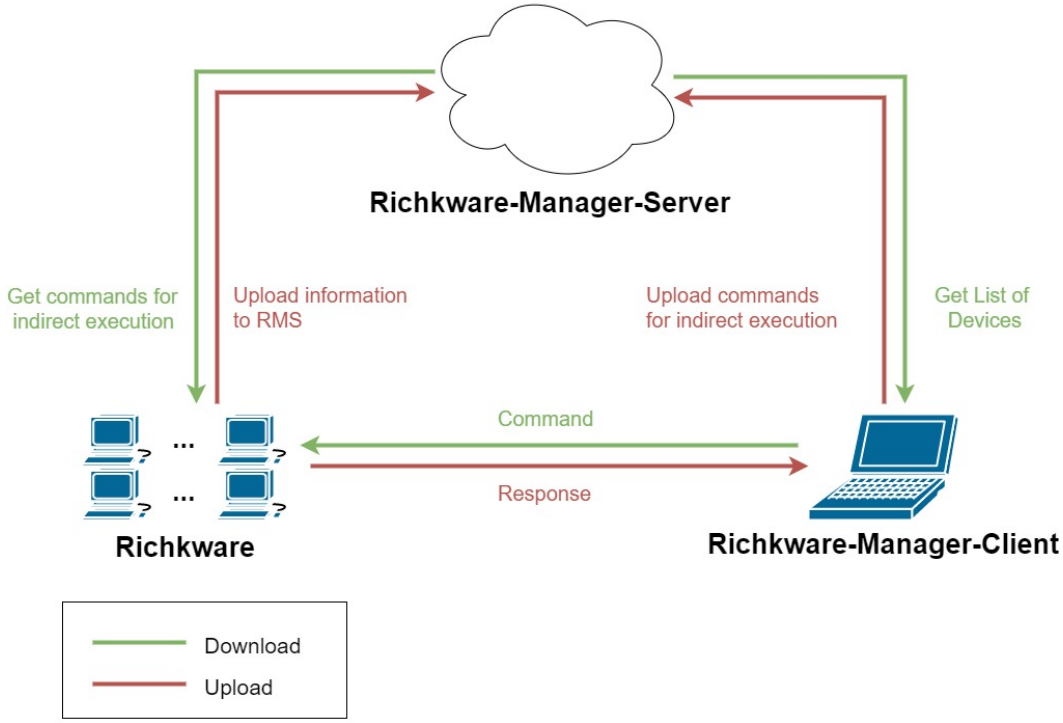


Figure 2.1: Project structure

2.2 Programming language

Richkware-Manager-Server and Richkware-Manager-Client are developed in Java, which makes them portable applications, so you can run them independently from the operating system you are using.

The libraries used in the projects are:

- **javax.servlet-api**: allows communication between servlet classes and runtime environment.
- **mysql-connector-java**: library for using MySQL databases
- **gson**: library for converting Java objects to JSON and vice versa

2.3 Server (Richkware-Manager-Server)

Service for management of hosts where is present a malware developed using Richkware framework.

It stores all malware informations in a SQL database:

- **Name**: name of device, where malware is present
- **IP**: malware IP address
- **Server Port**: TCP port opened by the malware, it allows the remote connection and the remote commands execution

- **Last Connection**: date and time of last malware connection
- **Encryption Key**: Server-side generated Encryption Key, is used from the malware to encrypt data.

The server interacts with Richkware, RMC, and browsers through the following servlet:

- **DevicesList**: shows the list of devices in an HTML page.
- **DevicesListAJAJ**: shows a JSON string containing the devices list, it supports encryption of the JSON string.
- **GetEncryptionKey**: when a Richkware instance contacts this servlet, it returns its encryption key.
- **LoadData**: used by Richkware instances to upload data to the server
- **RemoveDevice**: remove a particular device from the database.

The JSP (devices_list_AJAJ.jsp), allows you to make an asynchronous call to the DevicesListAJAJ servlet, then enter the data into the HTML from the JSON received through javascript functions.

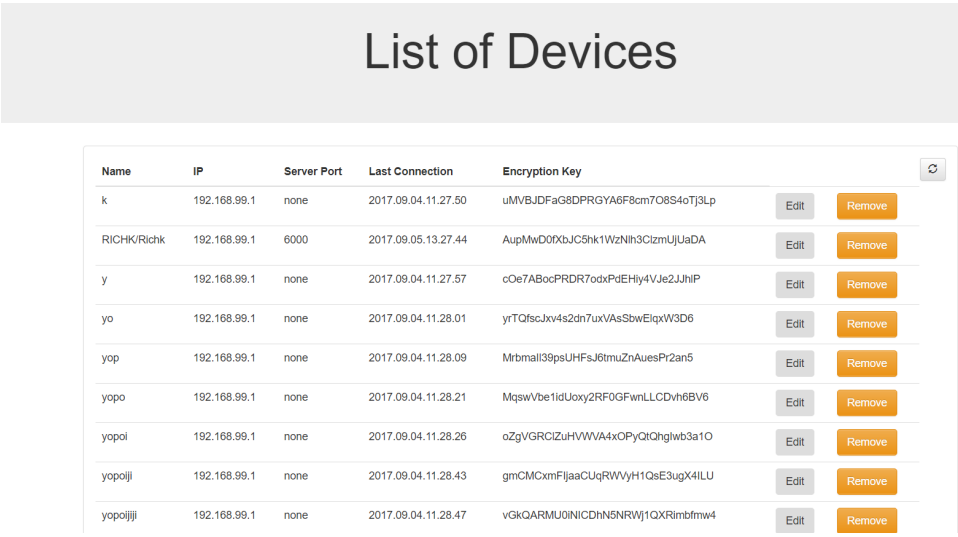


Figure 2.2: Richkware-Manager-Server, devices_list_AJAJ.jsp

2.4 Client (Richkware-Manager-Client)

Richkware-Manager-Server client, developed according to the Model-view-controller pattern, gets the list of all hosts from the server and allows you to send commands to run on the richkware instance via secure channel.

The View is a GUI (fig. Ref fig: Richkware-Manager-Client), entirely developed with the Swing library.

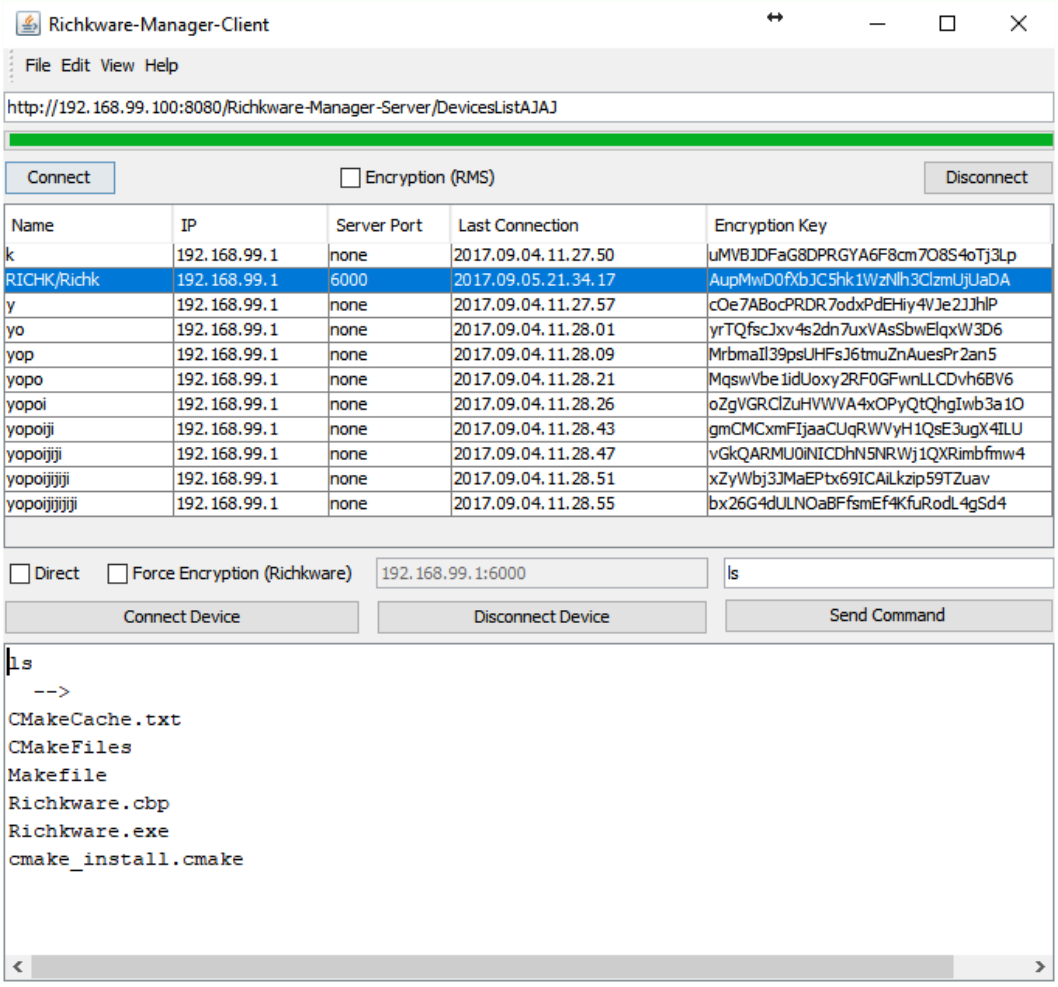


Figure 2.3: Richkware-Manager-Client GUI

To get the list of devices from the server, RMC calls to the DevicesListAJAJ servlet, which returns a JSON containing the information of all the devices stored in the database. The call in the RMC code is performed by the Network class, which encapsulates all network communication capabilities with the various parts. Within that class, there are also methods for sending commands to the Richkware instance according to a particular protocol, seen in the chapter (**??**).

2.5 Communication protocol between Richkware and RMC

In the protocol.h file there is the communication protocol used by RMC and Richkware to allow the RMC user to interact with the machine where the Richkware instance is installed.

Requests sent to the server created by Richkware are sent to a Dispatcher, which handles the request by a certain code, executes the request and returns the response, then the server can communicate it to the connected client. Dispatcher is implemented as follows:

Listing 2.1: Dispatcher comandi

```
...

switch (commandID) {
case 0:
response = "***quit***";
break;
case 1:
response = CodeExecution(command);
break;
case 2:
//...
break;
default:
response = "error: Command ID not found\n";
}
```

The syntax of a request is as follows:

Listing 2.2: Sintassi Protocollo di comunicazione Richkware e RMC

```
[[1]]ls
```

The previous command, having parameter 1, means that it is requesting the execution of the following string as a shell command, then "ls" will be executed by the Windows shell and its response sent to the client, like the response that would be printed on the shell.

Conclusion and future development

This report illustrates the open-source Richkware framework for creating malware, which can perform various functions.

Possible future developments are:

- creating a smart main that applies artificial intelligence concepts, such as hiding itself by antivirus and it takes decisions based on the external situation.
- extend the library with new features, such as creating a ransomware.

Bibliography

Alfieri, Roberto
2015 "Slide del corso di Reti degli elaborati, UNIPR".

AV-Test 2016 , av-test.org.

Cimato, Stelvio
2016 "Slide del corso di SSR, UNIMI".

Kaspersky Lab 2016 , kaspersky.com.

Securelist 2016 , securelist.com.

Silberschatz, Abraham
2014 *Sistemi Operativi. Concetti ed esempi*, Pearson.

Tanenbaum, Andrew S.
2011 *Reti di Calcolatori*, Pearson.

Wikipedia 2016 , wikipedia.com.