Progetto pratico 2: creazione di un malware per Windows

Michele Perlotto Gabriele Roncolato

I. Introduzione

Il progetto è composto da 5 elementi principali:

- **DesktopChanger**, un eseguible dal comportamento apparentemente innocuo che modifica lo sfondo del desktop e contiene tanuki_the_dropper come risorsa embeddata, scritto in .NET 6.0;
- **tanuki_the_dropper**, il malware dropper richiesto dalla specifica del progetto, scritto in .NET 6.0;
- **server_dropper**, il server C2 associato a tanuki_the_dropper richiesto dalla specifica del progetto, scritto in python;
- **tanuki_the_cryptor**, un semplice ransomware che viene scaricato ed eseguito sulla macchina della vittima da tanuki_the_dropper, scritto in .NET 6.0;
- **server_malware**, un server C2 associato a tanuki_the_cryptor che si occupa di memorizzare e comunicare la chiave di decifratura al ransomware, scritto in python.

I 3 programmi .NET sono stati compilati in modo che generassero un singolo file eseguibile contenente tutte le librerie utilizzate.

II. Analisi tanuki_the_dropper

Di seguito andremo ad analizzare un possibile attacco attuabile con **tanuki_the_dropper** rispetto alle varie fasi previste dalla Cyber Kill Chain.

A. Reconnaissance

Poiché il dropper è distribuito a potenziali vittime attraverso email si suppone che nella fase di reconnaissance gli attaccanti raccolgano grandi quantità di indirizzi email attraverso scan automatizzati di social network oppure attraverso campagne email. Ogni indirizzo email raccolto è individuato come potenziale target e usato come destinatario per un'email con allegato il malware.

B. Weaponization

Per rendere possibile l'attacco sono state sviluppate tre componenti principali: DesktopChanger, tanuki_the_dropper.exe e un server C2 in python che permette il download del ransomware attraverso l'invio di byte con socket. Il dropper è stato weaponizato usando social engineering ed embedding: il dropper è stato embeddato dentro l'eseguibile DesktopChanger, il cui comportamento è apparentemente benevolo, e si utilizza poi social engineering per portare la vittima a scaricare ed eseguire quest'ultimo. Per oscurare il download del ransomware da parte del dropper il canale di comunicazione fra quest'ultimo e il server viene oscurato con cifratura.

C. Delivery

Per la fase di delivery vengono raccolti tutti gli indirizzi email ottenuti nella fase di reconnaissance e vi si invia un'email contenente il seguente corpo:

"Hey there! It's been a long time since we talked. I hope this email finds you well. I just wanted to share with you this cute tanuki wallpaper that I found online. It's my new favorite and I think you would love it too! I've attached the file to this email, just download and run it and it will automatically set it as your wallpaper. It's super easy and harmless, I promise!"

L'email ha infatti allegato uno zip contenente DesktopChanger in formato exe.

D. Exploitation

Attraverso l'uso di social engineering si convince la vittima a scaricare l'allegato e ad eseguirlo nel proprio sistema: per fare questo si finge di essere un amico del destinatario e si descrive il comportamento dell'eseguibile come innocuo (fingendo che si limiterà a modificare lo sfondo del desktop). Una volta che la vittima ha eseguito il file si può procedere con la fase di installazione.

E. Installation

La fase di installazione si compone di due passi:

- Replicazione;
- Persistenza.

DesktopChanger, una volta avviato, procede con la modifica dello sfondo e con l'avvio del malware dropper. Una volta avviato il dropper verifica se, all'interno del sistema della vittima sono presenti le cartelle **Immagini** e **AppData\Local**. Se queste esistono procede nel creare una propria copia al loro interno (a patto che non esista già), settando l'attributo "Nascosto" nel nuovo file. Questo viene fatto per rendere più difficile ad un utente inesperto scoprire le copie.

Procede poi nel modificare il registro **HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run** aggiungendo due nuove entry, una per copia, come mostrato in figura 1. Grazie a questo le due copie verranno eseguite ad ogni avvio del sistema operativo.

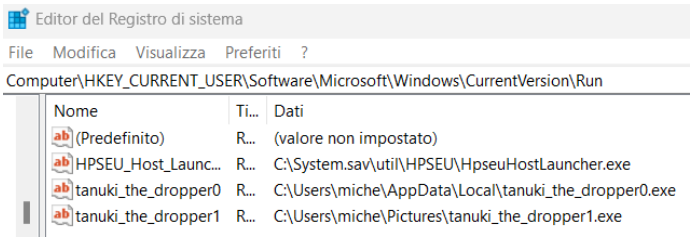

Fig. 1. Aggiunta delle entry.

F. Command and Control

Terminata la fase di installazione il dropper procede ad instaurare una comunicazione con il server C2, il cui indirizzo è hardcodato.

Per prima cosa verifica se è in grado di connettersi al server C2. In caso caso di insuccesso, attenderà un tempo stabilito prima di riprovare per un numero prestabilito di tentativi. Lo stesso comportamento è adottato nel caso in cui si verifichino successivi errori di comunicazione (ad esempio nella generazione delle chiavi).

Visto lo scopo del malware i tentativi sono stati impostati a tre, mentre il tempo di attesa fra ogni tentativo a 6 secondi. Una volta esauriti i tentativi il dropper termina la sua esecuzione.

I messaggi scambiati possono essere divisi in due gruppi:

- Messaggi per la generazione della chiave di AES a 256 bit da utilizzare per cifrare gli scambi successivi;
- Messaggi per la ricezione di dati.

Vediamo ora la parte di generazione della chiave. La chiave di AES viene generata a partire da una chiave a 1024 bit scambiata in modo sicuro con Diffie-Hellman (che abbrevieremo in DH) derivata poi con PBKDF2 per ricavare una chiave più piccola a 256 bit. Initialization vector (per AES) e salt value (per PBKDF2) sono prestabiliti. Il Dropper parte inviando al server un messaggio

{”Operation”: ”keyExchangeGen”}

per dare inizio alla generazione della chiave di DH. Il server risponde con

{”Operation”: ”keyExchangeAns”, ”Prime”: ”value”, ”Generator”: ”value”, ”Gx_server”: ”value”}

contenente rispettivamente il modulo, il generatore e la ”chiave pubblica” calcolata dal server. Il dropper calcola quindi la sua ”chiave pubblica” e la invia al server con il messaggio

{”Operation”: ”keyExchangeAns”, ”Gx_client”: ”value”}

A questo punto server e dropper applicano PBKDF2 per generare la chiave finale condivisa. Da questo momento, le comunicazioni successive verranno cifrate con AES usando la chiave ottenuta.

Nel caso in cui si verifichi un errore, il dropper chiude la connessione corrente e ne apre una nuova, rieseguendo da capo l’intero processo di generazione. Come scritto sopra, il verificarsi di un errore porta al ”consumo” di un tentativo.

Calcolata la chiave, il dropper elimina tutti i file associati al ”vero” malware, se presenti, e invia un nuovo messaggio al server

{”Operation”: ”exeSend”}

per dare inizio alla fase di download. Il server risponde inviando

{”Operation”: ”exeHash”, ”Hash”: ”hex_value”, ”DataLength”: value},

contente l’hash calcolato dal server con SHA256 dell’archivio zip contente il ransomware e la sua dimensione in byte. A seguito di questo messaggio il server invia l’archivio contenente il secondo malware diviso in blocchi (per rispettare la dimensione massima del buffer). Il dropper riceve e decifra il primo messaggio e l’archivio e verifica se la lunghezza di quest’ultimo e l’hash calcolato su di esso corrispondono a quelli ricevuti nel primo messaggio.

Se non corrispondono il dropper richiede le informazioni di controllo e l’archivio al server per un massimo di altre due volte. Se per ogni tentativo disponibile questi non corrispondono il dropper chiude la comunicazione e riparte dalla fase di generazione della chiave, ”consumando” uno dei tre tentativi rimanenti.

Se invece corrispondono, il dropper scrive l’archivio su disco, ne estrae il malware nella cartella **malware** creata al momento ed elimina lo zip. Il tutto viene effettuato all’interno della directory dove è presente l’eseguibile tanuki_the_dropper attualmente in esecuzione. Inoltre, sempre per rendere più difficile ad un utente inesperto scoprire il download del malware, la cartella **malware** e il suo contenuto avranno l’attributo ”Nascosto” settato. Fatto questo, il dropper esegue il malware.

Nel caso in cui si verifichino errori nell’estrazione/scrittura o esecuzione del malware, sempre come sopra, il dropper chiude la connessione corrente, ne apre una nuova e riparte dalla fase di generazione della chiave.

G. Actions and Objectives

L’obiettivo principale di tanuki_the_dropper, in quanto malware dropper, è quello di effettuare il download, la scrittura su disco e l’esecuzione di un secondo malware nel sistema attaccato oscurando la comunicazione con cifratura. Per raggiungere questo obiettivo il dropper si connette dalla macchina della vittima ad un server C2 di proprietà degli sviluppatori del malware e, dopo aver stabilito un canale di comunicazione cifrato, il server invia un archivio contenente il secondo malware affinché il dropper lo salvi nel file system per poi eseguirlo.

III. Analisi tanuki_the_cryptor

Analizziamo ora il funzionamento di tanuki_the_cryptor. Il suo comportamento può essere schematizzato come segue:

- Stabilisce un un canale di comunicazione cifrato con chiave asimmetrica con un server C2;
- Genera una chiave di cifratura simmetrica e la invia al server;

- Cifra i file di una cartella bersaglio sulla macchina della vittima;
- Chiede all'utente se vuole decifrare i file:
  - Se la risposta è affermativa, il malware richiede la chiave al server e decifra i file;
  - Se la risposta è negativa, il malware chiude la comunicazione e termina la sua esecuzione.

Per evitare sovrapposizioni di cifrature, dovute ad esempio ad esecuzioni in parallelo del malware da parte del dropper, abbiamo deciso di permette una sua sola esecuzione per volta.

La comunicazione tra malware e server è cifrata con RSA (a 2048 bit). Per permettere questo sono state generate due coppie di chiavi:

- La coppia chiave pubblica/privata del server, usata per cifrare i messaggi inviati dal malware al server;
- La coppia chiave pubblica/privata del malware, usata per cifrare i messaggi inviati dal server al malware.

Per comodità le chiavi usate dal malware (la chiave pubblica del server e la chiave privata del malware) sono state hardcodate al suo interno.

A. Avvio

Una volta avviato il malware verifica se all'interno del registro **HKEY CURRENT USER\Software** esiste la subkey **tanuki_the_cryptor** e in caso affermativo estrae i valori delle chiavi *id* ed *encrypted*:

- **id** contiene una sequenza di byte casuale che identifica, sul server, la vittima e la corrispettiva chiave di cifratura simmetrica usata per cifrare i file del sistema attaccato;
- **encrypted** esiste solo se, dopo aver generato la chiave simmetrica, i file bersaglio sono stati effettivamente cifrati (il valore che contiene non è importante ai fini dell'esecuzione).

Se *id* e *encrypted* esistono, significa che il malware è già stato eseguito in precedenza e che i file cifrati non sono stati decifrati, o per errore di comunicazione (ad esempio durante la richiesta della chiave) o perchè l'utente ha risposto negativamente alla richiesta del malware. Il malware passa quindi direttamente alla fase di richiesta al server della chiave di decifratura, saltando quelle precedenti.

Se *id* esiste e *encrypted* no, allora il malware ha inviato in precedenza una chiave simmetrica al server ma non ha mai iniziato a cifrare i file bersaglio. In questo caso il malware segue la sua esecuzione standard rigenerando una nuova chiave.

Se *id* e *encrypted* non esistono, il malware procede standard come sopra.

Per semplicità, abbiamo deciso di non considerare il caso in cui i file bersaglio vengono parzialmente cifrati (o decifrati).

B. Generazione della chiave simmetrica

Verificato che non esistono le chiavi, il malware una chiave per AES a 256 bit, un initialization vector e un id. Li invia poi al server assieme all'hash calcolato con SHA256 sulla loro concatenazione. Il server calcola a sua volta l'hash e ne verifica la corrispondenza. Se questa è vera, allora procede a memorizzare in un database mysql locale i dati ricevuti, inviando al malware un messaggio "keySuccess" per indicare l'avvenuta memorizzazione. Il malware crea quindi la subkey **tanuki_the_malware** e vi aggiunge al sui interno la chiave *id* con il valore generato sopra.

Se l'hash non corrisponde o vi sono errori legati alla scrittura su database il server invia un "keyFailure" e non memorizza nulla. Il malware si comporta similmente a quanto descritto per i tentativi di tanuki_the_dropper: chiude la connessione, ripete l'operazione di generazione e rinvia i dati al server, consumando un "tentativo". Consumati i tre tentativi (effettuati a distanza di 6 secondi) termina la sue esecuzione.

C. Cifratura dei file

Una volta inviata la chiave il malware procede a cifrare i dati contenuti nella cartella target con AES usando la chiave appena generata. Al termine della cifratura aggiungerà dentro la subkey **tanuki_the_malware** la chiave *encrypted* con valore *true* per indicare che i file sono stati cifrati con successo.

Fatto questo avvisa l'utente che i suoi file sono stati cifrati e chiede se vuole decifrarli o meno.

Se si dovessero verificare errori durante la cifratura il malware termina la sua esecuzione.

D. Ricezione della chiave simmetrica

Se l'utente conferma di voler decifrare i file il malware invia al server una richiesta di recupero della chiave accompagnata dal valore della chiave di registro *id* per identificare la vittima.

Nel caso in cui questo id non sia presente nel database il server chiude la connessione, altrimenti invia la chiave e l'initialization vector associati all'id insieme all'hash ottenuto dalla concatenazione dei due valori (sempre con SHA256).

Il malware calcola l'hash dei dati ricevuti e lo confronta con quello ricevuto dal server. Se non combaciano ripete la richiesta dopo aver aperto una nuova connessione per un massimo di altre due volte. Una volta consumati i tre tentativi termina l'esecuzione.

E. Decifratura dei file

Ricevuti i dati necessari il malware procede a decifrare i file precedentemente cifrati. Al termine della decifratura elimina la subkey **tanuki_the_malware** e termina la sua esecuzione.

In caso di errori durante la decifratura il malware termina la sua esecuzione.

IV. Test

I test sono stati effettuati su due macchine virtuali create con VirtualBox collegate alla stessa Rete con NAT. I server sono stati eseguiti su Ubuntu 20.04 mentre i malware sono stati eseguiti su Windows 10.