Uma Ferramenta para Detecção de Mineração de Criptomoedas em Redes de Computadores Institucionais

Arthur Takeshi Noronha Yoshikawa[1], **Roberto Samarone Dos Santos Araújo**[1]

[1] Instituto de Ciências Exatas e Naturais – Universidade Federal do Pará (UFPA)
Belém – PA – Brazil

arthur.yoshikawa@icen.ufpa.br, rsa@ufpa.br

Resumo. *A mineração ilegal de criptomoedas tem gerado o aumento do consumo energético e de recursos computacionais em redes públicas e institucionais. Atualmente, existem diversas aplicações distribuídas na Internet que conectam hosts a servidores de mineração. Diante disso, usuários maliciosos visam maximizar os ganhos financeiros ao procurar ambientes que possuem abundante potencial energético e computacional. Este trabalho apresenta uma ferramenta para identificar e bloquear atividades mineradoras em redes institucionais. O propósito de automatizar este tipo de bloqueio dá-se pelos ataques de mineração terem se tornado populares por conta do aumento do valor de mercado das criptomoedas. Por fim, o desenvolvimento desta ferramenta adiciona uma camada de segurança capaz de identificar ataques de mineração em redes institucionais, mitigando possíveis extravios de recursos computacionais e energéticos destes ambientes.*

Abstract. *Illegal mining of cryptocurrencies has been raising energetic fees and computer resources in public and institutional networks. Nowadays, many distributed applications that bind hosts to mining servers were developed and distributed across the Internet. That said, malicious users aim to maximize their financial gains targeting environments that offer a huge supply of energetic and computational resources. In this paper, it is presented a tool to identify and block mining activities at institutional networks. The main purpose to automate this blocking process has been the rising market price of cryptocurrencies given their abrupt popularity. Finally, the development of this tool inserts a security layer against mining activities at institutional networks and avoids huge computational and energy losses.*

1. Introdução

Desde setembro de 2017, ataques envolvendo o uso não autorizado de CPUs para mineração de criptomoedas foram os mais recorrentes [Segura 2018]. Estes ataques de mineração, também denominados *cryptojacking*, consistem na resolução de extensivos cálculos matemáticos para obtenção de moedas digitais que são convertidas em valor monetário [Pires et al. 2019].

Essa categoria de ataque extravia recursos computacionais de usuários remotos para realizar atividades de mineração, afetando fortemente redes institucionais e corporativas. Consequentemente, tais ataques aumentam o consumo energético e o uso de

recursos computacionais. Em geral, redes desse tipo possuem abundante concentração de energia e processamento [Ingols et al. 2009, Pires et al. 2019, Zimba et al. 2020].

A tecnologia de farejamento de rede ou *sniffing* tem sido empregada para vigilância e segurança de redes. Em razão dos ataques de mineração deixarem rastros de comunicação na rede, considera-se a suscetibilidade do método de farejamento para identificar possíveis atividades mineradoras. Consequentemente, as infraestruturas públicas são vulneráveis diante de tentativas de *cryptojacking*. Apesar de não existirem dados que englobam o total de incidentes nacionalmente, casos isolados assolam várias instituições [Pires et al. 2019].

Este trabalho apresenta o desenvolvimento de uma ferramenta capaz de detectar minerações de criptomoedas em redes institucionais por meio de técnicas de inspeção de rede e verificações textuais.

Este trabalho está organizado da seguinte forma: A seção 2 apresenta os trabalhos relacionados ao tema. A Seção 3 apresenta os fundamentos teóricos utilizados para obter as evidências de atividades mineradoras. A Seção 4 apresenta o *overview* e dependências do projeto. A Seção 5 apresenta a ferramenta desenvolvida. Os resultados obtidos são apresentados na Seção 6. Por fim, a conclusão está presente na Seção 7.

2. Trabalhos relacionados

O trabalho de Pires *et al.* monitorou máquinas comprometidas na rede local da Universidade Federal do Rio de Janeiro (UFRJ) [Pires et al. 2019]. Foi desenvolvido um programa de notificação que analisa os dados trafegados obtidos durante as auditorias físicas realizadas pelo time de segurança de rede da universidade, e os agrega para obter métricas e possíveis padrões de comportamento de ataques de mineração. A ferramenta apresentada neste artigo identifica os ataques de mineração por meio de informações presentes nos pacotes de rede durante o ato da mineração. Em contraste com o trabalho desenvolvido na UFRJ, as informações de mineração não são obtidas por meio de um processo de auditoria. O processo de coleta de dados e bloqueio é realizado internamente na ferramenta.

O *MineGuard* [Tahir et al. 2017] é uma ferramenta que detecta processos de mineração em máquinas virtuais. A ferramenta utiliza registradores situados em processadores modernos denominados contadores de desempenho de hardware (*Hardware Performance Counters* - HPCs) para rastrear operações de mineração em baixo nível em CPU e GPU com o mínimo de sobrecarga, oferecendo alta eficiência e precisão na detecção em tempo real de mineração de criptomoedas. O *MineGuard* sustenta na ideia que o atacante precisa executar repetidamente o algoritmo de *Proof-of-Work* (PoW) para minerar qualquer criptomoeda. A ferramenta apresentada neste artigo detecta a mineração com informações presentes nos pacotes de rede farejados em tempo real, o conteúdo dos pacotes são de alto nível por apresentarem informações próximos da linguagem humana. Consequentemente, não houve a realização de inspeções de *hardware*. Em razão da ferramenta não utilizar inspeções à nível de *hardware*, há relativa perda de performance e ganho em relação a facilidade de implementação e abstração de informações.

O *OutGuard* é um sistema de código-aberto (*open-source*) para detectar tentativas de *cryptojacking* usando classificação por aprendizado de máquina [Kharraz et al. 2019]. A base de dados do sistema foi obtida ao escanear os domínios do *Alexa Top Sites* e

coletado os recursos e traços de processamento em JavaScript. Ao submeter os dados a ferramentas de detecção de mineração, foi obtido aproximadamente 3006 vestígios de processos de mineração. A ferramenta apresentada neste artigo não aplicou treinamentos com base de dados, para manter a baixa complexidade de processamento do *software*. O procedimento de identificação é iniciado pela extração de informações de rede, e o bloqueio por consulta externa de um serviço *web* (remoto).

Existe um amplo acervo de trabalhos que relatam sobre a existência de ataques de *cryptojacking* [Kharraz et al. 2019, Zimba et al. 2020]. Os atacantes sequestram os recursos de uma máquina por meio de páginas *web* utilizando artifícios de engenharia social, no qual direcionam o usuário a executar um *script* de mineração que consome recursos de processamento da máquina sem autorização do proprietário. A ferramenta apresentada neste artigo considerou que o usuário malicioso detém do acesso físico às máquinas para iniciar o *cryptojacking*, pois não foi considerado nenhuma tentativa de ataque remoto dentro do período de desenvolvimento.

3. Fundamentação teórica

A seguir são apresentadas as tecnologias utilizadas nesse trabalho.

3.1. *Blockchain*

A arquitetura *blockchain* consiste em uma lista encadeada de blocos de transações conectados criptograficamente [Narayanan et al. 2016], onde cada bloco identificado por uma hash, aponta para o bloco anterior. Cada bloco possui além da *hash* do bloco anterior, os tempos e informações das transações. O único bloco que não faz referência ao anterior é a primeira unidade denominada Bloco Gênese (*Genesis Block*), cuja referência é nula por ser o primeiro bloco da estrutura. O esquema é ilustrado na Figura 1.

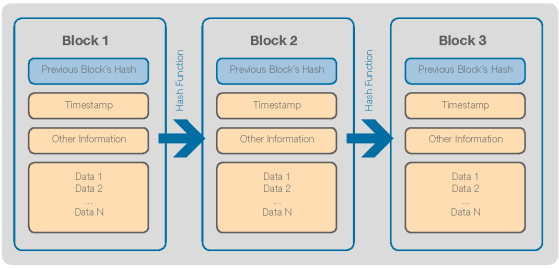

Figura 1. Esquemático estrutural de uma arquitetura de *blockchain*

Cada bloco na *blockchain* contém um conjunto de transações, que são operações ou movimentações de criptomoedas entre usuários que são registradas dentro de um bloco. Uma analogia a isso seriam as movimentações bancárias de depósito ou saque que alteram as quantias armazenadas do saldo de cada cliente. O esquema de transações difere que cada transação é assinada digitalmente para indicar o consentimento do usuário sobre as transações efetuadas. Na Figura 2, Alice recebe dinheiro de Joe e consome um serviço prestado por Bob, a saída final de cada transação é utilizada como entrada à nova operação. Alice possui a chave que provê a assinatura que acessa registros de transações antigas, o que prova que as operações são de sua autoria.

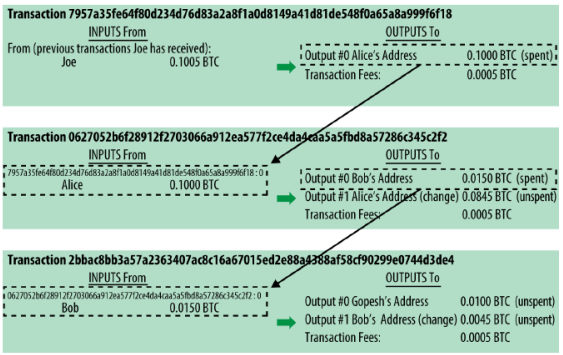

Figura 2. Corrente de transações, cujas saídas se tornam entradas de transações posteriores. [Bashir 2020]

As transações da *blockchain* precisam ser validadas pelos mineradores, dado que um novo conjunto de transações não significa necessariamente que este já pertence a *blockchain*. As movimentações são agrupadas em bloco e os mineradores da rede cedem o poder computacional para solucionar um problema matemático que representa a validação das transações realizadas, este processo provê segurança ao descartar operações inválidas.

Quando um minerador valida todas as transações de um bloco, outros mineradores da rede verificam a solução enviada, processo denominado regras de consenso. Caso a solução esteja correta, o minerador é recompensado com uma quantia em criptomoeda (e.g: *Bitcoin*) e o bloco calculado é inserido com sucesso na *blockchain*.

Cada bloco é adicionado a corrente que inicia desde o *genesis block* até a transação mais recente, caso um bloco não seja validado pelos mineradores então este é descartado. Por fim, a estrutura da *blockchain* é a base fundamental para implementar a lógica do livro-razão público.

3.2. Função Hash criptográfica

Uma função hash criptográfica é um algoritmo matemático que mapeia um dado de entrada de tamanho variável e retorna uma sequência de tamanho fixo como ilustrado na Figura 3. A função é descrita como $h = H(M)$, onde $H(M)$ representa a função hash H dado uma informação M de entrada e h sendo a sequência final gerada para M [Stallings 2013].

Figura 3. Entrada de um arquivo genérico com a sequência de saída processada pela função hash

No contexto da segurança computacional, a responsabilidade principal das funções hash criptográficas é conferir a integridade dos dados [Stallings 2013], dado que qualquer modificação (e.g. um *bit*) resulta em uma sequência de saída totalmente diferente, sendo conhecido como efeito avalanche. A função hash também foi projetada

para ser computacionalmente inviável, de forma que um atacante não consiga regenerar a informação de entrada de posse apenas do valor do hash ou encontrar uma entrada que gere a mesma saída.

A propriedade de resistência a colisão fraca em hash é essencial pois define a dificuldade que um atacante em posse de uma chave aleatória k encontrar um valor $x \mid x \neq y$ resultar em $H_k(x) = H_k(y)$ [Mironov et al. 2005]. Isto é, dado uma hash gerada a partir de uma informação de entrada, a probabilidade de existir outra entrada que resulte no mesmo valor de hash deve ser extremamente baixo.

Dentre outras propriedades de uma função de hash, tem-se a eficiência em calcular a hash para qualquer informação de entrada dentro de implementações de *software* ou *hardware*. E, a resistência a colisão forte que deve ser computacionalmente impossível encontrar um par(x, y) resultando em $H(x) = H(y)$, isto é, duas entradas distintas não podem gerar o mesmo hash [Stallings 2013].

Por fim, existem diferentes propriedades de segurança a serem atendidas dependendo do tipo de serviço a ser oferecido [Stallings 2013]. No caso da *blockchain*, cada bloco integrante da arquitetura está assinado juntamente com um valor de hash. Logo é mandatório que a mensagem original não possa ser reproduzida (mão única), não existir outras transações que gerem a mesma sequência (colisão fraca) e que seja impossível um terceiro indivíduo gerar uma informação para outra entidade assinar (colisão forte).

3.3. Criptomoedas

Uma criptomoeda é um recurso digital criado para funcionar como um meio de troca que registra as transações do usuário em um livro-razão/banco de dados, protegendo-as usando criptografia. O processo de criação de uma criptomoeda na rede é chamado de mineração, cujo processo é definido quando uma máquina cede o poder de processamento e energia para resolver problemas matemáticos e ser recompensada por essa resposta [Nakamoto 2009].

As criptomoedas não existem fisicamente nem são emitidas por uma organização central. Contrário a isso, esses recursos são gerenciados usando o controle distribuído ou descentralizado. Ele é definido como uma tecnologia que cada usuário da rede possui uma cópia do livro-razão de transações, permitindo operações financeiras confiáveis entre um ou mais pessoas, sem a presença de uma entidade intermediária no processo. Este esquema dificulta o esquema de fraude em qualquer transação na rede e garante confiabilidade em todo processo de validação das criptomoedas.

O processo de mineração pode ser classificado em dois tipos: transações individuais e de bloco. Na individual ou "*solo*", a recompensa gerada pelo processo de mineração depende unicamente do poder de processamento do computador. Já no processo em bloco comumente chamado "*pool*", vários computadores mineram juntos um mesmo bloco de transação e o processamento colaborativo aumenta a velocidade da resolução do bloco. Ao resolver todas as transações do bloco, a recompensa é dividida proporcionalmente ao processamento despendido por cada minerador envolvido.

3.4. Criptomoeda Monero

Lançada em 2014 como um *fork* da *Bytecoin*, atingiu valores de mercado de 5.7 bilhões de dólares em 2018. O Monero é uma criptomoeda de código-aberto descentralizada com

foco na privacidade e anonimato dos usuários [Li et al. 2021]. Como outras criptomoedas, o Monero pode ser usado como valor monetário para aquisição de bens e serviços. Todavia as transações são totalmente anonimizadas sendo de grande valor aos usuários que não desejam ter suas compras expostas.

Para prover anonimato e privacidade, o Monero é sustentado por dois conceitos: *Stealth address* e *Ring Signatures*. Em relação ao primeiro, todo remetente gera somente um único endereço público por transação. Todo usuário do Monero possui um par de chaves para visualização e transações (autorizar pagamentos).

O *Ring Signatures* é um conceito rebuscado da criptografia geral. Ele se refere à capacidade de uma carteira XMR armazenar um conjunto de chaves. Isso impossibilita qualquer usuário da rede saber qual a chave utilizada para assinatura, provendo anonimato. Assim como a moeda digital *bitcoin*, uma assinatura digital pode ser assinada por qualquer membro de um grupo que possua uma chave privada.

3.5. Farejamento de pacotes

O farejamento de pacotes ou *packet sniffing* consiste na técnica de capturar pacotes que trafegam em uma rede de computadores. As ferramentas de farejamento podem ser usadas para fins administrativos ou maliciosos, dependendo da índole do usuário [Ansari et al. 2003]. Existem ferramentas proprietárias ou *open-source* que oferecem o serviço de farejamento. Alguns exemplos notórios incluem o *Wireshark*, *tcpdump* e *CommView*.

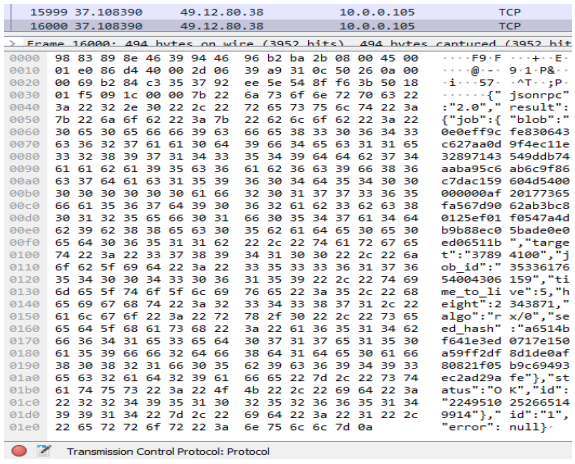

Figura 4. Pacote capturado com informações em texto claro.

As aplicações de farejamento permitem ao usuário acessar os pacotes provenientes da camada de redes do modelo TCP/IP. Essa técnica é comumente usada para fins de segurança ou performance da rede, um exemplo do funcionamento é apresentado na Figura 4. Utilizando o *Wireshark* durante a mineração, observa-se que alguns pacotes com informações de autenticação ou hash de bloco são exibidos.

3.6. Expressões Regulares

Desenvolvido sobre a influência de expressões regulares teóricas, a abreviação de expressões regulares é chamada de regex. Câmpeanu *et al.* define que o regex consegue

expressar matematicamente um grande número de subconjuntos de linguagens dentro de um contexto linguístico [CÂMPEANU et al. 2003]. Por exemplo, uma subsequência de caracteres definida como "aaabaaa" pode ser expressa como $L_1 = \{a^n b a^n \mid n \geq 0\}$.

A aplicação do regex foi importada no contexto de diversas linguagens de programação: *Python*, *Perl*, etc. A empregabilidade de uma expressão regular é capaz de fornecer ao desenvolvedor a busca de uma sequência específica, tal como as cadeias de *hash* criptográfica dos blocos de uma *blockchain*.

3.7. Ataque de Mineração (*Cryptojacking*)

O *cryptojacking* é um ataque em que um usuário malicioso infecta uma máquina hospedeira, de forma que os recursos computacionais do dispositivo são utilizados para executar atividades de mineração de criptomoedas [Pires et al. 2019]. Uma máquina atacada apresenta: lentidão de processamento; falhas de sistema; e uso excessivo de recursos computacionais, o que leva a redução da vida útil do dispositivo atacado.

Este tipo de ataque é difícil de ser detectado, tem em vista que os principais indícios não comprovam que a origem seja proveniente de um *cryptojacking*. Existem possibilidades do computador estar infectado por um *malware* ou executando programas de alta performance. Computacionalmente, a dificuldade de detectar tais ataques é extremamente maior, já que o *cryptojacking* parte de comunicações na rede até os componentes da máquina. Isso torna extremamente complexo o processo de investigação de pistas de mineração pelo grande número de etapas envolvidas.

Adicionalmente, outro fator de dificuldade em conter essa prática é a grande disponibilidade de serviços de mineração existentes na Internet. Por exemplo, *MinerGate*; *CoinHiver*; *EasyMiner*, etc. Existem usuários que mineram utilizando dispositivos pessoais, denominado *solo mining*. Apesar de não ser categorizado como um ataque, isso onera uma rede pública caso seja executado. Por fim, existem diversos métodos de transformar um dispositivo em cliente de mineração. Por exemplo a injeção de *scripts* na Internet utilizando engenharia social ou execuções manuais de mineradores em ambientes públicos.

4. Proposta da Ferramenta

A mineração de criptomoedas utiliza recursos computacionais da máquina do usuário para processar as atividades enviadas pelo servidor de mineração. A seguir, será descrita a metodologia do desenvolvimento da ferramenta para bloquear a atividade mineradora.

4.1. *Overview* da Ferramenta

Redes institucionais possuem mecanismos para monitorar e defender a rede. Todavia os ataques de mineração são difíceis de serem detectados. A causa disso é a dificuldade de se encontrar evidências conclusivas quando uma máquina hospedeira foi atacada. Por conta da inviabilidade de inspecionar manualmente um ambiente com vários dispositivos, o local ideal no ambiente da rede para executar a ferramenta seria em um servidor, como um *firewall* de borda da rede. Ao agir como intermediário entre as máquinas e a rede, o *firewall* verifica os pacotes passantes e os bloqueia. Isso ocorre caso o pacote verificado estabeleça comunicação com um servidor de mineração.

4.2. Limitações da Ferramenta

A ferramenta foi desenvolvida intensamente sobre o contexto da criptomoeda Monero. É necessário que os computadores estejam interligados fisicamente a rede institucional para captura de pacotes. Ou seja, tentativas de mineração com dispositivos pessoais ou conectados a rede sem fio (*wireless*) não foram consideradas. Ademais, tentativas de ataques remotos ou utilizando a tecnologia *Virtual private network* (VPN) também foram desconsideradas.

4.3. Requisitos da Ferramenta

A ferramenta desenvolvida deve ser amigável ao usuário final. Dado este contexto, foi considerado que:

1. A instalação da ferramenta deve residir em uma entidade central (servidor);
2. A ferramenta deve inspecionar interruptamente a rede;
3. Ao detectar um ataque, a ferramenta deve exibir um aviso desse evento;
4. Exibir o remetente do pacote juntamente ao aviso do ataque;
5. De posse do endereço remetente do pacote de mineração, enviar o endereço ao *firewall* do sistema para bloqueio imediato.

4.4. Esquema de rede e a ferramenta

Na Figura 5 é ilustrado um exemplo da rede onde a ferramenta deve ser instalada. Há diversos computadores conectados em uma rede intermediada por um servidor central. Então, a ferramenta de inspeção e bloqueio é instalada neste servidor. Supondo que neste ambiente há pelo menos um dispositivo se comunicando com um servidor de mineração. O pacote de rede estará enviando informações de transações e possivelmente referenciando a hash do bloco da transação pendente. Logo, o servidor intercepta os pacotes de rede suspeitos e efetua o bloqueio da comunicação entre a máquina sequestrada e o servidor de mineração.

Este esquema inicial pode não apresentar a performance máxima, visto que o servidor acessa somente informações de uma subrede do ambiente instuticional. É possível ampliar este alcance ao adicionar *switches* de rede para espelhar as portas de acesso da rede institucional.

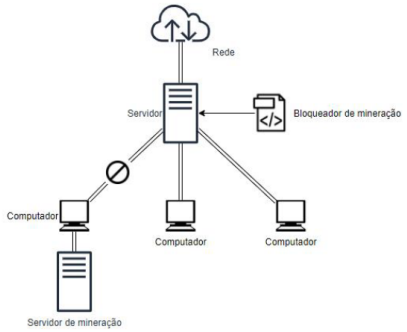

Figura 5. Esquemático do sistema de bloqueio

Baseado nos requisitos supracitados, a execução da ferramenta fecha um ciclo, que representa a execução ininterrupta do processo de inspeção da rede, ilustrado na Figura 6. Inicializando o programa, primeiramente é verificado se o pacote possui dados

adicionais além dos cabeçalhos, denominado *payload*. Com o *payload*, em seguida é extraído dentro do conteúdo do pacote a sequência hash do bloco enviado pelo servidor ao cliente, utilizando a busca com expressões regulares mencionada anteriormente. A sequência extraída alimenta a entrada da API do serviço localmonero, que serve como um explorador de blocos de transações, para averiguar se o hash pertence a *blockchain* Monero. Caso positivo, o endereço do servidor será apresentado ao usuário e o eventual bloqueio deste endereço.

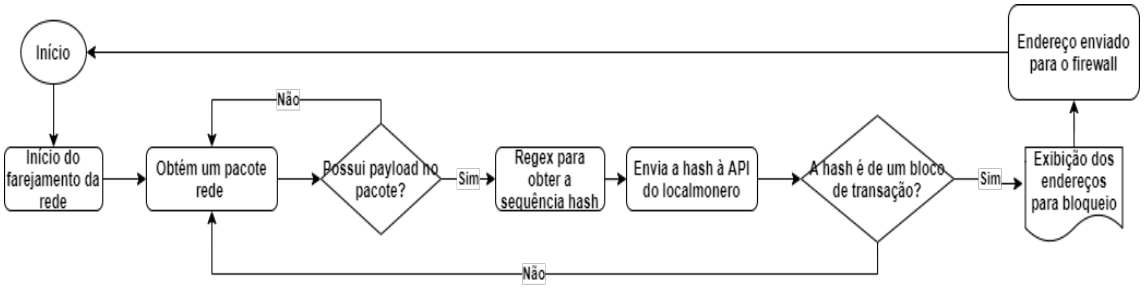

Figura 6. Fluxograma da ferramenta

4.5. Componentes de *Software*

Para desenvolver a ferramenta proposta, várias ferramentas e programas foram utilizados para construir o fluxo de processo da ferramenta supracitado na Figura 6. A seguir são apresentados os componentes utilizados na ferramenta.

4.5.1. Scapy

O *Scapy* é uma ferramenta de código aberto disponível para diversos sistemas operacionais, Linux, Windows, entre outros. Ele tem como foco a manipulação e o farejamento de pacotes de rede e está disponível na linguagem de programação *Python* [Scapy 2021]. A biblioteca concede acesso aos pacotes de rede trafegados pelas interfaces de rede. Com isso, o usuário é capaz de observar o fluxo de pacotes provenientes das camadas de apresentação, rede, enlace, dentre outras [Kurose e Ross 2012]. Além disso, é possível decodificá-los para fins de análise e investigação.

4.5.2. Re (*Regular Expression)*

Esta biblioteca é utilizada para construir expressões regulares como explicado na Subseção 3.6. Uma expressão regular busca um conjunto de strings pareadas às regras da expressão definida pelo programador. De forma breve, existindo um conjunto alfabético "ABCCDEF", ao definir com a biblioteca re pela expressão "/CCD", a saída proporcionada será a substring CCD [re 2021].

4.5.3. Pytest

O pytest é um *framework* de código aberto para escrever testes automatizados com objetivo de validar a saída de funções de programas e aplicações em Python [Pytest 2021]. Ele

possui diversos recursos como: informações detalhadas dos módulos de teste; escrita de um conjunto de testes para uma única função (parametrização); e, suporte para *plugins*.

4.5.4. MinerGate

O MinerGate é uma aplicação bastante conhecida para mineração criptomoedas como *bitcoin*, Monero, entre outras [MinerGate 2021]. Oferece facilidade em converter as máquinas de usuários em clientes de mineração. O programa minera na modalidade de "*pools*" de mineração, isto é, diversos clientes mineram coletivamente os blocos de transações e são recompensados proporcionalmente aos recursos dispendidos na mineração.

5. Desenvolvimento da Ferramenta

Nesta seção será apresentado o processo de desenvolvimento da ferramenta.

5.1. Processo de desenvolvimento

Existem quatro ambientes que participam no processo da ferramenta, como ilustrado no diagrama de sequência da Figura 7. Ele engloba quatro participantes durante todo funcionamento. O servidor onde a ferramenta reside inicializa a função inicial de bloqueio que executa ininterruptamente chamadas de farejamento na rede local. Ele retorna os pacotes capturados dentro de intervalos de 1 a 2 segundos.

De posse da lista de pacotes obtidos com o farejamento, é extraído as hashes presentes nos pacotes de dados de mineração. Estes são enviados à API remota para verificação da hash de entrada. Por fim, caso o pacote seja originário de um processo de mineração, a ferramenta extrai o endereço remetente. Isso quer dizer que é o servidor de mineração que enviou o pacote, e o endereço é enviado ao *firewall* para bloqueio imediato e o ciclo de funcionamento é repetido novamente.

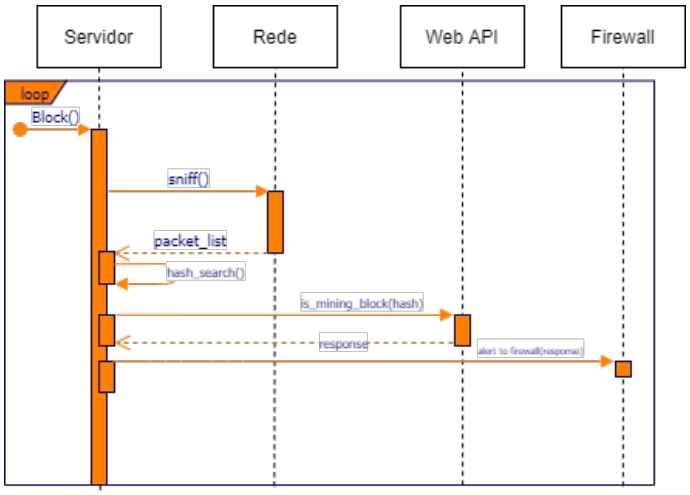

Figura 7. Diagrama de sequência do funcionamento da ferramenta

Em seguida, utilizando o *Scapy* para iniciar o farejamento da interface de rede, a instância do objeto de farejamento recebe o nome da interface de rede a ser inspecionada. Isso ocorre junto com um parâmetro de filtragem, declarado como *filter="tcp"*

para receber somente pacotes TCP. Em razão dos requisitos de um minerador, os pacotes precisam ter informações íntegras e serem entregues em ordem para evitar incongruências no processo de validação de transações.

Os pacotes obtidos possuem 4 campos: Ethernet, IP, TCP ou UDP e Raw, onde cada seção faz referência às camadas do modelo TCP/IP [Kurose e Ross 2012]. As informações de interesse estão situadas na camada de transporte e aplicação por fornecer o tipo de protocolo e dados visíveis. Todos os pacotes possuem endereços de remetente e destino que se traduzem no servidor de mineração e a máquina mineradora, e a presença de informações sobre hashes dos blocos ou *tasks*. Essas informações são enviadas pelo servidor que não sofrem ação de algoritmos criptográficos para reduzir sobrecargas de processamento, por parte do servidor de mineração.

Após a captura dos pacotes, a ferramenta verifica se um pacote possui um *payload* usando o método haslayer(Raw). Caso negativo, o pacote é descartado para minimizar o tempo de processamento da ferramenta. Em seguida, o *payload* é convertido em *string* usando o método linehexdump e codificando a informação para ASCII.

De posse do *payload* extraído anteriormente (ver Figura 8), foi aplicado o uso de regex para encontrar um hash SHA-256. Este é constituído de uma sequência de 64 caracteres hexadecimais que equivale a 256 bits dentro da estrutura JSON [Json 2021]. Ele está presente dentro dos pacotes farejados pelo Scapy.

Dado que uma hash é escrita na base hexadecimal [Stallings 2013], o formato buscado é uma cadeia completa de 64 caracteres de 0 a F. Com o módulo de expressões regulares esta definição é descrita como [A-Fa-f0-9]{64}, que equivale a uma cadeia com exatos 64 caracteres limitados em um conjunto de 0 a F (hexadecimal), isto é, o formato SHA-256. Dentre os cabeçalhos presentes no JSON, a hash do bloco pode ser extraída para investigar se existe alguma comunicação com um servidor de mineração.

Figura 8. Extração de *hash* SHA-02 usando uma expressão regular

Em seguida, as sequências hash encontradas são enviadas a um domínio *web* usando a biblioteca request do Python3 para requisitar comunicações HTTP. Ao conectar com a API do localmonero.co é possível descobrir se a hash encontrada é integrante de

uma *blockchain* da moeda Monero. Caso a sequência se trate de uma hash de um bloco de criptomoeda, então é extraído os endereços remetente e destinatário do pacote cuja hash foi encontrada.


```
{
  "status": "OK",
  "block_data": {
    "id": "0",
    "jsonrpc": "2.0",
    "result": {
      "block_header": {
        "depth": 234,
        "difficulty": 3658048014,
        "hash": "247575ca01657e2f61845dc2b6a64424c7e45952355757b950220b8def3fe5a0",
        "height": 1156509,
        "major_version": 3,
        "minor_version": 5,
        "nonce": 33,
        "orphan_status": false,
        "prev_hash": "692f714ee262e5ed9fd4c26e5ef80f85db8577f56890027085291de565b8c388",
        "reward": 10244800000000,
        "timestamp": 1476355796
      },
      "status": "OK",
      "tx_hashes": [
        "dbb1ddacd2ae0137752f0e761adcab64d463a0f74d1f506f49cd92a11b336bf1",
        "956e160424eb42f1780c04b1363fc7aa84553ff788f774da697201b9868253f9",
        "7da20edb196b5e10d5ee67dcfbcaa9f8fff7482a0647ebff231fc9ff46e1289b",
        "de2cfe5e140cb839f2fd4f63a4daed9960b96db013ed1e9a5c544914c64a0b8a",
        "2cb75dc5a36cc5d05efd38da3e08135b4340cc18a824d376ce07924d898bb9e4"
      ]
    }
  }
}
```


Figura 9. Consulta de um bloco válido na API.

Primeiramente, a requisição de consulta é iniciada ao acessar o endereço https://localmonero.co/blocks/api/get_block_data/{hash}, inserindo a hash extraída no campo {hash}. A API retorna uma estrutura JSON contendo um dado de estado "OK", ilustrado na Figura 9, indicando que a hash de entrada faz referência a um bloco de transação da criptomoeda. O localmonero armazena tanto blocos já validados por mineradores (antigos) quanto os transitivos que possuem transações pendentes para validação. O tipo analisado na ferramenta é o segundo, posto que as transações mais recentes ainda não são validadas. Os blocos de transição são verificáveis pois o cabeçalho do bloco não se altera, pois depende das informações relacionadas ao bloco anterior. Logo a unidade é catalogada no banco de dados da API, porém ainda não se trata de um bloco integrado à *blockchain*.

Após a verificação da hash, os endereços são enviados ao *firewall* nativo do sistema operacional Linux chamado iptables. Todos os endereços emissores das comunicações de mineração são inseridos dentro da tabela de bloqueio. Logo, qualquer tentativa de comunicação futura do servidor de mineração será recusada ao tentar estabelecer comunicação com o cliente.

```
def send_alert_to_firewall(ip):
    print("Blocking address", ip)
    run(['iptables', '-I', 'INPUT', '1', '-s', ip, '-j', 'DROP'])
```

Figura 10. Função de envio do endereço ao iptables

Com base no código ilustrado na Figura 10, a função run invoca comandos diretamente do sistema operacional, ao chamar pelo *firewall* com o argumento DROP, significa

que o endereço de entrada foi definido como regra de bloqueio na tabela. O indicador INPUT define o índice do bloqueio, por exemplo, se o endereço “162.190.0.1” for bloqueado o número que faz referência a esta regra de bloqueio é definido como 1. Este identificador é útil caso um bloqueio tenha sido erroneamente declarado e precise ser removido.

6. Testes e Resultados

Inicialmente, para conferir a corretude das funcionalidades criadas, a ferramenta utiliza o módulo de teste do Python3 denominado Pytest-3. Aqui considera-se duas hashes: uma SHA-256 de um arquivo comum e outra proveniente, de fato, de um bloco existente no ecossistema do Monero. A API adotada se comunica com a base do Monero.

As *strings* de entrada são enviadas ao objeto de requisição HTTP e a saída é o valor booleano indicando se a cadeia de caracteres está associada a uma atividade mineradora. A função is_mining_block é o módulo que envia a hash como entrada para requisição na API. O primeiro módulo de teste envia um hash não pertence de um bloco e o retorna como false, enquanto o segundo teste envia uma entrada válida, retornando True. Logo, caso qualquer mudança futura seja realizada na função de verificação da hash é fácil executar o lote de testes para verificar se a função ainda está retornando as saídas corretas.

```
def test_transaction_hash():
    assert(is_mining_block("44395a52d6722815be370fd739a743c8f7ec6f6f2fcd93083e220f9d2cf47e91") == False)

def test_with_a_mining_block_hash():
    assert(is_mining_block("1512a9cfba3cca2df4958f70af4dc87a5695a96d8f6683b9715023cce1f6dbee") == True)
```

Figura 11. Testes de verificação de hash

O segundo lote de testes determina a corretude da ferramenta em extrair hashes no formato SHA-256. Seguindo a Tabela 1, foram considerados 3 amostras de entrada: um *payload* completo; um blob, uma informação binária com 608 bits (152 em hexadecimal); e uma sequência hash incompleta de 30 caracteres.

No primeiro cenário, o *payload* possuía um blob e a hash do bloco, dado a definição de uma SHA-256, o blob foi descartado por possuir um comprimento de 152 caracteres apesar de seguir a variância de 0-F, o resultado final foi positivo (PASS). O segundo teste foi alimentado somente com uma sequência blob. Isso resultou em um retorno vazio, em razão que não foi encontrado uma informação congruente ao formato buscado pelo regex, resultando na aprovação do caso de teste, indicado como PASS. E o terceiro teste, a hash possui um comprimento irregular de 30 caracteres, novamente fora do padrão de 256 bits, resultando novamente em um retorno vazio com resultado PASS. Desta forma, é possível executar este lote de testes para qualquer usuário verificar se a função de extração de hash está funcionando corretamente.

Amostra	Esperado	Resultado
Payload Completo	”a6514bf641e3ed0717e150a59ff2df8d1de0af80821f05b9c69493ec2ad29afe”	PASS
Blob	vazio	PASS
Hash incompleta	vazio	PASS

Tabela 1. Testes de extração de hash

Para execução dos testes foi utilizado um computador equipado com um processador Intel Core i5-5200U a 2.20GHz com quatro núcleos e 8GB de memória RAM. A rede doméstica utilizada não conta com nenhum sistema prévio de bloqueio de domínios que pudesse coagir na execução da ferramenta. Utilizou-se a aplicação multiplataforma *MinerGate* para minerar a moeda Monero (XMR).

Como Ilustrado na Figura 12, o computador executa o *MinerGate* com a capacidade máxima de quatro núcleos de processamento. A cada fração de segundos é enviado um desafio ao cliente que consiste em validar o hash da transação do bloco com o informe "Share Accepted". Ao executar o código da ferramenta, o processo de inspeção ocorre em *background* em tempo contínuo. No momento de detecção de uma hash de bloco de mineração são apresentados os endereços do remetente e do destinatário do pacote com o desafio lançado. Devido ao envio contínuo de desafios ao minerador e ao intervalo de atraso (*delay*) existente na ferramenta, o resultado é a duplicação de detecções. A mensagem "QPainter::end" significa que as comunicações entre cliente-servidor foram interrompidas.

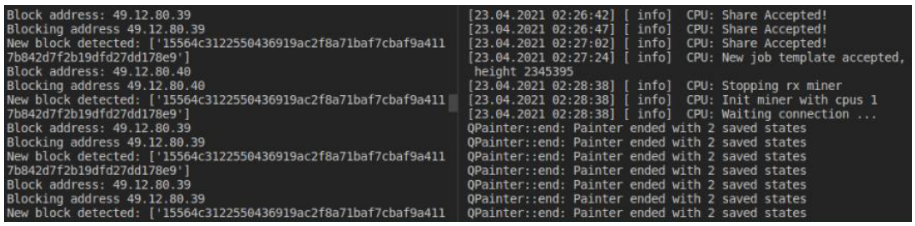

Figura 12. Detecção da hash do bloco e interrupção da comunicação

Após interromper com sucesso as comunicações entre cliente e servidor, é executado o comando para inspecionar a tabela de bloqueios do iptables apresentado na Figura 13. Cada entrada da tabela rotulada como DROP indica que se trata de uma regra de bloqueio e a coluna *source* indica o endereço cuja regra aplicada é de bloqueio ou permissão. Neste caso deseja-se somente bloquear comunicações. Os registros duplicados não são tratados pelo iptables, apesar de não oferecerem eventuais problemas. Todavia foi um erro não tratado devido à falta de testes adicionais na ferramenta.

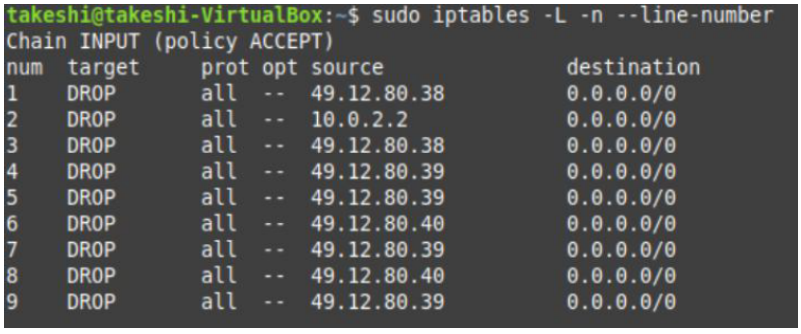

Figura 13. Tabela de endereços bloqueados do iptables

7. Considerações Finais e Trabalhos Futuros

Ataques de mineração podem ser realizados em máquinas públicas. Eles convertem o poder de processamento destes dispositivos em recursos financeiros a carteira do usuário malicioso. Contudo, esses ataques deflagram grandes problemas relativos à reputação das criptomoedas. Isto é, como a obtenção de criptomoedas é resultado de atividades de mineração, que estão recentemente atreladas ao sequestro de recursos em máquinas públicas, isso pode resultar em possíveis intervenções governamentais ou banimentos das moedas em alguns países.

Neste trabalho foi apresentado uma ferramenta para bloquear ataques de mineração. Ela objetiva conter demasiados gastos computacionais que podem comprometer a vida útil do *hardware* das máquinas atacadas, ou provocar gastos energéticos indesejados para organizações públicas. O trabalho também apresentou os resultados de como a ferramenta consegue em certa escala intervir uma atividade mineradora automaticamente em tempo real, sem a presença de fiscalização manual.

Este trabalho pode ser melhorado futuramente em termos de performance com o acréscimo de rastros de mineração por *hardware* (CPU e GPU) e testes em ambientes reais. A popularidade e facilidade de mineração da criptomoeda Monero foram as principais razões de testá-la inicialmente com a ferramenta. Consequentemente, precisam ser realizados mais testes na ferramenta para garantir e estender o bloqueio a outras criptomoedas. Adicionalmente, é necessário avaliar a ferramenta quanto ao consumo energético para verificar a viabilidade de empregá-la em um ambiente institucional de larga escala.

Referências

Ansari, S., Rajeev, S., e Chandrashekar, H. (2003). Packet sniffing: a brief introduction. *IEEE Potentials*, 21(5):17–19.

Bashir, I. (2020). Mastering blockchain 3rd edition.

CÂMPEANU, C., SALOMAA, K., e YU, S. (2003). A formal study of practical regular expressions. *International Journal of Foundations of Computer Science*, 14(06):1007–1018.

Ingols, K., Chu, M., Lippmann, R., Webster, S., e Boyer, S. (2009). Modeling modern network attacks and countermeasures using attack graphs. Em *2009 Annual Computer Security Applications Conference*, páginas 117–126.

Json (2021). `https://www.json.org/json-en.html`.

Kharraz, A., Ma, Z., Murley, P., Lever, C., Mason, J., Miller, A., Borisov, N., Antonakakis, M., e Bailey, M. (2019). Outguard: Detecting in-browser covert cryptocurrency mining in the wild. páginas 840–852.

Kurose, J. F. e Ross, K. W. (2012). *Computer Networking: A Top-Down Approach (6th Edition)*. Pearson, 6th edition.

Li, Y., Yang, G., Susilo, W., Yu, Y., Au, M. H., e Liu, D. (2021). Traceable monero: Anonymous cryptocurrency with enhanced accountability. *IEEE Transactions on Dependable and Secure Computing*, 18(2):679–691.

MinerGate (2021). `https://minergate.com/`.

Mironov, I. et al. (2005). Hash functions: Theory, attacks, and applications. *Microsoft Research, Silicon Valley Campus. Noviembre de*.

Nakamoto, S. (2009). Bitcoin: A peer-to-peer electronic cash system. *Cryptography Mailing list at https://metzdowd.com*.

Narayanan, A., Bonneau, J., Felten, E., Miller, A., e Goldfeder, S. (2016). *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton University Press, USA.

Pires, V., Coutinho, F., Menasché, D., e de Farias, C. (2019). Gatos virtuais: detectando e avaliando os impactos da mineração de criptomoedas em infraestrutura pública. Em *Anais do XIX Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*, páginas 57–70. SBC.

Pytest (2021). `https://docs.pytest.org/en/6.2.x/`.

re (2021). `https://docs.python.org/3/library/re.html`.

Scapy (2021). `https://scapy.net/`.

Segura, J. (2018). The state of malicious cryptomining. *Malware Bytes*.

Stallings, W. (2013). *Cryptography and Network Security: Principles and Practice*. Prentice Hall Press, USA, 6th edition.

Tahir, R., Huzaifa, M., Das, A., Ahmad, M., Gunter, C., Zaffar, F., Caesar, M., e Borisov, N. (2017). Mining on someone else's dime: Mitigating covert mining operations in clouds and enterprises. páginas 287–310.

Zimba, A., Wang, Z., Mulenga, M., e Odongo, N. H. (2020). Crypto mining attacks in information systems: An emerging threat to cyber security. *Journal of Computer Information Systems*, 60(4):297–308.