CryingJackpot: Fluxos de Rede e Indicadores de Desempenho contra o Cryptojacking

Gilberto Gomes[1,2] Luis Dias[1,2] Miguel Correia[2]
[1]CINAMIL, Academia Militar, Instituto Universitário Militar – Portugal
[2]INESC-ID, Instituto Superior Técnico, Universidade de Lisboa – Portugal
gomes.g@exercito.pt dias.lfxcm@exercito.pt miguel.p.correia@tecnico.ulisboa.pt

***Resumo*—*Cryptojacking* traduz a apropriação de recursos computacionais alheios sem conhecimento ou consentimento das vítimas para obter moedas criptográficas, um ataque generalizado relativamente fácil de implementar e difícil de detectar. Seja com recurso a um navegador Web ou através de um executável, o *cryptojacking* carece de soluções de detecção robustas e fiáveis.**

O presente artigo descreve o trabalho desenvolvido na prossecução de uma abordagem híbrida para a detecção de *cryptojacking*, onde não é necessário conhecimento prévio sobre os ataques ou dados de treino. A nossa abordagem de deteção, CRYINGJACKPOT, extrai e combina características/atributos baseadas em fluxo de rede e em indicadores de desempenho, agregando utilizadores com comportamentos semelhantes com recurso a algoritmos de aprendizagem automática não supervisionados. Avaliamos experimentalmente o desempenho da nossa abordagem com um conjunto de dados artificial e um conjunto híbrido, obtendo valores para a F-score de até 97%.

***Index Terms*—detecção de intrusões, clustering, cryptojacking, fluxo de rede, indicadores de desempenho, análise de segurança**

I. INTRODUÇÃO

Com o aparecimento da *Bitcoin* em 2009, surgiu uma nova actividade designada como **mineração**, da qual surgiu um novo tipo de ataque, o *cryptojacking*, que por definição traduz a apropriação ilícita do poder computacional dos utilizadores sem o seu conhecimento ou consentimento para a obtenção de moedas criptográficas. Este ataque pode ser executado segundo dois vectores de ataque: *scripts* em navegadores web e *malware* binário (executáveis) [1], [2].

A *Bitcoin*, uma representação digital de um valor, juntamente com a *blockchain*, um registo de transacções acessível para apêndice apenas, veio substituir os intermediários convencionais nas transacções utilizando uma rede descentralizada de nós [3]. Para um novo grupo de transacções, o processamento das mesmas passa por utilizar uma elevada fracção de poder computacional para validar, adicionar, e distribuir um novo bloco com estas transacções através da *blockchain*. O esforço computacional está associado à solução de um problema criptográfico, o qual passa pela solução da designada Prova de Trabalho (PoW). Por sua vez o mineiro (*miner*) que resolve a PoW e consegue anexar o bloco à *blockchain* é recompensado com novas moedas [4], [5].

Embora recente, a extracção ilícita de moedas criptográfica representa uma ameaça tanto imediata como a longo prazo, para organizações e utilizadores. Por exemplo, vários dos domínios mais visitados na internet, incluindo YouTube [6], Showtime, e The Pirate Bay [7], foram criticados por empregar *scripts* de mineração contra os seus utilizadores. A um nível diferente, *malware* para a extracção de Monero (XMR) foi encontrado num sistema aeroportuário europeu [8].

As abordagens mais comuns de detecção de intrusões são: **utilização indevida / detecção de assinaturas**, que utiliza o conhecimento prévio dos ataques para construir assinaturas num processo de correspondência que identifica comportamentos que partilham semelhanças com as assinaturas existentes. Normalmente gera um baixo número de falsos positivos, mas falha na detecção de ataques nunca antes vistos; **detecção de anomalias**, onde a ideia é identificar padrões que se desviam do comportamento normal, uma solução apelativa devido à sua capacidade de detectar ataques nunca vistos. No entanto, conduz frequentemente a elevadas taxas de falsos alarmes. [9], [10].

Ambas abordagens partilham uma desvantagem em comum: a necessidade de conhecimento prévio, quer dos ataques existentes, quer do que é o comportamento normal. Ambos tipos de informação podem ser difíceis de obter [11].

Os Sistemas de Detecção de Intrusões (IDSs) também podem ser classificados segundo a fonte dos dados, como (1) baseados nos utilizadores, (2) baseados em rede, ou (3) híbridos. O primeiro (1) tem em conta as características do utilizador para a detecção, o segundo (2) monitoriza o tráfego da rede, e o terceiro (3) – a nossa abordagem – combina ambas as fontes de dados [12].

As técnicas de Aprendizagem Automática (ML), onde modelos estatísticos são construídos com o conhecimento extraído a partir dos dados, através de um processo computacional, desempenham um papel fundamental em muitos IDSs. Estas técnicas podem ser divididas em duas grandes categorias: métodos supervisionados e não supervisionados. Nos **métodos supervisionados**, dados de treino etiquetados são utilizados por um algoritmo para construir um modelo. Um dos métodos supervisionados mais comuns é a classificação, onde o modelo é utilizado para rotular pontos de dados nunca antes vistos e determinar a que classe pertencem. A maioria dos métodos de ML utilizados nos sistemas de detecção por utilização indevida são supervisionados [10].

Aprendizagem não supervisionada – o conjunto de técnicas que utilizamos – é convincente para a detecção de intrusões porque, por definição, não precisa de dados etiquetados como entrada, nem de assinaturas, ou de treino.

Apresentamos o CRYINGJACKPOT, um nova abordagem para um IDS híbrido baseado em aprendizagem não supervisionada para a detecção de *cryptojacking*. A nossa abordagem combina uma técnica de *ensemble* com diferentes algoritmos de *clustering*, utilizando tanto características baseadas em utilizadores como

em rede. Fazendo uso de indicadores de desempenho de um sistema operativo os quais traduzem o estado ou actividade de um sistema e fluxos de rede, pretendemos desenvolver uma abordagem robusta e fiável que não necessitará de conhecimentos prévios ou dados de treino para a detecção de *cryptojacking* em navegadores web ou em executáveis. Combinando a natureza dual das nossas características, a nossa abordagem é capaz de detectar *cryptojacking*, inclusive com cargas de trabalho intensivas e sob tráfego de rede elevado, sendo resistente a diferentes graus de *throttling* (percentagens de consumo), ocultação de código, e o uso de *proxies*.

Testamos a nossa abordagem primeiro com um conjunto de dados público, (CSE-CIC-IDS 2018 [13]), limitado ao uso de atributos baseados em fluxo de rede, um conjunto de dados que possui outros ataques em adição ao *cryptojacking*. Também criámos um conjunto de dados híbrido para analisar fluxos de rede e indicadores de desempenho em simultâneo. O CRYINGJACKPOT obteve valores para a *F-score* de 82% para o conjunto de dados público e uma *F-score* de 97% para o conjunto de dados híbrido em várias experiências concebidas para cobrir um grande número de cenários do mundo real.

Os resultados parciais do presente trabalho encontram-se publicados como: G. Gomes, L. Dias e M. Correia em "CryingJackpot: Network Flows and Performance Counters against Cryptojacking" no 19th IEEE International Symposium on Network Computing and Applications (NCA), Novembro, 2020.

II. ANTECEDENTES

Esta secção começa por explicar a arquitectura dos ataques de *cryptojacking*, fornecendo alguns antecedentes sobre a utilização de *clustering* para a detecção de intrusões, e explica a nossa abordagem para a avaliação (detecção de *outliers*).

A. Cryptojacking

Para lidar com algumas das questões levantadas pela *blockchain*, tais como o consenso dentro da rede, a irreversibilidade das operações, entre outras, foi introduzida a ideia de PoW. Este algoritmo torna a tarefa de adicionar novos blocos à cadeia altamente dispendiosa em termos de consumo computacional [5]. Esta tarefa – definida como mineração – é feita repetidamente até que um número específico de zeros previamente definido para o valor da *hash* seja obtido. O primeiro mineiro capaz de resolver este problema é recompensado com novas moedas pelo esforço computacional utilizado. O novo bloco é divulgado através dos nós do sistema que o acrescentam à sua instância local da *blockchain* [4], [5].

Resultado da constante evolução tecnológica, não muito depois do aparecimento da *Bitcoin*, a prática da extracção desta moeda criptográfica com as máquinas de utilizadores correntes tornou-se inviável. Este paradigma não iria mudar até 2014 com o aparecimento de uma nova moeda criptográfica, o **Monero (XMR)**. XMR usa o *RandomX*, um algoritmo que torna as tarefas de mineração igualmente eficientes na Unidade de Processamento Central (CPU) e na Unidade de Processamento Gráfico (GPU), ao mesmo tempo que torna a prática menos eficaz em Application-Specific Integrated Circuits (ASICs). Sendo ideal para mineração pelas máquinas dos utilizadores do dia a dia, houve um aumento no desenvolvimento de *scripts* para este fim.

B. Clustering e detecção de intrusões

Clustering é um conjunto de técnicas para encontrar padrões em dados não etiquetados de alta dimensão [9]. A ideia geral é que objectos semelhantes são agrupados através da medição da distância entre os mesmos. Um algoritmo de *clustering* aceita como entrada um conjunto de entidades e devolve um grupo de subconjuntos (ou *clusters*) dessas entidades. No entanto, este processo não nos diz nada sobre a natureza dos subconjuntos/clusters individuais.

Para a detecção de intrusões ou anomalias, muitos trabalhos fazem uso da ideia de que grandes *clusters* representam comportamentos normais e que os *outliers* (e.g, pequenos aglomerados de entidades ou ruído) podem corresponder a um comportamento anómalo [10]. Os *outliers* são os pontos de um conjunto de dados que são altamente improváveis de ocorrer para um determinado modelo desses mesmos dados.

Neste trabalho, perseguimos a ideia de descobrir *outliers* como sendo *clusters* com uma só entidade, para o qual a escolha dos algoritmos traduz um passo fundamental no desenho da nossa abordagem. Devido à falta de literatura para a detecção de *cryptojacking* com métodos não supervisionados, decidimos prover a nossa abordagem de redundância, escolhendo três tipos diferentes de algoritmos e a implementação de uma técnica de *ensemble* influenciada por trabalhos anteriores [14], [15].

Os algoritmos de *clustering* que escolhemos podem ser classificados como baseados em partições, hierárquicos, e baseados em densidade. *K-means* ($O(nkdi)$), é um algoritmo popular baseado em partições que distribui os pontos de dados X em k *clusters*, onde cada ponto de dados é mais semelhante ao centróide que define o seu *cluster* do que aos centróides de *clusters* vizinhos.

Dentro do *clustering hierárquico (HC)*, os algoritmos organizam os dados numa estrutura hierárquica de acordo com uma matriz de proximidade. Os resultados do HC são normalmente representados por uma árvore binária ou dendrograma. Escolhemos o algoritmo *Agglomerative*, uma vez que a utilização de algoritmos divisórios é uma prática invulgar devido ao esforço computacional associado [16].

DBSCAN agrupa os pontos de dados em *clusters* que têm uma densidade mais elevada do que um certo valor limite ($MinPts$). Executa esta tarefa dentro de uma janela com um tamanho especifico definido pela distância até ao ponto de dados (eps)[10].

III. CRYINGJACKPOT, A ABORDAGEM

A nossa abordagem titulada CRYINGJACKPOT (Figura 1), visa identificar automaticamente outliers, i.e, entidades – utilizadores – alvo de *cryptojacking*, extraindo, combinando e processando um conjunto de características/atributos. Estas características são obtidas a partir de indicadores/contadores de desempenho relativos aos utilizadores [17] e fluxo de rede [18]. Em relação aos contadores, neste trabalho consideramos apenas contadores fornecidos pelo sistema operacional Windows, mas a abordagem é genérica.

A. Colecção de dados

Como mencionado anteriormente, o CRYINGJACKPOT obtém os atributos de duas fontes de dados diferentes, indicadores de desempenho e fluxo de rede.

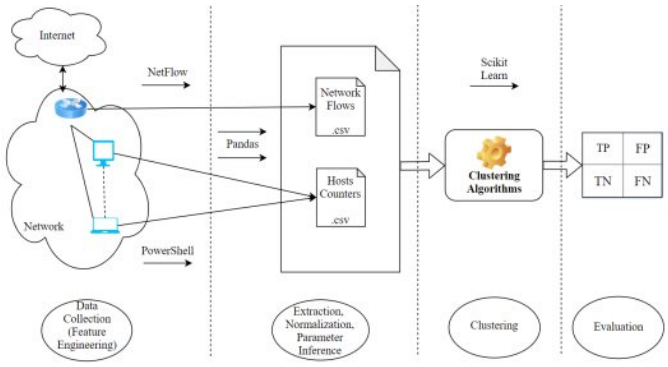

Fig. 1. Perspectiva geral do CRYINGJACKPOT.

Optamos por usar **indicadores de desempenho** para máquinas Windows, para explorar o comportamento do *cryptojacking* como um ataque computacionalmente intensivo. Acabamos com 123 contadores de desempenho, um conjunto de indicadores que descrevem **CPU, memória, utilização da rede, e processos em execução dentro de uma máquina.** Alguns exemplos:

```
\Processor(*)\% Processor Time
\Memory\Available Bytes
\Network Interface(*)\Bytes Sent/sec
```

Em relação às características baseadas no **fluxo de rede** as nossas escolhas foram influenciadas por obras que descrevem o protocolo Stratum e a detecção de *cryptojacking* [19].

Do ponto de vista da fonte, o CRYINGJACKPOT começa por extrair 2 características fixas que exploram os padrões de dados recebidos: (1) `SrcBytesPktBwd`, número de bytes por pacote recebido como SrcIP; (2) `SrcMeanBytesBwd`, número médio de bytes recebidos como SrcIP.

Implementamos também uma técnica que melhorou significativamente os nossos resultados. Extraímos um conjunto de 11 características de fluxos cujos IPs de destino tinham sido, durante uma determinada janela de tempo, contactados apenas por um IP hospedeiro/fonte, i.e., um destino raro. Esta decisão visou explorar as ligações feitas com os domínios exclusivos das piscinas de mineração. Além disso, o CRYINGJACKPOT também define 5 características relacionadas com portas de rede, HTTP/80 e HTTPS/443, utilizando neste caso, as portas de rede de origem como chaves de agregação.À semelhança da avaliação realizada para os contadores de desempenho, também pesámos o nosso conjunto inicial de características, terminado com 37 características baseadas em fluxo de rede.

O CRYINGJACKPOT é semelhante desde do ponto de vista do destinatário, diferindo nas características relacionadas com portas de rede.

B. Extracção, normalização, e parâmetros iniciais

A nossa abordagem pode processar dados em diferentes janelas de tempo pré-definidas, de duração $\mathcal{W} = \{w_1, w_2, ..., w_n\}$, e.g., 10 minutos, 1 hora, 1 dia. Esta abordagem efectua o *clustering* para cada janela de duração $w_i \in \mathcal{W}$, para os períodos de dados disponíveis. Para cada janela de tempo de duração $w_i \in \mathcal{W}$, são extraídas características de todos os fluxos e contadores relativamente à janela de tempo em análise seguindo a lógica da Secção III-A.

Esta etapa resulta em dois vectores de características para cada anfitrião, um contendo as características obtidas dos contadores de desempenho e o segundo com características dos fluxos da rede.

Para cada janela de tempo $w_i \in \mathcal{W}$, os atributos têm de ser normalizados antes da etapa de *clustering*. A normalização é realizada escalonando cada característica a um intervalo comum, e é importante não perder informação durante este processo.

Avançando para a fase de *clustering*, uma decisão crítica está associada aos parâmetros de inicialização dos algoritmos. Estes parâmetros ditam os resultados obtidos no que é conhecido como processo de tentativa e erro, por não existir um conjunto específico de parâmetros pré-definidos que se saiba que gerem resultados ideais.

Em relação aos parâmetros do *K-means* e do *Agglomerative*, ambos distribuem um conjunto de objectos em K *clusters*, pelo que K tem de ser definido. O melhor valor de K pode ser encontrado através de força bruta, um método que é inviável na prática, devido ao esforço computacional necessário. Para tal, utilizamos dois algoritmos, o algoritmo **do cotovelo** e o algoritmo **silhueta**, para encontrar o valor de K.

Em relação ao *DBSCAN*, este requer que a densidade numa vizinhança de um determinado objecto seja suficientemente alta para ter um *cluster* atribuído [16]. Temos de escolher o número de amostras numa vizinhança para que um ponto possa ser considerado como um ponto central ($min_samples$) e a distância máxima entre duas amostras para que uma seja considerada como na vizinhança da outra (eps) [20]. eps pode ser escolhido como a alteração mais pronunciada, quando as distâncias entre cada entidade e os seus vizinhos são calculadas e ordenadas [21]. Os três algoritmos mencionados podem ser implementados utilizando a distância euclidiana.

C. Clustering e avaliação

Após a extracção e normalização das características, os vectores de características são fornecidos como entrada para os algoritmos de *clustering*. A ideia é agrupar máquinas com comportamento semelhante, com base nos valores das características. O CRYINGJACKPOT é capaz de executar esta tarefa utilizando as características relacionadas com a máquina/utilizador independentemente das baseadas em fluxo, sendo o objectivo principal a utilização simultânea de ambos os conjuntos. Este é um passo importante na concepção porque dota a nossa abordagem de flexibilidade para futuras implementações.

O resultado da etapa de *clustering* é um conjunto de *clusters/clusters* com anfitriões para cada janela de tempo de duração $w_i \in \mathcal{W}$. Cada anfitrião aparece num *cluster* para cada janela de tempo.

Vários métodos podem ser utilizados para a classificação. Se necessário, uma inspecção manual pode ser realizada por um analista em segurança, começando com os *clusters* mais pequenos. Contudo, para automatizar a identificação de anomalias, consideramos que **um outlier é uma entidade isolada num cluster**. A desvantagem desta abordagem é que, para várias vítimas em simultâneo, as nossas capacidades de detecção deterioram os resultados, uma vez que as vítimas tendem a partilhar *clusters*.

IV. Conjunto de dados híbrido

Devido à natureza da nossa abordagem, não fomos capazes de descobrir um conjunto de dados que satisfizesse as nossas necessidades de ter fluxos de rede e indicadores de desempenho, combinados com ataques de *cryptojacking*. Por conseguinte, criámos um conjunto de dados híbrido, onde dados do mundo real foram recolhidos e adaptados para serem depois utilizados nos nossos testes.

Tal como foi afirmado por Sharafaldin et al. [22], existe um conjunto de considerações a ter em conta ao utilizar ou criar um conjunto de dados. Tivemos estas observações em consideração pois precisávamos de obter um conjunto de dados específicos para os nossos testes de detecção. Esta tarefa foi conduzida em duas fases: Recolha de dados benignos (perfis-B) e recolha de dados maliciosos (perfis-M).

A nossa abordagem para a criação do conjunto de dados é descrita como se segue. Decidimos obter um pequeno conjunto de dados do mundo real, sem filtragem. Extraímos toda a informação, relativa ao tráfego de rede (pacotes) e indicadores de desempenho, para um conjunto de utilizadores cujas impressões digitais reflectiram as actuais tendências computacionais do mundo real. Este conjunto inicial de dados foi então utilizado para replicar a restante estrutura da rede (Secção IV-B) utilizando uma abordagem que visou assegurar que os dados se aproximassem do que seria um conjunto de dados do mundo real. Os dados do perfil M (Secção IV-A) foram obtidos com uma abordagem semelhante, utilizando amostras reais de malware combinadas com o tráfego real dos utilizadores.

A. Recolha de amostra maliciosas

Encontrar amostras de *cryptojacking* foi uma das tarefas mais importantes, pois queríamos garantir que apenas amostras maliciosas activas eram recolhidas, e que tínhamos coberto o maior número possível de variantes do ataque. O nosso objectivo era encontrar scripts de utilização em navegadores web, bem como malware executável.

Em relação ao *cryptojacking web*, utilizámos um motor de pesquisa de código-fonte (https://publicwww.com/) para procurar etiquetas específicas que identificassem código de scripts de mineração incorporados em diferentes domínios.

Tabela I
Amostras de cryptojacking baseadas em navegador web.

Etiqueta de Busca	Domínio	Script de Mineração
minero.cc	tibumpiscinas.com.br	minero.cc/lib/minero.min.js
minero.cc	naftkala.com	minero.cc/lib/minero.min.js
hostingcloud	serieshdpormega.com	hostingcloud.racing/8vVW.js
hostingcloud	vstars.pl	hostingcloud.racing/BBlJ.js
webminepool	oslafo.com	webminepool.com/lib/base.js

Da nossa pesquisa seleccionámos 5 amostras baseadas em navegador web que verificamos estarem ativas. A Tabela I resume a informação sobre as amostras e a Figura 2 mostra um extracto de código-fonte para um dos domínios maliciosos. Estas amostras também funcionam com diferentes níveis de *throttle*, ou seja, limite para o consumo efectivo de recursos, o que nos permite explorar diferentes cenários durante a detecção. Conseguimos mostrar que mesmo ataques com consumo reduzido de CPU são detectados.

```
<script src="https://www.hostingcloud.racing/Aync.js"></script>
<script>
      var _client = new
  Client.Anonymous('03f0ff94450082fe747c62c0d24b9c90d4d5f06d581643c1ebe2740356f36321',
  {
          throttle: 0 , c: 'w' , ads: 0
      });
       _client.start();
</script>
<div style="display:none">
    </div>
```

Fig. 2. Código-fonte do domínio www.serieshdpormega.com

Também desenvolvemos a nossa abordagem para poder detectar **amostras de *cryptojacking* executável**. Para encontrar estes binários, procurámos através de repositórios de *malware* como https://urlhaus.abuse.ch/ e https://dasmalwerk.eu/. Recolhemos 5 amostras de *malware* executável para *cryptojacking* estes também com diferentes valores de *throttle* (Tabela II).

Tabela II
Amostras de cryptojacking executável.

Amostra
down.gogominer.com/sex_Live1.5.0.1099.exe
14.141.175.107/cryptominerbros/wordpress/
14.141.175.107/cryptominerbros/wordpress/wp-content/Vh/
149.28.24.180/miner/bashd32
ivansupermining.info/bin/minbuild.exe

B. Criação das extracções base

Com as amostras maliciosas necessárias, criámos as extracções de base que mais tarde foram utilizadas para criar o conjunto de dados híbridos pretendido. Os perfis B visam encapsular os comportamentos das entidades dos utilizadores nas suas tarefas normais. Este é um dos grandes desafios dos conjuntos de dados artificiais, o de alcançar o comportamento complexo de um utilizador do mundo real. Para as extracções do nosso conjuntos de dados definimos 3 tipos de perfis de comportamento para os anfitriões: (1) perfil baixo (L); (2) perfil médio (M); (3) perfil alto (H). Perfil baixo representa anfitriões com baixa utilização dos recursos do sistema e tráfego de rede ligeiro. Perfil médio representa anfitriões com um uso intermédio de recursos focados na produtividade (Microsoft Office, trabalho em nuvem). O perfil alto visa reproduzir cargas computacionais intensas (máquinas virtuais, Adobe Photoshop, downloads, múltiplos separadores web). A nossa abordagem visou a criação de um conjunto de dados que simulassem diferentes tipos de redes, i.e., permitir a detecção de *cryptojacking* para diferentes cargas de trabalho, e mais importante, para compreender se as cargas de trabalho de alto perfil escondem o comportamento do ataque.

Tendo a nossa estratégia definida, utilizámos 9 computadores Windows 10 Enterprise com CPU Intel(R) Pentium(R) G3250 @3.20GHz, 3200 Mhz, 2 Core e 8,00 GB de RAM para executar as nossas extracções. Estas máquinas foram divididas pelos três perfis, e cada máquina tinha a ferramenta Wireshark instalada para obter os pacotes de rede, e um script PowerShell por nós desenvolvido para a recolha dos indicadores de desempenho.

Seguindo um calendário para que as expectativas para os perfis fossem satisfeitas, acabamos por ter o equivalente a um dia de extracções relativas a actividade benigna, com 9 ficheiros CSV três para cada perfil e de forma semelhante, 9 ficheiros PCAP com o tráfego da rede. Aos ficheiros PCAPs aplicámos o *CICFlowMeter* disponível em https://github.com/ahlashkari/

CICFlowMeter para obter fluxos de rede partir dos pacotes recolhidos.

Seguindo a mesma abordagem, mas agora utilizando as amostras maliciosas anteriormente mencionadas, utilizámos 10 máquinas. 5 delas foram utilizadas para executar os scripts web seguindo calendarização e as outras 5 para as amostras binárias a executar permanentemente no segundo plano das vítimas. Além disso, também aqui as máquinas foram divididas por perfis. No final da nossa extracção diária, obtivemos ficheiros equivalentes como na extracção benigna.

Como passo final para criar o conjunto final de dados, criámos 10 dias de extracções, um dia para cada uma das vítimas que tínhamos criado. Definimos o número de máquinas que queríamos para cada um dos dias, um número que variou entre 300 e 400 máquinas. O processo para criar estas máquinas a partir das extracções base foi:

- definir o número de máquinas por perfil;
- reproduzir a quantidade de máquinas pretendidas a partir das extracções base com a adição de ruído.

Relativamente a aplicação de ruído, o objectivo era clonar máquinas/anfitriões alterando os valores de base por uma percentagem aleatória (Equação 1).

$$x' = x + random([-1, 1]) \times \frac{x}{random([3, \alpha])}, \alpha \in [15, 30] \quad (1)$$

Os valores predefinidos foram escolhidos para estabelecer um limiar para o qual não alterássemos o tipo de perfil para as réplicas. O nosso conjunto de dados final terminou com 10 dias de dados, contendo os primeiros 6 dias predominantemente máquinas de baixo perfil, um comportamento comum em redes do mundo real, e os outros 4 dias com máquinas predominantemente de alto perfil , sendo a principal razão destas a avaliação da nossa abordagem na detecção de *cryptojacking* entre elevadas cargas de trabalho. Além disso, durante estes 10 dias, metade têm vítimas sob *cryptojacking* web, i.e., o ataque existe para janelas de tempo específicas durante o dia. A outra metade tem *malware* binário estando o *cryptojacking* a correr em segundo plano.

V. Avaliação experimental

Para desenvolver e avaliar o CryingJackpot, utilizámos maioritariamente Python (v3) [23]. Além disso, utilizámos bibliotecas populares tais como *Pandas* [24] para manipulação de dados, e *Scikit-learn* [20] para o processamento de dados e os algoritmos de *clustering*. Todas as experiências foram conduzidas com uma máquina Windows (4-Core Intel Core i7 2.6GHz com 16GB de RAM). O objectivo das experiências é o teste contra diferentes cenários possíveis do mundo real e avaliação de desempenho.

A. Métricas

Tabela III
Indicadores baseados na matriz de confusão.

Indicador	Descrição
Verdadeiros Positivos (TP)	entidades correctamente classificadas como *outliers*
Falsos Positivos (FP)	entidades incorrectamente classificadas como *outliers*
Verdadeiros Negativos (TN)	entidades correctamente classificadas como *inliers*
Falsos Negativos (FN)	entidades incorrectamente classificadas como *inliers*

Considerámos um *outlier* – um positivo – como sendo um anfitrião isolado num *cluster*, identificado por um endereço IP, tendo sido assinalado pelo CryingJackpot. Com base na matriz de confusão da Tabela III, foram utilizadas as seguintes métricas na avaliação:

- Precisão (PREC) – fracção de outliers que são reais (i.e., verdadeiros positivos): $PREC = TP/(TP + FP)$
- Recall (REC) – fracção de outliers corretamente classificados pelo detetor: $REC = TP/(TP + FN)$
- F-score – pontuação global de deteção: $FScore = 2 \times (PREC \times REC)/(PREC + REC)$

Para obter os valores dos indicadores da Tabela III, analisámos os resultados do *clustering* relativos apenas às janelas de ataque (*malware* persistente implica a consideração de todas as janelas de tempo).

B. Caracterização dos conjuntos de dados

Foram utilizados dois conjuntos de dados. O primeiro conjunto de dados, por nós desenvolvido, é descrito na Secção IV.

O segundo, *CIC-IDS 2018* [13], que designámos como o conjunto de dados artificial, foi criado para testar e avaliar IDS em rede. Utilizámos este conjunto de dados combinado com os ataques de *cryptojacking* que criámos para testar a nossa abordagem em diferentes circunstâncias (o conjunto de dados original não continha ataques de *cryptojacking*).

Ao contrário do conjunto de dados híbrido, o conjunto artificial já possuía ataques estando estes limitados a informação relativa a fluxo de rede, sobre os quais adicionamos informações relacionadas com o *cryptojacking*.

C. Resultados com o conjunto híbrido

Os nossos primeiros testes consistiram em combinar características baseadas em fluxo com características baseadas em contadores, para identificar *cryptojacking* utilizando os algoritmos de *clustering* já mencionados. Também realizámos a detecção utilizando ambos os tipos de características de forma independente. Uma primeira observação foi que **a combinação de características baseadas em anfitriões e baseadas em fluxo gerou melhores resultados do que a sua utilização individual**. Estas diferenças traduzem-se em resultados muito mais consistentes para todos os algoritmos implementados e melhorias nos resultados globais (Figura 3). Da utilização individual de atributos baseados em fluxo, obtivemos um aumento de 39% para a melhor F-score, e um aumento de 4,5% para a utilização de contadores, resultado do uso combinado de características.

Relativamente aos resultados gerais do método, **os três algoritmos implementados tiveram um comportamento semelhante, com o DBSCAN a ter um desempenho ligeiramente superior.** Para o K-means e o Agglomerative, a silhueta contribuiu para melhores resultados do que o algoritmo do cotovelo. Contudo, o mesmo não se aplicou ao conjunto de dados artificial (cf. Secção V-D), o que poderá ter estado relacionado com a utilização exclusiva de características baseadas em fluxo de rede.

D. Resultados com o conjunto artificial

O conjunto de dados artificiais tem apenas características baseadas em fluxo de rede, mas inclui mais ataques do que o

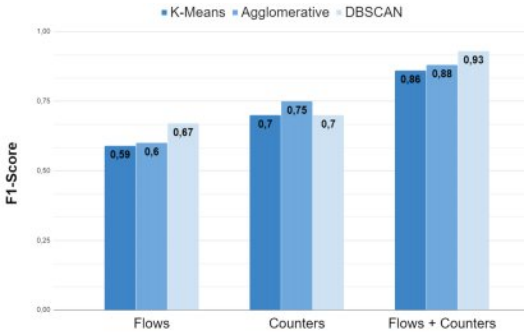

Fig. 3. Comparação de F-scores para a utilização de fluxo de rede, indicadores de desempenho, e o uso simultâneo de ambos tipos de características (conjunto híbrido).

cryptojacking. Por conseguinte, aqui avaliamos o desempenho do CRYINGJACKPOT com essa restrição e perante a presença de outros ataques.

Para os resultados obtidos da deteção de *cryptojacking* para as duas melhores janelas de tempo, observamos que **a nossa abordagem de clustering com *ensemble* melhora os resultados** que são um relativamente inconsistentes para os três algoritmos implementados (e.g., baixa precisão mas alta recall para o Agglomerative). Além disso, o melhor desempenho na detecção de *cryptojacking* foi obtido com o K-means combinado com o método de cotovelo, com uma média de F-score para todas as janelas de tempo de 0,68.

VI. CONCLUSÃO

Apresentamos o CRYINGJACKPOT, uma abordagem híbrida de detecção de *cryptojacking*, baseada em aprendizagem automática não supervisionada, que detecta ataques indefinidos sem assinaturas e dados de treino. A nossa abordagem baseia-se em *clustering*, i.e., na agregação de utilizadores com padrões de dados semelhantes, utilizando para este fim a combinação de características de dados baseados em fluxo de rede e indicadores de desempenho para máquinas Windows. Para a avaliação, foram utilizados dois conjuntos de dados. Em primeiro lugar, desenvolvemos um conjunto de dados híbrido fiável com dados baseados em fluxo e baseados nos utilizadores, onde a detecção de *cryptojacking* foi realizada com excelentes resultados. Depois utilizámos um conjunto de dados público para testar as nossas capacidades quando limitados ao fluxo de rede, estudando a detecção de *cryptojacking* na presença de outros ataques, tendo obtido resultados positivos. O CRYINGJACKPOT provou ser uma solução robusta e flexível para a detecção de *cryptojacking* com valores de precisão, recall, e F-score entre 0,92 e 1.

REFERÊNCIAS

[1] S. Eskandari, A. Leoutsarakos, T. Mursch, and J. Clark, "A first look at browser-based cryptojacking," in *Proceedings - 3rd IEEE European Symposium on Security and Privacy Workshops, EURO S and PW 2018*, Jul. 2018, pp. 58–66.

[2] H. L. J. Bijmans, T. M. Booij, and C. Doerr, "Inadvertently making cyber criminals rich: A comprehensive study of cryptojacking campaigns at internet scale," *Proceedings of the 28th USENIX Security Symposium*, pp. 1627–1644, 2019.

[3] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system satoshi nakamoto," Tech. Rep., 2008.

[4] M. E. Peck, "Blockchains: How they work and why they'll change the world," *IEEE Spectrum*, vol. 54, no. 10, pp. 26–35, Oct. 2017.

[5] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *Proceedings - 2017 IEEE 6th International Congress on Big Data, BigData Congress 2017*, Sep. 2017, pp. 557–564.

[6] D. Goodin, *Now even YouTube serves ads with CPU-draining cryptocurrency miners*, 2018. [Online]. Available: https://arstechnica.com/information-technology/2018/01/now-even-youtube-serves-ads-with-cpu-draining-cryptocurrency-miners (visited on 11/08/2019).

[7] The Guardian, *Ads don't work so websites are using your electricity to pay the bills*, 2017. [Online]. Available: https://www.theguardian.com/technology/2017/sep/27/pirate-bay-showtime-ads-websites-electricity-pay-bills-cryptocurrency-bitcoin (visited on 11/05/2019).

[8] S. Gatlan, *European airport systems infected with monero-mining malware*, 2019. [Online]. Available: https://www.bleepingcomputer.com/news/security/european-airport-systems-infected-with-monero-mining-malware (visited on 11/08/2019).

[9] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys and Tutorials*, vol. 18, no. 2, pp. 1153–1176, Apr. 2016.

[10] S. Dua and X. Du, *Data Mining and Machine Learning in Cybersecurity*. 2011, vol. 66, pp. 37–39.

[11] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, 2012.

[12] F. Sabahi and A. Movaghar, "Intrusion detection: A survey," *2008 Third International Conference on Systems and Networks Communications*, pp. 23–26, 2008.

[13] *A realistic cyber defense dataset (cse-cic-ids2018) - registry of open data on aws*. [Online]. Available: https://registry.opendata.aws/cse-cic-ids2018 (visited on 08/12/2020).

[14] A. Strehl and J. Ghosh, "Cluster ensembles: A knowledge reuse framework for combining multiple partitions," *Journal of Machine Learning Research*, vol. 3, no. Dec, pp. 583–617, 2002.

[15] L. Dias, S. Valente, and M. Correia, "Go with the flow: Clustering dynamically-defined netflow features for network intrusion detection with DynIDS," 2020, in submission.

[16] R. Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE Transactions on Neural Networks*, vol. 16, no. 3, pp. 645–678, 2005.

[17] *Windows performance monitor - Microsoft Docs*. [Online]. Available: https://docs.microsoft.com/en-us/windows/win32/perfctrs/about-performance-counters (visited on 08/08/2020).

[18] B. Claise, "Cisco systems netflow services export version 9," RFC 3954, Oct. 2004.

[19] D. Draghicescu, A. Caranica, A. Vulpe, and O. Fratu, "Crypto-mining application fingerprinting method," *2018 International Conference on Communications*, pp. 543–546, 2018.

[20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[21] N. Rahmah and I. S. Sitanggang, "Determination of optimal epsilon (eps) value on DBSCAN algorithm to clustering data on peatland hotspots in Sumatra," in *IOP Conference Series: Earth and Environmental Science*, vol. 31, Feb. 2016.

[22] I. Sharafaldin, A. Gharib, A. H. Lashkari, and A. A. Ghorbani, "Towards a reliable intrusion detection benchmark dataset," *Software Networking*, vol. 2017, no. 1, pp. 177–200, 2017.

[23] G. van Rossum, "Python tutorial release 3.8.1," Tech. Rep., 2020.

[24] W. McKinney *et al.*, "Data structures for statistical computing in python," in *Proceedings of the 9th Python in Science Conference*, vol. 445, 2010, pp. 51–56.