Homework 4

Daniele Ferrarelli

Contents

1 Analisi Iniziale

- Processor: x86
- Endian: Little Endian
- Format Portaable Executable

Si inizia la fase di analisi del malware eseguendolo su una macchina virtuale, appositamente preparata e scollegata da internet, per osservare il funzionamento del malware. Il malware in questione è un ransomware, poiché andrà a criptare file della macchina virtuale se eseguito. Successivamente mostrerà una pagina HTML ed una immagine contenenti le istruzioni per ottenere il programma di decodifica. Si può notare che viene generato un ID del sistema corrente da utilizzare per ottenere il programma di decodifica dal sito malevolo. Nelle informazioni mostrate dal malware si vede che indica l'utilizzo di RSA-2048 ed AES-128 come funzioni criptografiche. Inoltre il malware si cancella una volta terminato il lavoro. Al momento dell'avvio del malware, se non si disattiva Windows Defender, viene rilevato come il ransomware Locky.

2 Unpack del Malware

Si inizia l'analisi del malware con il tool PE Bear, tramite questo strumento si notano le sezioni UPX0 ed UPX1, questo indica che il malware fa' uso del packer UPX. Si tenta, utilizzando l'eseguibile UPX, di fare l'unpack automatico del malware, tuttavia l'unpacking non ha successo, questo può indicare l'utilizzo di modalità di packing diverse da quelle originali di UPX. Per questo motivo si necessita di fare l'unpacking manuale.

Per prima cosa si prova ad utilizzare i plugin di OllyDbg per trovare l'Original Entry Point, tuttavia non si ottengono risultati. Si cerca allora di utilizzare OllyDbg per generare un dump del programma unpackato. Si prova a trovare un long jump su Ghidra che può corrispondere al salto verso l'OEP. La ricerca dei jump produce 8 possibili jump da analizzare, tuttavia solamente il jump ad indirizzo 0x004A20D0 con target 0x0040EA13 sembra soddisfare le richieste, essendo un long jump.

Utilizzando un breakpoint sull'accesso in memoria all'indirizzo 0x0040EA13 si può vedere come venga modificata la memoria tramite l'algoritmo di compressione utilizzato da UPX. Questo fa' pensare che effettivamente il jump sia il jump verso l'Original Entry Point. Si posiziona un breakpoint prima

del jump e poi si produce un dump della memoria, successivamente tramite ImpRec si ricostruisce l'IAT. Infine si porta il dump su Ghidra e lo si analizza.

Si notano all'interno dell'applicazione la presenza di due stringhe "lwpqabklbeiutlcti" e "vickhgfqkhpdvlnva" e l'utilizzo della chiamata IsBadStringPtrW. Si può vedere come vengano eseguite una serie di operazioni su queste due stringhe. Questa serie di operazioni non sembra far parte dei meccanismi osservati del malware, questo può indicare che le istruzioni appartenenti al malware siano offuscate. Al termine di queste operazioni verrà utilizzata una chiamata a VirtualAlloc per preparare una zona di memoria dove verranno copiati dei byte. Verrà eseguito un salto all'interno di questa area di memoria e tramite debugger si può notare come venga cambiato l'address space originale. Questo può indicare un ulteriore meccanismo di packing che maschera il malware come una applicazione innocua.

Al momento del salto in questa area di memoria viene passato sullo stack il riferimento ad Kernel32.GetModuleHandleA in modo da non dover risolvere i riferimenti alle API chiamate successivamente. Da questo riferimento vengono risolte le altre API necessarie per il funzionamento di questa zona di memoria. Viene chiamata una nuova VirtualAlloc e si copierà il contenuto della zona di memoria in quella nuova, per poi continuare ad eseguire nella nuova zona. Al suo interno verranno utilizzate le chiamate a VirtualProtect per cambiare i permessi di accesso alla memoria (in particolare alla sezione .text) e riscrivere i byte dell'address space utilizzando le istruzioni REP STOS e REP MOVS. Successivamente si andrà ad reimpostare i livelli di protezione della sezione appena modificata. Infine viene deallocata la memoria utilizzata in precedenza e si andrà a saltare nell'address space appena modificato.

Il nuovo entry point dell'applicazione è all'indirizzo 0x00402D8F, come fatto in precedenza si crea un dump dell'applicazione tramite OllyDump per poi risolvere l'IAT tramite ImpRec. A questo punto analizzando il file su Ghidra si possono notare un numero maggiore di API utilizzate, inoltre le API sono legate al funzionamento osservato del malware, come per esempio ADVAPI.dll che include le API per criptare dati. Infine verrà disattivata l'ASLR di windows e si procede con l'analisi del malware.

3 Analisi Malware

Iniziando l'analisi, utilizzando Ghidra, si nota la presenza di chiamate a funzioni con la convenzione thiscall, questo può indicare l'utilizzo di un linguaggio ad oggetti all'interno del malware. Durante l'analisi si nota la presenza di un gran numero di NOP, JUMP e LEA che vengono utilizzate per ostacolare la fase di analisi statica e dinamica andando a complicare il flusso

delle istruzioni. Il passaggio di molti parametri alle funzioni del malware viene oscurato utilizzando determinati offset rispetto ad EBP, questo ha reso l'analisi più ardua poiché per risolvere molti parametri si è reso necessario utilizzare il debugger e seguire i vari valori in memoria. Si è fatto grande uso del decompilatore di Ghidra poiché rende più leggibile il flusso di esecuzione rispetto al disassemblatore, questo per la grande presenza di NOP e JUMP all'interno delle istruzioni assembler.

Una volta terminata la fase di deoffuscamento si trova l'entry point originale del malware, al suo interno si andrà a cercare la funzione main. Una funzione che potrebbe essere il main è FUN_00429EA, controllando la segnatura di funzioni main nella documentazione Windows si può vedere come una possibile segnatura per C++ abbia i parametri int, char* e char*. Questo corrisponde con i parametri della funzione appena trovata, per questo si continua l'analisi rinominando FUN_00429EA a main. La funzione che chiamerà il main viene rinominata a mainWrapper, questa è di particolare interesse in fasi successive dell'analisi, poiché conterrà al suo interno una funzione essenziale per il funzionamento del malware. Varie funzioni presenti all'interno del malware non vengono analizzate, poiché sono funzioni legate alla gestione degli oggetti. Inoltre sono presenti molte funzioni che incapsulano la chiamata all'api vera e propria, come per esempio FUN_0x0041F680 che incapsula GetModuleFileNameW. Per facilitare l'analisi molte funzioni che incapsulano chiamate ad API vengono rinominate su Ghidra, questo permette velocizzare la fase di analisi statica.

3.1 MainWrapper

Iniziando l'analisi del main si nota la presenza di una area di memoria che viene utilizzata per mantenere dei dati utilizzati dal main. Si nota che questa area di memoria ha un indirizzo non appartenente alle sezioni trovate su Ghidra. Questo fa' pensare che viene allocata da funzioni chiamate in precedenza. Tra queste funzioni si nota la presenza della funzione mainWrapper che invocherà FUN_0047705, al suo interno sono presenti chiamate a VirtualAlloc, questo fa' pensare che in questa zona viene preparata la zona di memoria. Per questo motivo la funzione viene rinominata a prepareMemoryArea. Dopo aver chiamato la funzione prepareMemoryArea verrà chiamato il main effettivo del malware.

3.1.1 PrepareMemoryArea

All'interno di questa funzione verrà allocata una porzione di memoria utilizzata poi dalla funzione main. Inoltre viene fatto un controllo sulla PEB

per controllare la presenza di un debugger in esecuzione. Questo controllo viene eseguito accedendo, tramite il segmento FS, alla flag BeingDebugged presente nella PEB. Se il malware esegue sotto debugger verranno utilizzate le chiamate ad GetModuleHandle e GetProcAddress per ottenere il riferimento all'API della DLL kernel32 AllocConsole, che viene copiato nella zona di memoria precedentemente allocata.

Se non è presente il debugger in questa zona verranno inseriti dei byte di controllo ed una serie 0. Controllando il contenuto del resto dell'area di memoria, si può notare una serie di byte che corrispondono al linguaggio HTML, questo può indicare che verranno utilizzati per creare la pagina HTML mostrata quando il computer viene infettato. Viene inoltre trovata una sequenza di byte che potrebbe essere utilizzata per creare la bitmap mostrata insieme alla pagina HTML. All'interno è presente anche il testo mostrato all'utente, contenente le istruzioni per decriptare i file nel sistema. Si può notare l'ID placeholder "FF000000" che verrà sostituito con un ID generato dal malware per identificare la macchina infetta. In questa zona di memoria vengono inseriti dei byte di configurazione, questi vengono utilizzati per cambiare il comportamento del malware, come per esempio l'utilizzo di internet e scritture su registri di sistema. Queste informazioni sono presenti all'interno dell'eseguibile, poiché non vengono accessi file esterni o servizi di comunicazione con internet. In questa funzione queste informazioni vengono deoffuscate e copiate nell'area di memoria, probabilmente corrispondono a configurazioni del sistema che vengono inserite a tempo di compilazione.

Per facilitare l'analisi viene disattivata la misura anti-debug modificando l'eseguibile, in questo modo è possibile analizzare il comportamento del malware senza che rilevi la presenza di un debugger. Successivamente si passa all'analisi della funzione main del malware (FUN_00429EA0).

3.2 Main

Iniziando l'analisi della funzione main si nota la presenza di una funzione (FUN_00401018) che scrive su FS:[0], questa funzione viene chiamata all'inizio di molte altre funzioni, questo fa pensare che si possa trattare di una funzione inserita dal compilatore, per questo motivo non viene analizzata.

Successivamente si nota la presenza della chiamata a SetErrorMode, questo viene fatto utilizzando le flag SEM_FAILCRITICALERRORS, SEM_NOGPFAULTERRORBOX, SEM_NOOPENFILERRORBOX. Questo set della gestione di errore permette di non mostrare finestre di dialogo all'utente in caso di errore (file non trovato, Windows Error Reporting, critical-error-handler), questi errori vengono direttamente passati al processo. In questo modo il malware può nascondersi anche in caso di errore. Suc-

cessivamente viene usata l'api SetUnhandledExceptionFilter per cambiare il gestore di eccezioni top-level, impostando la funzione FUN_0041F820 (malwareExceptionHandler).

3.2.1 Gestione delle eccezioni

Questo gestore di eccezioni chiuderà due handle (handle che fanno riferimento a mutex dichiarati successivamente) utilizzando la chiamata a CloseHandle. Poi utilizzerà una chiamata a GetModuleFileNameW, questa funzione permette di ottenere il path del file che contiene il modulo indicato da un handle passato alla funzione. Se questo handle è NULL, come nel caso di questa chiamata, allora questa funzione otterrà il path dell'eseguibile che corrisponde al processo corrente. Una volta ottenuto il path verrà chiamata la funzione FUN_0042EC10 (rinominata a spawnProcess). All'interno di questa funzione verrà chiamata la API CreateProcessW che permette di creare un processo, come input a questa chiamata viene passato il path ottenuto in precedenza. Infine utilizzerà la chiamata ad ExitProcess per terminare il processo corrente. Questo fa' pensare che se il malware incontra una eccezione che non può andrà a lanciare una nuova istanza del malware per poi chiudere l'istanza corrente.

3.2.2 Settaggio Token

Successivamente nel main viene chiamata la funzione FUN_0042CF00 , questa funzione andrà, tramite le API GetCurrentProcess e OpenProcessToken, ad ottenere un token per il processo corrente. Infine andrà ad impostare, tramite SetTokenInformation, la flag TokenVirtualizationEnabled.

3.2.3 Ricerca della lingua del sistema

All'interno del malware si notano chiamate ad una serie di API per ottenere gli identificativi della lingua usata per UI e per il sistema. Una volta ottenuti gli identificativi viene applicato un AND con 0x3ff e si controlla se il risultato corrisponde ad 0x19. Se il risultato è diverso da 0x19 si salterà in un'altra porzione di codice, altrimenti si eseguirà la funzione FUN_00421DE0. Per scoprire in che caso la lingua del sistema vada a cambiare il comportamento del malware vengono cambiati i risultati delle varie chiamate alle API per ottenere l"ID della lingua. In questo modo se la lingua è corretta il risultato con l'AND precedente dovrebbe essere 0x19. Considerando la provenienza di alcuni malware si prova con l'ID della lingua russa e si vede che in questo caso il risultato delle operazioni corrisponde ad 0x19, si nota che successivamente

si va a chiamare la funzione FUN_00421DE0. Si passa ad analizzare questa funzione (rinominata a deleteMalware).

3.2.4 Delete malware

In questa funzione verranno prelevati i path dell'eseguibile corrispondente al processo corrente e della cartella temporanea di Windows. Successivamente si utilizzerà la chiamata MoveFileEx per spostare il malware nella cartella temp, viene anche cambiato il nome dell'eseguibile generandone uno casuale, utilizzando GetTempFileNameW, a cui viene aggiunto il prefisso "sys". Successivamente si utilizzerà di nuovo MoveFileExW con la flag MOVEFILE_DELAY_UNTIL_REBOOT per spostare il file temporaneo appena generato in una destinazione nulla. Utilizzando questa destinazione e flag è possibile cancellare un file al momento del reboot del sistema. Infine si nota la presenza di una stringa offuscata, analizzando con OllyDbg è possibile vedere che questa stringa corrisponde al comando "cmd.exe /C del /Q /F". Questo comando verrà lanciato utilizzando la funzione spawnProcess analizzata in precedenza. In questo modo si lancerà una istanza di cmd.exe che eseguirà il comando "del" in modalità quiet (con la flag Q) e forced (con la flag F), cancellando il file temporaneo contenente il malware. Da questo si può capire che questa funzione andrà a rimuovere il malware dalla macchina corrente. Questa funzione verrà utilizzata sia nel caso in cui il malware ha terminato la sua esecuzione sia nel caso si deve cancellare poiché la lingua del sistema è in russo. Questa funzione si trova all'interno di un while loop insieme al resto delle istruzioni del main, questo poiché al termine dell'esecuzione delle istruzioni del malware si andrà a saltare in questa funzione, cancellando il malware.

Andando a vedere la mappa nel mondo delle infezioni del ransomware Locky, su cui è basato il malware corrente, si può notare come le infezioni non siano presenti in russia. Questo perché il malware viene cancellato se la lingua del sistema è il russo.

3.2.5 Sleep del malware

Continuando l'analisi del malware si nota la presenza di uno sleep per un numero di millisecondi dipendente da dati scritti nella memoria precedentemente allocata. Nel caso non sia presente il debugger lo sleep è di 12 secondi, mentre se è presente il debugger è di un tempo dettato dal valore dei byte che rappresentano il riferimento ad AllocConsole presenti in memoria. Per continuare l'analisi del malware senza dover aspettare lo sleep il malware verrà modificato in modo da utilizzare un tempo di 0s modificando il valore

di una PUSH prima della chiamata alla funzione Sleep.

4 Comportamento con debugger

Continuando l'analisi del malware, dopo la fase di sleep c'è un if in cui si andranno a controllare dei byte presenti nella zona di memoria settata all'inizio dell'esecuzione. Se in questa zona alcune porzioni non sono 0 significa che il malware ha rilevato la presenza di un debugger. In questo caso il malware andrà ad eseguire una serie di istruzioni che sono diverse dal comportamento normale del malware. Andando ad analizzare questa serie di istruzioni si può notare la presenza di una stringa offuscata, tramite debugger è possibile vedere che questa stringa è "svchost.exe". Questo fa pensare che il malware possa ripartire mascherandosi come il programma svchost.exe. Successivamente il malware andrà a copiarsi, tramite CopyFileW, nella cartella temp di Windows con il nome svchost.exe. Si nota inoltre che dopo il malware andrà a cancellare il file nella cartella temp svchost.exe:ZoneIdentifier, questo perché l'estensione :Zone.Identifier comprende dei metadati che gestiscono alcune feature di sicurezza di Windows per file scaricati da internet. Poi verrà chiamata la funzione spawnProcess che farà partire svchost.exe dalla cartella temp, questo eseguibile è in realtà quello del malware appena copiato. Infine il malware ritornerà all'inizio della funzione main, poiché è presente un ciclo while, e si cancellerà utilizzando la funzione deleteMalware. Questo indica che se il malware rileverà la presenza di un debugger andrà in sleep per poi lanciarsi come svchost.exe andando a aggirare il debugger.

5 Comportamento normale

Se il malware non rileva la presenza di debugger avrà un comportamento diverso al caso precedente. In questo caso la memoria contenente i byte di controllo non è più occupata dal riferimento ad AllocConsole ma contiene informazioni importanti per il funzionamento del malware. Viene controllato se all'offset 0xC il valore è 0, se sì si andrà ad eseguire questa serie di istruzioni. Per prima cosa verrà generato l'ID che andrà ad identificare la macchina una volta che è stata infettata. Questo viene fatto utilizzando la funzione FUN_00431440 rinominata a generateHash. Successivamente si utilizzerà la funzione FUN_00426A40 (rinominata a prepareMutex) per creare due semafori per la sincronizzazione di thread, da questo si può capire che il malware utilizzerà più thread per il suo funzionamento.

5.1 GenerateHash

All'interno di questa funzione verrà generato l'hash che identifica la macchina, questo viene fatto tramite una serie di informazioni del sistema. Per prima cosa si preleverà la Windows directory (in questo caso C:\Windows), con questa informazione si preleverà il GUID della partizione utilizzando la funzione GetVolumeNameForVolumeMountPointA. Una volta prelevato il GUID, si preparerà una CSP (cryptographic service provider), con tipologia PROV_RSA_AES e flag CRYPT_VERIFYCONTEXT. Con queste flag si preparerà un provider di funzioni di crittografia per generare l'hash del GUID. Successivamente si creerà un hash con la funzione CryptCreateHash, questa utilizzerà l'algoritmo CALG_MD5 specificato tramite un parametro della funzione. Infine si utilizzerà la funzione CryptHashData per creare un hash MD5 del GUID. Questo poi viene convertito ad una stringa che contiene solo valori esadecimali per poi essere tagliato in modo da contenere 16 caratteri. Questo ID della macchina verrà utilizzato poi per dare parte del nome dei file criptati e per identificarla se invierà dei dati ad un server remoto. Queste informazioni vengono trovate andando ad analizzare lo stack durante l'esecuzione passo passo del malware. Tuttavia si nota che questo ID è diverso da quello finale utilizzato per identificare la macchina. Questo si otterrà successivamente nell'esecuzione del malware.

5.2 Ricerca Atom

Si nota la presenza della funzione FUN_004270F0, questa funzione (rinominata a searchAtoms) viene chiamata dal main, dopo aver generato l'hash all'interno di un IF. Andandola ad analizzare si nota la presenza di chiamate a funzioni per trovare Atom nel sistema. La tabella atom è una tabella definita dal sistema che mantiene stringhe ed identificatori. Tramite debugger si può notare che viene ricercato il valore "˜˜˜ID ˜˜˜", dove ID è l'identificativo della macchina. Viene ricercata sia la tabella globale del sistema che quella specifica all'applicazione tramite le API GlobalFindAtom e FindAtom. Questo atom potrebbero essere aggiunti più avanti nell'esecuzione e potrebbero indicare se la macchina è già stata infettata. Se vengono trovati si terminerà l'esecuzione del malware. Tuttavia portando il malware ad eseguire questa ricerca si nota che viene cercato un identificatore diverso da quello generato al termine dell'esecuzione del malware.

5.3 Comunicazione internet disattivata

Il malware nell'esecuzione normale può avere due comportamenti distinti. Nel caso in cui il byte di configurazione ad offset 0xF sia zero il malware eseguirà senza connettersi ad internet. Nel caso sia disattivata la connessione, si andrà ad eseguire la funzione FUN_00423740 (rinominata a generateHash2). All'interno di questa funzione si vedono query ad informazioni di sistema utilizzando GetUserDefaultUILanguage, GetVersionEx, GetSystemMetrics e DsRoleGetPrimaryDomainInformation. Si nota anche la presenza della stringa offuscata "YBNDRFG8EJKMCPQX0T1UWISZA345H769". Eseguendo questa funzione si vede che viene ritornata sul registro EAX il riferimento ad una area dello stack contenente l'ID finale della macchina. Questo fa' pensare che qui viene terminata la generazione dell'ID della macchina utilizzando le informazioni prelevate in precedenza.

Infine si eseguiranno una serie di funzioni per impostare classi usate successivamente dal sistema, per poi andare ad eseguire la parte fuori dall'IF comune ai due modi di esecuzione.

5.4 Comunicazione internet attiva

Se il byte controllato in precedenza è diverso da 0, il malware andrà ad eseguire una serie di istruzioni alternative a quelle descritte precedentemente. In questa zona viene anche settata una variabile globale, tramite quest'ultima (rinominata a withInternet) si può vedere più avanti nell'analisi che si va a modificare il comportamento del thread che andrà a criptare i file.

Una funzione che sembra essere interessante è la funzione FUN_00421720 (rinominata getSystemInfo). Al suo interno si possono vedere una serie di funzioni che raccolgono informazioni sulla macchina corrente, in particolare GetVersionExA. Si nota la presenza di una serie di stringhe che contengono vari nomi di versioni del sistema operativo Windows. Al termine di questa funzione si nota la chiamata alla funzione FUN_004396E0, rinominata a gatherDataAndSend. Questa funzione viene vista anche all'interno del main dopo l'esecuzione della funzione getSystemInfo. Si passa ad analizzare la funzione gatherDataAndSend.

5.4.1 gatherDataAndSend

All'interno di questa funzione si notano chiamate a funzioni per la generazione di dati casuali (CryptGenRandom). Successivamente si nota la presenza della funzione FUN_00434FA0, quest'ultima contiene chiamate ad altre funzioni per creare un hash, utilizzando per esempio CryptCreateHash

e CryptSetHashParam. Tramite un breakpoint all'indirizzo 0x0043AA0C è possibile vedere i parametri della funzione che andrà chiamare la CryptHashData. Quello che verrà hashato è una stringa contenente i parametri di una richiesta http. Un esempio visto da un esecuzione con il debugger contiene:

- Id generato
- azione
- lingua
- versione
- se è un server
- se è un computer appartenente ad una azienda
- se è una architettura x64

Queste informazioni vengono anche criptate con la funzione encryptWithAESENC trovata successivamente. Si notano all'interno della funzione FUN_00436CB0, rinominata a communicateWithC&C, api per eseguire richieste HTTP, come per esempio HTTPSendRequest. Il formato delle informazioni informazioni criptate ed hashate potrebbe indicare la presenza di parametri nella richiesta HTTP. Si nota la presenza della stringa "POST" ed "HTTP 1.1", da questo si può evincere che le richieste vengono fatte tramite il metodo POST. Per questo motivo si fa eseguire il malware e si osserva, tramite Wireshark, le chiamate che vengono eseguite. All'interno del log di wireshark si notano una serie di pacchetti DNS e TLS. Alcune di queste richieste sono sb.scorecardresearch.com, img-s-msn-com.akamized.net, api.msn.com, c.msn.com, c.bing.com e assets.msn.com, tuttavia potrebbero essere dei tracker che vengono contattati da Edge quando aprirà asasin.htm.

Tramite esecuzione con debugger si può osservare che le funzioni che andranno a far partire la richiesta http prenderanno l'indirizzo dallo stack. Osservando questo indirizzo si può notare che è incompleto, contenendo solamente "http:///", questo può indicare che nella configurazione del malware non è stato inserito nessun indirizzo IP. Eseguendo in questo modo e forzando l'esecuzione delle istruzioni che andranno ad eseguire richieste http si può notare che falliscano e facendo terminare il malware, per questo motivo per analizzare le chiamate successive si sostituiscono le chiamate problematiche con NOP. Si tenta anche di inserire manualmente un indirizzo di una macchina locale per far connettere il malware ad un server locale, però senza successo. Per fare ciò si lancia su un'altra macchina il comando netcat e si imposta sullo stack del malware l'indirizzo di questa macchina. Dopo svariati

tentativi non si riesce a far connettere il malware, questo potrebbe essere dovuto ad altre configurazioni necessarie per far connettere il malware con la rete.

Uno dei parametri visti in precedenza è il parametro "act", questo potrebbe indicare l'azione da eseguire quando ci si connette ad un server. Tra le azioni viste durante l'esecuzione è presente l'azione getkey. Questo potrebbe indicare che il malware andrà a connettersi con un server di C&C con cui comunicherà l'avvenuta infezione e otterrà dati, come per esempio la chiave da usare per criptare una chiave AES generata casualmente. Si può ipotizzare, quindi, che nella prima richiesta si mandino le informazioni della macchina insieme ad una richiesta della chiave da utilizzare per criptare. Altre azioni viste sono quelle di gettext, che probabilmente si riferisce ad una richiesta per ricevere il testo contenente le istruzioni per ottenere il programma di decriptazione nella lingua corretta per il sistema corrente.

5.5 Zona comune

Dopo aver eseguito le azioni eseguite con o senza internet si andrà ad eseguire la funzione FUN_0046C640 (rinominata a getAllDrives). In questa funzione si nota la presenza delle chiamate GetLogicalDrives, GetDriveType, GetDiskFreeSpaceExW e GetVolumeInformationW. Alcune di queste funzioni vengono eseguite all'interno di un ciclo, questo fa pensare che si stanno raccogliendo delle informazioni sui drive della macchina virtuale. Eseguendo questa funzione si nota come venga prodotta una stringa contenente "c:". Questa stringa può indicare il drive principale di Windows, quindi si suppone che questa funzione vada a cercare i drive della macchina virtuale corrente, per questo viene rinominata a getAllDrives.

Successivamente verrà chiamata la funzione FUN_00476B50, questa funzione viene rinominata a CreateMalwareThread.

5.5.1 Create Malware Thread

Al suo interno si notano numerose chiamate alla funzione FUN_00474820. In questa funzione si otterrà un security token utilizzando GetCurrentProcess e OpenProcessToken per poi utilizzare LookupPrivilegeValueA per risolvere una stringa ad un determinato ID del permesso. Successivamente si userà la chiamata AdjustTokenPrivileges per settare il token precedentemente ottenuto con il permesso appena trovato. Questa funzione, rinominata a setupPrivilege, viene chiamata 4 volte andando a settare i privilegi di "SeDebugPrivilege", "SeTakeOwnershipPrivilege", "SeBackupPrivilege", "SeRestorePrivilege". Settando il token con "SeDebugPrivilege" si otterrano

permessi amministratore permettendo al malware di eseguire chiamate ad API privilegiate. Successivamente si nota la chiamata a GetModuleHandleA con parametro la stringa "ntdll.dll", poi si utilizzeranno chiamata a GetProcAddress per risolvere 3 API di ntdll.dll. Le api risolte sono NtQueryInformation, NtDuplicateObject e NtQueryObject, questi riferimenti verranno mantenuti come variabili globali. Infine si farà partire un thread con la chiamata CreateThread che eseguirà dalla funzione FUN_004768D0 (rinominata a threadFunction).

Analisi threadFunction All'interno di questa funzione si nota un ciclo while infinito con la chiamata della funzione FUN_00476150 (rinominata a closeOtherHandles) e uno sleep di 2 secondi.

Si passa ad analizzare la funzione closeOtherHandles. All'interno di questa funzione verrà chiamata la funzione FUN_004749C0 (rinominata a getSystemHandleInfo). Al suo interno verrà chiamata la API NtQuerySystemInformation che a seconda dei parametri può ritornare diverse tipologie di strutture contenenti informazioni sul sistema. In questo caso viene passato il parametro 0x10 che corrisponde a SystemHandleInformation. Con questo parametro la chiamata andrà a produrre la struttura SYSTEM_HANDLE_INFORMATIONS. Questa struttura tuttavia non è documentata nel manuale ufficiale di Windows, quindi alcuni dettagli potrebbero non essere correttamente analizzati.

```
struct SYSTEM_HANDLE_INFORMATION {
ULONG NumberOfHandles;
SYSTEM_HANDLE_TABLE_ENTRY_INFO Handles [ANYSIZE_ARRAY];
}
```

Le entry presenti nell'array sono del tipo SYSTEM_HANDLE_TABLE_ENTRY_INFO. Questa struttura contiene informazioni sulle handle, come per esempio UniqueProcessorId, HandleAttributes, HandleValue. Queste strutture non sono documentate da Windows, quindi alcuni campi non sono conosciuti.

Successivamente la funzione andrà a prendere l'ID attuale del processo, per poi utilizzare la funzione FUN_00474C30 rinominata openAndSwap, per ogni handle contenuta nella struttura precedente. All'interno della funzione si utilizzeranno le chiamate ad OpenProcess, con permesso di PROCESS_DUP_HANDLE, GetCurrentProcess e NtDuplicateObject (funzione non documentata che tuttavia viene chiamata dall'API DuplicateHandle). Quello che andrà a fare il thread è duplicare il riferimento delle handle presenti nel sistema per permettere al malware di eseguire operazioni su queste

handle. La chiamata NtDuplicateObject permettere di duplicare un handle di un determinato processo in modo da poterla utilizzare anche nel processo target. In questo caso verranno aperti i riferimenti di altri processi per poi duplicare le loro handle con target il malware. Poi verrà fatto partire un altro thread che utilizzerà NtQueryObject per ottenere il nome dei file che hanno come riferimento le varie handle, una volta che questo thread ha completato la sua ricerca viene terminato.

Successivamente fuori da questa funzione verrà controllato che per ogni file aperto con una handle faccia parte dell'insieme di file che il malware vuole criptare. Per fare ciò vengono utilizzate le funzioni che verrano analizzate successivamente blackListDirectory e whitelist. Se ci sarà un match all'interno di queste liste si andrà ad eseguire nuovamente la funzione openAndSwap. Questa volta si passerà il parametro DUPLICATE_CLOSE_SOURCE per la funzione NtDuplicateObject in modo da chiudere le handle. Questo viene fatto poiché lasciare handle aperte da altri processi potrebbe portare a qualche problema nel momento in cui i file verranno criptati e rinominati.

5.5.2 Creazione thread

Successivamente all'interno del main ci sarà una nuova chiamata a CreateThread, questa chiamata si trova all'interno di un ciclo for. Si suppone che venga creato un thread per ogni drive nel sistema, essendo un thread per drive non si necessitano di meccanismi di sincronizzazione. Si passa ad analizzare la funzione FUN_00429DF2 che viene eseguita dal thread (threadMalwareCrypt).

Analisi threadMalwareCrypt All'interno di questa funzione si notano chiamate a SetThreadPriority e GetCurrentThread, queste vengono fatte poiché viene cambiata la priorità del thread. All'inizio questa priorità viene impostata a THREAD_PRIORITY_LOWEST. Viene controllato un determinato dato non riconosciuto da Ghidra, se si supera il controllo verrà impostata la modalità THREAD_BACKGROUND. Questo può essere utile per nascondere l'esecuzione del malware visto che potrebbe utilizzare molte risorse. Successivamente viene chiamata la funzione FUN_0046BF70 (rinominata a findFilesAndCreateClasses), si passa ad analizzare questa funzione.

Ricerca dei file Si inizia l'analisi della funzione findFilesAndCreateClasses e si nota al suo interno diverse chiamate ad altre funzioni. Tuttavia, andandole ad analizzare, solamente la funzione FUN_0046B310 (rinominata a findFiles) risulta essere interessante poiché sono presenti al suo interno di-

verse chiamate a funzioni di DLL, mentre le altre funzioni sembrano essere di gestione di classi C++. Si passa ad analizzare la funzione findFiles.

All'interno di questa funzione si notano molte chiamate a funzioni di ricerca di file, come per esempio FindFirstFileW e FindNextFileW. Tramite FindFirstFileW è possibile enumerare i vari file all'interno del sistema, viene passato come punto di inizio la cartella di base del drive e si andrà ad enumerare i vari file. Verranno controllati i permessi tramite la funzione FUN_00462B10 (rinominata a checkPermission). Questa utilizzerà GetFileSecurityW e AccessCheck per controllare che si abbiano i permessi di accesso ai vari file enumerati. Successivamente si entrerà in un ciclo while in cui si utilizzerà la funzione FindNextFileW per avanzare con l'enumerazione. All'interno di questo ciclo si eseguiranno le funzioni FUN_0043C160 e FUN_0043EAD0. Una volta eseguita la prima si nota la presenza sullo stack di una serie di nomi di cartelle di sistema di windows (come per esempio Windows, Boot, System Volume Information, etc), questo può indicare una black list di cartelle da non prendere in considerazione per avere il loro contenuto criptato, per questo questa funzione viene rinominata a blackListDirectory. Oltre a cartelle si nota la presenza anche di 3 file all'interno di questa lista. Questi file sono _Locky_recover_instructions.txt, _Locky_recover_instructions.bmp e _HELP_instructions.txt, questo fa' pensare che non sia solamente una blacklist di directory ma anche di file.

Successivamente con la chiamata FUN_0043EAD0 si notano sullo stack una serie di estensioni di file, questa potrebbe essere la whitelist delle estensioni di file da dover criptare, per questo motivo viene rinominata a whitelist. Si nota la presenza dell'estensione .txt, per questo si prova a far partire il malware e si controlla se un file .txt appositamente creato viene criptato. Una volta eseguito si nota che questo file creato viene criptato, mentre altri file con estensioni .exe non vengono modificati, questo fa' pensare che la funzione appena trovata sia effettivamente la funzione che genera la whitelist.

Per ogni file trovato il malware verifica che non corrisponda ad uno degli elementi della blacklist, e che abbia una estensione nella whitelist, se sì verranno eseguite una serie di funzioni che sembrano allocare memoria, questo può significare che viene allocata una classe ed aggiunta in qualche struttura dati.

Funzione di criptazione Dopo la ricerca dei file viene chiamata la funzione FUN_00427AC0, questa funzione viene rinominata a cryptFiles. Al suo interno verrà chiamata la funzione FUN_0041A310 (rinominata a prepareCSP). Quest'ultima andrà a preparare il CSP e importerà la chiave mantenuta all'interno della memoria preparata all'inizio dell'esecuzione, questo

viene fatto perché successivamente verrà usato questo CSP per generare parte del nome del criptato e per criptare la chiave generata casualmente.

Successivamente si entra in un ciclo for, questo ciclo probabilmente eseguirà una volta per ogni file da criptare. Al suo interno si trova la funzione FUN_00413BE0, rinominata a encryptFile. Poi si nota la presenza di un IF con una chiamata alla funzione FUN_0042E7D0. Quest'ultima avrà al suo interno la creazione di un file tramite la chiamata CreateFileW e la scrittura di un contenuto al suo interno tramite WriteFile. Prima della chiamata si trova l'estensione .htm che corrisponde ad uno dei file creati dal malware una volta terminato l'attacco. Questo può indicare che in questa zona viene creato il file .htm da mostrare all'utente. Si nota, tramite l'esecuzione del malware, che all'interno delle cartelle viene creato un nuovo file asasin.htm, questo fa' pensare che in questa zona viene creato questo file per le varie cartelle presenti nel sistema.

Si passa ad analizzare la funzione encryptFile. Questa funzione viene eseguita per tutti i file da criptare che sono stati trovati in precedenza. Al suo interno si utilizzerà la funzione GetFileAttributesExW con il parametro "GetFileExInfoStandard" per ottenere dei metadati sul file (come per esempio le dimensioni). Il nome dei file criptati è composto da l'identificatore della macchina e da un valore casuale, entrambi sono composti solamente da caratteri esadecimali. In questa serie di istruzioni saranno generati valori casuali per poi essere convertiti a stringa. Si nota la presenza della chiamata a CryptEncryt, questa potrebbe essere utilizzata per criptare la chiave generata casualmente utilizzando la chiave RSA del malware. Successivamente il file verrà aperto tramite la chiamata CreateFileW e letto per poi essere criptato dalla funzione FUN_00473830 (rinominata ad encryptWithAESENC). Una volta criptati i dati verranno riscritti nel file, per poi modificare il file time utilizzando SetFileTime per impostarlo al valore di GetSystemTimeAsFileTime. Questi file avranno come momento di creazione il momento in cui sono stati criptati. Infine si va a flushare il contenuto del file per poi chiudere la handle.

All'interno della funzione encrypWithAESENC si notano tramite Ghidra la presenza di istruzioni AESENC. Queste istruzioni sono disponibili in versioni più recenti di CPU ed offrono il supporto hardware alla cifratura di dati. Si è tentato di analizzare dinamicamente questa funzione, tuttavia le istruzioni AESENC non vengono riconosciute da OllyDbg e portano allo scorretto funzionamento del debugger. Tuttavia il loro funzionamento si può derivare dal manuale Intel e dal contesto in cui vengono utilizzate. La chiave utilizzata da queste istruzioni potrebbe essere stata generata casualmente, per poi criptarla utilizzando la chiave RSA inclusa nel malware. Questo può essere confermato andando a far girare il più volte malware con file identici.

Quello che si può osservare è che i file risultano essere differenti ogni volta. Inoltre osservando i file generati si può vedere che viene creato un file in più rispetto a quelli presenti sul desktop, questo può essere la chiave generata casualemente e criptata con RSA.

Connessione ad internet attiva All'interno della funzione cryptFiles, se un byte di controllo è uguale a 0 si andranno ad eseguire una serie di istruzioni aggiuntive. Si ritrova la chiamata alla funzione analizzata in precedenza gatherDataAndSend. Andando a vedere i parametri di questa funzione si può notare una stringa contenente i parametri da inviare tramite la richiesta http, in questa stringa troviamo:

- ID della macchina
- l'azione da eseguire: in questo caso contiene la stringa stats
- numero di file criptati
- numero di file in cui si ha avuto un fallimento durante la cifratura
- lunghezza: probabilmente indica quanti byte sono stati criptati

Con questa richiesta si andranno ad inviare al server di C&C statistiche sulla infezione in corso. Per riuscire ad arrivare a debuggare questa zona è stato necessario modificare i salti condizionali fatti dal malware.

5.5.3 Richieste IPC

Dopo la creazione dei thread si vede nel main la chiamata alla funzione FUN_0040FA10 (rinominata a deleteShadow). Analizzando questa funzione si vedono chiamate alla libreria COM che permette di invocare funzioni appartenenti ad altri processi. Si nota la presenza di una stringa offuscata, questa stringa contiene il comando "vssadmin.exe Delete Shadows /Quiet /All". Questo comando può essere utilizzato per eliminare i shadow volumes presenti in windows. Questi volumi contengono una versione precedente dei dati contenuti in una cartella, quindi è di interesse del malware cancellarli per impedire di recuperare i dati dopo l'infezione. Si nota anche la presenza della funzione FUN_00478420, questa funzione andrà a caricare la vssapi.dll e le api CreateVssBackupComponentsInternal, VssFreeSnapshhotPropertiesInternal. Probabilmente vengono utilizzate queste istruzioni come metodo alternativo per eliminare shadow volumes.

Infine si nota la chiamata della funzione FUN_0040D5C0, al suo interno si vede l'utilizzo della libreria COM per chiamare funzioni tramite IPC. Tuttavia tramite debugger non è possibile vedere che funzioni vengono chiamate. Nonostante ciò si può notare che vengono chiamate funzioni di taskchd. Quest'ultimo viene utilizzato per gestire le task in windows.

5.5.4 Persistenza del malware

Continuando con l'analisi del main, si notano una serie di chiamate a funzioni per gestire chiavi di registri di Windows. Si nota la presenza della stringa "Software\Microsoft\Windows\CurrentVersion\Run". Andando a vedere sulla documentazione si nota che questa è una sottochiave di HKEY_CURRENT_USER che viene utilizzata per far partire programmi all'avvio del sistema. Si nota la presenza di una chiamata a RegSetValueExW, questo fa' pensare che il malware si aggiunga ai programmi da far partire al boot. Tramite debugger si vedono che i parametri di RegSetValueExW sono "opt321" con tipo REG_SZ ed il path del malware. Questo indica che il malware imposterà il nome "opt321" per impostarsi allo startup.

Queste operazioni sui registri vengono fatti solamente se un determinato valore della zona di memoria allocata in precedenza è diversa da 0. Questo può indicare che a seconda della configurazione il malware si può avviare al boot del sistema. Nel caso di esecuzione normale queste azioni non vengono eseguite.

Continuando l'analisi si nota anche la presenza di una chiamata a RegDeleteValueA, questo cancellerà la chiave precedentemente aggiunta.

Inoltre si nota la presenza della funzione FUN_004273C0, questa funzione viene rinominata a addAtoms. Al suo interno verranno chiamate le API per l'aggiunta di atoms nella tabella globale ed in quella dell'applicazione. Queste funzioni aggiungeranno l'atom "~~~ID~~~" dove ID è l'ID generato in precedenza. Tuttavia si nota che l'ID aggiunto risulta essere diverso da quello cercato dalla funzione searchAtom. Il valore ricercato corrisponde con l'ID generato tramite la prima funzione di generazione dell'ID, mentre nell'esecuzione normale del malware si nota che viene eseguita una seconda funzione per generarlo. Tramite debugger si nota che l'ID aggiunto corrisponde con quello generato tramite la seconda funzione.

5.5.5 Creazione immagini e cambio wallpaper

L'ultima funzione importante analizzata nel main è la funzione FUN_00425870, rinominata a createImagesAndEditWallpaper. Al suo interno si possono notare una serie di stringhe che possono corrispondere a chiavi di registro,

queste chiavi trovate possono essere utilizzate per cambiare wallpaper al sistema. Questo è una azione eseguita dal malware, quindi si pensa che venga fatta all'interno di questa funzione. Si nota la chiamata alla funzione FUN_0042E6B0, al suo interno si trova la chiamata a FindFirstFileW per ricercare un file nel sistema, per questo motivo viene rinominata a searchFile. Sono presenti due chiamate a searchFile, la prima per trovare asasin.htm presente nel desktop, se non viene trovata si va a chiamare la funzione FUN_0042E7D0 che creerà la pagina HTML del malware. Successivamente c'è una nuova chiamata alla funzione searchFile che andrà a cercare la bitmap asasin.bmp. Questa bitmap è quella che viene aperta una volta che il sistema è stato infettato. Se questa bitmap non viene trovata si andrà a generare tramite la funzione FUN_004248B0, al suo interno si possono notare molte chiamate a funzioni di Windows per generare bitmap, come per esempio CreateCombatibleBitmap.

Una volta ottenuta la bitmap (creandola o ricercandola) si andranno a modificare le chiavi di registro contenute in HKEY_CURRENT_USER\Control Panel\Desktop. Le chiavi modificate sono WallpaperStyle, che viene impostata a 0, ed TileWallpaper, che viene impostata al percorso della bitmap. In questo modo il malware può cambiare lo sfondo del desktop mostrando l'immagine che mostra l'avvenuta infezione all'utente.

Infine si notano le chiamate alla funzione ShellExecuteW, utilizzando il debugger è possibile vedere che venga eseguito il comando "open" sulla bitmap e sulla pagina .htm. Con queste due chiamate si andranno a mostrare all'utente la pagina .htm e la bitmap.

6 Riassunto

Il malware analizzato è una versione del ransomware Locky. Da quanto ottenuto si può capire il suo funzionamento generale. Il malware inizierà preparando una area di memoria. Questa area di memoria potrebbe essere impostata a seconda di come è stato configurato il malware. Nel caso analizzato si ha un malware che non andrà a comunicare con internet. Una volta generato l'ID che identifica la macchina, il malware andrà a chiudere le handle di altri programmi verso i file tramite l'esecuzione di alcune funzioni di ntdll, queste operazioni vengono fatte da un thread che rimane attivo durante il ciclo di vita del malware. Successivamente andrà a far partire un thread per ogni drive del sistema. Questo thread andrà a cercare quali file dovrà criptare il malware utilizzando una whitelist ed una blacklist. Una volta trovati i file di interesse al malware utilizzerà delle istruzioni assembler per criptarli, utilizzando probabilmente una chiave casuale per poi criptarla con

la chiave presente all'interno della zona di memoria preparata in precedenza. Infine utilizzerà delle facility del sistema operativo per cancellare i shadow volume per poi generare una bitmap impostandola come sfondo del desktop.