Università del Molise

Dipartimento di Bioscienze e Territorio

Proposta di progetto

Attacco e difesa di Cryptojacking basato su GPU

Author:
Federico Zappone

Networking security and software security

Dicembre 01, 2020

Introduzione

In quest'ultimo decennio si è molto discusso di monete digitali e delle così dette criptovalute in seguito all'avvento delle blockchain. Quest'ultima è una tecnologia basata sul concetto di database distribuito, ovvero un sistema che utilizza un registro condiviso per salvare informazioni, esso è accessibile solo ai nodi della stessa rete ed è rappresentabile come una successione di blocchi contenenti le informazioni. Contemporaneamente alle blockchain è nata anche la prima criptovaluta, più precisamente, il 3 gennaio 2009 veniva scritto il primo blocco di *Bitcoin* [1] dal valore complessivo inferiore a un dollaro statunitense. In seguito il 6 novembre 2010 *Bitcoin* si presentava con un valore di $0,50, in meno di due anni il prezzo era aumentato di 625 volte, e da allora in poco più di sette anni, raggiunse il suo massimo storico di quasi $20.000 ovvero circa 40.000 volte in più [2] [3]. Divenuto un caso più unico che raro, *Bitcoin* si è posto da apripista a più di 5000 altre criptovalute [4] fino ad essere definito come l'oro del XXI secolo. Questa definizione non è dovuta solo all'incredibile aumento del prezzo negli ultimi anni ma anche ad una delle caratteristiche chiave che sia l'oro che la maggior parte delle criptovalute condivide, il processo di "estrazione" definito appunto come *mining* nel campo delle monete digitali.

Il *mining* di criptovalute consiste nel creare monete virtuali attraverso la risoluzione di alcune funzioni crittografiche necessarie per la validazione delle transazioni e dei blocchi che compongono una blockchain. Questa operazione termina con un compenso da parte della blockchain al *miner*, il quale, avendo offerto la sua potenza di calcolo viene premiato con la stessa criptovaluta minata. Questi calcoli vengono eseguiti da sistemi informatici dedicati a questo specifico processo, essi sono divisi in due grandi macro sezioni: quelli che sfruttano la Central Processing Unit (CPU) e quelli che sfruttano la Graphics processing unit (GPU). I primi prendono il nome di *ASIC*, acronimo di *Application Specific Integrated Circuit*, ovvero circuiti costruiti per la risoluzione di un calcolo ben specifico che risultano però molto inefficienti su altri tipi di algoritmi. I sistemi basati su GPU sono invece spesso molto più prestanti, le schede video riescono infatti a effettuare più calcoli al secondo rispetto alle CPU che risultano quindi meno redditizie nella maggior parte dei casi di *mining*. D'altro canto per le GPU risulta essere molto più difficile la gestione delle temperature e inoltre comportano costi maggiori sia in termini di assemblaggio che di costi energetici per il mantenimento. Proprio questi ultimi aspetti sono quelli di notevole impatto per il *mining*: il costo di acquisto delle componenti è nettamente aumentato negli ultimi anni, questo sia a causa delle nuove scoperte tecnologiche più performanti, sia a causa delle grandi farm di *mining* che costantemente acquistano nuove schede video.

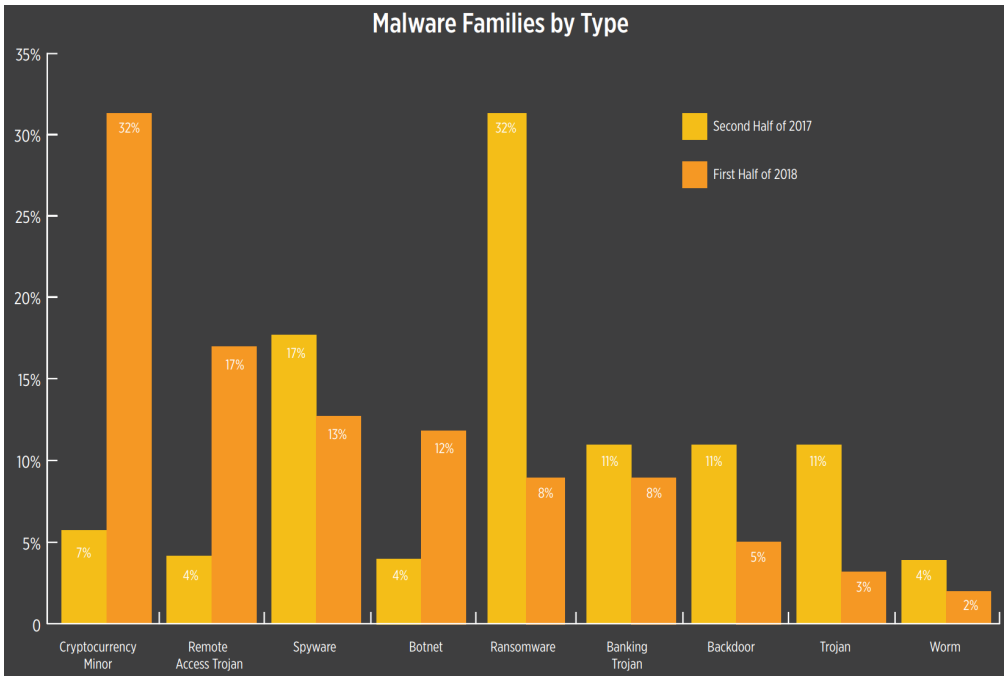
Con l'aumentare dei costi di *mining*, e allo stesso tempo delle opportunità di guadagno offerte dal

mondo in crescita delle blockchain, le criptovalute hanno iniziato a risaltare agli occhi del crimine informatico. Più precisamente la tecnologia blockchain, e di conseguenza le criptovalute, posseggono, per lo meno la maggior parte di esse, una caratteristica molto importante per i cyber criminali, ovvero garantiscono una certa forma di anonimato. Le criptovalute infatti non sono in alcun modo collegabili a un individuo diversamente da un conto bancario, l'unico ad averne pieno controllo è colui che le possiede. Questa caratteristica ha portato a un enorme utilizzo delle criptovalute per lo svolgimento di azioni illecite, nel 2017 infatti si è riscontrato un netto aumento di *ransomware* [5], ovvero quei virus che una volta bloccato il sistema al quale accedono chiedono un riscatto per il suo sblocco. L'incremento di questi attacchi è dovuto principalmente alla pratica dei cyber criminali di richiedere il riscatto sotto forma di criptovalute per mantenere l'anonimato. Successivamente si è arrivati a una forma di attacco più intelligente e meno invasiva che utilizza i *cryptominers*. Definito appunto come "l'anno dei cryptominers", il 2018 vede la stessa impennata di casi di *ransomware* dell'anno precedente sotto una nuova forma di attacco che si appropria di criptovalute in modo illegale ma sfruttando un'operazione lecita come quella del *mining* 1.1. Questa tecnica detta *cryptojacking* consiste in programmi che utilizzano il computer della vittima per l'estrazione di criptovalute ma, diversamente dal *mining* legittimo, il guadagno di questo processo viene poi attribuito non ai possessori del calcolatore bensì all'attaccante. Il tutto avviene seguendo un basso profilo, tramite programmi infetti che vengono installati sul computer o, come più recentemente si è affermato, utilizzando script malevoli presenti all'interno di pagine web. La sostanziale differenza tra le tecniche sta principalmente nella loro rintracciabilità, infatti quelli che operano attraverso il web sono più difficili da individuare e analizzare rispetto a programmi che vengono installati e che sono quindi sotto l'occhio diretto di antivirus e delle politiche dettate dal sistema operativo.
Le GPU grazie alla loro potenza offrono quindi nella maggior parte dei casi dei ricavi maggiori nel *mining* rispetto alle CPU, di contro però non sono facilmente utilizzabili in ambito web. Ciò ha portato infatti ad uno sviluppo maggiore dei *cryptominers* basati su CPU e, di conseguenza, a sistemi di difesa che si concentrano su di essi, trascurando le minacce derivanti da attacchi pensati per Graphics Processing Unit. Infatti sia Musch et al. [6] che Saad, Khormali e Mohaisen [7] mostrano la diffusione di *cryptominers* basati su CPU nei siti web, inoltre analizzano l'efficacia di tecniche difensive di blacklisting che risultano però essere una protezione insufficiente e poco pratica. Wang et al. [8] forniscono in seguito un metodo di analisi di firme basate su istruzioni della CPU durante l'esecuzione dei moduli *WebAssembly*. Konoth et al. [9] introducono invece *MineSweeper* basato anch'esso sul precedente principio di firme ma aggiunge inoltre il rilevamento di eccessive chiamate di sistema di natura crittografica durante l'esecuzione di un programma. Kharraz et al. [1] dimostrano invece che come l'analisi della CPU generi una grande quantità di falsi positivi all'interno della navigazione web. Un differente approccio viene fornito da Tahir et al. [10] che presentano *MineGuard*, un sistema che tramite l'analisi delle prestazioni hardware rileva l'esecuzione di alcuni algoritmi di *mining*. *MineGuard* monitora costantemente sia CPU che GPU ed inoltre offre un'ottima soluzione sia in termini di efficienza che di utilizzo di risorse per la sua esecuzione. Belkin, Gelernter e Cidon [11] offrono infine una soluzione dedicata esclusivamente a *cryptominers* di GPU che risulta essere più pratica in ambito web rispetto alle altre, questo analizzando le pagine web che utilizzano *WebGL* [12]. Quest'ultima è una delle librerie web basate sulla specifica *OpenGL* (*Open Graphics Library*) considerata ormai lo standard per quanto riguarda la grafica tridimensionale in

sistemi operativi *Unix-like*. In particolare *OpenGL* è stata pensata per architetture parallelizzabili come le GPU e definisce delle *API* per applicazioni che operano in ambienti 3D. Su questo standard sono basate le librerie che permettono di utilizzare la GPU in ambito web. Più precisamente *WebGL* è una *Web-based Graphics Library* ovvero una libreria pensata per la gestione di elementi grafici all'interno del web, essa fornisce delle *API* per grafica 3D in un contesto *HTML5* tramite l'utilizzo del *Document Object Model (DOM)*. *GPU.js* è invece una libreria scritta interamente in *JavaScript*, pensata per sfruttare l'accelerazione hardware delle GPU basata a sua volta su *WebGL*.
L'idea di base di questo progetto è quindi quella di sviluppare un *cryptominer* che, attraverso lo standard *OpenGL* [13] e l'utilizzo di un linguaggio di scripting, sia in grado di effettuare *mining* di criptovalute sfruttando la GPU. L'utilizzo di un linguaggio di scripting web è necessario visto che per portare a termine l'attacco saranno sfruttate le vulnerabilità al *Cross Site Script (XSS)*. Gli attacchi di tipo *XSS* sono tra i più diffusi nel web, infatti, secondo il report di Positive Technologies Security [15], la percentuale di questi attacchi è salita dall'occupare il 77,9% nel 2017 all'88,5% di tutte le tipologie di attacchi web registrate nel 2018. Secondo Precise Security [16] invece, in rapporto a tutti i tipi di attacchi sferrati verso le grandi compagnie di Europa e Nord America nel 2019, la percentuale di attacchi *Cross Site Script* risulta essere del 39%, superando più del doppio la percentuale degli attacchi di *SQL Injection*.
L'*XSS* consiste nell'introdurre del codice arbitrario lato client all'interno dei siti web con il fine di eseguire una serie di attacchi rivolti ai visitatori. Questa vulnerabilità affligge i siti dinamici che non effettuano un controllo sugli input lato client, il che porta ad una "fusione" tra il payload, ovvero il codice inserito dall'attaccante, e il codice reale della pagina. Gli attacchi di *Cross Site Script* si suddividono in tre gruppi:

- **XSS reflected:** il payload viene eseguito solo nel momento dell'iniezione;
- **XSS stored:** il payload viene salvato all'interno della struttura dell'applicativo ed eseguito ad ogni accesso del contenuto;
- **XSS dom-based:** il codice sorgente e la risposta del server non vengono modificate, il payload viene eseguito a runtime senza inoltrare richieste al server ma utilizzando il codice già presente nella pagina.

FIGURA 1.1: Top Malware Families by type, 2018, *Skybox Vulnerability Report Trends* [5]



Proposta di progetto

Lo scopo ultimo del progetto è quello di sviluppare un attacco web basato sulla tecnica del *cryptojacking* che operi su GPU e successivamente sviluppare una difesa per attacchi simili.
Per il raggiungimento di tale scopo è necessario innanzitutto sviluppare uno script che effettui il *mining* attraverso l'utilizzo della GPU in ambito web. Successivamente creare un sistema che permetta al codice malevolo di raggiungere le vittime attraverso pagine web e testarne l'efficacia in un ambiente simile a quello reale. Una volta sviluppata e testata l'efficienza del sistema di attacco, sarà poi possibile analizzarlo e identificare una modalità di difesa efficiente e, se possibile, utilizzarla per implementare un sistema contro attacchi dello stesso genere.
Per il raggiungimento dei due obiettivi principali si è pensato all'utilizzo combinato di librerie che permettono l'interazione con la GPU in ambito web e tipologie di attacco che permettono l'iniezione di codice malevolo all'interno di applicativi web.
Si è quindi preso in considerazione l'utilizzo di librerie come *WebGL*, in particolare *GPU.js* che, come dimostrato proprio dagli sviluppatori [17], computa circa quindici volte più velocemente della sola CPU il caricamento di elementi grafici complessi. Operando in un contesto di grafica tridimensionale, queste implementazioni permettono di effettuare operazioni anche complesse attraverso la GPU dell'utente che visualizza la pagina web dove sono utilizzate. L'idea è quindi quella di sviluppare un *cryptominer* attraverso l'utilizzo di queste librerie e del linguaggio *Javascript*. La scelta di questo linguaggio non è dovuta solo al fatto che è uno dei più utilizzati e supportati all'interno del web [18], ma anche al fatto che si presta benissimo per l'iniezione di codice malevolo come mostrato da OWASP [19].
Infatti, per poter raggiungere le vittime ed effettuare l'operazione di *mining* tramite pagine *HTML*, è necessario che lo script creato sia presente nella struttura della pagina visitata dalle vittime. Per fare ciò si potrebbe creare un'applicazione web con il codice malevolo ma questa non sembra essere la soluzione migliore, in quanto, oltre a raggiungere un numero molto limitato di vittime, aumenterebbe il rischio di essere scoperti nel caso si stesse effettuando un attacco reale. Proprio per rendere l'attacco il più veritiero possibile, si è pensato invece di effettuare una simulazione dell'inserimento del codice malevolo all'interno di siti web vulnerabili realmente esistenti, questo sfruttando attacchi di tipo *XSS*.
Esistono vari strumenti in grado di analizzare la struttura delle pagine web e di identificare vulnerabilità *XSS*, come ad esempio *XSStrike* [20], *Traxss* [21] e *XSSer* [22]. Questi sono tutti tool open-source di analisi sviluppati in *Python*, il più interessante è però *XSSer*, uno dei tool preinstallati presenti nelle distribuzioni *Linux* atte al *penetration testing*. Si è quindi pensato di utilizzare

uno di questi tool per individuare in modo semi-automatico o del tutto automatico le vulnerabilità *XSS* all'interno di siti web. Per l'identificazione dei siti target si utilizzerà una lista pubblica di siti vulnerabili, ad esempio quella fornita da xssed [23]. Una volta individuato un buon numero di siti vulnerabili si creerà un ambiente di lavoro dove effettuare i test di *injection*. L'ambiente di test dell'attacco sarà infatti composto da copie dell'intera struttura dei siti vulnerabili identificati riprodotte sulla macchina locale, questo sempre per rendere i test il più veritieri possibile senza realmente effettuare l'attacco verso i siti originali.
Una volta testata l'efficacia del sistema di attacco, sarà effettuata un'analisi per individuare una possibile difesa attuabile dagli utenti degli applicativi web. Una possibile soluzione sarebbe quella di sviluppare un'estensione per il browser che individui la presenza di librerie web che utilizzano la GPU e avvertire di ciò l'utente che sta accedendo alla pagina. In questo modo si potrebbe evitare che l'utente acceda al contenuto della pagina sospetta o, che dopo un'attenta verifica, scelga se disabilitare o meno alcune funzionalità della pagina. Questa soluzione potrebbe però non essere efficiente se gli attacchi saranno effettuati con l'aggiunta di tecniche di offuscamento del codice come quella sviluppata da Sharif et al. [24]. In tal caso si potrebbe valutare l'opzione di monitorare le prestazioni dell'hardware per individuare lo svolgimento dell'attacco e bloccarne l'esecuzione. Non saranno effettuate analisi dal punto di vista del *Cross Site Script* in quanto modi per evitare questo tipo di vulnerabilità sono già stati studiati ed analizzati a fondo negli ultimi anni, come è stato fatto proprio da Bisht e Venkatakrishnan [25].
Lo sviluppo dei sistemi avverrà con l'ausilio di una macchina virtuale, questa scelta non è dovuta tanto ai danni che lo sviluppo potrebbe portare sul sistema, ma al fatto che non si è a conoscenza delle operazioni effettuate dai siti vulnerabili che si andranno a riproporre in copia locale. Inoltre l'utilizzo di una macchina virtuale fornirà un ambiente privo di agenti esterni che possano interferire con l'esecuzione dei sistemi. Il progetto sarà reso pubblico sulla piattaforma *GitHub*[1] nel quale saranno presenti lo script del *cryptominer* iniettabile, il sistema semi-automatico o automatico di *injection* tramite *XSS* e se possibile l'implementazione di un metodo di difesa contro attacchi simili a quello sviluppato.

[1] https://github.com/ZappaBoy/hi-jacket

Bibliografia

[1] Amin Kharraz et al. «Outguard: Detecting in-browser covert cryptocurrency mining in the wild». In: *The World Wide Web Conference*. 2019, pp. 840–852.

[2] *Bitcoin Wiki*. https://en.bitcoinwiki.org/wiki/Bitcoin_history.

[3] *Wired - Trasformazione di Bitcoin*. https://www.wired.it/economia/finanza/2019/01/03/bitcoin-2009-trasformazione-storia/.

[4] *Coinlore all Coins*. https://www.coinlore.com/all_coins.

[5] *Skybox Vulnerability Report Trends*. https://lp.skyboxsecurity.com/rs/440-MPQ-510/images/Skybox_Report_Vulnerability_Threat_Trends_2018_Mid-Year_Update.pdf.

[6] Marius Musch et al. «Web-based Cryptojacking in the Wild». In: *arXiv preprint arXiv:1808.09474* (2018).

[7] Muhammad Saad, Aminollah Khormali e Aziz Mohaisen. «End-to-end analysis of in-browser cryptojacking». In: *arXiv preprint arXiv:1809.02152* (2018).

[8] Wenhao Wang et al. «Seismic: Secure in-lined script monitors for interrupting cryptojacks». In: *European Symposium on Research in Computer Security*. Springer. 2018, pp. 122–142.

[9] Radhesh Krishnan Konoth et al. «Minesweeper: An in-depth look into drive-by cryptocurrency mining and its defense». In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 2018, pp. 1714–1730.

[10] Rashid Tahir et al. «Mining on someone else's dime: Mitigating covert mining operations in clouds and enterprises». In: *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer. 2017, pp. 287–310.

[11] Alex Belkin, Nethanel Gelernter e Israel Cidon. «The Risks of WebGL: Analysis, Evaluation and Detection». In: *European Symposium on Research in Computer Security*. Springer. 2019, pp. 545–564.

[12] The Khronos Group Inc. *WebGL*. https://github.com/KhronosGroup/WebGL.

[13] The Khronos Group Inc. *OpenGL*. https://www.opengl.org/.

[14] gpujs. *GPU.js*. https://github.com/gpujs/gpu.js.

[15] Positive Technologies Security. *Penetration testing of corporate information systems: statistics and findings, 2019*. https://www.ptsecurity.com/ww-en/analytics/corp-vulnerabilities-2019/.

[16] Precise Security. *Cross-Site Scripting (XSS) Makes Nearly 40% of All Cyber Attacks in 2019*. https://www.precisesecurity.com/articles/cross-site-scripting-xss-makes-nearly-40-of-all-cyber-attacks-in-2019/.

[17] GPU.js community. *GPU.js benchmark*. https://gpu.rocks/#/benchmark.

[18] W3Techs. *Usage statistics of JavaScript as client-side programming language on websites*. https://w3techs.com/technologies/details/cp-javascript.

[19] OWASP. *Cross Site Scripting (XSS)*. https://owasp.org/www-community/attacks/xss/.

[20] s0md3v. *XSStrike*. https://github.com/s0md3v/XSStrike.

[21] M4cs. *Traxss*. https://github.com/M4cs/traxss.

[22] epsylon. *XSSer*. https://github.com/epsylon/xsser.

[23] xssed. *xssed*. http://www.xssed.com/pagerank.

[24] Monirul I Sharif et al. «Impeding Malware Analysis Using Conditional Code Obfuscation.» In: *NDSS*. 2008.

[25] Prithvi Bisht e VN Venkatakrishnan. «XSS-GUARD: precise dynamic prevention of cross-site scripting attacks». In: *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer. 2008, pp. 23–43.