Universitat Oberta de Catalunya

uoc.edu

# Disseny i implementació d'un RAT i el seu CnC

**Daniel Pedrós Oliver**
Màster Universitari en Seguretat de les Tecnologies de la Informació i de les Comunicacions
Sistemes d'autenticació i autorització

**Enric Hernández Jiménez**
**Víctor García Font**

4 de juny de 2019

## FITXA DEL TREBALL FINAL

| | |
|---|---|
| **Títol del treball:** | *Disseny i implementació d'un RAT i el seu CnC* |
| **Nom de l'autor:** | *Daniel Pedrós Oliver* |
| **Nom del consultor/a:** | *Enric Hernández Jiménez* |
| **Nom del PRA:** | *Víctor García Font* |
| **Data de lliurament (mm/aaaa):** | *06/2019* |
| **Titulació o programa:** | *Màster universitari en Seguretat de les Tecnologies de la Informació i de les Comunicacions* |
| **Àrea del Treball Final:** | *Sistemes d'autenticació i autorització* |
| **Idioma del treball:** | *Català* |
| **Paraules clau** | *Control remot, evasió d'antivirus, xifrat* |


**Resum del treball**

Vivim a un món on les TIC juguen un paper fonamental en la vida de les persones. D'igual manera, la seguretat d'aquestes és un aspecte clau per tal d'assegurar la seguretat i privacitat de la vida digital de totes aquestes persones. Aquest projecte es planteja amb la intenció de facilitar l'evolució i maduració de les solucions de seguretat actuals, proporcionant una eina de control remot discreta i funcional als equips de seguretat ofensiva, encarregats de buscar les limitacions de les solucions de seguretat actuals amb la finalitat de millorar-les.

L'objectiu d'aquest projecte consisteix en el desenvolupament d'una eina de control remot i el seu servidor de comandament i control, a més de la implementació de diferents tècniques d'evasió de solucions de seguretat. Tot aquest procés es desenvolupa seguint una metodologia capaç d'adaptar-se dinàmicament a noves condicions, assegurant així l'adaptabilitat de la solució desenvolupada. Aquest document tracta des de l'anàlisi inicial i el disseny de la solució, fins a la implementació, evasió de solucions de seguretat i proves d'aquesta, proporcionant una visió completa de tot el procés de desenvolupament.

**Abstract**

We live in a world where ICT plays a fundamental role in people's lives. Similarly, the security of these is a key aspect to ensure the security and privacy of digital life for all these people. This project is intended to facilitate the evolution and maturation of current security solutions, providing a discrete and functional remote control tool to the offensive security teams, responsible

for seeking the limitations of current security solutions in order to improve them.
The objective of this project consists of the development of a remote control tool and its command and control server, as well as the implementation of different techniques to evade security solutions. This entire process is developed following a methodology capable of dynamically adapting to new conditions, thus ensuring the adaptability of the solution developed. This document covers from the initial analysis and design of the solution, to the implementation, evasion of security solutions and testing of the developed system, providing a complete view of the entire development process.

# Índex

1 Introducció ..... 1
1.1 Context i justificació del treball ..... 1
1.2 Objectius del treball ..... 2
1.3 Metodologia ..... 2
1.4 Definició de tasques i planificació temporal ..... 3
1.5 Descripció de la memòria ..... 5
2 Context tecnològic ..... 6
2.1 Entorn i llenguatge de desenvolupament ..... 6
2.1.1 Visual Studio 2017 ..... 6
2.1.2 .NET Framework i C# ..... 7
2.2 Control de versions i gestió del projecte ..... 8
2.2.1 GitHub ..... 8
2.2.2 Asana ..... 9
3 Anàlisi del problema ..... 10
3.1 Especificació de requisits ..... 10
3.1.1 Requisits arquitectònics ..... 10
3.1.2 Requisits no funcionals ..... 11
3.1.3 Requisits funcionals ..... 11
3.2 Arquitectura de la solució ..... 12
3.3 Criptografia ..... 13
3.3.1 Criptografia de clau simètrica ..... 13
3.3.2 Criptografia de clau asimètrica ..... 14
3.3.3 Intercanvi de claus Diffie-Hellman ..... 16
3.4 Casos d'ús ..... 17
4 Disseny de la solució ..... 19
4.1 Modularització dels components ..... 19
4.2 Establiment de la clau simètrica de xifrat ..... 20
4.3 Interacció entre components ..... 21
4.3.1 Interacció Agent-CnC ..... 21
4.3.2 Interacció Consola-CnC ..... 22
4.4 Disseny de la base de dades ..... 24
5 Implementació ..... 26

5.1 Implementació dels components ..... 26
5.1.1 Agent ..... 26
5.1.2 CnC ..... 27
5.1.3 Consola ..... 29
5.2 Implementació de les comunicacions ..... 30
5.2.1 Comunicació entre Agent i CnC ..... 30
5.2.2 Comunicació entre Consola i CnC ..... 32
5.3 Implementació de la base de dades ..... 33
5.3.1 SQLite + DB Browser for SQLite ..... 33
5.3.2 Estructura de la base de dades ..... 35
5.4 Evasió d'AV i solucions de seguretat ..... 37
5.4.1 AMSI ..... 37
5.4.2 HIDS ..... 38
5.4.3 NIDS ..... 39
6 Proves ..... 41
6.1 Entorn de proves ..... 41
6.1.1 Sistema ..... 41
6.1.2 Antivirus ..... 42
6.2 Proves de funcionament ..... 43
6.2.1 Agent ..... 43
6.2.2 CnC ..... 43
6.2.3 Comunicacions ..... 43
6.2.4 Evidències ..... 44
6.3 Proves de detecció ..... 49
6.3.1 Execució de l'agent ..... 50
6.3.2 Execució d'eines externes ..... 51
7 Conclusions ..... 57
7.1 Objectius assolits ..... 57
7.2 Objectius no assolits ..... 57
7.3 Futurs projectes relacionats ..... 58
7.4 Valoració personal ..... 58
8 Glossari ..... 60
9 Bibliografia ..... 61

## Llista de figures

Il·lustració 1: Cicle de vida del desenvolupament iteratiu i incremental .....3
Il·lustració 2: Planificació temporal de les tasques .....4
Il·lustració 3: Logotip de Visual Studio.....6
Il·lustració 4: Interfície de Visual Studio.....7
Il·lustració 5: Logotip de .NET Framework .....7
Il·lustració 6: Logotip de GitHub .....8
Il·lustració 7: Logotip d'Asana.....9
Il·lustració 8: Arquitectura bàsica del sistema .....13
Il·lustració 9: Esquema basat en criptografia de clau simètrica.....14
Il·lustració 10: Esquema basat en criptografia de clau asimètrica.....15
Il·lustració 11: Diagrama de l'intercanvi de claus Diffie-Hellman .....16
Il·lustració 12: Casos d'us del sistema a desenvolupar .....17
Il·lustració 13: Mòduls bàsics de cada component del sistema.....19
Il·lustració 14: Disseny del protocol d'establiment de la clau de xifrat.....21
Il·lustració 15: Diagrama d'interacció Agent-CnC .....22
Il·lustració 16: Diagrama d'interacció Consola-CnC .....23
Il·lustració 17: Diagrama de classes del component Agent.....26
Il·lustració 18: Diagrama de classes del component CnC .....28
Il·lustració 19: Logotip d'SQLite .....33
Il·lustració 20: Interfície de DB Browser for SQLite .....34
Il·lustració 21: Panell d'administració de l'antivirus McAfee .....42
Il·lustració 22: Registre de la màquina víctima amb l'identificador de l'Agent ...44
Il·lustració 23: Registre de l'Agent a la base de dades del CnC.....44
Il·lustració 24: Execució correcta d'una acció enviada pel CnC .....45
Il·lustració 25: Peticions rebudes al CnC i resultat d'execució d'una acció .....46
Il·lustració 26: Resultat de l'execució d'accions a la base de dades .....46
Il·lustració 27: Tràfic HTTPS entre Agent i CnC .....47
Il·lustració 28: Prova de tràfic HTTP entre Agent i CnC .....48
Il·lustració 29: Configuració xifrada al body d'una resposta HTTP .....49
Il·lustració 30: Resultat negatiu de l'anàlisi d'amenaces de l'Agent .....50
Il·lustració 31: Execució de l'Agent i consola d'events de l'antivirus .....51
Il·lustració 32: Detecció de Meterpreter per l'antivirus.....52
Il·lustració 33: Configuració i arranc del handler de Meterpreter.....53
Il·lustració 34: Inserció d'ordres a la base de dades .....54
Il·lustració 35: Connexió a la màquina víctima mitjançant Meterpreter .....55
Il·lustració 36: Execució de Meterpreter no detectada per l'antivirus .....56

# 1 Introducció

En aquest capítol es presenta una introducció al projecte desenvolupat. Entre altres, es tracten aspectes com la justificació d'aquest a més de posar-lo en context i plantejar els seus objectius principals. També es presenta la metodologia utilitzada per al seu desenvolupament, així com la planificació temporal d'aquest i una descripció dels productes obtinguts amb el desenvolupament del projecte. Per últim, es comenta també l'estructura de la memòria, presentant els diferents capítols que formen part d'aquesta.

## 1.1 Context i justificació del treball

Avuí en dia, la seguretat de les Tecnologies de la Informació i la Comunicació juga un paper fonamental en les vides de les persones. El món s'està digitalitzant cada dia més i més i, açò, en moltes ocasions, suposa una millora per a la societat, però també introdueix riscs i nous problemes per a aquesta.

Existeixen usuaris que utilitzen les Tecnologies de la Informació i la Comunicació de forma malintencionada. Aquests, cada dia més especialitzats, utilitzen les TIC amb interessos com el robatori d'informació, l'estafa o l'espionatge.

Les solucions de seguretat per a les TIC, com els antivirus o els HIDS (*Host Intrusion Detection System*), també són cada dia més efectius. No obstant, els atacants també s'adapten als nous escenaris i aconsegueixen sempre anar un pas per davant d'aquestes solucions de seguretat. Per això, és necessari dedicar esforços en la millora d'aquestes solucions.

Els analistes de seguretat que prenen el paper d'atacant per tal de comprovar la seguretat dels sistemes de les organitzacions juguen un paper fonamental en la millora de la seguretat d'aquestes i, en general, en la millora de les eines de detecció i solucions de seguretat aplicables a la societat en general. Aquests desenvolupen noves eines ofensives amb la intenció d'identificar les limitacions de les solucions de seguretat actuals i ajudar així a l'ampliació i millora d'aquestes, preparant-les per a ser capaces d'identificar i defensar-se front a noves tècniques que pogueren utilitzar els atacants.

Així doncs, i amb la finalitat de contribuir en la millora de les solucions de seguretat disponibles i, especialment, amb els equips de seguretat ofensiva que es dediquen a buscar els punts dèbils d'aquestes, es proposa el desenvolupament d'un RAT (*Remote Access Trojan*) i el seu servidor de comandament i control. El desenvolupament d'aquest projecte permet aprofundir en la comprensió del funcionament de solucions de seguretat, com els antivirus, i obtenir una ferramenta (2019) que serà utilitzada pels equips de seguretat ofensiva amb la finalitat de comprovar la validesa de les solucions de seguretat existents i contribuir així a la seua constant millora.

## 1.2 Objectius del treball

Per tal de donar una solució a la problemàtica exposada a l'apartat anterior es defineix el principal objectiu del projecte: dissenyar i desenvolupar un RAT (*Remote Access Trojan*) i el seu servidor CnC (*Command & Control*). Aquest RAT serà utilitzat en la realització de tests d'intrusió. Per això mateix, el seu disseny i implementació inclouran també l'aplicació de diferents tècniques d'evasió de solucions de seguretat com antivirus o NIDS (*Network Intrusion Detection Systems*).

Donades les necessitats i/o característiques del RAT que es proposa, aquest objectiu principal es pot dividir en diferents subobjectius. Els subobjectius que es proposen són els següents:

- Disseny d'un sistema funcional basat en 3 components:
    - Component que s'implanta a la màquina de la víctima
    - Servidor de comandament i control
    - Consola d'usuari per a l'administració, mitjançant el servidor de comandament i control, dels diferents components implantats a màquines víctima
- Implementació del component implantable i del servidor de comandament i control
- Aplicació de tècniques d'evasió d'antivirus i altres mesures de seguretat al component implantable

## 1.3 Metodologia

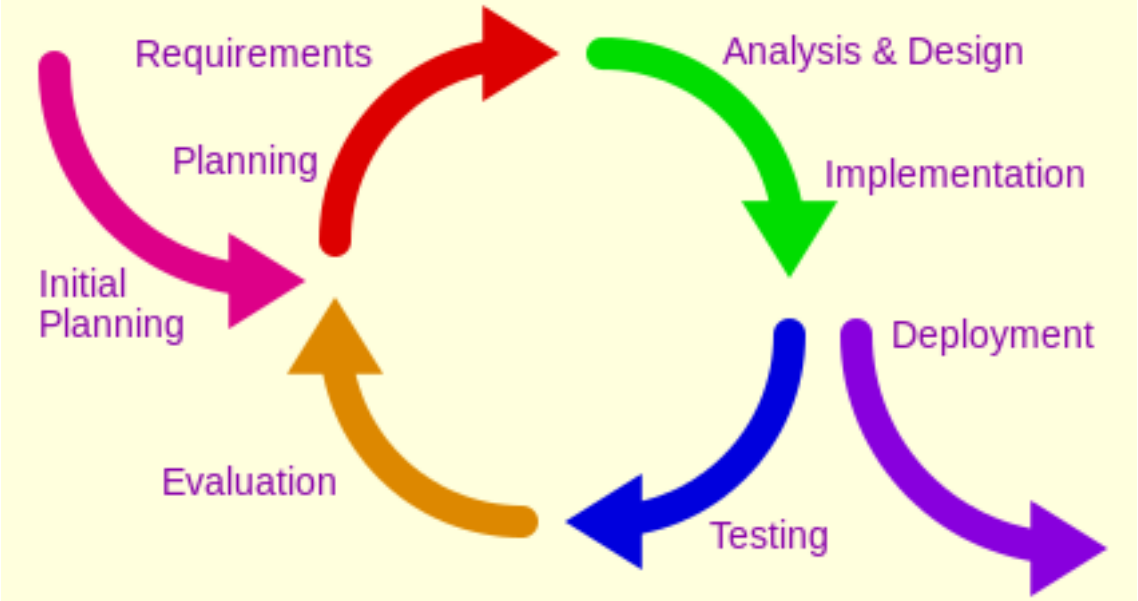
Sobre la metodologia proposada per al desenvolupament del projecte destaquen dos aspectes principals:

- Desenvolupament iteratiu incremental
- Programació modular basada en components

El desenvolupament iteratiu incremental consisteix, bàsicament, en el desenvolupament basat en xicotets increments de la funcionalitat de la solució. D'aquesta manera, en cada iteració del procés s'obté una versió completament funcional del producte que ofereix un increment en les funcionalitats implementades o en la qualitat del producte [4].

Aquesta metodologia és bona per a projectes que necessiten canvis a curt termini o han d'adaptar-se a condicions canviants durant el procés de desenvolupament. En aques projecte tenen importancia aquests dos factors ja que el resultat ha d'adaptar-se a les continues millores de seguretat que ofereixen tant els antivirus com els propis sistemes operatius, etc. Així doncs, a

la següent figura s'exemplifica el cicle de vida d'un projecte on s'aplica la metodologia basada en desenvolupament iteratiu i incremental.




**Il·lustració 1: Cicle de vida del desenvolupament iteratiu i incremental**

La programació modular basada en components és una tècnica de disseny de programari que consisteix en separar les funcionalitats d'un programa en diferents mòduls independents [1]. Aquests mòduls contenen estrictament només el necessari per a executar un aspecte, o conjunt d'aspectes molt relacionats, de la funcionalitat del programa. Açò permet que aquests components siguin intercanviables per altres mòduls que ofereixen la mateixa funcionalitat, independentment de la implementació d'aquesta. Açò permet realitzar una clara diferenciació entre la definició de les funcionalitats que ofereix el programa i la implementació d'aquestes.

## 1.4 Definició de tasques i planificació temporal

En aquest apartat s'enumeren i es descriuen les diferents tasques que es duen a terme durant el desenvolupament del projecte. S'inclouen, entre altres, tasques relacionades amb el desenvolupament de programari o la recerca d'informació i documentació. A més, una vegada definides aquestes, es presenta la planificació temporal seguida, situant sobre el calendari les dates d'inici i de finalització de cadascuna de les tasques.

Les diferents tasques plantejades per al desenvolupament del projecte són les següents:

- **Estudi de la situació actual**: Aquesta tasca consisteix en realitzar un estudi de la situació actual de conceptes relacionats amb el projecte. Des d'altres tipus de programari semblant fins a l'estudi de les tècniques més rellevants utilitzades pels antivirus i per a la detecció i identificació de programari maliciós.

- **Anàlisi de requeriments**: La finalitat d'aquesta tasca és identificar i detallar els diferents requeriments i necessitats que ha de satisfer el programari que es desenvolupa.

- **Especificació de la solució**: Una vegada determinats els requeriments del programari i especificades les seues funcionalitats, la finalitat d'aquesta tasca és dissenyar una solució que sigui capaç de satisfer els requeriments mitjançant les funcionalitats especificades.

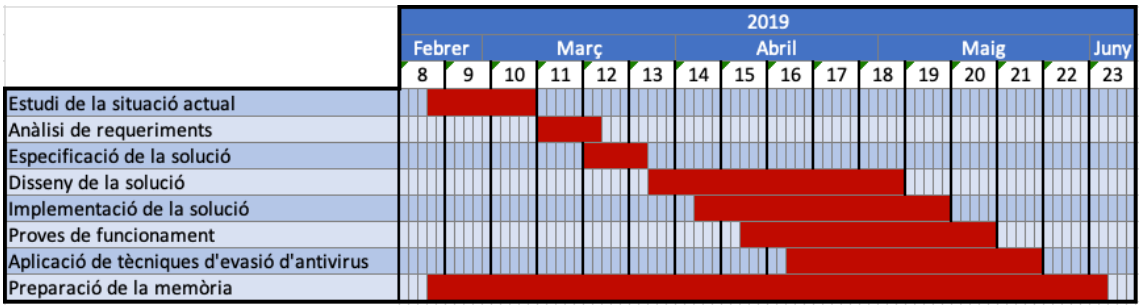
- **Implementació de la solució**: Aquesta tasca busca plasmar el disseny proposat en una solució real i funcional.

- **Proves de funcionament**: Tal i com es pot deduir del nom de la tasca, una vegada obtinguda la solució funcional, la finalitat d'aquesta tasca és provar el correcte funcionament de les funcionalitats d'aquesta.

- **Aplicació de tècniques d'evasió d'antivirus i altres solucions de seguretat**: La finalitat d'aquesta tasca és dotar a la solució implementada del mecanismes necessaris per tal de fer-la capaç d'evadir solucions de seguretat, com per exemple antivirus o NIDS.

- **Preparació de la memòria**: Aquesta última tasca consisteix en la documentació del procés de desenvolupament a la memòria del projecte (aquest document). També s'inclou la preparació de la presentació de resultats.

Una vegada definides les tasques a realitzar, al següent diagrama de Gantt es pot observar la distribució temporal d'aquestes:




**Il·lustració 2: Planificació temporal de les tasques**

A la columna de l'esquerre, les diferents tasques que formen part del desenvolupament del projecte. A la dreta, numerades des de la 8 a la 23, les setmanes de l'any 2019 agrupades també pel més al qual corresponen. Cadascuna d'aquestes setmanes està dividida en 5 parts, que corresponen als dies laborals de la setmana.

Així doncs, la planificació temporal estima:

- Estudi de la situació actual: des del 20 de febrer al 8 de març
- Anàlisi de requeriments: de l'11 al 19 de març
- Especificació de la solució: del 18 al 26 de març
- Disseny de la solució: del 27 de març al 3 de maig
- Implementació de la solució: del 2 d'abril al 10 de maig
- Proves de funcionament: del 9 d'abril al 17 de maig
- Aplicació de tècniques d'evasió d'antivirus: del 16 d'abril al 25 de maig
- Preparació de la memòria: del 20 de febrer al 4 de juny

Cal destacar dos aspectes de la planificació temporal. El primer d'ells és que la preparació de la memòria està planificada durant tot el procés de desenvolupament del projecte ja que contínuament es van documentant detalls d'aquest. El segon aspecte que destaca és la paral·lelització de les tasques de disseny, implementació, proves i aplicació de tècniques d'evasió d'antivirus a la solució. Aquesta paral·lelització és deguda a la metodologia basada en el desenvolupament iteratiu i incremental comentada a l'apartat de metodologia.

## 1.5 Descripció de la memòria

Aquest apartat comenta molt breument el contingut del present document. Aquest s'ha dividit en 5 capítols que tracten els aspectes més rellevants del desenvolupament del projecte. Aquests aspectes són:

- Context tecnològic: exposició del context tecnològic sobre el que es desenvolupa el projecte, prestant especial atenció a l'entorn i llenguatge de desenvolupament així com també a la gestió i seguiment d'aquest.
- Anàlisi del problema: exposició dels trets principals del projecte. S'analitzen diferents alternatives plantejades per al desenvolupament del projecte i es prepara un esbós general de com serà aquest.
- Disseny de la solució: a partir dels aspectes analitzats al capítol anterior, se seleccionen els més òptims i es proposa un disseny concret per a ser implementat.
- Implementació: procés dut a terme per tal de plasmar el disseny proposat al capítol anterior en una solució real i funcional.
- Proves: comprovació del correcte funcionament de la solució implementada i validació del compliment dels objectius del projecte.

# 2 Context tecnològic

La finalitat d'aquest capítol és presentar el context tecnològic sobre el qual s'ha desenvolupat el projecte. Es presenten diferents ferramentes utilitzades, així com la justificació de la seua elecció. Amb aquesta informació es pretén situar el projecte en un context real i donar-li així un punt de sortida des del qual començar a construir la solució.

## 2.1 Entorn i llenguatge de desenvolupament

En aquest apartat es comenten, tal i com el títol indica, l'entorn i el llenguatge de desenvolupament utilitzats per a la producció de la solució. Als següents subapartats es comenten les principals característiques d'aquests i es justifica la seua elecció.

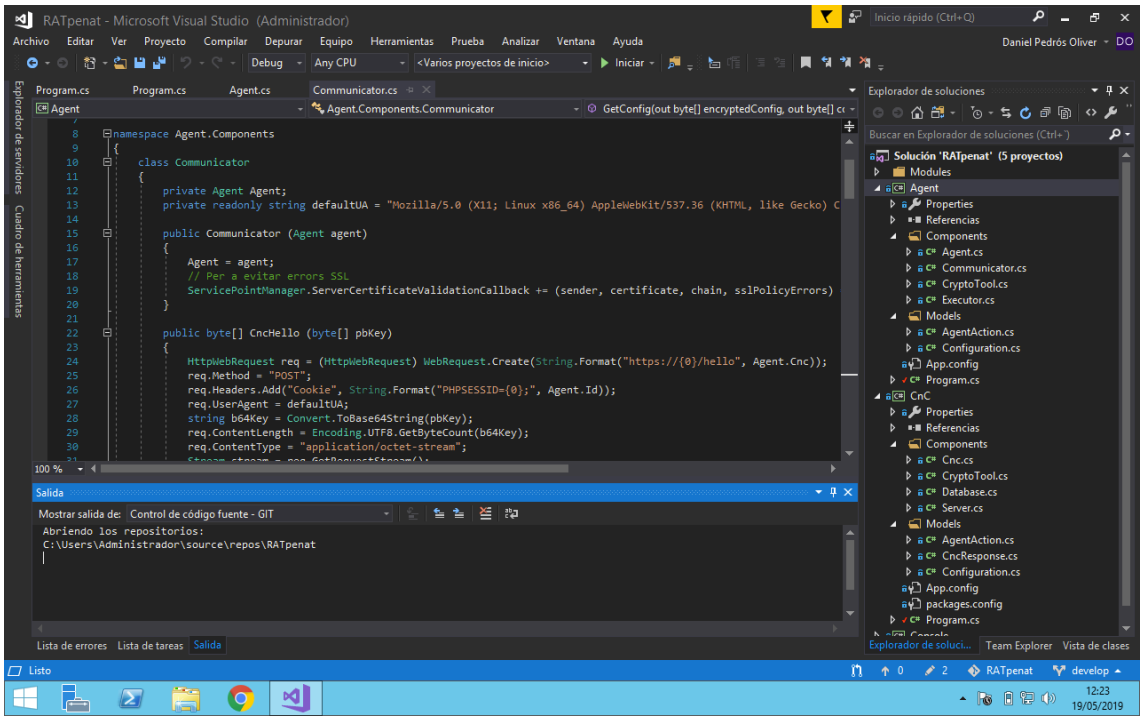
### 2.1.1 Visual Studio 2017

Microsoft Visual Studio és un entorn de desenvolupament integrat (IDE) que permet programar aplicacions (denominades solucions) en llenguatges com C++, C#, Visual Basic, Java, Python, etc., a més de llocs i aplicacions web [2]. Disposa de diferents versions (*Community, Professional o Enterprise*), i en aquest projecte s'ha utilitzat la *Community*, ja que és la que millor s'adapta a les necessitats del projecte.


**Il·lustració 3: Logotip de Visual Studio**

Entre altres característiques, Visual Studio proporciona diferents eines que faciliten al desenvolupador la tasca de programar. Entre elles, s'inclou l'editor de codi amb suport per a l'autocompleció (*IntelliSense*), un *debugger* integrat, que facilita la tasca de correcció d'errors i *debugging*, o altre eines per al disseny de part de la solució, com per exemple el disseny de la base de dades.

A més d'aquestes característiques, Visual Studio permet la integració amb eines externes utilitzades per al control de versions, com per exemple Git o *Subversion*. Sobre Git, concretament GitHub, utilitzada també per al desenvolupament del projecte, es parla al pròxim apartat. La interfície gràfica de Visual Studio permet aprofitar al màxim la integració de totes aquestes eines, facilitant així el procés de programació de la solució que s'estigui implementant. A continuació es mostra la interfície principal de Visual Studio.




**Il·lustració 4: Interfície de Visual Studio**

## 2.1.2 .NET Framework i C#

.NET *Framework* és un *framework* desenvolupat per Microsoft inicialment per a entorns Windows, però ara ja disponible per a altres sistemes operatius com macOS o Linux [3]. Aquest framework *proporciona* interoperabilitat entre diferents llenguatges de programació i un entorn d'execució multiplataforma.

El programari desenvolupat per a .NET *Framework* no s'executa directament sobre el maquinari, sinó que s'executa sobre una capa d'abstracció anomenada CLR (sigles per a *Common Language Runtime*), proporcionant així el suport multiplataforma comentat. CLR és una màquina virtual per a l'execució de les aplicacions desenvolupades per a .NET *Framework*. Aquesta proporciona serveis de seguretat, gestió de memòria o gestió d'excepcions.


**Il·lustració 5: Logotip de .NET Framework**

C# és un llenguatge de programació orientada a objectes fortament tipat capaç d'executar-se sobre la plataforma .NET *Framework*. És un llenguatge molt semblant a Java, en el que respecta a la seua sintaxis, però també per la forma d'executar-se, ja que Java ho fa sobre la màquina virtual de Java i C# ho

fa sobre la màquina virtual CLR comentada a l'anterior paràgraf. C# és utilitzat per al desenvolupament d'aplicacions per a Windows i destaca, entre altres aspectes, per la senzillesa que ofereix per al desenvolupament d'aplicacions de tipus client-servidor.

Donat que la solució desenvolupada al projecte s'executa sobre entorns Windows i presenta una estructura de tipus client-servidor, desenvolupar el projecte escollint C# com a llenguatge de programació s'ha considerat l'elecció més encertada. A més, com es comenta en posteriors capítols, C# ofereix algunes característiques que seran de profit per a l'evasió d'antivirus i solucions de seguretat, com per exemple la càrrega directament en memòria d'un binari.

## 2.2 Control de versions i gestió del projecte

En aquest apartat es parla d'algunes ferramentes utilitzades durant el desenvolupament del projecte. Es presenta una eina utilitzada per a la realització d'un correcte control de versions del programari desenvolupat i una altra utilitzada per al seguiment de les diferents tasques que formen part del projecte.

### 2.2.1 GitHub

GitHub és una plataforma pensada per al desenvolupament de codi de forma col·laborativa. Aquesta allotja el codi font i altres arxius de projectes de desenvolupament de programari i permet realitzar, de forma visual i senzilla, un correcte control de versions. Per tal de realitzar aquest control, GitHub fa servir un programari per al control de versions anomenat Git.


**Il·lustració 6: Logotip de GitHub**

Git, i per tant també GitHub, fan servir una estructura de dades local anomenada repositori, on es guarden els arxius, directoris i metadades del projecte que s'està desenvolupant. Les modificacions realitzades sobre aquests arxius van quedant emmagatzemades al repositori local. Per tal d'aplicar les modificacions realitzades al repositori remot (que seria el que s'allotja a GitHub) cal fer un PUSH utilitzant Git.

A pesar que aquesta eina està pensada per al treball col·laboratiu, en aquest cas s'ha decidit utilitzar-la a mode de còpia de seguretat i per tal de separar les versions estables del programari desenvolupat de les que estan de desenvolupament encara. Així doncs, al repositori s'utilitzen dues branques, una anomenada *develop* per a les modificacions durant el procés de desenvolupament i una altra, la principal, anomenada *master* on es publiquen les versions estables del producte.

### 2.2.2 Asana

Asana és una aplicació web i mòbil la funció de la qual és facilitar la gestió de les diferents tasques que formen part d'un projecte. Està pensada per a la gestió de projectes col·laboratius ja que incorpora moltes funcionalitats pensades per a proporcionar una fàcil coordinació entre els membres de l'equip.


**Il·lustració 7: Logotip d'Asana**

Aquesta està formada per diferents àrees de treball. Aquestes àrees de treball es poden entendre com a equips de treball, ja que cadascuna d'aquestes està associada a un conjunt d'usuaris (integrants del grup). A cadascuna de les àrees de treball es poden crear diferents projectes. Cada projecte està format per un conjunt de tasques que es poden organitzar en format de llista o en columnes, un format molt interessant i utilitzat per la metodologia Kanban.

Les diferents tasques disposen d'una descripció i poden tenir assignat un responsable i una data límit. A més, és possible també afegir arxius adjunts o comentaris, la qual cosa resulta interessant per tal de centralitzar la informació necessària o interessant per al desenvolupament.

En aquest projecte, Asana s'ha utilitzat per tal de fer un seguiment de l'estat del projecte, així com poder determinar de forma ràpida i visual, la planificació de les tasques. S'ha utilitzat una organització de les tasques en format de columnes, concretament 3 columnes: tasques pendents, tasques en desenvolupament i tasques finalitzades.

# 3 Anàlisi del problema

En aquest capítol es realitza una anàlisi en profunditat del problema que es pretén resoldre i es presenten les primeres pinzellades de la solució a implementar. Es tracten diferents aspectes, des de l'especificació dels diferents requisits de la solució fins a la identificació dels diferents casos d'ús del sistema, passant per la definició de l'arquitectura de la solució o la utilització de diferents tipus de criptografia.

## 3.1 Especificació de requisits

En aquest apartat es comenten els diferents requisits que ha de complir la solució que es pretén desenvolupar. Aquests s'han dividit en 3 tipus principals: els requisits arquitectònics de la solució, requisits funcionals i requisits no funcionals. Tots tres tipus es plantegen per tal d'encaminar la solució i descriure el com serà i què farà aquesta.

### 3.1.1 Requisits arquitectònics

Aquest tipus de requisit busca descriure com serà l'arquitectura de la solució a desenvolupar. Per al cas d'aquest projecte, els requisits arquitectònics de la solució inclouen:

- **Arquitectura dividida en 3 components independents**: el sistema ha d'estar format per 3 components independents capaços d'interactuar entre ells. Aquests 3 components han de ser:
    - **Component implantable a la màquina víctima (o Agent)**: és la part de la solució que s'implanta a l'ordinador de la víctima i s'encarrega d'executar les accions necessàries. Es comunica amb el servidor de comandament i control per a obtenir tota la informació necessària per a dur a terme la seua funció.
    - **Component d'interacció entre l'usuari i el sistema (o Consola)**: és el component amb el que interactua l'usuari que utilitza la solució desenvolupada. Aquest component també es comunica amb el servidor de comandament i control per a establir les funcions que ha de realitzar un determinat Agent.
    - **Servidor de comandament i control (o CnC)**: és la part de la solució desplegada a un servidor controlat i que s'encarrega d'interactuar amb els altres dos tipus de components. Aquest component interpreta les accions que estableix un determinat usuari mitjançant la Consola i les proporciona a l'Agent concret que ha d'executar les accions en qüestió.

### 3.1.2 Requisits no funcionals

Els requisits no funcionals descriuen com ha de ser la solució que es va a desenvolupar. Aquest tipus de requisit sol utilitzar-se per tal d'explicar aspectes de la solució relacionats amb el rendiment, la seguretat o la fiabilitat d'aquesta, entre altres. Així doncs, per al cas d'aquest projecte, s'han identificat els següents requisits no funcionals:

- **Totes les comunicacions del sistema han d'estar xifrades**: qualsevol interacció entre els diferents components del sistema ha de ser protegida. Per això, és necessari que totes aquestes comunicacions es realitzen utilitzant un canal segur xifrat.

- **Les comunicacions han de simular tràfic web legítim**: per tal d'evitar alçar sospites a un sistema de detecció d'intrusos, les comunicacions que s'estableixen entre l'Agent i el CnC han de ser el menys anòmales possible i simular una navegació web estàndard.

- **La implementació del sistema ha de ser modular**: els diferents components del sistema i els mòduls d'aquests han de poder ser substituïts, millorats i/o ampliats sense comprometre el funcionament dels altres components o mòduls.

- **Les funcionalitats del sistema han de ser modulars i ampliables**: les funcionalitats que implementa el sistema han de poder ser ampliables i/o modificables per tal de que aquest pugui adaptar-se a noves condicions del context.

### 3.1.3 Requisits funcionals

Els requisits funcionals estableixen la base per al funcionament de la solució. Aquests defineixen el com ha de funcionar el sistema, entrant un poc més en els detalls de la part tècnica d'aquest. Els diferents requisits funcionals que es busca tindre presents a la solució final inclouen:

- **Les comunicacions han d'implementar-se utilitzant HTTPS**: HTTPS proporciona una primera capa de protecció per a les comunicacions ja que estableix un canal de comunicació xifrat utilitzant SSL o TLS. A més, és un protocol molt habitual entre el tràfic de qualsevol ordinador o xarxa.

- **L'Agent ha d'implementar tècniques d'evasió d'antivirus i solucions de seguretat**: per tal d'evitar que aquest component sigui detectat i marcat com a maliciós pels antivirus o qualsevol altra solució de seguretat, és necessari que aquest implementi mesures per tal d'evadir aquestes solucions de seguretat.

- **L'Agent no ha d'emmagatzemar al disc cap informació que no sigui estrictament necessària**: minimitzar la utilització de

l'emmagatzematge evita que les solucions de seguretat, com per exemple l'antivirus o un HIDS (*Host Intrusion Detection System*), puguin analitzar la informació emmagatzemada per l'Agent i ajuda a que aquest component passi més desapercebut als ulls d'un usuari víctima.

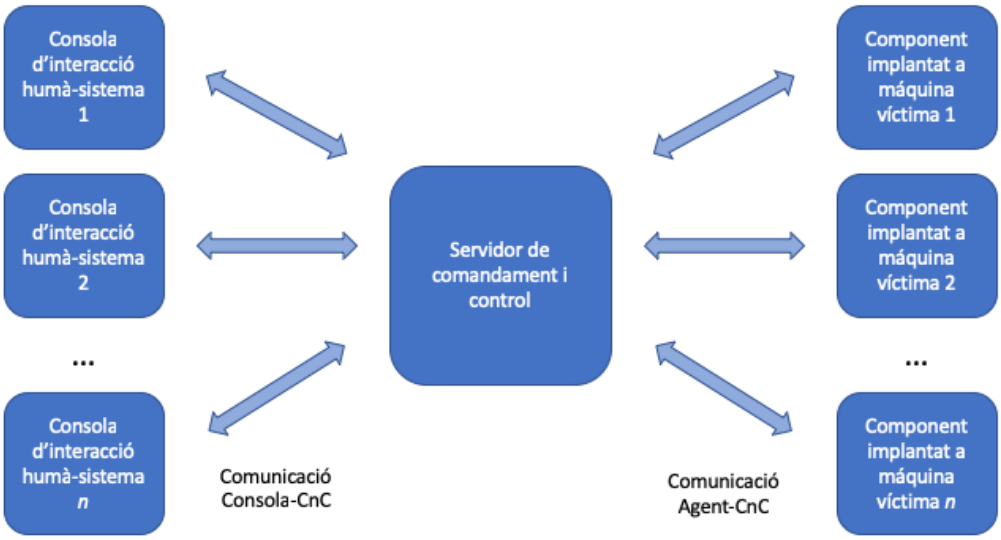
- **L'Agent només s'encarrega de sol·licitar periòdicament accions i executar-les**: evitar que el propi component implantable implementi totes les funcionalitats permet evadir solucions de seguretat més fàcilment i evitar que el component sigui detectat com a una amenaça. Açò és degut a que si un antivirus obté la firma del component amb totes les funcions implementades caldria modificar tot el component per a evadir aquesta firma. No obstant, si un antivirus obté i marca com a maliciosa la firma d'un mòdul, només caldria modificar el mòdul per a evadir esta firma, deixant intacte el propi component.

- **Cada Agent només pot ser controlat per 1 únic usuari al mateix temps**: evitar que dos o més usuaris del sistema interactuen al mateix temps amb un mateix Agent s'evita que un usuari pugui pertorbar el treball que altre usuari està realitzant amb eixe mateix Agent.

## 3.2 Arquitectura de la solució

En aquest apartat es presenta l'arquitectura bàsica de la solució implementada. Aquesta està basada en les especificacions comentades a l'apartat d'especificació de requisits i, com ja s'ha comentat, està formada per 3 components principals. Aquesta arquitectura basada en 3 components independents aporta la modularitat desitjada a la solució, permetent la substitució o reimplementació de qualsevol d'ells sense afectar al funcionament i implementacions dels altres.

Seguint amb l'estructura descrita als requisits arquitectònics, es proposa una solució basada en 1 únic servidor de comandament i control, *n* potencials consoles d'interacció amb el sistema i *n* potencials components implantats a màquines víctima. Al següent esquema es veu representada aquesta arquitectura.




**Il·lustració 8: Arquitectura bàsica del sistema**

A l'anterior esquema es representen les *n* potencials Consoles que interactuen amb l'únic servidor CnC i els *n* potencials Agents que també interactuen amb l'únic servidor CnC. Així doncs, i encara que no forma part de l'estructura en sí, a l'esquema s'observen els dos tipus d'interacció que existeixen al sistema.

## 3.3 Criptografia

En aquest apartat es parlen de diferents tipus de criptografia que són interessants de cara al disseny de la solució. Concretament, es parla de la criptografia basada en clau simètrica i la criptografia basada en clau pública-privada (asimètrica). Per a cadascuna d'aquestes, s'explica el concepte, de forma general, i es comenta la seua utilitat en el marc del projecte que s'està desenvolupant.

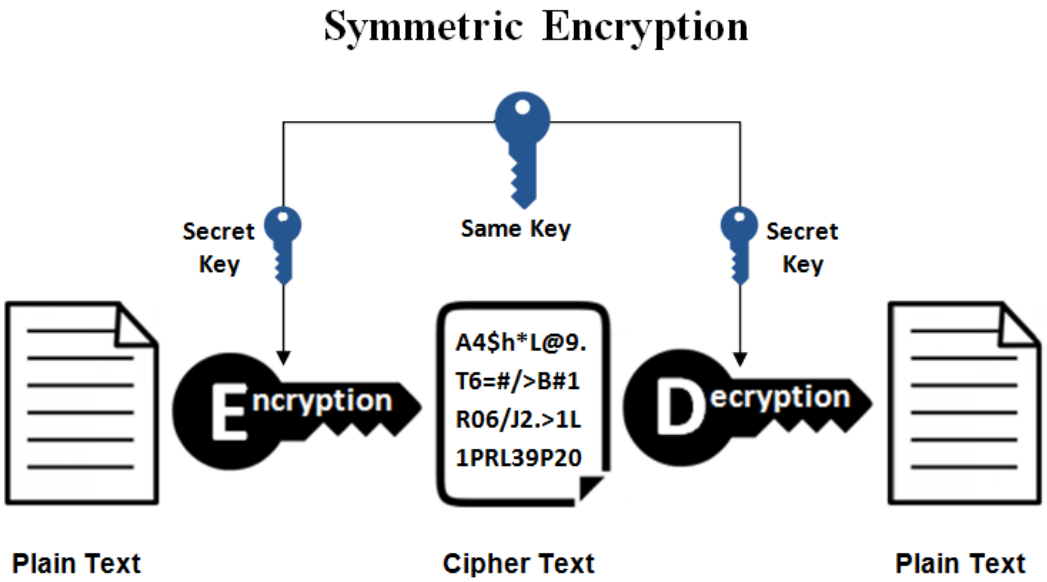
### 3.3.1 Criptografia de clau simètrica

La criptografia de clau simètrica, o també coneguda com a criptografia de clau compartida, es caracteritza per la utilització d'una única clau per al xifrat i desxifrat de la informació. Aquest tipus de criptografia permet a un emissor i un receptor comunicar-se de forma segura si ambdós comparteixen la mateixa clau, la qual cosa dona nom a la criptografia de clau compartida. En aquest cas, l'emissor seria qui xifra la informació utilitzant la clau compartida i el receptor seria qui la desxifraria, utilitzant la mateixa clau.

Quan es parla de l'ús de la criptografia de clau simètrica, assumim que les dues parts que intervenen en la comunicació han acordat la clau compartida de forma segura. És a dir, assumim que ningú que no siguin els dos implicats coneix la clau. D'açò es parla en el subapartat dedicat a l'intercanvi de claus.

La utilització d'aquest tipus de criptografia ofereix alguns avantatges, com per exemple, la possibilitat de xifrar i desxifrar dades més ràpidament que amb la utilització d'altres esquemes criptogràfics [5]. No obstant, un dels inconvenients més rellevants que presenta és que cal mantenir una clau per cada parell d'implicats en la transmissió de la informació. Així doncs, en una comunicació on existeixen *n* interventors on tots interactuen amb tots, cadascun d'ells haurà de gestionar i emmagatzemanr *n-1* claus.

Al següent esquema es pot observar la utilització de la clau per al xifrat i desxifrat de la informació en una arquitectura basada en criptografia de clau simètrica. Es pot observar que la clau utilitzada al procés de xifrat i desxifrat és la mateixa i que aquesta és compartida entre qui realitza cadascun d'aquests processos.




**Il·lustració 9: Esquema basat en criptografia de clau simètrica**

Exemples d'algorismes que utilitzen aquest tipus de criptografia n'hi ha molts. Els més coneguts són l'algorisme DES (*Data Encryption Standard*) o l'algorisme AES (*Advanced Encryption Standard*) [7].

### 3.3.2 Criptografia de clau asimètrica

La criptografia basada en clau asimètrica també coneguda com a criptografia de clau pública o de clau pública-privada, es caracteritza per la utilització d'una clau formada per dues components: la component pública i la component privada. Aquestes dues components de la clau estan molt relacionades entre elles i només tenen sentit si s'utilitzen juntes, ja que les dades xifrades amb la component pública només poden ser desxifrades utilitzant la component privada.

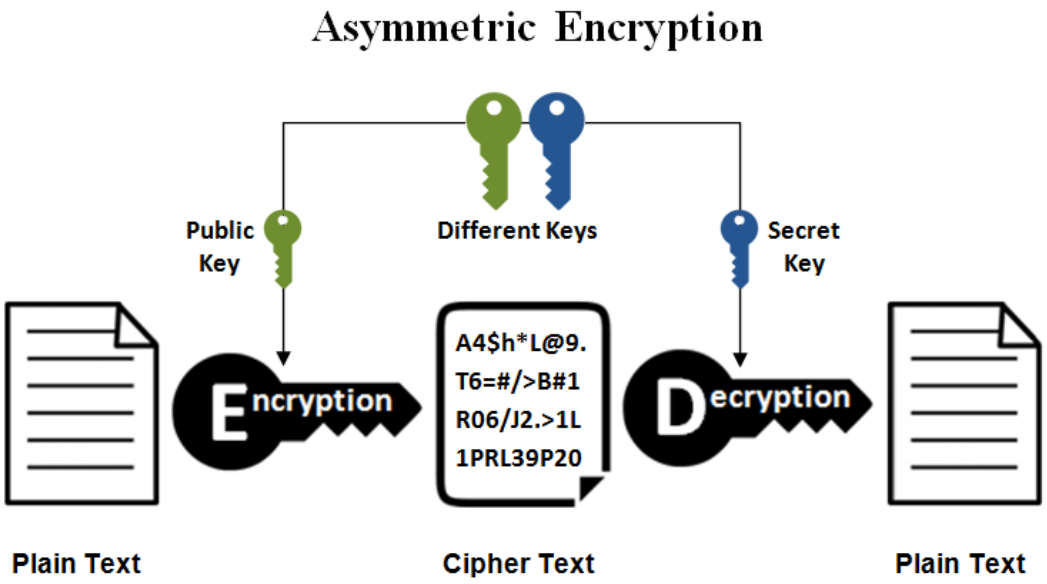
D'aquestes dues components, com és de suposar, la component pública és la que pot compartir-se obertament i la component privada és la que, com el

seu nom indica, s'ha de mantindré en secret. Com s'acaba de comentar, en aquesta arquitectura criptogràfica, la component pública de la clau pot ser comunicada públicament sense que això suposi una amenaça per a la confidencialitat de la informació que es xifra.

El que es desprèn de les característiques d'aquest esquema criptogràfic és que, per a un sistema on existeixen *n* components interactuant amb un únic component central, aquests només hauria d'emmagatzemar i gestionar una única clau (la component privada) i els altres components només haurien d'emmagatzemar i gestionar una única clau (la component privada).

Per les arquitectures criptogràfiques basades en clau asimètrica no és un problema la comunicació de la clau pública als components que volen transmetre informació xifrada al posseïdor de la clau privada. Com que la component pública només permet el xifrat de la informació, no hi ha ningun problema en que sigui coneguda per tothom.

A continuació es mostra un diagrama on es pot observar l'arquitectura en qüestió. Es pot apreciar l'ús de claus diferents per al procés de xifrat i el procés de desxifrat. Aquestes dues claus són les dues components d'una clau asimètrica, la component pública de la qual és utilitzada per a xifrar informació, i la privada per a desxifrar-la.




**Il·lustració 10: Esquema basat en criptografia de clau asimètrica**

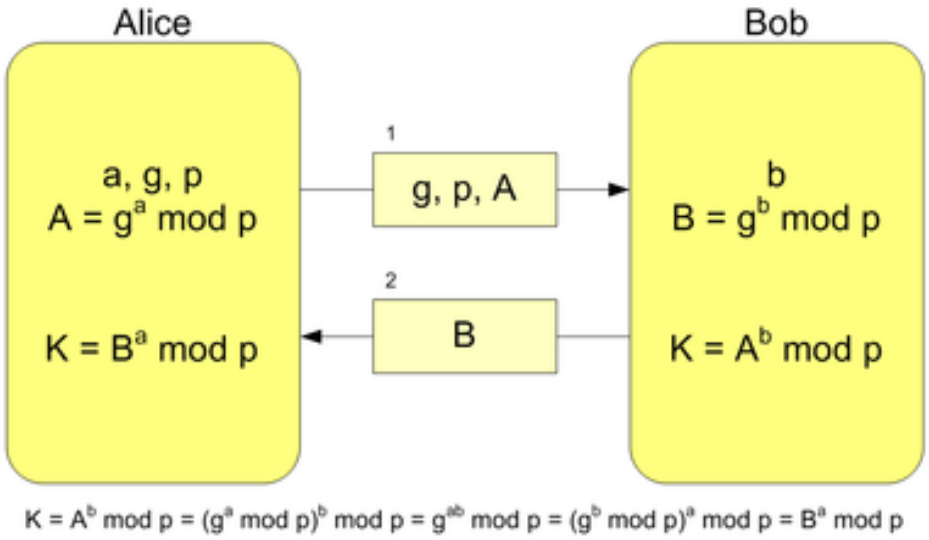
Existeixen molts algorismes que utilitzen aquest tipus de criptografia. Un dels més rellevants i utilitzat en l'actualitat és l'algorisme RSA (*Rivest-Shamir-Adleman*) [6].

### 3.3.3 Intercanvi de claus Diffie-Hellman

Tal i com es comenta a l'apartat de criptografia basada en clau simètrica, un aspecte important de la seguretat d'aquest esquema criptogràfic és la compartició de la clau secreta de forma segura. Aquesta no és una tasca trivial ja que és complicat disposar d'un canal segur per tal de compartir i establir la clau de xifrat abans de començar a xifrar informació que es transmet per un canal insegur.

Diffie i Hellman proposen un model que permet compartir una clau de forma segura a través d'un canal insegur. Els fonaments d'aquest model estan basats en la criptografia de clau asimètrica, utilitzant una component pública i altra privada, i fan ús de funcions unidireccionals [9]. Aquestes funcions presenten unes propietats matemàtiques que asseguren que són senzilles de calcular en un sentit però computacional o temporalment inabastables en sentit contrari.

Sense entrar massa en els detalls matemàtics d'aquest model, Diffie i Hellman proposen la utilització de l'exponenciació modular com a funció unidireccional. Aquesta permet calcular ràpidament el resultat de la seua aplicació. No obstant, per tal de calcular la inversa d'aquesta funció i obtindré els components inicials a partir del resultat caldria resoldre el problema del Logaritme Discret. Encara que existeixen diferents aproximacions que redueixen el temps empleat, com per exemple aplicant la factorització [8], la resolució d'aquest problema segueix sent temporalment inviable, ja que no és possible obtenir la solució en un temps raonable.




**Il·lustració 11: Diagrama de l'intercanvi de claus Diffie-Hellman**

A l'anterior esquema es pot observar la interacció entre dos components per tal de derivar de forma segura una clau simètrica que utilitzaran més tard en l'esquema de criptografia simètrica. Bàsicament, en primer lloc Alice genera la seua clau privada *a* i Bob la seua clau privada *b*. Alice tria els paràmetres *g* i

*p*, nombres primers molt grans, que són el generador i el mòdul utilitzats a la funció unidireccional. Alice calcula la seua clau pública *A* a partir d'aquests paràmetres i la seua clau privada i la transmet a Bob, juntament amb el generador i el mòdul. Bob, amb els paràmetres *g* i *p*, juntament amb la seua clau privada, calcula la seua clau pública *B* i la transmet a Alice.

Amb aquesta informació, ambdós calculen la clau secreta compartida combinant la seua clau privada amb la clau pública de l'altre, obtenint així el mateix valor, tal i com es demostra a la fórmula presentada a la il·lustració. Aquest valor obtingut és el que s'empra per a xifrar i desxifrar la informació en un esquema criptogràfic basat en clau simètrica.

La utilització d'aquest model permet aprofitar els avantatges de la criptografia de clau asimètrica i també els de la de clau simètrica. En primer lloc, permet transmetre informació de forma segura sobre un canal insegur, aprofitant el potencial de la utilització d'una clau pública. I en segon lloc, una vegada derivada la clau, permet utilitzar l'esquema de criptografia de clau simètrica, que ofereix un ràpid xifrat i desxifrat de la informació, estalviant temps i recursos.

## 3.4 Casos d'ús

Per tal de donar una visió clara de les funcionalitats que ofereix el sistema per als usuaris es presenta el següent esquema. Aquest inclou les diferents funcions disponibles al sistema, els diferents tipus d'usuaris d'aquest i la relació entre les funcions i els usuaris, que determina qui pot utilitzar quina funció. Aquestes funcions representades són d'alt nivell i només proporcionen una idea bàsica de què és el que ofereix la solució que es pretén desenvolupar.


**Il·lustració 12: Casos d'us del sistema a desenvolupar**

Es pot observar al diagrama anterior l'existència de dos usuaris. En el cas de l'usuari "Component Agent" no s'està representant el concepte d'usuari com a persona física que interactua amb el sistema, sinó com a entitat consumidora de les funcionalitats oferides pel sistema.

Així doncs, es poden identificar dos tipus de consumidors de funcionalitats, que serien l'usuari del sistema (a través de la Consola) i els components Agent. El primer d'aquests pot realitzar 3 tipus bàsics d'acció: autenticar-se, llistar els Agents disponibles per a ser controlats i interactuar amb un Agent en concret per a configurar algun paràmetre o executar accions. En el cas dels components Agent, aquests poden establir una clau de xifrat, sol·licitar la seua configuració o accions a executar i enviar el resultat de l'execució d'alguna acció.

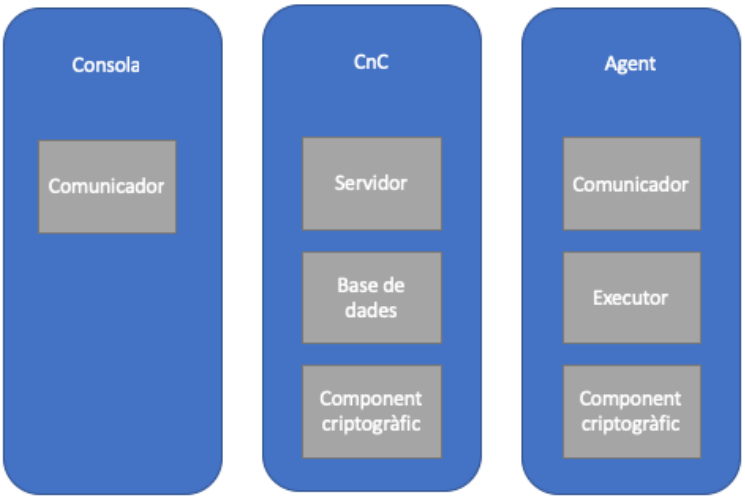
# 4 Disseny de la solució

A partir de l'anàlisi del problema realitzat, en aquest apartat es concreta ja el disseny de la solució. L'objectiu és determinar els aspectes necessaris de cara a la implementació de la solució. Els aspectes més rellevants del disseny comentats en aquest capítol se centren en la modularització dels components del sistema, el protocol utilitzat per a establir la clau de xifrat, els diferents tipus d'interacció entre els components del sistema i el disseny de la base de dades.

## 4.1 Modularització dels components

En aquest apartat es descriu de forma bàsica la modularització dels diferents components del sistema. Aquesta modularització és necessària per tal d'assegurar la independència de cadascuna de les funcionalitats dels components respecte de la seua implementació, així com el complement dels requisits de la solució.

Cadascun dels 3 components principals del sistema, especificats a l'apartat d'arquitectura del capítol anterior, estan formats per diferents mòduls que són els que proporcionen al component totes les funcionalitats que aquest necessita. Açò és possible gràcies a la utilització de la programació modular, que separa les diferents funcionalitats que implementa un component software i les agrupa en diferents mòduls. Es podria dir que cada component principal és només una agrupació de mòduls software que implementen les funcionalitats d'aquest.




**Il·lustració 13: Mòduls bàsics de cada component del sistema**

Al l'anterior esquema es poden observar els diferents mòduls que formen cada component del sistema. Aquests mòduls implementen funcions com, per

exemple, la interacció amb la base de dades, les comunicacions o el xifrat i desxifrat de dades. A més, altres mòduls software modelen característiques pròpies del component principal, com per exemple accions o configuracions. A continuació es detalla la funcionalitat de cadascun dels mòduls representats al diagrama.

- **Mòdul comunicador**: ofereix les funcionalitats necessàries per a establir les comunicacions entre els diferents components del sistema. Aquest està present als components Agent i Consola. Encara que aquests es comuniquen amb el component CnC, aquest no disposa de mòdul comunicador com a tal, sinó que, per les seues característiques i funcionalitats, s'utilitza el mòdul servidor.

- **Mòdul servidor**: és el mòdul utilitzat pel component CnC per a gestionar les comunicacions amb els mòduls Agent i Consola. Així doncs, existeixen dues implementacions per al mòdul servidor, una per a cada tipus de component que ha d'atendre.

- **Mòdul criptogràfic**: ofereix les funcionalitats requerides per a xifrar i desxifrar informació, a més de generar claus públiques i privades. Aquest està present als components Agent i CnC ja que són els que, a més d'utilitzar HTTPS, xifren la informació transmesa.

- **Mòdul executor**: ofereix la funcionalitat necessària per a executar accions. Només el component Agent necessita aquestes funcionalitats, així que només aquest component incorpora el mòdul.

- **Mòdul base de dades**: ofereix la funcionalitat requerida per a interactuar i gestionar la base de dades. L'únic component que necessita la base de dades és CnC, així que només aquest incorpora el mòdul.

## 4.2 Establiment de la clau simètrica de xifrat

Per a la derivació de la clau simètrica entre components Agent i CnC que serà utilitzada per a transmetre's informació de forma segura entre ells es planteja la utilització del protocol d'intercanvi de claus de Diffie-Hellman. Concretament, es proposa la utilització de la versió d'aquest protocol basada en criptografia de corba el·líptica (ECDH). De forma molt general, aquest intercanvi de claus es duu a terme de la següent manera:

1. Tant el component Agent com el component CnC generen un parell de claus (privada i pública) cadascun.

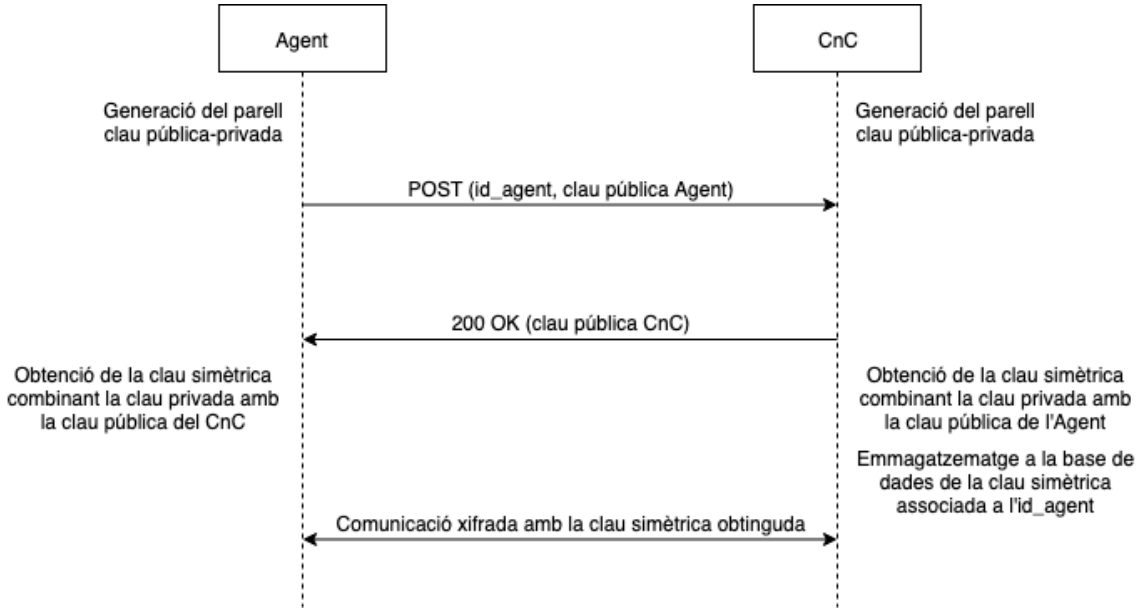
2. El component Agent, una vegada arrancat, envia al component CnC la seua clau pública i el seu identificador.

3. El component CnC rep la informació que el component Agent li ha transmès i combina la clau pública d'aquest amb la seua clau privada,

obtenint un valor que guarda a la seua base de dades associat a l'identificador del component Agent. Una vegada fet açò, el component CnC envia al component Agent la seua clau pública.

4. Una vegada el component Agent ha rebut la clau pública del component CnC la combina amb la seua pròpia clau privada, obtenint com a resultat el mateix valor que el component CnC obté al tercer pas. Només queda guardar aquest valor i, a partir d'aquest moment, utilitzar el valor per a xifrar i desxifrar la informació que comparteixen ambdós components.

A continuació es presenta un esquema bàsic que representa aquest protocol d'intercanvi de claus públiques per tal de derivar una clau simètrica:




**Il·lustració 14: Disseny del protocol d'establiment de la clau de xifrat**

## 4.3 Interacció entre components

En aquest apartat es descriu com interactuen entre ells els diferents components del sistema. Tal i com s'ha pogut veure a l'esquema que mostra l'arquitectura del sistema al capítol anterior, existeixen 2 tipus de comunicacions: les que s'estableixen entre els components Consola i el CnC i les que s'estableixen entre els components Agent i el CnC.

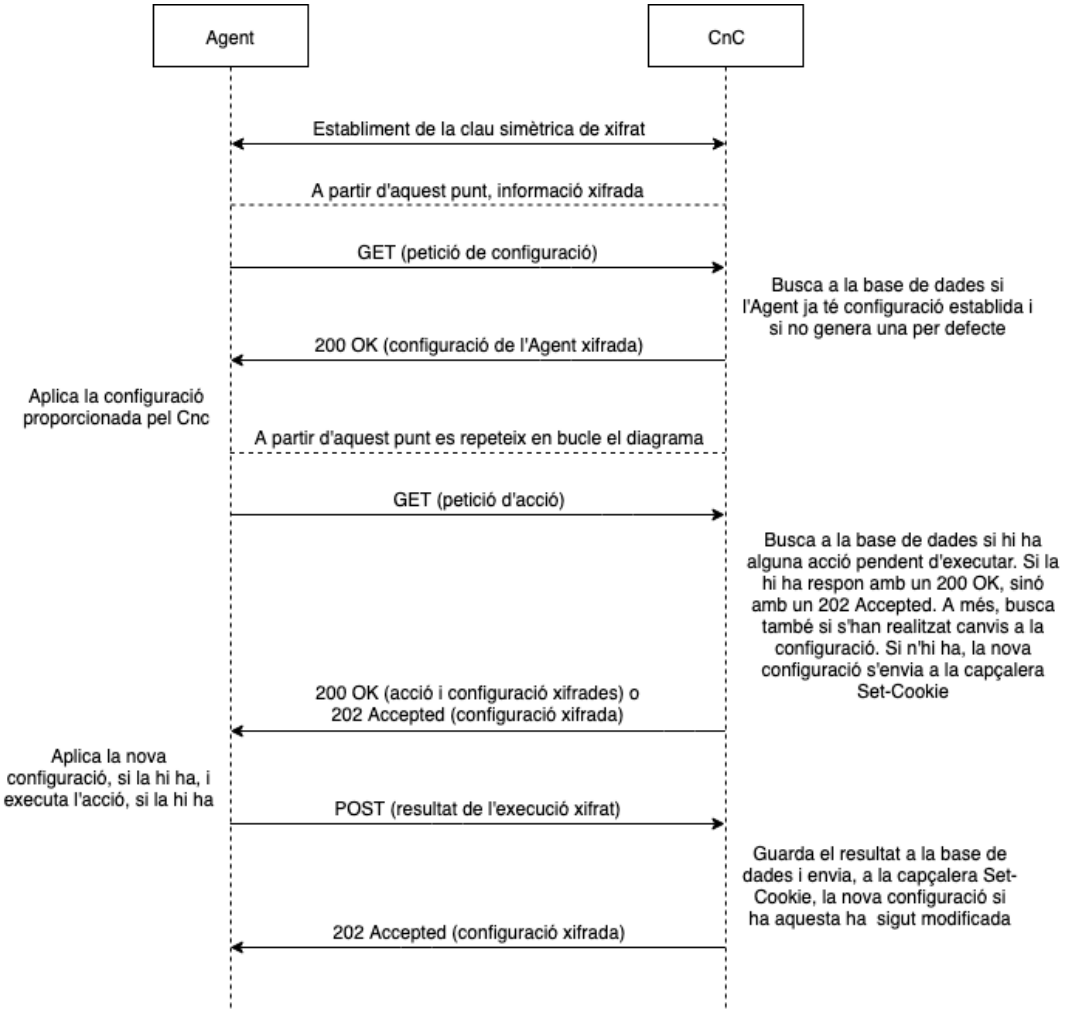
### 4.3.1 Interacció Agent-CnC

La interacció entre Agents i el CnC s'inicia quan s'executa el component Agent. A continuació es mostra el diagrama que representa aquesta interacció i s'exposen els diferents passos que formen part d'aquesta.

1. El component Agent i el CnC deriven la clau simètrica que s'utilitzarà per als processos criptogràfics tal i com s'ha explicat a l'esquema de l'apartat anterior.

2. Una vegada obtinguda aquesta clau simètrica, el component Agent sol·licita al CnC una configuració. Quan rep la resposta del CnC, l'Agent aplica la configuració.

3. Amb l'Agent configurat, aquest comença un bucle de peticions al servidor de comandament i control (CnC) on sol·licita accions a dur a terme. El servidor transmet aquestes accions a l'Agent, el qual les executa, espera un temps (determinat a la configuració) i torna a repetir aquest terces pas.




**Il·lustració 15: Diagrama d'interacció Agent-CnC**

### 4.3.2 Interacció Consola-CnC

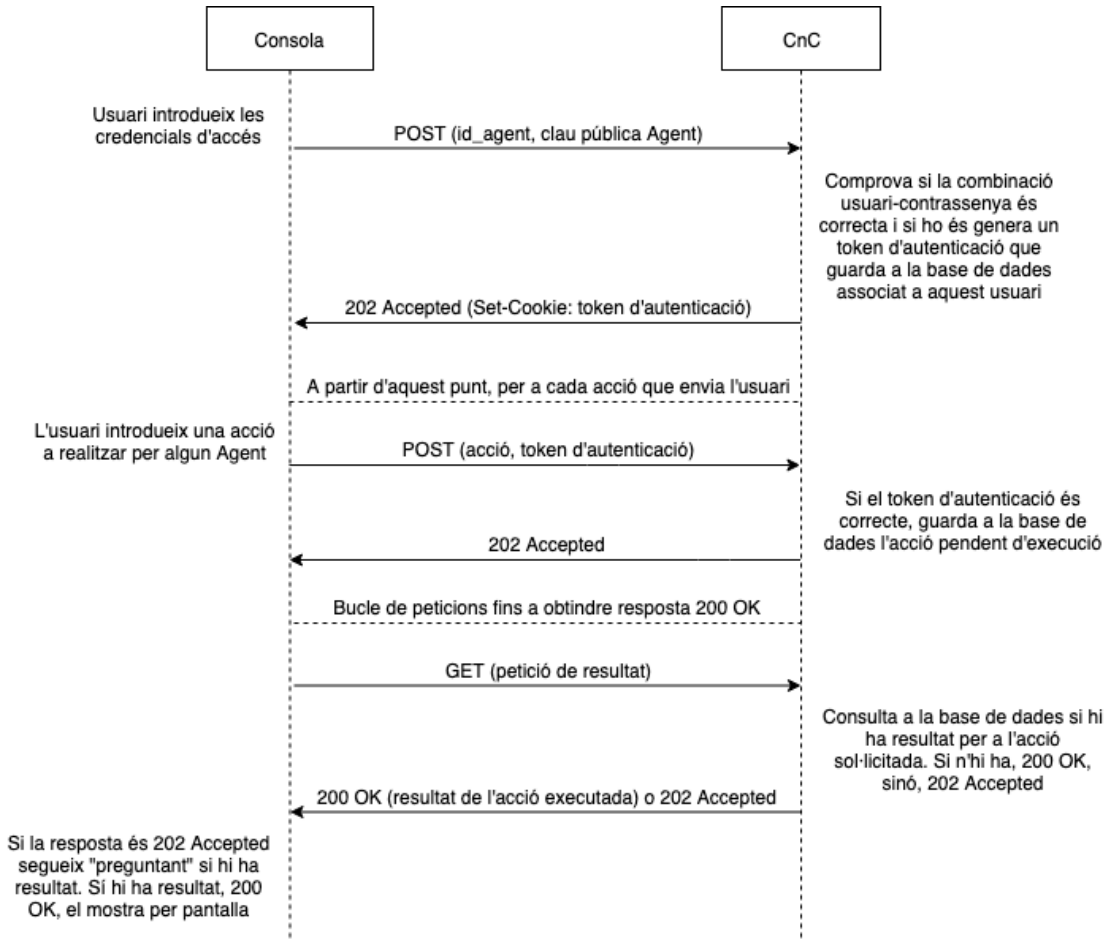
La interacció entre un component Consola i el CnC s'inicia quan s'executa el component Consola. Aquest component, que és utilitzat per l'usuari per tal

d'indicar al CnC quines accions ha de dur a terme un determinat Agent, interactua amb el CnC de la següent forma.

1. El component Consola fa una petició d'autenticació al servidor CnC i aquest, si l'autenticació és vàlida, li proporciona un *token* d'autenticació.

2. Una vegada autenticat, quan un usuari introdueix una acció a executar, el component Consola, utilitzant el *token* d'autenticació del pas anterior, notifica al CnC què és el que ha d'executar un determinat Agent. Aquest pas es repeteix tantes vegades com accions introdueixi l'usuari.

3. El component Consola consulta cada cert temps al CnC si ja disposa del resultat de l'execució de l'acció indicada. La resposta del servidor de comandament i control és el resultat de l'execució de l'acció. Aquest pas es realitza cada vegada que s'executa el pas 2.




**Il·lustració 16: Diagrama d'interacció Consola-CnC**

## 4.4 Disseny de la base de dades

Sobre el disseny de la base de dades, en aquest apartat es comenten els diferents camps i taules necessaris per a representar tota la informació que gestiona el sistema. No s'entra en detall de per a què s'utilitza cada camp, només es proposa l'estructura general de la base de dades i es comenta molt breument el significat de cada camp. Així doncs, les taules i camps necessaris per a la representació de la informació gestionada pel sistema són:

- Taula **agent**: modela la informació que es guarda dels components Agent
    - Camp **id**: identificador únic d'un component Agent
    - Camp **alias**: (opcional) nom assignat a l'Agent
    - Camp **shared_key**: clau simètrica derivada amb l'Agent
    - Camp **active**: indica si l'Agent segueix actiu o no. Es considera que no està actiu si entre l'última petició i el moment actual han passat més de 30 minuts
    - Camp **last_req**: *timestamp* de l'última petició realitzada per l'Agent
- Taula **user**: modela la informació que es guarda dels usuaris del sistema
    - Camp **id**: identificador únic de l'usuari
    - Camp **username**: nom d'usuari (ha de ser únic)
    - Camp **password**: contrasenya de l'usuari
    - Camp **active**: indica si l'usuari ha fet logout o continua actiu al sistema
    - Camp **last_login**: *timestamp* de l'últim login de l'usuari
- Taula **session**: associació entre usuari i agent controlat, a més del *token* d'autenticació per a l'ús del sistema
    - Camp **id**: identificador únic de la sessió
    - Camp **agent**: identificador de l'Agent controlat
    - Camp **user**: identificador de l'usuari autenticat al sistema
    - Camp **token**: *token* d'autenticació de l'usuari

- Taula **action_history**: representació de la informació sobre les diferents accions executades i pendents d'executar o de llegir el resultat.
    - Camp **session**: identificador de la sessió a la que s'associa l'acció
    - Camp **action**: acció a executar
    - Camp **result**: resultat de l'acció executada
    - Camp **completed**: indica si l'acció s'ha executat i es pot llegir ja el resultat d'aquesta
- Taula **configuration**: representació de la informació associada a la configuració dels components Agent
    - Camp **agent**: identificador de l'Agent
    - Camp **req_delay**: temps d'espera entre cada petició d'acció que realitza l'Agent associat
    - Camp **user_agent**: *User-Agent* utilitzat per l'Agent associat
    - Camp **modified**: indica si la configuració ha estat modificada des de l'última vegada que l'Agent la va aplicar

# 5 Implementació

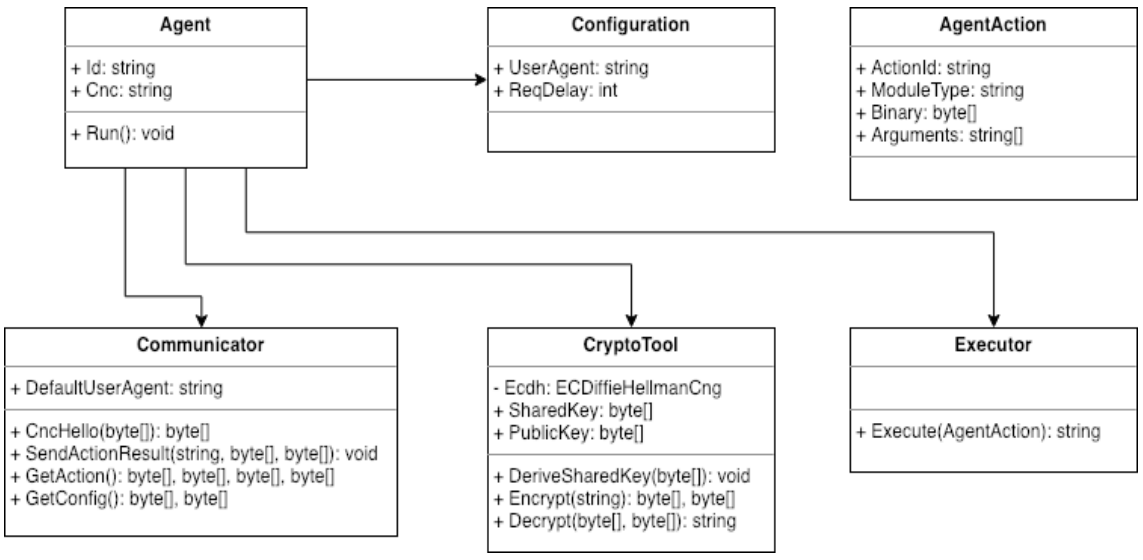
Aquest capítol se centra en descriure la implementació dels diferents components que formen part del sistema, així com els detalls més tècnics que expliquen com es realitza la interacció entre aquests. Els components del sistema, com ja s'ha comentat en capítols anteriors, estan formats per diversos mòduls degut a l'aplicació del desenvolupament modular o basat en components. Sobre les comunicacions, s'ha modelat una interacció molt semblant a la navegació web, els detalls de la qual es comenten en aquest capítol.

## 5.1 Implementació dels components

Tal i com ja s'ha comentat, aquest apartat descriu la implementació de cadascun dels components del sistema. A més d'una descripció detallada de cadascun dels mòduls que formen part d'aquests, es presenta també la representació en forma de diagrama de classes de cada component. D'aquesta forma s'aconsegueix donar una visió completa de la implementació de cada component i dels mòduls de cadascun d'aquests.

### 5.1.1 Agent

A continuació es mostra el diagrama de classes que representa els diferents mòduls que formen part del component Agent, així com les relacions que s'estableixen entre aquests. Més avall es presenta la descripció detallada de cadascun d'aquests mòduls i les funcionalitats que implementen.




**Il·lustració 17: Diagrama de classes del component Agent**

- **Agent**: Mòdul principal del component. Aquest modela el que consideraríem el component Agent de la solució. Bàsicament

implementa el mètode *Run* que arranca tota la funcionalitat de l'Agent. Dintre d'aquest mètode es fa ús de tots els altres mòduls i bàsicament s'autentica amb el servidor CnC, obté la configuració necessària i comença un bucle de peticions al servidor CnC on demana accions per a executar.

- **Communicator**: Aquest mòdul s'encarrega de la comunicació amb el servidor de Comandament i Control. El mètode *CncHello* s'utilitza per a l'autenticació amb el CnC, *GetConfig* obté la configuració per a l'Agent, *GetAction* obté les accions que l'Agent ha d'executar i *SendActionResult* s'utilitza per a enviar al servidor CnC el resultat de les diferents accions executades per l'Agent.

- **CryptoTool**: Aquest mòdul s'encarrega de la criptografia utilitzada a l'Agent. El mètode *DeriveSharedKey* s'utilitza per a obtenir la clau compartida amb el servidor de Comandament i Control a partir de la clau pública d'aquest i *Encrypt* i *Decrypt*, com és previsible, s'utilitzen per a xifrar i desxifrar, respectivament, utilitzant la clau simètrica derivada amb el CnC.

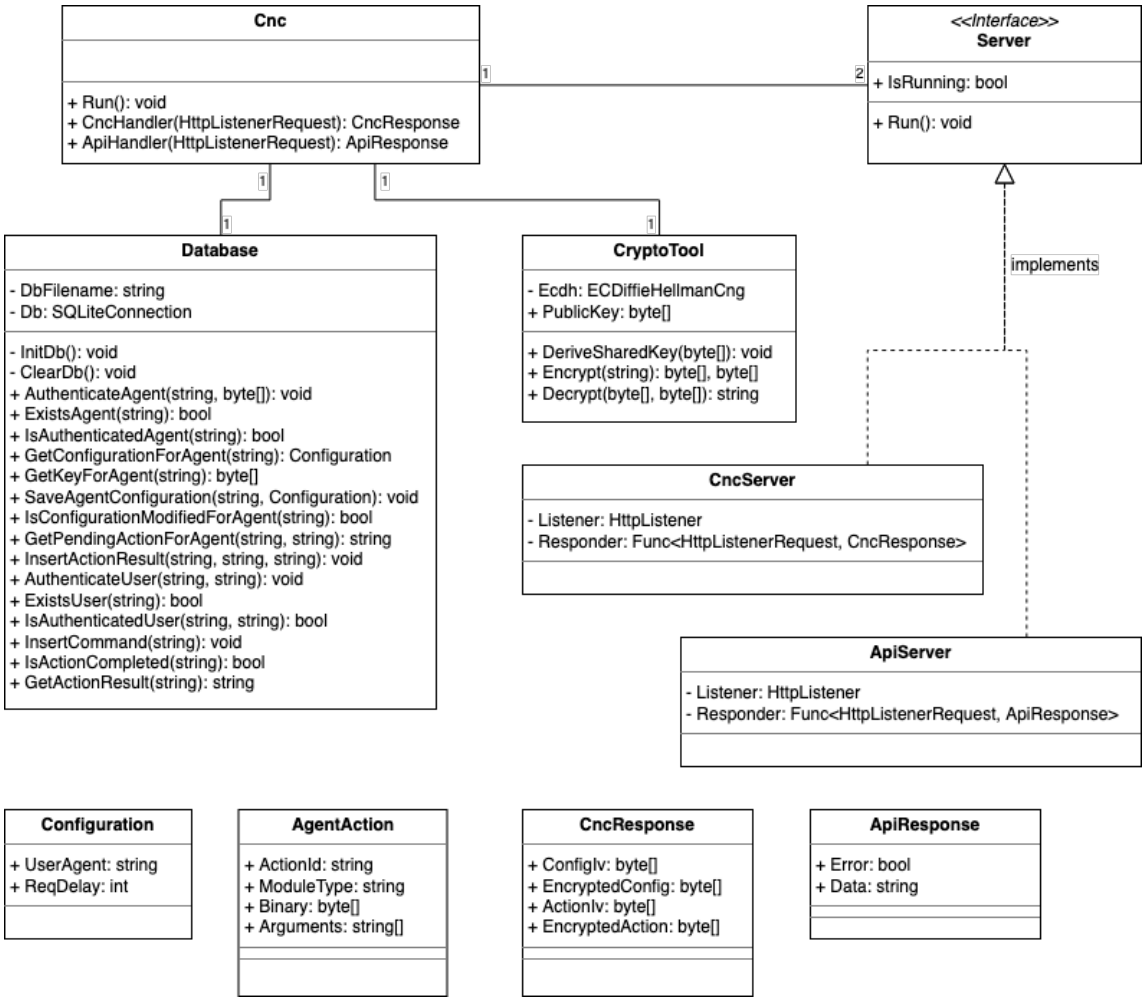
- **Executor**: És el mòdul encarregat d'executar les diferents accions que encarrega el servidor CnC. Només implementa el mètode *Execute* al qual se li proporciona un objecte *AgentAction* i retorna el resultat de l'execució d'aquest.

- **AgentAction**: Aquest mòdul modela les accions executables per un Agent. Inclou el binari, els arguments necessaris per a l'execució i altres dades com un identificador de l'acció i el tipus d'aquesta.

- **Configuration**: Aquest mòdul modela els aspectes més rellevants de la configuració d'un Agent. En aquest cas s'utilitzen només un *UserAgent*, per a les peticions web, i el temps d'espera entre cada petició que realitza l'Agent.

### 5.1.2 CnC

A continuació es mostra el diagrama de classes que representa els diferents mòduls que formen part del component CnC, així com les relacions que s'estableixen entre aquests. Més avall es presenta la descripció detallada de cadascun d'aquests mòduls i les funcionalitats que implementen.




**Il·lustració 18: Diagrama de classes del component CnC**

- **Cnc**: Aquest és el mòdul principal del component i representa el que considerem el servidor de Comandament i Control de la solució. El seu mètode *Run* arranca totes les funcionalitats d'aquest. Als mètodes *CncHandler* i *ApiHandler* s'implementen les funcionalitats necessàries per a atendre les peticions dels Agents i de les Consoles, respectivament.

- **Database**: És el mòdul encarregat d'interactuar amb la base de dades. En aquest cas s'utilitza una base de dades SQLite, degut a la fàcil integració d'aquesta amb C#, el llenguatge emprat per a programar la solució. Bàsicament, cadascun dels mètodes implementats en aquest mòdul donen solució a la necessitat d'obtenir o guardar determinada informació a la base de dades, fent possible al Cnc gestionar la informació necessària per al funcionament del servidor de Comandament i Control.

- **CryptoTool**: A l'igual que amb el mòdul CryptoTool de l'Agent, aquest s'encarrega d'implementar la criptografia necessària per al funcionament del servidor de Comandament i Control.

- **Server**: Aquesta interfície declara les funcionalitats i propietats que han d'implementar els diferents servidors web utilitzats al servidor CnC, que són els *CncServer* i *ApiServer*.

- **CncServer**: Aquest servidor implementa la comunicació que s'estableix amb els Agents. Aquest utilitza el mètode *CncHandler* implementat al mòdul principal Cnc per a generar les respostes que s'envien als Agents.

- **ApiServer**: Aquest servidor implementa la comunicació que s'estableix amb les Consoles. Aquest utilitza el mètode *ApiHandler* implementat al mòdul principal Cnc per a generar les respostes que s'envien al les Consoles.

- **Configuration**: A l'igual que amb el mòdul *Configuration* de l'Agent, aquest modela la configuració aplicable als components Agent.

- **AgentAction**: A l'igual que amb el mòdul *AgentAction* de l'Agent, aquest modela les accions que poden executar els components Agent.

- **CncResponse**: Aquest mòdul representa les respostes que s'envien als Agent. Incorpora la informació de la configuració i accions xifrades i els seus respectius vectors d'inicialització, utilitzats per a desxifrar la informació.

- **ApiResponse**: Aquest mòdul modela les respostes que s'envien als components Consola. Incorpora informació sobre si s'han produït errors i la informació que retorna el servidor de Comandament i Control.

### 5.1.3 Consola

La implementació del component Consola queda fora dels objectius de l'actual projecte. Tal i com es comenta a l'apartat d'objectius, en aquest projecte sí que es dissenya el sistema pensat per a la utilització d'aquest tercer component, però no es desenvolupa. No obstant, en aquest apartat es comenten els aspectes més representatius a tindre en compte per a la implementació d'aquest component.

El component Consola és el que presenta la funcionalitat més senzilla d'implementar. Aquest pot ser implementat en qualsevol llenguatge d'scripting, com per exemple Python o Bash. Independentment de la seua implementació, l'únic aspecte que s'ha de tenir en compte és que la informació que es transmet al CnC ha d'estar en el format correcte. És a dir, simplificant molt, només s'ha de tenir en compte que aquest component i el CnC s'han d'entendre.

La implementació d'aquest component deuria incloure un login, per tal de realitzar l'autenticació de l'usuari. Una vegada autenticat l'usuari, aquest ha de poder seleccionar un Agent per a controlar i enviar les accions o configuracions necessàries. La implementació d'aquest component ofereix tantes possibilitats

que es possible, fins i tot, implementar-lo amb una interfície gràfica o una interfície web per a facilitar el seu ús a usuaris no familiaritzats amb les interfícies de consola.

## 5.2 Implementació de les comunicacions

En aquest apartat es presenten els detalls més tècnics de la implementació de la comunicació entre els diferents components del sistema. En primer lloc es parla de la interacció entre els components Agent i CnC i posteriorment Consola i CnC. A continuació es presenten els detalls de la implementació per a cadascuna d'aquestes interaccions, estructurant-les tenint en compte la finalitat de la comunicació i marcant entre claus {} els diferents valors que s'utilitzen.

### 5.2.1 Comunicació entre Agent i CnC

La interacció que s'estableix entre l'Agent i el servidor de Comandament i Control es pot dividir en 5 tipus d'intercanvi d'informació. Aquests intercanvis d'informació són: establiment de la clau de xifrat, configuració de l'Agent, execució d'accions i enviament dels resultats d'aquestes execucions. A continuació es mostra el detall tècnic dels missatges que s'intercanvien aquests dos components.

#### 5.2.1.1 Establiment de la clau simètrica de xifrat

- Estructura de la petició:

```
POST /hello HTTP/1.1
User-Agent: {User-Agent per defecte}
Cookie: PHPSESSID={ID de l'Agent};

{Clau pública de l'Agent codificada en Base64}
```

- Estructura de la resposta:

```
HTTP/1.1 200 OK
{Clau pública del CnC codificada en Base64}
```

#### 5.2.1.2 Obtenció de la configuració de l'Agent

- Estructura de la petició:

```
GET /conf HTTP/1.1
User-Agent: {User-Agent per defecte}
Cookie: PHPSESSID={ID de l'Agent};
```

- Estructura de la resposta:

```
HTTP/1.1 200 OK
Set-Cookie: {Vector d'inicialització codificat en Base64};

{Configuració serialitzada, xifrada amb la clau simètrica
derivada amb l'Agent i codificada en Base64}
```

#### 5.2.1.3 Obtenció d'accions per a executar

- Estructura de la petició:

```
GET /{cadena de text aleatòria que serveix com a
identificador d l'acció} HTTP/1.1
User-Agent: {User-Agent establit a la configuració de l'Agent}
Cookie: PHPSESSID={ID de l'Agent}
```

- Estructura de la resposta:
    - Si hi ha acció disponible per a executar, codi de resposta 200 OK, sinó 202 Accepted
    - Si la configuració ha estat modificada i ha d'aplicar-se de nou a l'Agent, capçalera *Authentication* amb valors iv per al vector d'inicialització per a desxifrar la configuració codificat en Base64 i c per a la configuració serialitzada, xifrada amb la clau simètrica per a l'Agent i codificada en Base64
    - Si hi ha acció disponible per a executar, capçalera *Set-Cookie* inclou el vector d'inicialització per a desxifrar l'acció codificat en Base64 i el body de la resposta inclou l'acció serialitzada, xifrada amb la clau simètrica per a l'Agent i codificada en Base64

```
HTTP/1.1 {200 OK o 202 Accepted}
Authentication: iv={Vector d'inicialització per a desxifrar la
configuració codificat en Base64}; c={Configuració
serialitzada, xifrada amb la clau simètrica i codificada en
Base64};
Set-Cookie: {Vector d'inicialització per a desxifrar l'acció
codificat en Base64}

{Acció per a executar serialitzada, xifrada amb la clau
simètrica i codificada en Base64}
```

#### 5.2.1.4 Enviament del resultat de l'execució d'una acció

- Estructura de la petició:

```
POST /result/{ID de l'acció que s'ha executat} HTTP/1.1
User-Agent: {User-Agent establit a la configuració de l'Agent}
Cookie: PHPSESSID={ID de l'Agent}
Authentication: {Vector d'inicialització per a desxifrar el
resultat de l'execució codificat en Base64}

{Resultat de l'execució xifrat amb la clau simètrica de l'Agent i
codificat en Base64}
```

- Estructura de la resposta:

```
HTTP/1.1 {200 OK o 202 Accepted}
Authentication: iv={Vector d'inicialització per a desxifrar la
configuració codificat en Base64}; c={Configuració
serialitzada, xifrada amb la clau simètrica i codificada en
Base64};
```

### 5.2.2 Comunicació entre Consola i CnC

La interacció que s'estableix entre la Consola i el servidor de Comandament i Control és bastant més senzilla que la interacció Agent-CnC. En aquest cas només es realitza un tipus d'intercanvi d'informació, que es mostra a continuació.

#### 5.2.2.1 Establiment de la clau simètrica de xifrat

- Estructura de la petició:

```
POST / HTTP/1.1

{
  authentication: {Token d'autenticació de l'usuari},
  data: {Informació que s'envia al CnC per a realitzar diferents
accions}
}
```

- Estructura de la resposta:

```
HTTP/1.1 200 OK
```

```
{
  error: {1 si s'ha produït algun error o 0 si no},
  data: {Informació sobre el resultat obtingut}
}
```

## 5.3 Implementació de la base de dades

En aquest apartat es presenten els detalls de la implementació de la base de dades utilitzada pel servidor de comandament i control. En primer lloc es presenta la tecnologia utilitzada per a la implementació d'aquesta, es parla sobre les seues característiques i es justifica la seua elecció. En segon lloc es mostra la implementació interna de la base de dades dissenyada a l'anterior capítol, on s'especifiquen detalls com el tipus de dades, claus primàries, etc.

### 5.3.1 SQLite + DB Browser for SQLite

SQLite és una base de dades relacional continguda en una biblioteca programada en C. Aquesta presenta una clara diferència respecte als sistemes de gestió de bases de dades tradicionals, com per exemple Oracle o MySQL i és que aquests estan basats en una arquitectura client-servidor mentre que SQLite s'integra directament en les aplicacions que la implementen.


**Il·lustració 19: Logotip d'SQLite**

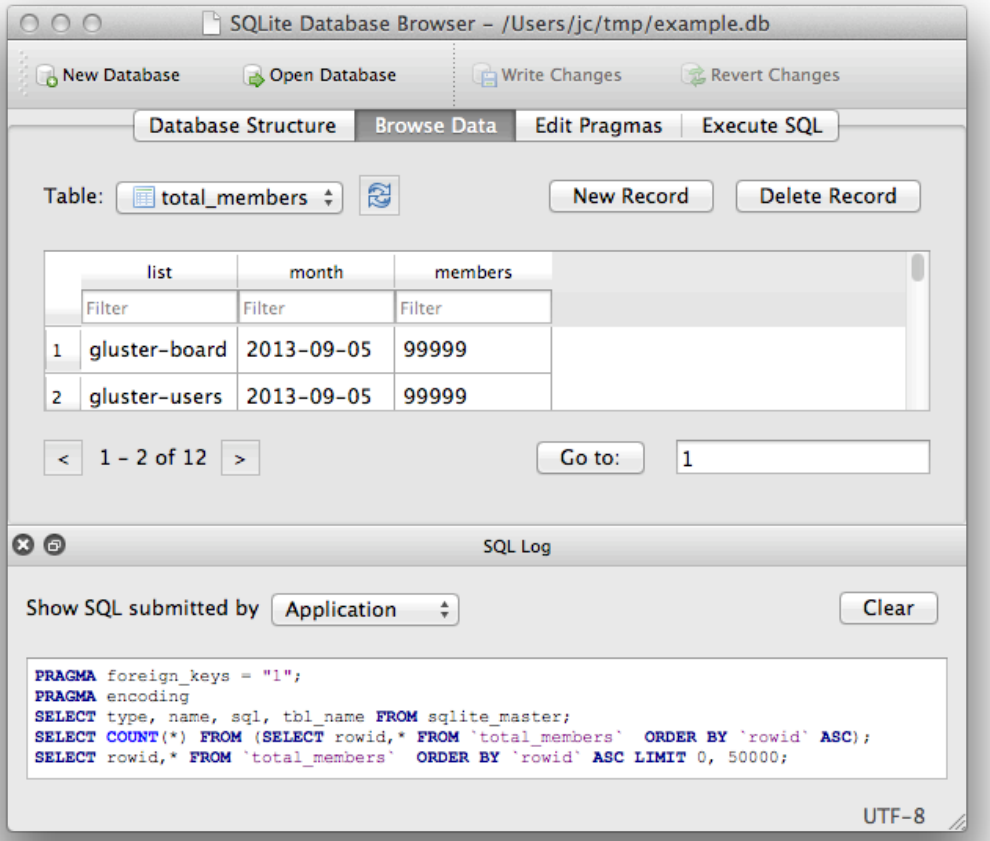
Les bases de dades SQLite poden arribar a tenir fins a 2 terabytes de tamany. A més, aquestes permeten treballar amb camps de tipus BLOB, que representen dades extenses en format binari. Tota la informació gestionada per aquestes bases de dades s'emmagatzema en un únic fitxer.

SQLite treballa amb el llenguatge per excel·lència de les bases de dades relacionals: SQL. Aquesta implementa moltes de les funcionalitats oferides per aquest llenguatge, com per exemple les transaccions atòmiques, la consistència, l'aïllament, els triggers o les consultes complexes.

En aquest projecte s'ha decidit implementar la base de dades utilitzant SQLite degut a la seua fàcil integració amb el llenguatge de programació emprat. A més, la informació gestionada pel servidor CnC no és tan extensa com per a haver d'utilitzar altres bases de dades que busquen la optimització de temps o recursos. Per tot açò, i tenint en compte que SQLite implementa

totes les funcions requerides a la implementació del CnC s'ha escollit aquesta base de dades per a la implementació de la solució.

DB Browser for SQLite és una eina de codi lliure que permet carregar els arxius de base de dades d'SQLite i consultar l'estructura, contingut, etc. de la pròpia base de dades. A més, ofereix una forma molt visual, en forma de taules, de representar la informació de la base de dades i permet, a més, modificar aquesta informació i aplicar els canvis.




**Il·lustració 20: Interfície de DB Browser for SQLite**

Aquesta aplicació s'ha utilitzat, tant en el procés de desenvolupament com en les proves de funcionament, per tal de verificar la correcta estructura de la base de dades, així com per a poder accedir i manipular les dades contingudes en aquesta. Per exemple, durant les proves d'execució de les ordres per part dels Agents, s'ha utilitzat aquesta eina per tal d'inserir a la base de dades la informació necessària per tal de que s'enviï l'ordre a executar a l'Agent corresponent. A més, una vegada executada l'ordre, aquesta eina s'ha utilitzat també per a verificar la correcta recepció i emmagatzematge a la base de dades del resultat de l'execució.

### 5.3.2 Estructura de la base de dades

En aquest subapartat es presenta l'estructura implementada a la base de dades. En primer lloc es mostra un esquema de les taules de la base de dades, així com els diferents camps que contenen i el tipus d'aquests. Una vegada presentat l'esquema, es detallen les diferents sentències SQL utilitzades per a generar i inicialitzar la base de dades.

L'estructura detallada de la base de dades és la següent:

- Taula **agent**
  - Camp **id**: tipus text, clau primària
  - Camp **alias**: tipus text
  - Camp **shared_key**: tipus text, no pot ser nul
  - Camp **active**: tipus integer, no pot ser nul, ha de ser 0 o 1
  - Camp **last_req**: tipus text, no pot ser nul
- Taula **user**
  - Camp **id**: tipus integer, clau primària
  - Camp **username**: tipus text, no pot ser nul
  - Camp **password**: tipus text, no pot ser nul
  - Camp **active**: tipus integer, no pot ser nul, ha de ser 0 o 1
  - Camp **last_login**: tipus text
- Taula **session**
  - Camp **agent**: tipus text, clau primària, clau secundària
  - Camp **user**: tipus text, clau primària, clau secundària
  - Camp **token**: tipus text, clau primària
- Taula **command_history**
  - Camp **agent**: tipus text, clau primària, clau secundària
  - Camp **action_id**: tipus text
  - Camp **action**: tipus text, no pot ser nul
  - Camp **completed**: tipus integer, no pot ser nul, 0 o 1
  - Camp **result**: tipus text

- Taula **configuration**
    - Camp **agent**: tipus text, clau primària, clau secundària
    - Camp **user_agent**: tipus text
    - Camp **req_delay**: tipus integer, no pot ser nul
    - Camp **modified**: tipus integer, no pot ser nul, ha de ser 0 o 1

A més d'aquestes taules, i amb la finalitat de mantenir actualitzat el camp *active* de la taula *agent*, que indica si un Agent segueix actiu o no, s'han definit dos triggers que s'executen quan es realitza una inserció o actualització d'algun registre de la taula *agent*. Així doncs, el resum d'aquests triggers és el següent:

- Actualització o inserció d'un registre a la taula *agent*
    - S'executa després de realitzar-se l'acció que el desencadena
    - Comprova si la diferència entre el valor del camp *last_req* i l'hora actual és major de 30 mins.
    - Si es compleix la condició, estableix el valor del camp *active* a 0

A continuació es mostren les diferents sentències SQL executades per tal de crear i inicialitzar la base de dades del servidor de comandament i control:

```
CREATE TABLE IF NOT EXISTS agent (id TEXT PRIMARY KEY, alias TEXT, shared_key TEXT NOT NULL, active INTEGER NOT NULL CHECK (active == 0 OR active == 1), last_req TEXT NOT NULL);

CREATE TABLE IF NOT EXISTS user (id INT PRIMARY KEY, username TEXT NOT NULL, password TEXT NOT NULL, active INTEGER NOT NULL CHECK (active == 0 OR active == 1), last_login TEXT);

CREATE TABLE IF NOT EXISTS session (agent TEXT, user INT, token TEXT, PRIMARY KEY (agent, user, token), FOREIGN KEY (agent) REFERENCES agent(id), FOREIGN KEY (user) REFERENCES user(id));

CREATE TABLE IF NOT EXISTS command_history (agent TEXT PRIMARY KEY, action_id TEXT, action TEXT NOT NULL, completed INTEGER NOT NULL CHECK (completed == 0 OR completed == 1), result TEXT, FOREIGN KEY (agent) REFERENCES agent(id));

CREATE TABLE IF NOT EXISTS configuration (agent TEXT PRIMARY KEY, user_agent TEXT, req_delay INT NOT NULL, modified INTEGER NOT NULL CHECK (active == 0 OR active == 1), FOREIGN KEY (agent) REFERENCES agent(id));
```

```
CREATE TRIGGER active_agent_update AFTER INSERT ON agent BEGIN
UPDATE agent SET active=(last_req >= DATETIME('now', '-30 minutes'));
END;

CREATE TRIGGER active_agent_insert AFTER UPDATE ON agent BEGIN
UPDATE agent SET active=(last_req >= DATETIME('now', '-30 minutes'));
END;
```

## 5.4 Evasió d'AV i solucions de seguretat

En aquest apartat es parla de les diferents solucions de seguretat que poden estar implantades a la màquina o a la xarxa víctimes. Una vegada presentada la solució de seguretat, es parla de les diferents tècniques implementades per tal d'evadir la detecció i catalogació de l'Agent com a una amenaça per part d'aquestes solucions de seguretat.

En primer lloc es parla de la integració de les aplicacions de Windows amb l'antivirus o el servei antimalware. En segon lloc es parla dels sistemes de detecció d'intrusions en host (màquines d'usuari o servidors). Per últim, es tracten els sistemes de detecció d'intrusions en xarxa. A més de comentar cadascun d'aquests, es presenten les mesures implementades per tal d'evadir aquestes tecnologies.

### 5.4.1 AMSI

El concepte AMSI fa referència a l'*Antimalware Scan Interface* desenvolupada per Microsoft per al seu sistema operatiu Windows [11]. AMSI és una interfície que permet la integració entre les diferents aplicacions i serveis del sistema i el programari antimalware (o antivirus) instal·lat al sistema.

Aquesta interfície permet desacoblar la implementació de les diferents aplicacions per al sistema operatiu Windows de la implementació de cadascuna de les solucions antimalware existents. Així doncs, qualsevol aplicació que implementi aquesta interfície podrà utilitzar determinades funcions del programari antimalware instal·lat a l'equip, independentment de quin sigui aquest.

AMSI ofereix diferents funcionalitats per a les aplicacions que la implementen. Destaquen la possibilitat d'analitzar arxius, buffers de memòria, streams, IPs i URLs, etc. A més, aquesta treballa utilitzant sessions, la qual cosa li permet correlar diferents events i obtenir conclusions com a resultat de l'anàlisi d'un conjunt d'anàlisis, i no només d'un simple anàlisi d'un fitxer, URL, etc.

AMSI està implementada en diferents aplicacions natives de Windows. Aquestes són l'UAC (*User Access Control*), PowerShell, Windows Script Host, JavaScript, VBScript i les macros VBA per a Office. Aquestes aplicacions, i qualsevol altra que implemente aquesta interfície, poden invocar les funcions

AmsiScanBuffer o AmsiScanString, entre altres, per tal de realitzar una anàlisi d'un buffer de memòria o d'un string emprant el programari antimalware del sistema i determinar si es tracta d'una amenaça de seguretat.

Així doncs, és necessari que el component Agent no sigui detectat per les eines de seguretat disponible, entre les que s'inclou AMSI. Com s'implementa l'evasió d'AMSI? Bé, existeixen moltes maneres d'evadir la detecció d'AMSI. No obstant, en aquest projecte s'ha optat per l'execució de codi que suposa una amenaça per a la seguretat del sistema directament en memòria.

Açò és degut a que les versions anteriors a la 4.8 de .NET Framework no realitzen una anàlisi del buffer de memòria que es carrega a l'utilitzar la funció Assembly.Load(Byte[]) [10]. Açò permet que el codi maliciós pugui carregar-se i executar-se directament en memòria sense que el programari antimalware l'analitze per tal de determinar si conté alguna amenaça per a la seguretat, sempre i quan la versió de .NET Framework sigui menor que 4.8, la qual cosa és molt habitual.

### 5.4.2 HIDS

El concepte HIDS fa referència a *Host-based Intrusion Detection System*. Bàsicament, es tracta d'un programari la funció del qual és revisar les activitats d'una màquina en busca d'anomalies. En el cas que aquestes anomalies suposen un risc potencial per al sistema, aquest és capaç de prendre mesures per tal de protegir la seguretat del sistema.

Aquests sistemes són capaços de detectar canvis en determinats fitxers del sistema, o registres, unitats de xarxa, etc. A més, també poden detectar dins del propi sistema les firmes de programari maliciós ja conegut. Açò suposa que, en el moment que alguna solució de seguretat catalogui l'Agent com a programari maliciós, generi una firma de detecció d'aquest i la transmeti a les altres solucions de seguretat, qualsevol HIDS implantat a qualsevol màquina serà capaç d'identificar i prendre mesures sobre l'executable de l'Agent.

Una bona forma d'evitar que es puguin obtenir firmes de detecció de l'executable és dissenyar i implementar un programari mutable. Per al desenvolupament del projecte, considerem mutable aquell executable que és capaç de canviar la seua "aparença" o "forma" davant d'un HIDS, encara que no ho faci automàticament.

Per tal de dotar a l'Agent d'aquesta capacitat de mutar, en aquest projecte s'ha utilitzat la programació modular, que, com ja s'ha explicat, consisteix en dissenyar la solució de tal forma que les seues funcionalitats estiguin separades en diferents mòduls i que aquests siguin independents de la seua implementació. Aquesta independència el que permet és la reimplementació o modificació de qualsevol mòdul de l'Agent.

Amb aquesta capacitat d'intercanviar els diferents mòduls s'entén que l'executable és mutable, ja que pot ser recompilat havent substituït la implementació d'un o més mòduls d'aquest, evadint així la detecció basada en firmes que aplica un HIDS. Per exemple, si una solució de seguretat marca

com a maliciós l'executable i la firma de detecció inclou el seu hash MD5, si es modifica la implementació d'un mòdul i es recompila l'executable el seu hash MD5 serà diferent, aconseguint així evitar la detecció d'un HIDS.

### 5.4.3 NIDS

El concepte NIDS fa referència a *Network-based Intrusion Detection System*. A l'igual que els HIDS, la funció d'aquests també consisteix en la identificació d'anomalies, però en aquest cas a la xarxa. Un sistema NIDS analitza, en temps real, tot el tràfic de xarxa i és capaç de detectar atacs de denegació de servei, escanejos de ports, intents d'explotació de vulnerabilitats, etc.

Existeixen dos tipus diferents de sistemes NIDS: els basats en regles i els basats en anomalies. Els primers disposen, a l'igual que els HIDS, en un conjunt extensible de regles de detecció i, bàsicament, el que fan és buscar coincidències entre aquestes regles i el tràfic de xarxa que estan analitzant. Donat que aquests només comproven regles per a detectar amenaces ja conegudes, no són capaços d'identificar atacs o intrusions que utilitzen noves tècniques que no hagen sigut firmades encara.

Els segons, en canvi, no disposen d'aquest conjunt de coneixement, sinó que són capaços d'aprendre l'estat normal d'una xarxa i, a partir d'eixa informació, buscar anomalies en el comportament o l'estat d'aquesta. Per exemple, si en una oficina no es treballa mai diumenge i de sobte un diumenge es registra tràfic de xarxa entre X equips, un NIDS basat en anomalies registraria aquesta activitat i generaria una alerta. En canvi, un NIDS basat en regles, si no disposa d'una regla específica que identifiqui aquest comportament, el sistema no seria capaç d'identificar-lo per ell mateix.

Per tal d'evadir la detecció per qualsevol dels tipus de sistema de detecció d'intrusos basat en xarxa, s'han aplicat diferents tècniques al disseny i implementació de la solució desenvolupada en aquest projecte. En primer lloc, per tal d'evitar que un NIDS sigui capaç d'analitzar el contingut del tràfic de xarxa intercanviat entre el servidor de comandament i control i l'Agent, les comunicacions s'han implementat mitjançant una connexió SSL. Com que el tràfic de xarxa està xifrat d'extrem a extrem, un NIDS que intercepti aquest tràfic no serà capaç d'entendre'l i analitzar-lo degut a que està xifrat.

No obstant, existeixen organitzacions que "trenquen" el tràfic SSL. És a dir, el tràfic de xarxa que utilitza SSL es xifra entre la màquina client i el NIDS. Aquest el desxifra, l'analitza, el torna a xifrar i l'envia a la seua destinació. Així s'aconsegueix que el NIDS pugui analitzar tot el tràfic de la xarxa, estigui xifrat o no. En aquest cas, per tal d'evitar generar anomalies que puguin fer sospitar a un NIDS del tràfic intercanviat entre el servidor de comandament i control i l'Agent, aquests simulen una comunicació HTTP normal i corrent. S'ha camuflat el tràfic amb aquest protocol perquè és el que major percentatge d'aparició té en una xarxa normal.

A més, per tal d'evitar la generació de firmes que puguin identificar i marcar aquest tràfic, s'ha implementat de tal forma que les peticions realitzades

per l'Agent al CnC incorporen un camp generat aleatòriament. Açò dificulta, encara que no impossibilita, la creació de regles que puguin diferenciar propietats de la comunicació entre l'Agent i el CnC i puguin ser utilitzades per a detectar l'activitat d'un Agent a la xarxa.

Per últim, i donat que molts NIDS són capaços d'identificar i detectar executables o codi maliciós en el contingut dels diferents paquets interceptats s'ha implementat el xifrat de la informació transmesa entre el servidor de comandament i control i l'Agent mitjançant una clau simètrica establida entre aquests. Així doncs, d'aquesta forma s'aconsegueix obtenir els avantatges que proporciona l'ús de SSL encara que l'organització estigui aplicant el trencament d'SSL, ja que en aquest cas el xifrat s'aplica d'extrem a extrem (Agent-CnC).

# 6 Proves

Degut a l'aplicació del procés de desenvolupament iteratiu i incremental, part del procés de prova de la solució es realitza en paral·lel al propi desenvolupament d'aquesta. És a dir, mentre van desenvolupant-se cadascuna de les funcionalitats del sistema aquestes van provant-se i validant-se. Per tal de donar una visió general d'aquest procés de proves, aquest apartat es divideix en dos subapartats: el primer parla sobre l'entorn de proves sobre el que es valida la solució i el segon sobre les principals proves que s'han realitzat.

## 6.1 Entorn de proves

Per a les proves de la solució proposada s'utilitzen dos entorns diferents, un per al component Agent i l'altre per al servidor de Comandament i Control. Com que el projecte busca obtenir una eina de control remot per a màquines Windows, ambdós entorns de prova són Windows, com cabia esperar.

### 6.1.1 Sistema

En el cas de l'Agent, que, recordem, és el component que s'executa a la màquina víctima, s'ha decidit realitzar les proves sobre una màquina virtual Windows 10 actualitzada a la última versió. Aquesta està connectada a una subxarxa local que no té connexió a Internet.

S'ha decidit utilitzar aquest sistema operatiu perquè és el més nou ofert per Microsoft i és el més utilitzat entre les organitzacions i particulars. S'impedeix que la màquina tingui connexió a Internet amb la finalitat d'evitar que pugui enviar mostres dels arxius analitzats per l'antivirus a Microsoft. Açò és convenient ja que el sistema desenvolupat està pensat per a utilitzar-se en projectes de test d'intrusió al món laboral i, per això, es juga amb un avantatge si Microsoft, o qualsevol altre proveïdor de serveis de seguretat, no disposa de firmes de la solució desenvolupada.

Per a les proves del servidor de Comandament i Control s'ha decidit emprar una altra màquina virtual Windows Server 2012 R2, també connectada a la mateixa subxarxa que la màquina Windows 10 amb la finalitat de que tinguin visibilitat entre elles. S'ha decidit utilitzar aquest sistema operatiu a pesar de que no és el més actual degut a que ja s'havia treballat amb ell anteriorment i suposava un avantatge conèixer un poc més del seu funcionament. A més, i com que no forma part dels objectius del projecte, treballar amb aquest sistema operatiu ha facilitat la configuració del mateix degut al coneixement previ que ja es tenia sobre aquest.

### 6.1.2 Antivirus

En quant a la solució de seguretat, o antivirus, utilitzada per a les proves s'ha emprat *McAfee Endpoint Security with Adaptative Threat Protection* versió 10.6.0. S'ha decidit utilitzar aquesta ferramenta ja que ofereix una versió gratuïta de prova i és també una solució àmpliament utilitzada a les organitzacions, tal i com s'ha pogut comprovar amb l'experiència com a professional de la seguretat.

Aquesta solució de seguretat s'ha instal·lat a la màquina Windows 10 amb la finalitat de comprovar si el component Agent és capaç d'evadir la detecció de l'antivirus. A més de l'anàlisi d'arxius, *McAfee Endpoint Security* incorpora altres eines de seguretat, com per exemple el *firewall* o l'eina de control de contingut web. Aquestes es poden veure i configurar al panell d'administració de l'antivirus, que és el que es mostra a continuació.


**Il·lustració 21: Panell d'administració de l'antivirus McAfee**

Alguns aspectes rellevants de la configuració de l'antivirus són:

- Anàlisi en temps real
- Prevenció d'exploit
- Integració amb AMSI

## 6.2 Proves de funcionament

Sobre les proves de funcionament de la solució cal comentar que s'han intentat comprovar tots els casos de prova possible. A continuació s'enumeren els més significants, agrupats pel tipus de component al qual afecten. A més, s'afegeix un últim subapartat que recull els resultats de les diferents proves, evidenciant el correcte funcionament del sistema implementat.

### 6.2.1 Agent

Per al component Agent s'han realitzat les següents proves:

- **Primera execució**: cal comprovar que genera correctament el seu identificador i el guarda a la clau de registre corresponent
- **Execucions posteriors**: cal comprovar que l'identificador que s'assigna i envia és el mateix que va generar a la primera execució i que està guardat al registre de Windows
- **Execució d'accions**: cal comprovar que realment s'executen les accions a la màquina víctima (Windows 10), per exemple, obrint una finestra del navegador, etc.

### 6.2.2 CnC

- **Primera execució**: cal comprovar que es crea i s'inicialitza la base de dades
- **Execucions posteriors**: cal comprovar que carrega adequadament la informació de la base de dades
- **Execució d'accions**: cal comprovar que el servidor de comandament i control obté el resultat de les accions executades als Agent i el guarda correctament a la base de dades

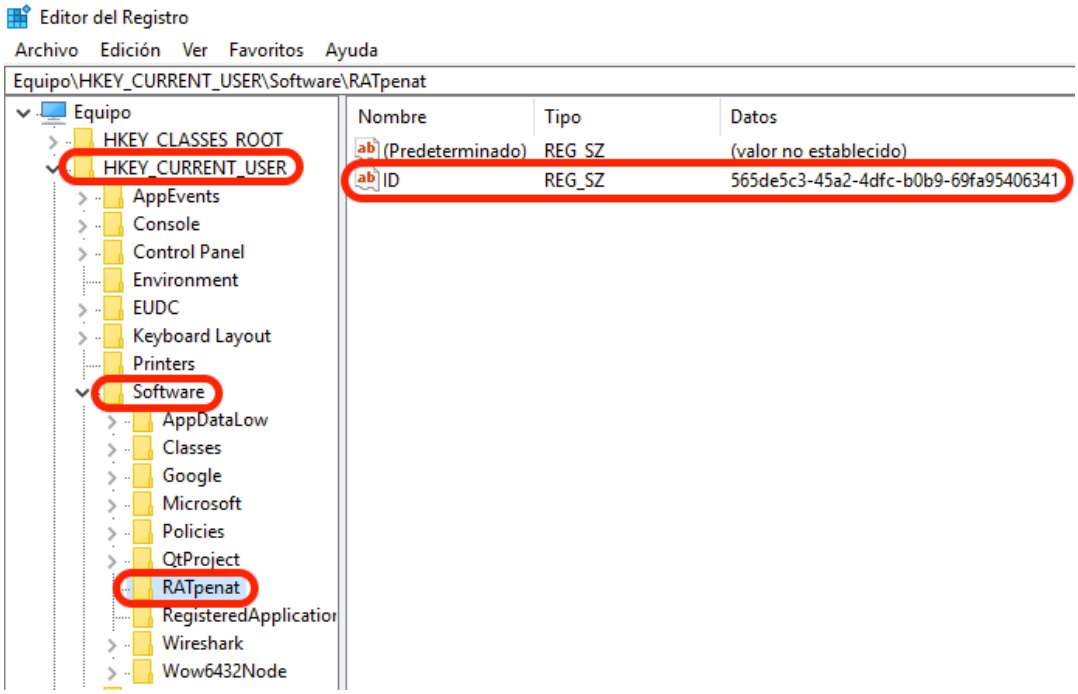
### 6.2.3 Comunicacions

- Cal comprovar que les comunicacions, per defecte, s'estableixen sobre un canal segur, HTTPS, i que no es pot conèixer el contingut d'aquesta comunicació perquè aquesta està xifrada, degut a la utilització del protocol SSL
- Cal comprovar el format de les peticions i respostes que s'intercanvien els components és el correcte i que no és possible conèixer el contingut perquè està xifrat amb la criptografia de clau simètrica, per exemple, utilitzant el protocol HTTP i interceptant el tràfic

### 6.2.4 Evidències

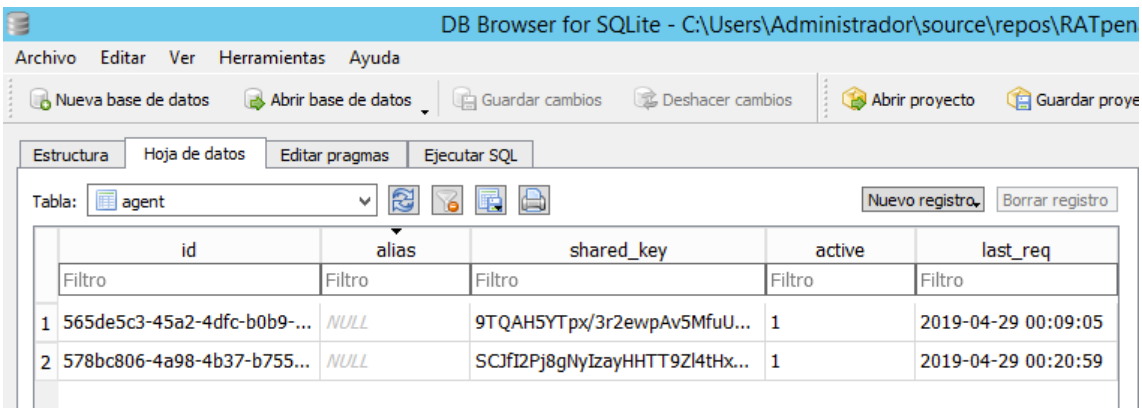
La presentació de les diferents evidències està ordenada de forma que resulti senzill entendre el funcionament del sistema. Cal recordar que l'execució del component Agent s'ha realitzat amb l'antivirus deshabilitat per tal d'evitar interferències en les proves de funcionament.

En primer lloc, després de l'execució del component Agent, es pot veure a la següent captura de pantalla que aquest guarda (o consulta, si no és la primera execució) al registre de Windows el seu identificador.



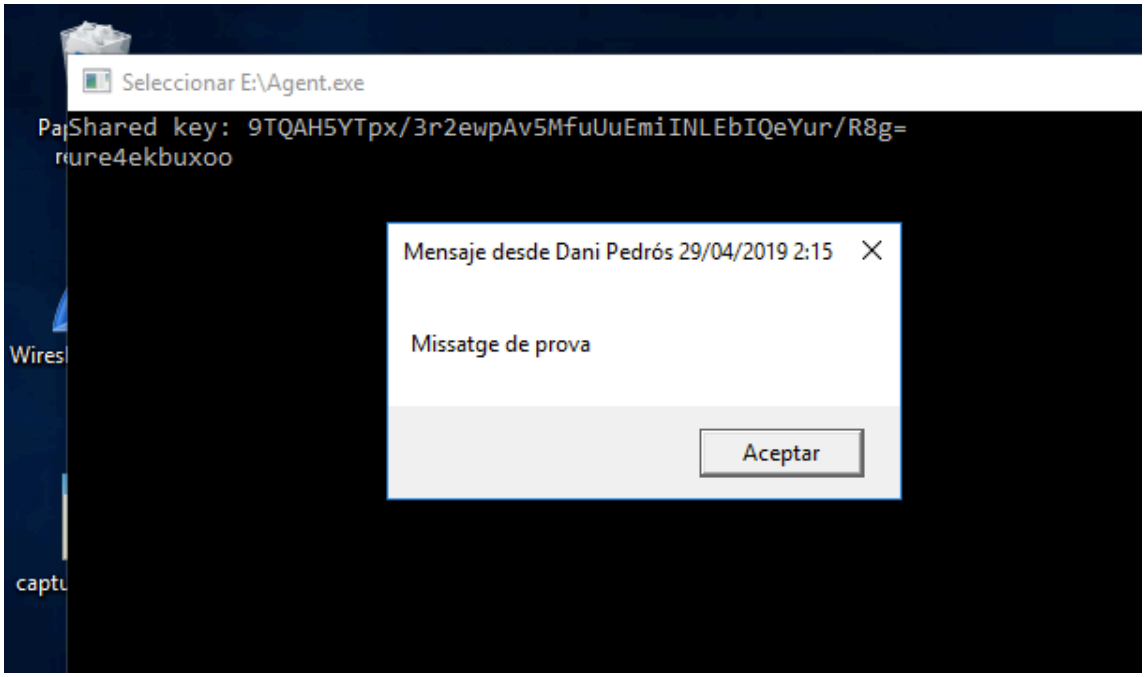

**Il·lustració 22: Registre de la màquina víctima amb l'identificador de l'Agent**

Veiem a la base de dades del component CnC, el servidor de comandament i control, que ha quedat registrat l'Agent amb l'identificador que es pot observar a l'anterior imatge i que disposa ja d'una clau simètrica, codificada en format Base 64.




**Il·lustració 23: Registre de l'Agent a la base de dades del CnC**

Una vegada introduïda alguna ordre per a que un determinat Agent l'executi, aquesta s'emmagatzema a la base de dades, com es mostra a una imatge que es comenta més endavant. Quan l'Agent demana les accions que té pendents d'executar, a aquesta entrada de la base de dades se li assigna un identificador i s'envia a l'Agent, el qual l'executa, com es pot comprovar a la següent captura de pantalla.

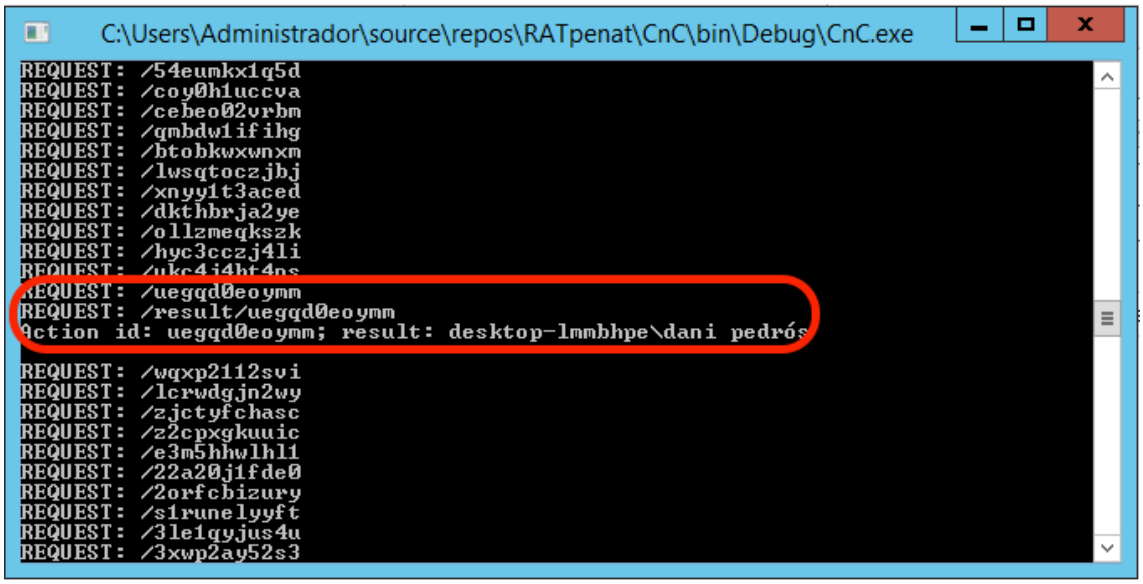



**Il·lustració 24: Execució correcta d'una acció enviada pel CnC**

A l'anterior evidència, a més de comprovar que l'ordre s'executa correctament a la màquina víctima, es pot observar a la consola de darrere la clau derivada amb el component CnC. Si es consulta la segona evidència d'aquest apartat, es pot comprovar que el component CnC té la mateixa clau guardada a la base de dades i associada a aquest Agent.

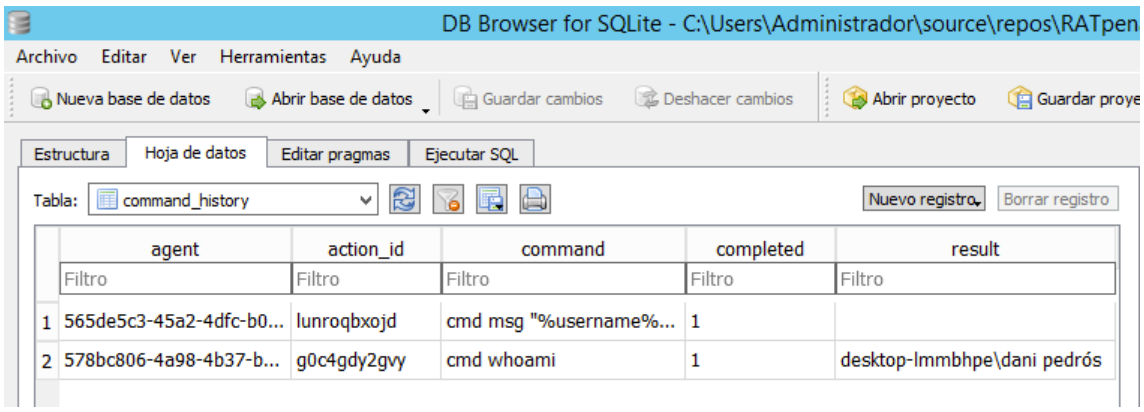
A la següent captura de pantalla es pot observar la recepció, per part del component CnC, del resultat de l'execució d'una acció a un segon Agent. A la imatge s'observen les diferents peticions que rep el component CnC de l'Agent, on aquest li consulta si disposa d'alguna acció per a executar. Marcada en roig apareix la petició d'acció on s'envia a l'Agent l'acció a executar, seguida de la petició POST on l'Agent envia el resultat de l'execució d'aquesta al CnC i, per últim, l'escriptura a la consola del resultat.

Recordar que la consola que es mostra a la imatge és la del servidor CnC i no la de l'Agent on s'executen les ordres, la qual cosa evidència la correcta recepció per part del CnC dels resultats de l'execució de les accions als diferents Agents.




**Il·lustració 25: Peticions rebudes al CnC i resultat d'execució d'una acció**

A la següent evidència es demostra que el resultat de les accions es guarda correctament a la base de dades, a l'espera de que la Consola llegeixi aquest resultat. Així doncs, també s'observa la correcta associació entre l'Agent que l'executa, l'identificador de l'acció, quina acció és i l'indicador de si ja ha estat completada o no.

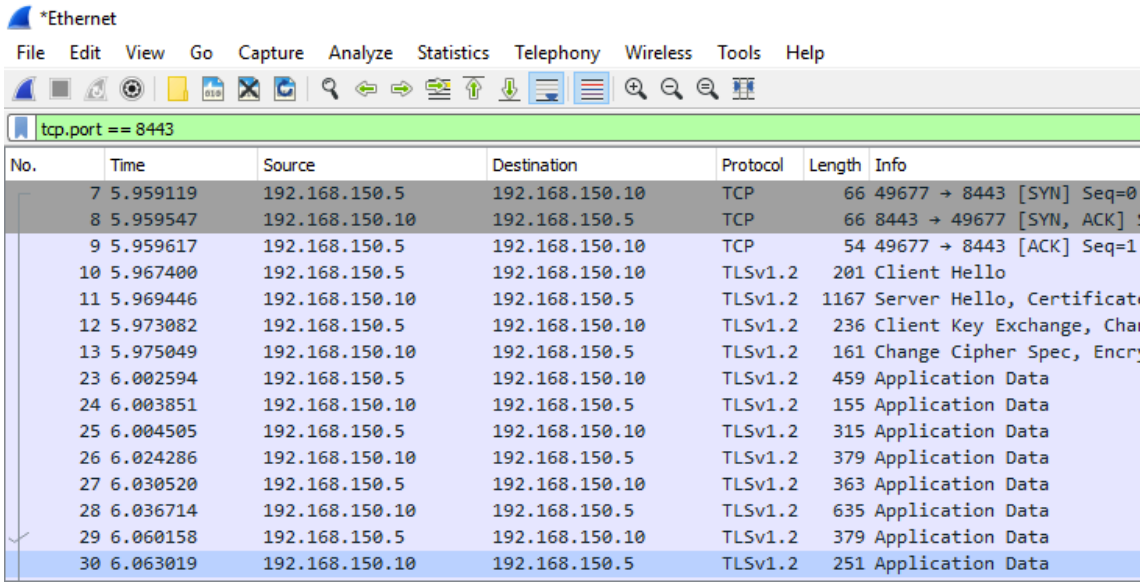



**Il·lustració 26: Resultat de l'execució d'accions a la base de dades**

En aquesta última evidència es pot comprovar que l'identificador de l'acció es correspon al vist a les peticions de l'evidència quarta, així com també que el resultat de l'execució és el mateix que hi apareix en aqueixa evidència. Amb aquests exemples s'ha demostrat el correcte funcionament del sistema.

No obstant, és possible aprofundir més encara en la comprovació del correcte funcionament de la solució. Per a això cal enfocar les proves des de la perspectiva de la comunicació entre l'Agent i el CnC. A les següents evidències es mostren els detalls de la comunicació interceptada entre aquests dos components.

En primer lloc es mostra que la comunicació es realitza completament mitjançant HTTPS i, per tant, és impossible veure el contingut d'aquesta. A la següent captura de pantalla es mostra la comunicació establida entre l'Agent (amb IP 192.168.150.5) i el servidor CnC (amb IP 192.168.150.10), on es veu clarament l'establiment de la connexió TCP, el Hello del protocol SSL i, a partir d'ací, paquets amb informació d'aplicació, que no és més que informació que WireShark no és capaç d'interpretar perquè està xifrada.

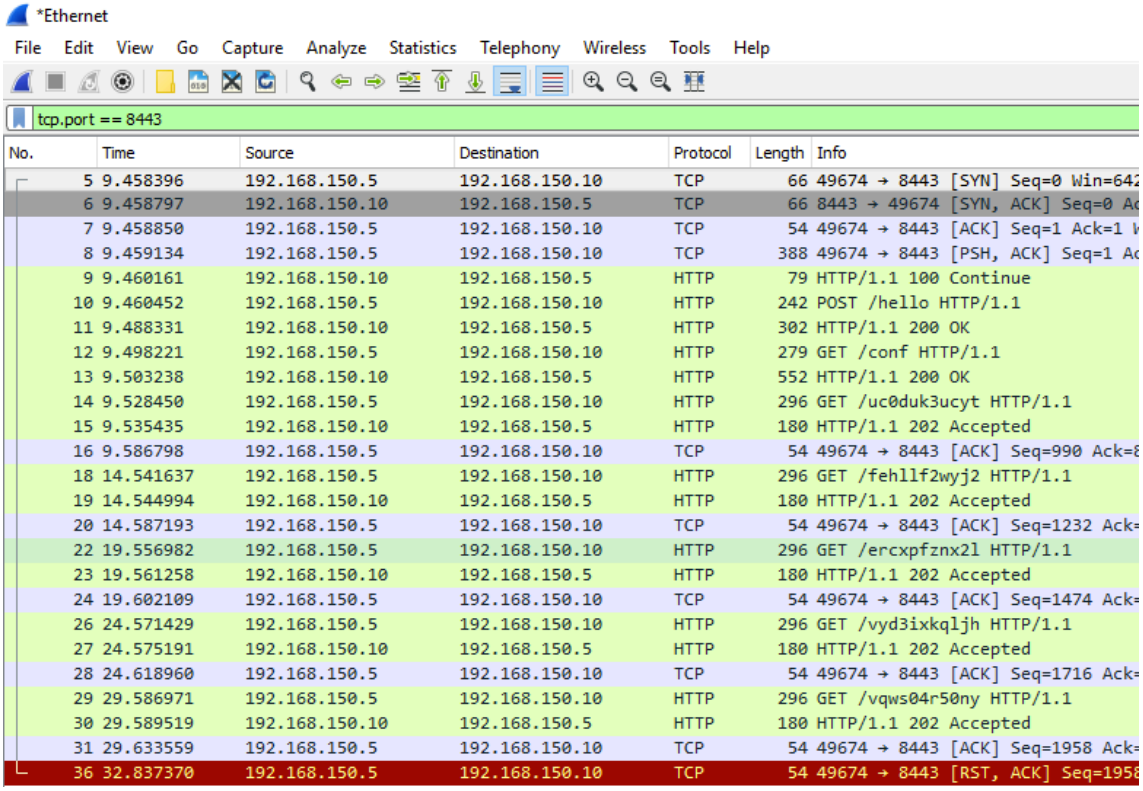



**Il·lustració 27: Tràfic HTTPS entre Agent i CnC**

Per tal d'evidenciar l'estructura de les peticions que realitza el component Agent i les respostes del CnC, s'han modificat aquests per a que estableixin la comunicació mitjançant HTTP en lloc de HTTPS. Al no estar xifrades les peticions, a la següent captura de pantalla és possible identificar les diferents peticions HTTP explicades al capítol d'implementació.

Es poden observar tots els tipus d'interacció definits. Després de l'establiment de connexió TCP es pot veure la petició POST per a l'establiment de la clau simètrica de xifrat. Seguidament, una petició GET on es demana la configuració que l'Agent ha d'aplicar. I, per últim, a partir d'aquest punt, un bucle de peticions a un recurs aleatori que serveix com a identificador de la possible acció a executar.

Amb aquesta evidència es demostra que la solució implementada compleix amb els detalls d'implementació detallats en aquest capítol. Des dels diferents passos definits fins a l'estructura de cada petició i resposta intercanviades per aquests components, la solució implementada s'ajusta al detall de la solució proposada.



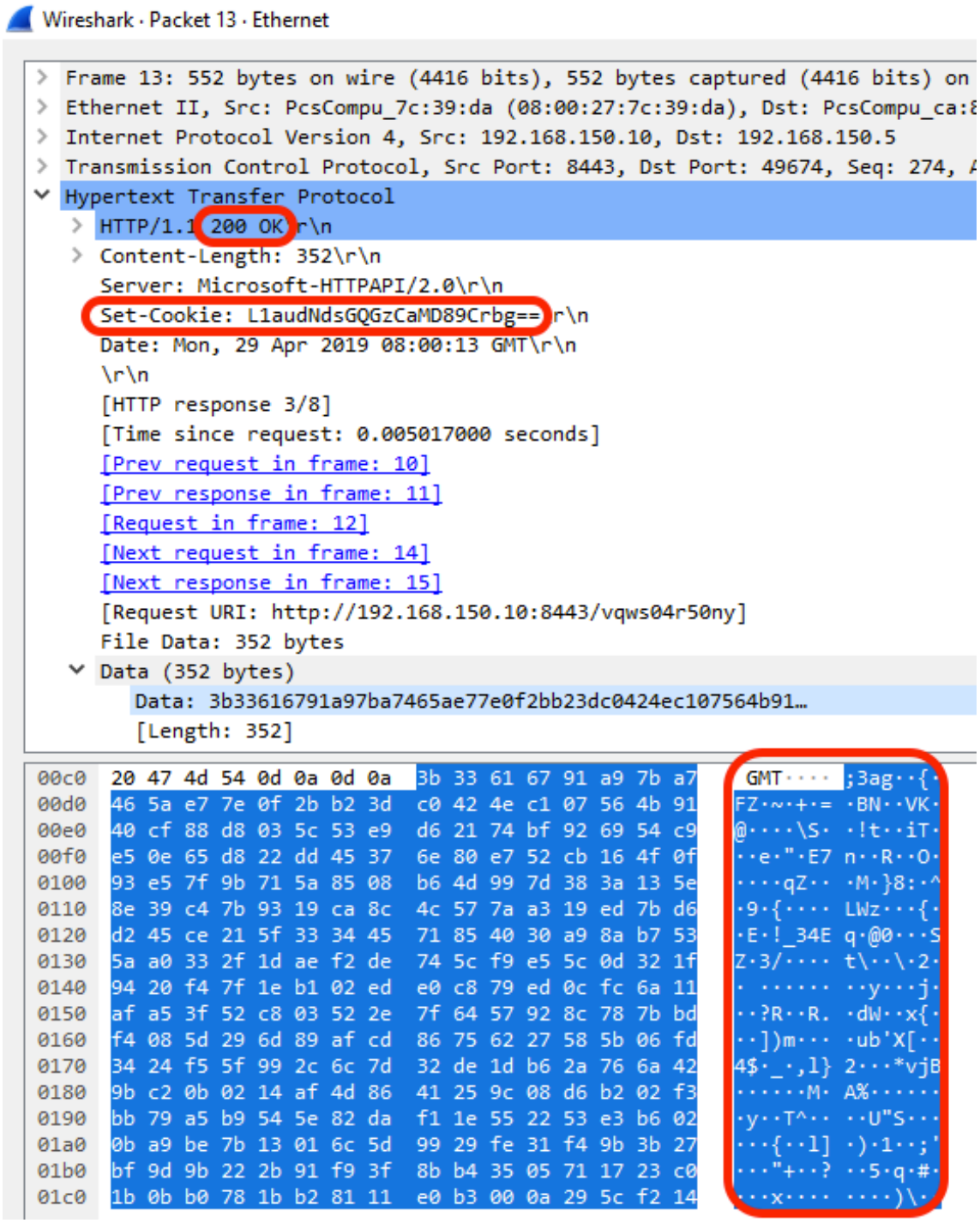

**Il·lustració 28: Prova de tràfic HTTP entre Agent i CnC**

La intercepció de tràfic HTTP permet inspeccionar el contingut de cadascuna d'aquestes peticions. Per tal de demostrar que s'aplica correctament el xifrat de la informació que es transmeten Agent i CnC es pot accedir a la resposta a la petició GET /conf, on el servidor CnC envia a l'Agent la configuració que aquest ha d'aplicar.

Si es consulta el body d'aquesta petició es pot comprovar que conté informació que , aparentment a simple vista, no té ningun sentit. Açò és degut a que, a pesar que s'ha forçat l'ús d'HTTP per a que no es xifren les comunicacions, la informació que comparteixen l'Agent i el CnC es xifra amb la clau simètrica que aquests dos components han establert en primer lloc.

Es poden veure el contingut de les capçaleres HTTP, però no es pot obtenir la informació, en aquest cas, de la configuració transmesa. D'igual manera, tampoc podria obtenir-se la informació relativa a les accions a executar per part d'un Agent o al resultat de l'execució d'aquestes.

Amb aquesta última evidència es demostra que la informació utilitzada tant pels Agent com pel CnC es xifra utilitzant la clau compartida entre aquests. Açò proporciona la seguretat de que, encara que la solució de defensa tingui la capacitat de trencar el túnel segur HTTPS, aquesta no podrà conèixer de ninguna manera la informació que es transmeten ambdós components. Açò permet minimitzar les possibilitats de que un NIDS, per exemple, detecti eines sospitoses circulant per la xarxa.




**Il·lustració 29: Configuració xifrada al body d'una resposta HTTP**

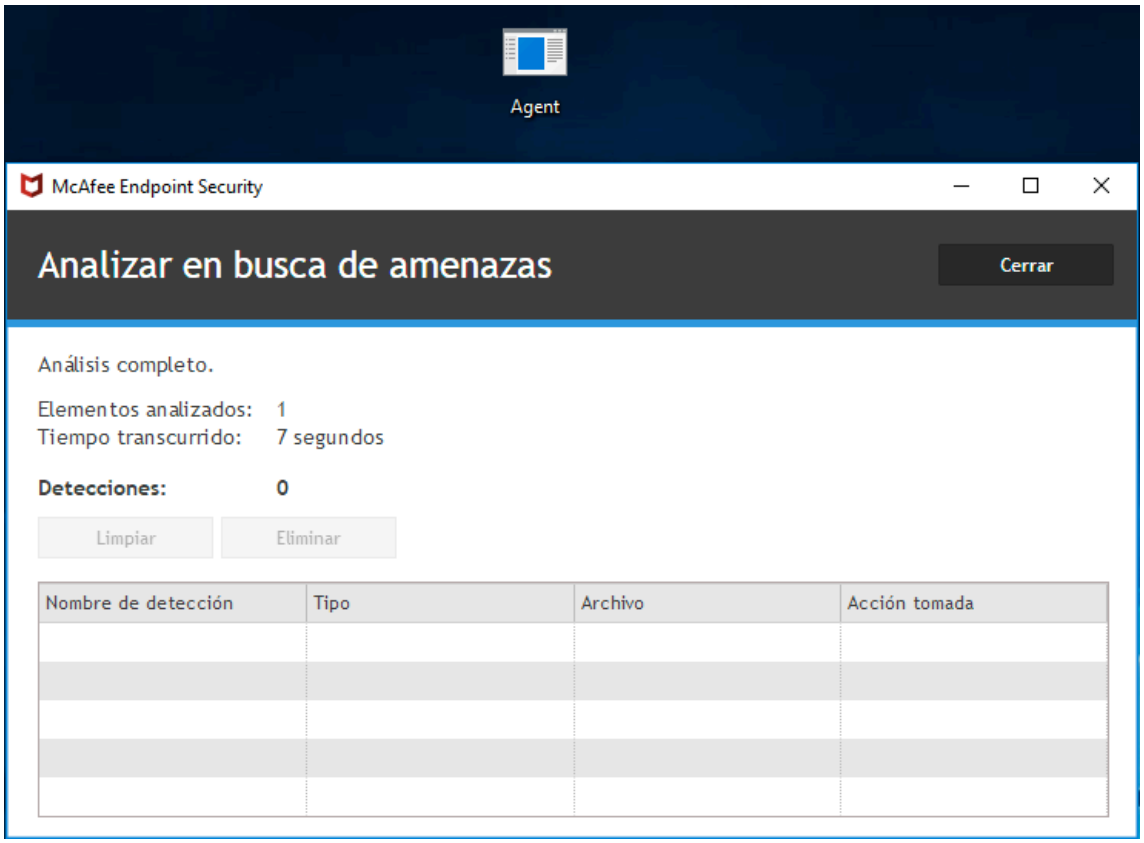
## 6.3 Proves de detecció

En aquest apartat es parla de les proves realitzades per tal d'evidenciar que la solució desenvolupada evadeix efectivament la detecció de l'antivirus. El primer subapartat se centra en comprovar que no salta ninguna alerta a l'antivirus instal·lat quan s'executa l'Agent a la màquina víctima i aquest es comunica amb el servidor de comandament i control. El segon subapartat

presta atenció a comprovar que és possible executar eines externes detectables per l'antivirus, sense que aquestes siguin detectades.

### 6.3.1 Execució de l'agent

Per tal de demostrar que el component Agent de la solució implementada no és detectable per l'antivirus, en primer lloc s'ha realitzat una anàlisi sobre l'executable generat. Al menú contextual (botó dret del ratolí) s'ha seleccionat l'opció oferta per McAfee i s'ha realitzat una anàlisi del binari. El resultat d'aquesta anàlisi ha resultat negatiu, tal i com es mostra a la següent captura de pantalla, la qual cosa demostra que l'executable pot emmagatzemar-se en disc sense que l'antivirus detecti ninguna amenaça de seguretat.

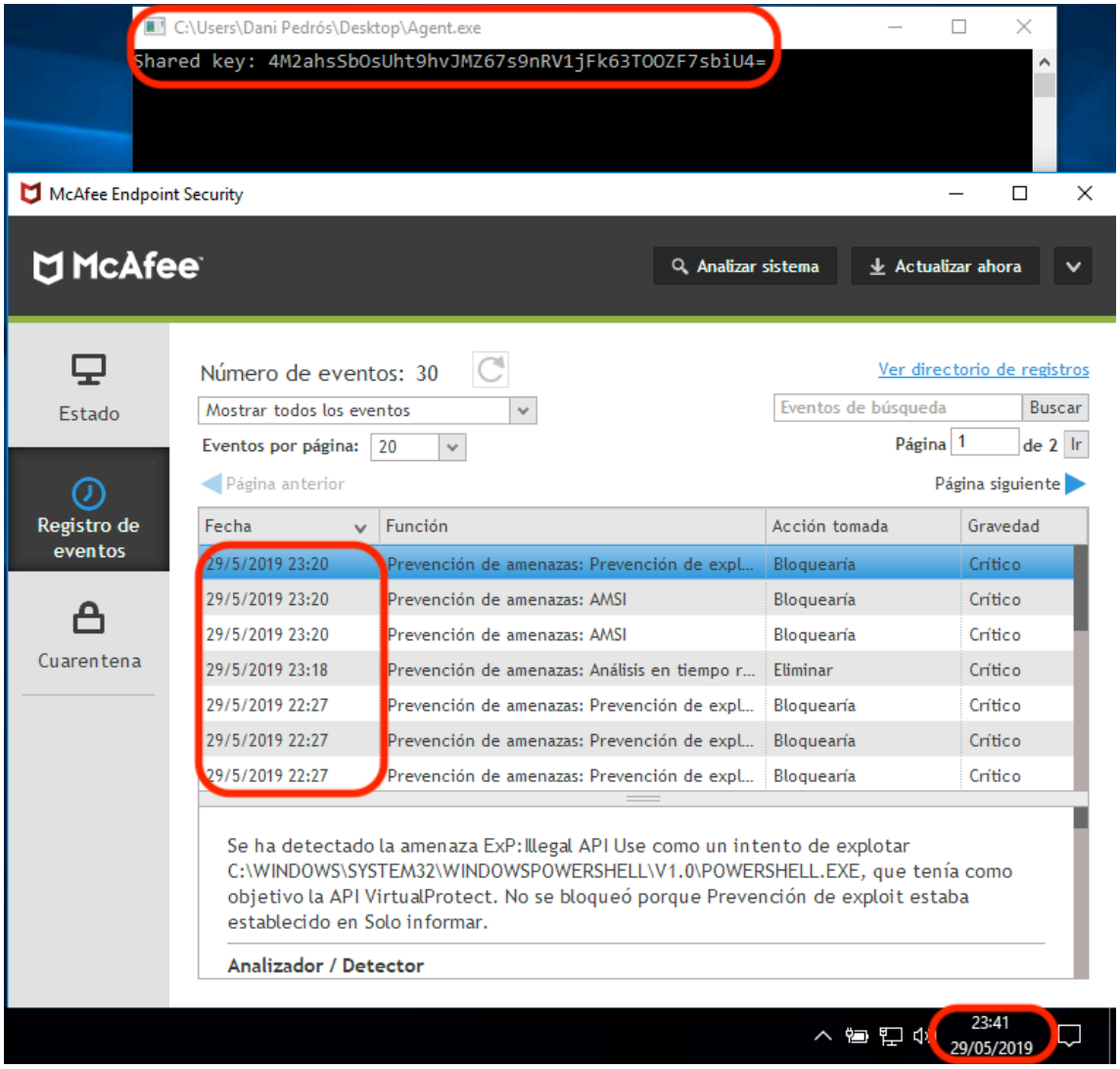



**Il·lustració 30: Resultat negatiu de l'anàlisi d'amenaces de l'Agent**

No obstant, les eines antivirus actuals són capaces de realitzar anàlisi dinàmiques durant l'execució dels diferents programes de l'equip. Així doncs, per tal de demostrar que la pròpia execució del component Agent, així com la comunicació que estableix amb el servidor de comandament i control, no desperten sospita per a l'antivirus, s'ha realitzat una execució d'aquest component i s'ha deixat que interactuï un temps amb el CnC mentre es revisen els events de l'antivirus.

Tal i com es pot observar a la següent captura de pantalla, l'execució de l'Agent no ha fet saltar ninguna alerta a l'antivirus. A la part superior es pot observar que el component està funcionant amb normalitat i ha obtingut la clau

de xifrat compartida amb el CnC. A la part inferior s'observa l'hora actual en la que s'ha realitzat l'execució. I a la part central es distingeixen les últimes alertes identificades per l'antivirus. Es pot veure que aquestes alertes corresponen a altres proves realitzades 20 minuts abans, demostrant així que l'execució de l'Agent tampoc desperta ninguna sospita a l'antivirus.




**Il·lustració 31: Execució de l'Agent i consola d'events de l'antivirus**

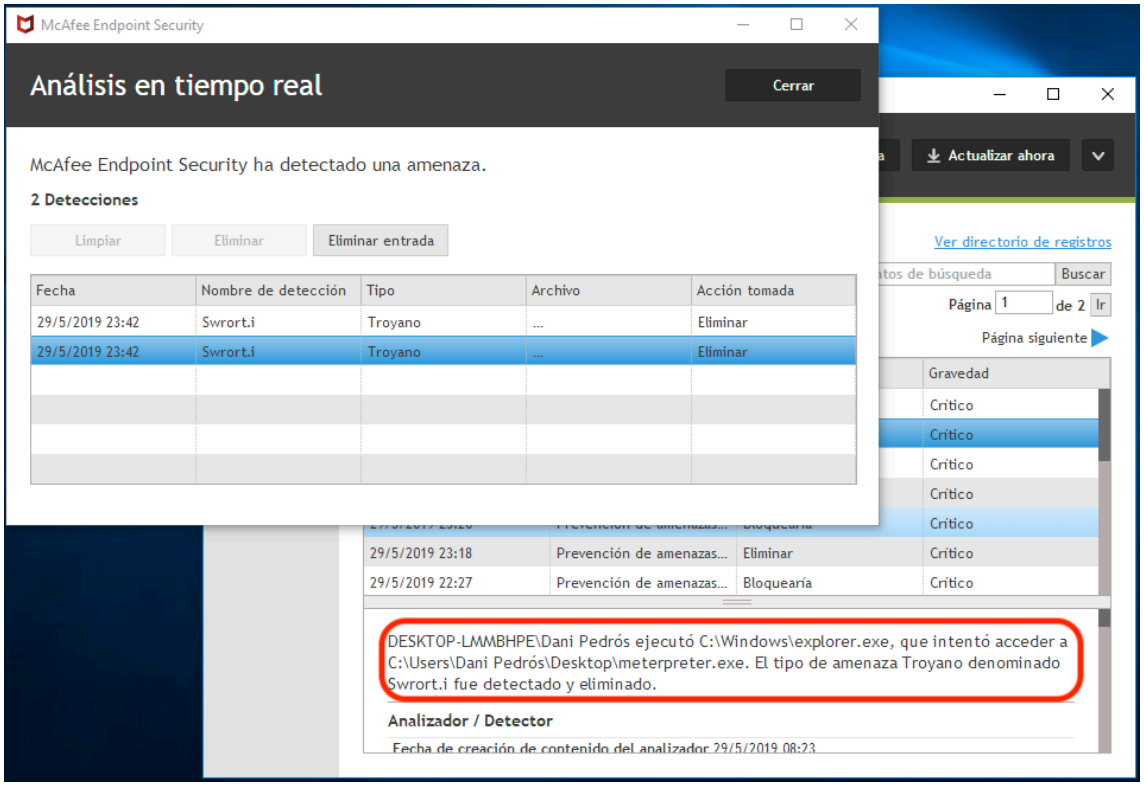
### 6.3.2 Execució d'eines externes

Una vegada demostrat que és possible emmagatzemar el component Agent en disc i executar-lo sense que sigui detectat com a una amenaça per l'antivirus, en aquest subapartat es demostra que és possible executar eines externes. Aquestes eines poden ser detectables per l'antivirus. No obstant, es demostra que executant-les mitjançant l'Agent, s'aconsegueix evadir la detecció d'aquestes.

En primer lloc s'ha afegit una tercera màquina a la xarxa de proves. Es tracta d'una màquina Debian 9 de 64 bits, la qual s'utilitza per a executar la part

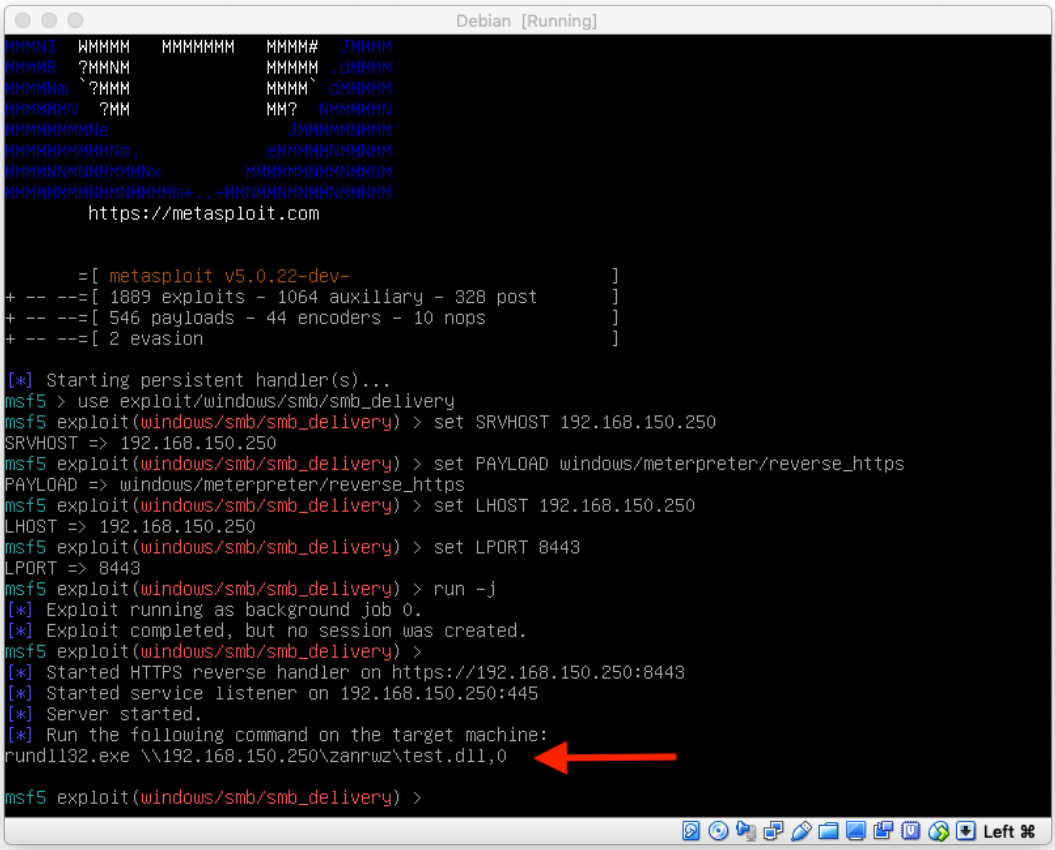
servidor de l'eina externa. S'ha decidit utilitzar com a eina externa un Meterpreter, l'eina de control remot del framework Metasploit. Aquesta decisió s'ha pres per diversos motius: ja es disposava de coneixements en l'ús d'aquesta, és una eina molt completa i utilitzada en entorns reals i és una eina que tots, o quasi tots, els antivirus identifiquen, eliminen i alerten.

Per tal de demostrar que aquesta eina és fàcilment detectable pels antivirus, s'ha generat un executable que inclou aquesta eina i s'ha deixat a la màquina víctima, la qual incorpora l'antivirus. Tal i com es pot veure a la següent captura de pantalla, l'antivirus ha detectat i eliminat l'amenaça tant de la seua ubicació com de la paperera de Windows.



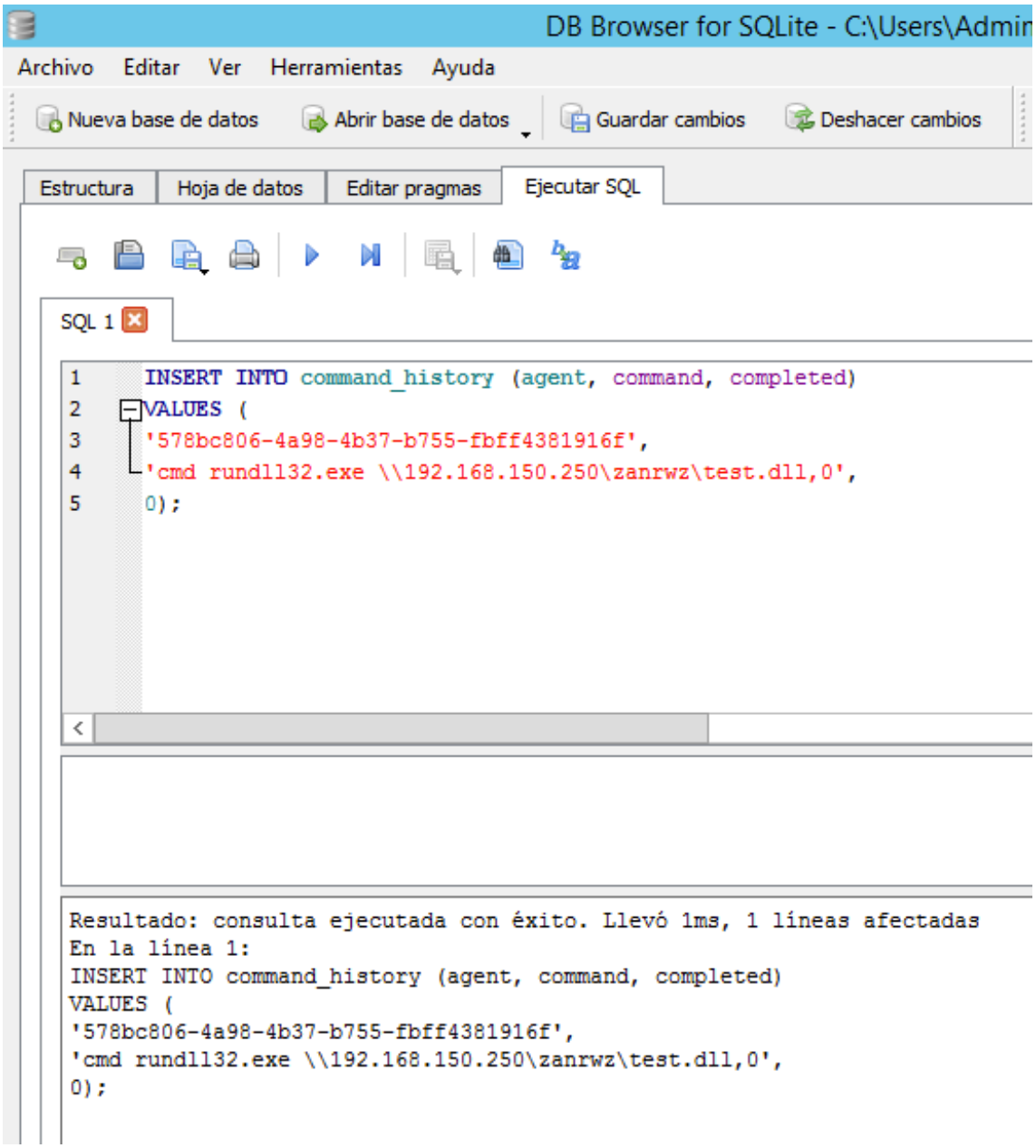

**Il·lustració 32: Detecció de Meterpreter per l'antivirus**

Aquest Meterpreter identificat i neutralitzat per l'antivirus es tracta d'un payload de tipus *reverse_https*. A la màquina Debian s'ha utilitzat aquest payload per a posar a l'escolta un handler que espera les connexions de la màquina víctima. A la següent captura de pantalla es pot observar la configuració bàsica d'aquest i, senyalada amb una fletxa roja, l'ordre que ha d'executar-se a la màquina víctima per tal de connectar amb la màquina atacant.




**Il·lustració 33: Configuració i arranc del handler de Meterpreter**

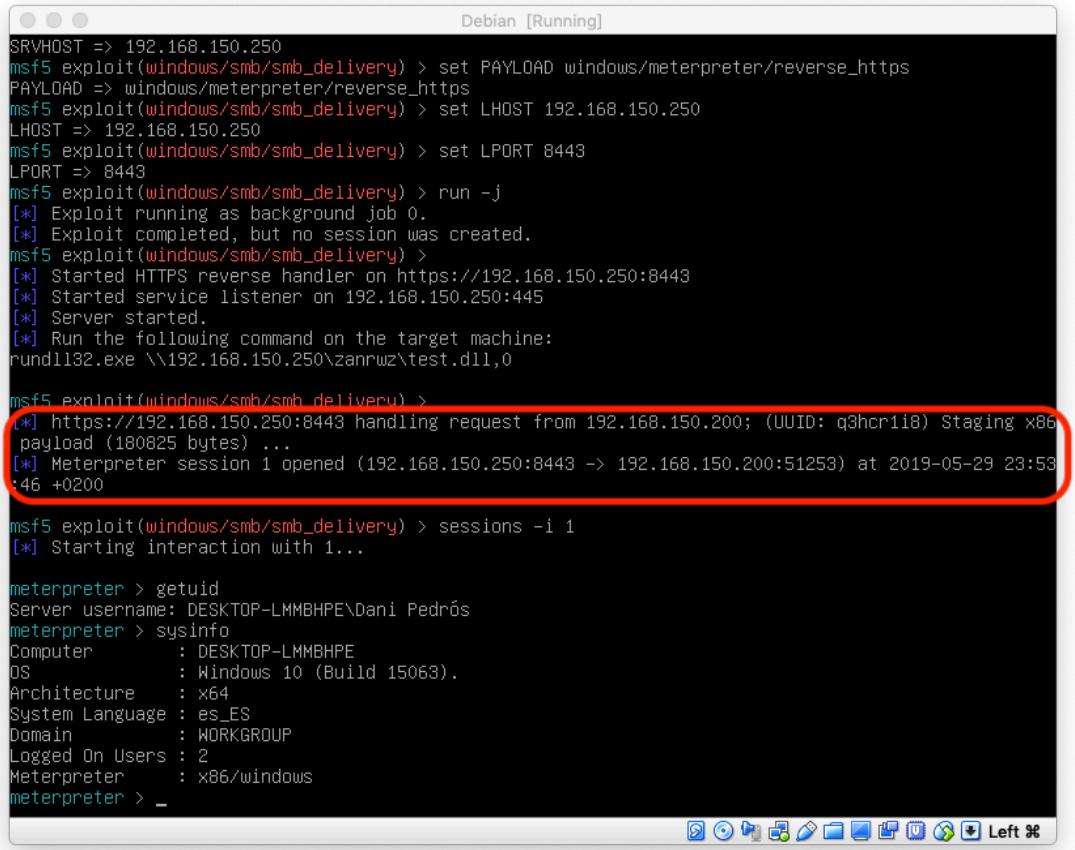
S'observa a la següent captura de pantalla com s'introdueix a la base de dades del servidor de comandament i control la informació necessària per tal que l'Agent implantat a la màquina víctima execute el comandament que s'indica a l'anterior captura de pantalla. El primer dels paràmetres fa referència a l'identificador únic de l'Agent que ha d'executar l'ordre, el segon a la pròpia ordre, que coincideix amb la que indica el handler de Meterpreter, i per últim el valor 0, que indica que encara no s'ha executat l'ordre.




**Il·lustració 34: Inserció d'ordres a la base de dades**

Una vegada l'agent rep i executa l'ordre inserida a la base de dades, es pot observar el resultat a la màquina atacant, on s'està executant el handler de Meterpreter. Es veu a la següent captura de pantalla que la màquina atacant ha rebut la connexió de la màquina víctima. Ressaltat en roig s'observen els detalls de l'execució del Meterpreter a la màquina víctima, com per exemple l'hora d'execució, que és 23:53:46 del dia 29/05/2019.

A més, es pot observar també que s'han executat diferents ordres de Meterpreter, la qual cosa demostra que la connexió és funcional i que l'atacant ha obtingut control de la màquina víctima utilitzant aquesta eina externa. Les ordres utilitzades mostren el nom de l'usuari i informació sobre el sistema.

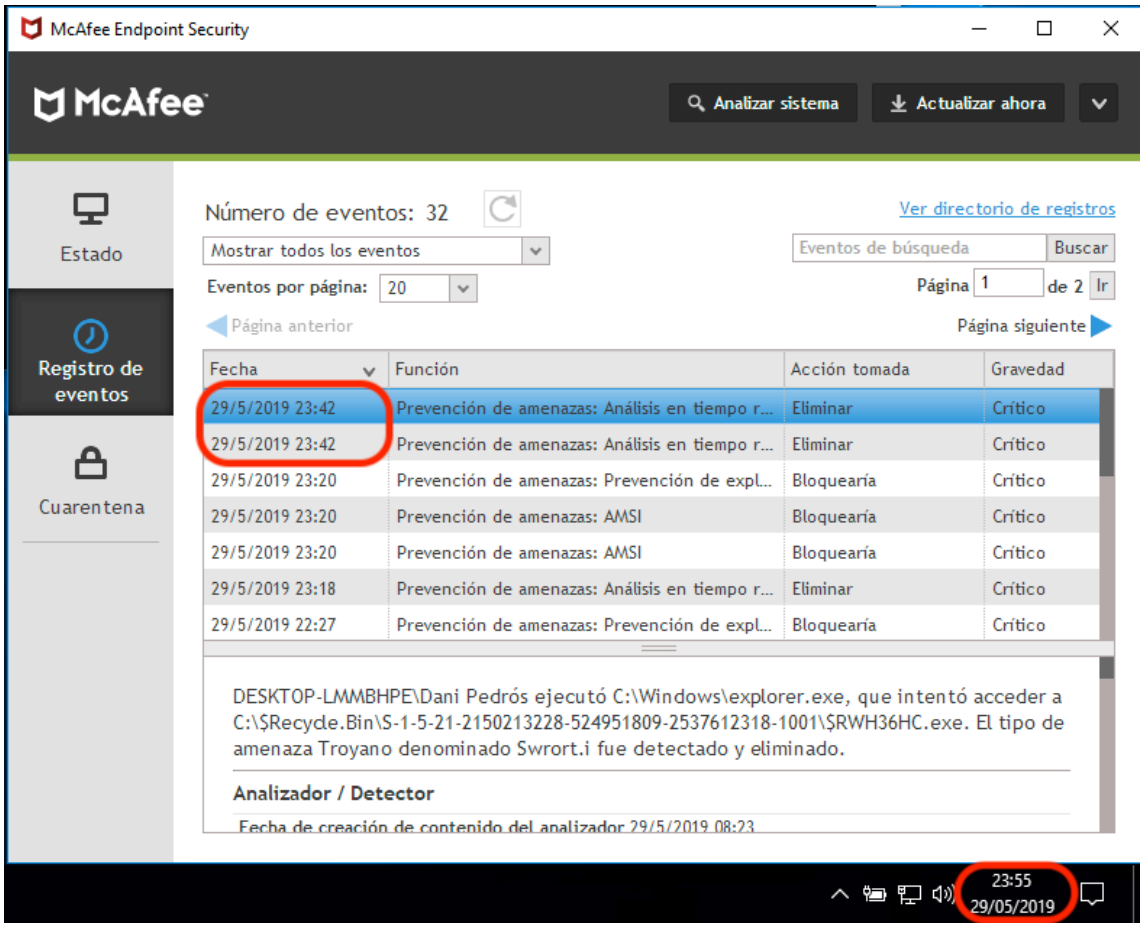



**Il·lustració 35: Connexió a la màquina víctima mitjançant Meterpreter**

Una vegada demostrat que és possible obtenir el control de la màquina víctima utilitzant eines externes mitjançant l'Agent desenvolupat, només queda comprovar si aquesta activitat ha sigut identificada per l'antivirus. A la següent captura de pantalla es pot observar clarament que aquesta activitat no ha sigut identificada, a pesar d'haver utilitzat una eina que, com s'ha demostrat al principi, és detectada per l'antivirus.

Es pot veure a la part central que les últimes alertes generades per l'antivirus corresponen a la detecció de Meterpreter realitzada a l'inici d'aquest subapartat. Si es comparen aquestes alertes i l'hora a la que s'han generat amb les de la primera il·lustració d'aquest apartat es pot comprovar que coincideixen. A més, a la part inferior de la pantalla es veu l'hora a la qual s'han consultat les alertes, que correspon a les 23:55 del mateix dia.

Així doncs, es demostra que l'execució de Meterpreter mitjançant l'Agent, realitzada a les 23:53, no ha sigut identificada per l'antivirus, donat que l'últim event generat per aquest és anterior a aquesta execució i l'hora de consulta dels events és posterior a l'execució de Meterpreter. Queda demostrat, doncs, que l'execució d'eines externes mitjançant l'Agent desenvolupat no és detectable per l'antivirus.




**Il·lustració 36: Execució de Meterpreter no detectada per l'antivirus**

# 7 Conclusions

Es presenten a continuació les conclusions que es desprenen del desenvolupament de l'actual projecte. En aquest capítol es parla dels objectius assolits en el desenvolupament, així com també dels que no han pogut ser assolits. També es proposen diferents projectes relacionats amb el tema tractat en el present projecte i que resulten interessants de cara al seu desenvolupament en un futur. I per últim, per tal de donar tancament al present document i al projecte en si, es presenta la valoració personal d'aquest.

## 7.1 Objectius assolits

A partir de les consideracions realitzades al capítol introductori d'aquest document, es pot considerar que tots els objectius plantejats han sigut assolits. Si es considera l'objectiu principal plantejat, que consisteix en el disseny i desenvolupament d'un RAT i el seu CnC, al contingut del present document s'evidencia tot el procés de desenvolupament i implementació.

Però si es tracten per separat els 3 subobjectius derivats de l'objectiu principal, i presentats també al capítol d'introducció, és possible entendre més fàcilment que aquests han sigut assolits. Es presentaven els següents subobjectius: disseny d'un sistema basat en 3 components (Agent, CnC i Consola), implementació de l'Agent i el CnC i aplicació de tècniques d'evasió d'antivirus a l'Agent.

Respecte al disseny d'un sistema basat en 3 components, en aquest document es detalla la funcionalitat oferta per cadascun dels 3 components proposats, així com també la interacció que s'estableix entre ells. En quant a la implementació de l'Agent i el CnC, s'ha materialitzat tot el procés de disseny d'aquests dos components en una solució real capaç de ser executada i provada. I per últim, al capítol d'implementació s'expliquen algunes eines de detecció utilitzades pels programaris antivirus i les tècniques implementades al component Agent per tal d'evadir aquestes eines. A més, al capítol de proves és possible comprovar, amb un exemple real, el correcte funcionament, tant dels components com de les tècniques d'evasió d'antivirus.

## 7.2 Objectius no assolits

Tal i com s'explica a l'apartat anterior, tots els objectius plantejats per al desenvolupament d'aquest projecte han sigut assolits satisfactòriament. No obstant, i ja que aquest és un projecte desenvolupat aplicant la metodologia basada en el disseny iteratiu i incremental, en aquest punt podrien proposar-se nous objectius que seguiren ampliant funcionalitats o millorant la solució.

Per exemple, podrien proposar-se nous objectius com l'addició de noves funcionalitats a la solució, com la creació de noves funcions d'alt nivell que, invocades per l'usuari del sistema, permeteren que un Agent desplegat en alguna màquina aconseguís obtenir persistència. Altra possible funció podria ser l'autodestrucció de l'Agent, la qual, invocada per un usuari, provoqués que

l'Agent eliminés proves de la seua existència a la màquina i es desinstal·lés automàticament.

Aquest apartat està, en certa manera, relacionat amb el següent ja que aquestes tasques proposades podrien entendre's com a una extensió del propi projecte o diferents projectes nous. No obstant, al següent apartat es presenten, a més d'aquestes propostes, altres considerades projectes independents.

## 7.3 Futurs projectes relacionats

A l'anterior apartat es parla, no dels objectius no assolits, sinó de possibles ampliacions dels objectius per a l'actual projecte, les quals podrien ser desenvolupades en un futur. En aquest apartat, a més d'aquestes propostes, es presenten idees per al desenvolupament de futurs projectes relacionats amb aquest.

En primer lloc, es considera especialment rellevant proposar, a partir del disseny proposat en aquest projecte, la implementació del tercer component del sistema, la Consola. Aquest projecte consistiria en desenvolupar un component de programari capaç d'integrar-se amb el servidor de comandament i control desenvolupat a l'actual projecte. Aquest hauria d'implementar les funcions definides als casos d'ús i, a més comunicar-se de forma efectiva amb el servidor de comandament i control.

Altre projecte que es proposa és la implementació de noves funcionalitats amb la intenció d'ampliar les funcionalitats oferides actualment pel sistema. A l'anterior apartat s'han proposat un parell de noves funcionalitats, encara que qualsevol ampliació de les funcionalitats actuals del sistema podria entendre's com un projecte nou independent.

Per últim, resulta interessant també proposar la realització de més proves amb més antivirus i, fins i tot, en una xarxa simulada o un entorn real. Aquestes proves, enteses com a projecte independent, permetrien continuar desenvolupant el sistema i adaptant-lo a les noves condicions de l'entorn. Així doncs, a part del propi projecte centrat en la realització de proves més extensives, és probable que sorgiren nous projectes enfocats a la implementació de noves tècniques d'evasió d'antivirus.

## 7.4 Valoració personal

Com a conclusió d'aquesta memòria es presenta la valoració general del projecte i el seu desenvolupament. Aquesta valoració parla dels coneixements obtinguts tant a les assignatures cursades, com a l'entorn professional, així com també de les tecnologies descobertes, errors comesos i dificultats. Per últim, es parla també dels coneixements apresos i extrets del desenvolupament d'aquest projecte.

Algunes de les assignatures del màster han sigut especialment rellevants per al desenvolupament d'aquest projecte. Les més significatives han sigut

Seguretat en Sistemes Operatius i Seguretat en Xarxes. La primera d'aquestes ha aportat coneixements essencials per a la instal·lació i configuració bàsica del servidor Windows, així com també de la gestió dels certificats (necessaris per a la comunicació HTTPS) i la criptografia de clau simètrica i asimètrica. En quant a l'assignatura Seguretat en Xarxes, aquesta ha aportat una visió en profunditat sobre el funcionament d'alguns sistemes de detecció d'intrusos, com SNORT. Aquesta informació ha sigut rellevant per tal de desenvolupar tècniques d'evasió d'aquests sistemes NIDS.

A més, el desenvolupament d'aquest projecte ha permès adquirir coneixements de tecnologies i llenguatges de programació que no s'han estudiat durant la docència del màster. Destaca l'aprenentatge de la programació en C#, llenguatge que no havia utilitzat mai i que ha sigut important per al desenvolupament del projecte. No només la pròpia programació, sinó l'ús correcte de llibreries i funcions com *Assembly.Load* han permès aprendre com s'integren les biblioteques dinàmiques de Windows (DLLs) amb el programari que les utilitza.

Sobre els errors comesos, i tenint en compte el que s'acaba d'exposar a l'anterior paràgraf, destacar la falta de temps dedicada inicialment a l'aprenentatge en profunditat del llenguatge de programació C#. Açò ha provocat alguns retards en el desenvolupament, deguts a no tenir clar l'existència d'algunes funcions, etc. No obstant, aquests inconvenients han sigut suplerts mitjançant la lectura i comprensió de la documentació aportada per Microsoft sobre aquest llenguatge. Així doncs, aquest error inicial ha permès aprendre a buscar de forma adequada entre l'extensa documentació disponible a Internet.

A més del que s'ha aprés dels errors, el desenvolupament del projecte també ha permès conèixer i aprendre nous recursos que poden ser aplicats en un futur, tant en la vida personal com en la professional. Destaca l'afiançament dels coneixements relacionats amb la programació orientada a objectes, així com també l'aplicació de metodologies adaptades a projectes canviants. Més relacionat amb l'aprenentatge personal, encara que també aplicable al món professional, destaca l'aprenentatge relacionat amb l'establiment d'objectius realistes que permeten evitar i/o controlar la frustració i realitzar un càlcul adequat de l'esforç requerit.

La realització del projecte presentat en aquest document m'ha aportat un creixement significatiu tant com a professional de la seguretat de les tecnologies de la informació i les comunicacions, com a persona. Haver paral·lelitzat el desenvolupament d'aquest projecte amb totes les assignatures del segon semestre i una jornada laboral completa de 8h al dia ha requerit un gran esforç i una gran optimització i gestió del temps. Tot açò ha despertat la curiositat i interès necessaris per a continuar formant-me i descobrint el món de les tecnologies de la informació i les comunicacions, i en especial, la seguretat d'aquestes. Presentat tot el procés de desenvolupament, les proves i les conclusions, puc assegurar que estic molt orgullós del projecte i de tots els resultats obtinguts d'aquest.

# 8 Glossari

- **TIC**: Tecnologies de la informació i les comunicacions. Agrupen els elements i les tècniques utilitzades en el tractament i la transmissió de la informació, principalment d'informàtica, Internet i telecomunicacions. Designen, també, el sector d'activitat econòmica.

- **RAT**: *Remote Access Trojan*. És un tipus de programari maliciós que permet a un usuari controlar el sistema informàtic d'una víctima mitjançant una connexió remota a través d'Internet. Normalment, un RAT s'instal·la sense el coneixement de la víctima i tracta d'ocultar la seua activitat, tant a la víctima com al programari de seguretat (antivirus, NIDS, HIDS, etc.).

- **CnC**: *Command & Control*. Fa referència als servidors utilitzats per al control de programari maliciós. Aquests servidors poden pertànyer directament als operadors del programari maliciós o estar instal·lats a servidors de tercers que hagen sigut compromesos.

- **HIDS**: *Host-based Intrusion Detection System*. La funció d'aquests consisteix en la identificació de riscs potencials i possibles intrusions en un sistema mitjançant la detecció d'anomalies revisant l'activitat de la màquina on està instal·lat.

- **NIDS**: *Network-based Intrusion Detection System*. La funció d'aquests consisteix, a l'igual que els HIDS, en la identificació de riscs potencials i possibles intrusions, en aquest cas en una xarxa, mitjançant la detecció d'anomalies en l'activitat i tràfic de la xarxa.

- **AV**: Antivirus. Programari dissenyat per a identificar, eliminar i informar de les amenaces per a la seguretat de la màquina on han sigut instal·lats. Aquests identifiquen les amenaces de diferents formes, com per exemple la firma digital, l'heurística, anàlisi estàtica o anàlisi dinàmica.

- **SSL**: *Secure Sockets Layer*. Conjunt de protocols que permeten la comunicació segura a través d'Internet mitjançant el xifrat de la informació.

- **HTTP**: *Hyper Text Transfer Protocol*. Protocol dissenyat per a l'intercanvi de documents d'hipertext i multimèdia a Internet.

- **HTTPS**: *Hyper Text Transfer Protocol Secure*. Versió d'HTTP que treballa sobre SSL, aportant seguretat a les comunicacions.

- **AMSI**: *AntiMalware Scan Interface*. Interfície que permet a les diferents aplicacions de Windows integrar-se amb el programari antimalware instal·lat al sistema i realitzar anàlisis i comprovacions de seguretat.

# 9 Bibliografia


[1] B. H. LISKOV, «A design methodology for reliable software systems,» [En línea]. Available: https://pdfs.semanticscholar.org/d420/c8b473a23b80241fd7c90757becb59b1136c.pdf. [Último acceso: 18 Maig 2019].

[2] «Visual Studio 2017,» [En línea]. Available: https://docs.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2017. [Último acceso: 19 Maig 2019].

[3] «.NET Framework,» [En línea]. Available: https://docs.microsoft.com/en-us/dotnet/framework/get-started/. [Último acceso: 19 Maig 2019].

[4] «Desenvolupament Iteratiu Incremental,» [En línea]. Available: https://proyectosagiles.org/desarrollo-iterativo-incremental/. [Último acceso: 18 Maig 2019].

[5] D. R. Stinson, Cryptography: Theory and Practice, Chapman and Hall/CRC, tercera edició, 2006.

[6] P. C. v. O. i. S. A. V. A. J. Menezes, Handbook of applied Cryptography, CRC Press, 1996.

[7] J. Daemen y V. Rijmen, The Design of Rijndael: AES - The Advanced Encryption Standard, Springer, 2002.

[8] J. G. L. d. Lacalle, «Factorización polinomial de números enteros,» 2004. [En línea]. Available: http://gaceta.rsme.es/abrir.php?id=102. [Último acceso: 29 05 2019].

[9] S. Pasini, «Secure Communications over Insecure Channels Using an Authenticated Channel,» 06 09 2006. [En línea]. Available: https://lasec.epfl.ch/pub/lasec/doc/Pas05.pdf. [Último acceso: 28 05 2019].

[10] Microsoft, «What's new in the .NET Framework,» 18 04 2019. [En línea]. Available: https://docs.microsoft.com/en-us/dotnet/framework/whats-new/. [Último acceso: 31 05 2019].

[11] Microsoft, «Antimalware Scan Interface,» 27 02 2019. [En línea]. Available: https://docs.microsoft.com/es-es/windows/desktop/amsi/antimalware-scan-interface-portal. [Último acceso: 31 05 2019].