MineCap: Detecção de Mineração de Criptomoedas em Redes Corporativas com Aprendizado de Máquina e Prevenção de Abusos com Redes Definidas por *Software*

Helio N. Cunha Neto[1], **Natalia C. Fernandes**[1], **Diogo M. F. Mattos**[1]

[1]MídiaCom - PPGEET/TET/UFF
Universidade Federal Fluminense (UFF) – Brasil

{heliocunha,natalia,menezes}@midiacom.uff.br

Resumo. *A mineração não autorizada de criptomoedas implica o uso de valiosos recursos de computação e o alto consumo de energia. Este trabalho propõe o mecanismo MineCap, um mecanismo dinâmico e em linha para detectar e bloquear fluxos de mineração não autorizada de criptomoedas, usando o aprendizado de máquina e redes definidas por software. O MineCap desenvolve a técnica de super aprendizado incremental, uma variante do super learner aplicada ao aprendizado incremental. O super aprendizado incremental proporciona ao MineCap precisão para classificar os fluxos de mineração ao passo que o mecanismo aprende continuamente com os dados recebidos. Os resultados revelam que o mecanismo alcança 98% de acurácia, 99% de precisão, 97% de sensibilidade e 99,9% de especificidade e evita problemas relacionados ao desvio de conceito. Os resultados desse trabalho foram submetidos e aceitos em um congresso internacional, um congresso nacional, um minicurso, uma revista indexada e, ainda, há um artigo em processo de revisão em uma revista.*

Abstract. *Covert mining of cryptocurrency implies the use of valuable computing resources and high energy consumption. In this work, we propose MineCap, a dynamic online mechanism for detecting and blocking covert cryptocurrency mining flows, using machine learning on software-defined networks. MineCap uses a novel technique called super incremental learning, a variant of the super learner with incremental learning. Hence, we design an accurate mechanism to classify mining flows that learn with incoming data with an average of 98% accuracy, 99% precision, 97% sensitivity and 99.9% specificity and avoid concept drift-related issues. The results of this work were submitted and accepted at one international congress, one national congress, one short course, one journal, and an under review paper in a journal.*

1. Introdução

As redes corporativas são alvos constantes de uma grande variedade de ameaças [Andreoni Lopez et al. 2017].Assim, há a necessidade crescente de ferramentas para proteger informações estratégicas, recursos de rede e poder de processamento. Existem diversos tipos de sistemas que protegem as redes de computadores, tais como *Firewalls*, *IDS*, *IPS*, Antivírus etc. Existem também propostas na literatura que utilizam

Este trabalho foi realizado com recursos da CNPq, CAPES, FINEP, FUNTTEL, FAPERJ, TAESA e do programa de P&D ANEEL (PD-07130-0053/2018).

técnicas de aprendizado de máquina e de *Big Data Analytics* para classificar fluxos de rede em tráfego normal e tráfego malicioso, com a intenção de identificar ataques de rede [Andreoni Lopez et al. 2017].

Com a popularização das criptomoedas, como por exemplo Bitcoin e Ethereum, usuários maliciosos exploram vulnerabilidades da rede e de dispositivos em redes corporativas para realizarem o processo de mineração, mesmo quando não há o consentimento dos gestores da rede. Além de usuários internos de uma rede corporativa que podem, sem permissão, realizar o processo de mineração, existem algumas ameaças, categorizadas como *cryptojacking*, em que invasores desenvolvem aplicativos maliciosos que são embutidos em páginas Web para minerar criptomoedas sem o consentimento do usuário. A maioria dos processos de mineração utilizam o mecanismo de prova de trabalho (*Proof of Work* - PoW). No mecanismo de PoW, o minerador cria um novo bloco, que consiste de uma ou mais transações utilizando uma criptomoeda que deve ser adicionada à cadeia de blocos. Para isso, o minerador deve, por força bruta, encontrar um *nonce* que resolve o desafio de processar um valor *hash* começando com um número pré-determinado de zeros. O minerador pode processar um bloco individualmente ou pode se juntar a um grupo de mineradores para processar um bloco, esse grupo de mineradores é chamado de *pool* de mineração. Cada *pool* tem seu próprio endereço IP e porta.

1.1. Caracterização do Problema e Motivação

A mineração não autorizada de criptomoedas torna-se um inconveniente, pois o processo de mineração acaba onerando a rede, recursos computacionais e elétricos e, atualmente, os principais mecanismos de proteção contra mineração não autorizada de criptomoedas protegem apenas um *host* específico, deixando, assim, a rede vulnerável a dispositivos sem o mecanismo de proteção instalado. Como consequência, observa-se, atualmente, inúmeros exemplos de ataques de mineração não autorizada. Por exemplo, em 2018, um hacker invadiu milhares de roteadores para realizar mineração não autorizada da criptomoeda Monero, utilizando o código *CoinHive* para realizar o processo de mineração[1]. O *site* PirateBay também foi identificado utilizando o código *CoinHive* para realizar mineração sem o consentimento de seus usuários. Nesse caso, ao acessar o *site* PirateBay, o usuário executa em segundo plano um código de mineração escrito em JavaScript[2]. Existe também a possibilidade de usuários mal-intencionados se aproveitarem da vulnerabilidade de um *site* com bastante acesso e implantar um código de mineração. Esse caso aconteceu em um dos *sites* do governo de São Paulo, o Portal do Cidadão. O código malicioso foi retirado da página assim que os administradores foram notificados[3].

1.2. Objetivos e Contribuições

O objetivo principal desse trabalho é desenvolver um sistema de detecção e prevenção de mineração de criptomoedas não autorizada, chamado MineCap [4], capaz de

[1]Notícia disponível em https://g1.globo.com/economia/tecnologia/blog/altieres-rohr/post/2018/08/02/hackers-atacam-roteadores-mikrotik-no-brasil-para-minerar-criptomoedas-na-web.ghtml. Acessado em 13 de maio de 2020.

[2]Disponível em https://www.bbc.com/news/technology-41306384. Acessado em 13 de maio de 2020.

[3]Disponível em https://g1.globo.com/tecnologia/noticia/site-do-governo-de-sp-usou-computador-de-visitante-para-minerar-moeda-virtual.ghtml. Acessado em 13 de maio de 2020.

[4]Disponível em https://github.com/helioncneto/MineCap

identificar e bloquear fluxos de mineração através de chamadas a um controlador de redes definidas por Software (Software Defined Networking - SDN), responsável por instalar fluxos nos comutadores OpenFlow [Neto et al. 2019a, Neto et al. 2019b, Neto et al.].

O trabalho propõe uma nova técnica de aprendizado, chamada de super aprendizado incremental [Neto et al. 2019b, Neto et al.], que propõe uma modificação na técnica super *learner* [Van der Laan et al. 2007] e utiliza aprendizado incremental [Medeiros et al. 2019] para melhorar a eficiência do MineCap. O super aprendizado incremental foi desenvolvido para contornar o desvio de conceito, desafio comum em sistemas de processamento de grandes massas de dados em linha, quando há mudanças nas estatísticas das características dos dados anualizados que influenciam no resultado final da classificação.

2. O Sistema MineCap

O MineCap é um sistema de detecção de mineração de criptomoedas, que além de detectar a mineração de criptomoedas, realiza o bloqueio do fluxo de mineração direto no comutador OpenFlow, sendo assim, o mais próximo possível da fonte. O sistema proposto possui uma arquitetura com três camadas: captura, processamento e bloqueio. A *camada de captura* tem como objetivo capturar os dados e prepará-los para a camada superior. Os pacotes de rede são capturados através da biblioteca `libpcap`. A camada de aplicação emprega uma aplicação desenvolvida na linguagem Python, baseada na aplicação `flowtbag`[5], para realizar a abstração de pacotes em fluxos. Os fluxos são publicados no serviço de mensagens Apache Kafka, um sistema de publicação e assinatura (*publisher/subscriber*) que garante o armazenamento e a entrega confiável das mensagens. A *camada de processamento* é responsável por consumir, processar e classificar os fluxos da rede que o Apache Kafka fornece como dados em fluxo. O mecanismo MineCap adota o Apache Spark *Streaming* como plataforma de processamento de fluxo em linha. A *camada de bloqueio* recebe a saída do classificador e instala uma regra de bloqueio para fluxos rotulados como mineração de criptomoeda. O OpenFlow 1.3 [The OpenFlow Consortium 2012] é o protocolo de interface sul (*Southbound API*) de rede definida por *software* utilizado no MineCap.

3. Super Aprendizado Incremental

A proposta de uma nova técnica de aprendizado de máquina chamada de super aprendizado incremental [Neto et al. 2019b, Neto et al.] visa, além de ter um super modelo alimentado pelos melhores classificadores de um determinado problema, conseguir aprender com novos dados recebidos em linha. A capacidade de aprender com novos dados sem a necessidade de retreinar o modelo inteiro é uma proposta para solução do desvio de conceito, um problema que ocorre quando há uma mudança no comportamento estatístico na variável de saída que o modelo está tentando prever [Medeiros et al. 2019]. Na proposta, modelos de aprendizado de máquina são usados como modelos candidatos que alimentam o super modelo. Os modelos candidatos são treinados e realizam as previsões da probabilidade da amostra ser ataque ou normal e então as entrega como entrada para treinamento do super modelo. Então, posteriormente, o super modelo é treinado parcialmente com novos dados utilizando o aprendizado incremental. Os modelos candidatos geram a lista $W = (w_{i0}, w_{i1}, y)$, onde w_{i0} é a probabilidade do iésimo modelo

[5]Disponível em https://github.com/DanielArndt/flowtbag. Acessado em 13 de maio de 2020.

candidato retornar como saída a classe tráfego normal e w_{i1} é a probabilidade do iésimo modelo candidato retornar como saída a classe de tráfego de mineração. Finalmente, y é a saída desejada de cada amostra, ou seja, a classe alvo dessa amostra. A lista W gerada pelos modelos candidatos é passada como entrada para treinar o super modelo. Após a fase de treinamento, o sistema está pronto para receber novos fluxos de rede em linha para classificação. Os novos fluxos de rede são entregues aos modelos candidatos para classificação, que computam as probabilidades do fluxo ser normal ou ataque, i.e. w_{i0}, w_{i1}, e enviam como entrada para o super modelo realizar a previsão. Em seguida, o super modelo armazena esses novos dados classificados temporariamente. Quando um número k suficiente de amostras é atingido, o processo incremental verifica cada amostra armazenada e coleta somente as amostras que possuem maior probabilidade de serem fluxo de mineração e menor probabilidade de serem fluxo normal, ou o inverso, para realizar a aprendizagem parcial.

O conjunto de dados utilizado na avaliação do MineCap e do algoritmo proposto foi criado pelos autores no laboratório MídiaCom. Para tanto, foram utilizados diversos softwares de mineração em uma rede, de modo a gerar tráfego real de mineração para o treino dos modelos. Os aplicativos de mineração utilizados foram o MinerGate[6] e o GuiMiner[7], no sistema operacional Windows, para criar o conjunto de dados de treino, e o xmrig e cpuminer, utilizando o sistema operacional Linux, para realizar as avaliações. O tráfego foi capturado em um ambiente controlado com um *switch*, computadores atuando como mineradores e outros utilizando diferentes perfis de navegação como *streaming* de vídeo, descarga de arquivos, acesso a sítios Web e outros.

4. Resultados Obtidos

O sistema MineCap executa tanto em uma estação (*host*) como em um *cluster* separado do controlador de rede. A execução do MineCap em uma ou mais estações separadas do controlador de rede evita a sobrecarga no controlador. O controlador da rede redireciona todos os pacotes de saída da rede local para o MineCap, aplicando a técnica de espelhamento de porta na saída da rede. A avaliação foi realizada em uma rede emulada, usando a plataforma de emulação `Mininet`, com 16 *hosts* em uma topologia em árvore personalizada, usando sete comutadores executando o protocolo OpenFlow 1.3 [The OpenFlow Consortium 2012].

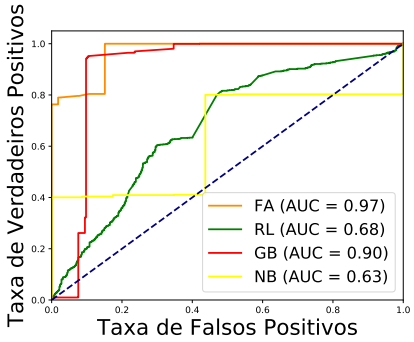
Na primeira avaliação, entre os dezesseis *hosts* do ambiente, quatro deles executavam aplicativos de mineração de criptomoedas[8] e o restante estava repetindo o tráfego de 30 minutos contido em um arquivo de captura real de tráfego de rede, usando `tcpreplay`[9]. Nessa primeira etapa, é avaliado o desempenhos de quatro algoritmos de aprendizado de máquina, sendo eles a Floresta Aleatória, a Regresão Logística, o *Naive Bayes* e o *Gradient Boosted Tree*. Nos testes de avaliação, utilizaram-se *pools* e mineradores diferentes dos utilizadas no conjunto de dados de treinamento. Antes de particionar o conjunto de dados em blocos de treinamento e validação, o conjunto de dados foi embaralhado para evitar qualquer elemento de viés ou padrões nos conjuntos de dados. O

[6]Disponível em https://minergate.com. Acessado em 13 de maio de 2020.

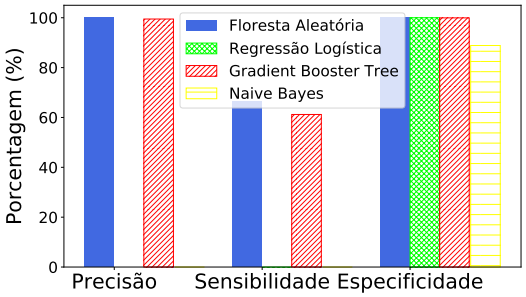
[7]Disponível em https://guiminer.org. Acessado em 13 de maio de 2020.

[8]O tráfego de mineração usado para avaliar os algoritmos de aprendizado de máquina se originam da execução dos aplicativos de mineração `cpuminer` e `xmrig`.

[9]Disponível em https://github.com/appneta/tcpreplay. Acessado em 13 de maio de 2020.



(a) Curva de características operacionais do receptor (ROC).



(b) Precisão, sensibilidade e especificidade dos algoritmos de classificação avaliados.

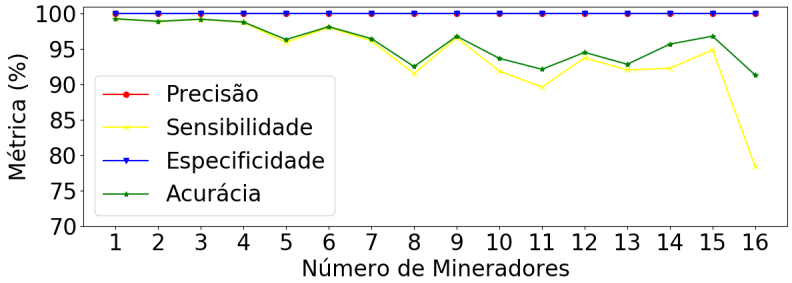
Figura 1. Avaliação dos algoritmos de aprendizado de máquina. a) O algoritmo de Floresta Aleatória (FA) apresenta uma Área Abaixo da Curva *AUC* de 0,97 e, assim, apresenta a melhor relação de compromisso entre sensibilidade e especificidade. b) O *Gradient Boosted Tree* (GB) também apresenta altas taxas de sensibilidade e especificidade, porém são inferiores às da Floresta Aleatória.

conjunto de dados passou pelo processo de *oversampling*, pois a técnica é frequentemente usada para balancear a quantidade de classes do conjunto de dados [Luengo et al. 2011]. O conjunto de dados criado possui abstrações de fluxo de rede, que são conjuntos de pacotes com a mesma quíntupla: IP de origem, porta de origem, IP de destino, porta de destino e protocolo de transporte. No total, cada amostra possui 46 características. Para evitar que os modelos aprendam os endereços *IP* e portas utilizadas no conjunto de dados utilizado para treinamento dos modelos, essas características são removidas na etapa de pré-processamento. Isso é feito para ocorrer a generalização dos modelos, caso contrário os modelos ficarão específicos para as *pools* de mineração utilizadas no conjunto de dados de treinamento.

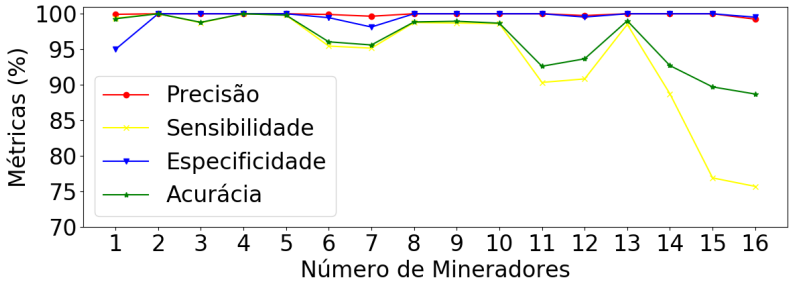
A Figura 1 mostra a curva de características operacionais do receptor ROC, a precisão, a sensibilidade e a especificidade de cada algoritmo de classificação testado. A curva ROC mede e especifica o desempenho dos algoritmos testados através do relacionamento entre a taxa verdadeiros positivos e falsos positivos em diversos pontos de corte na probabilidade de uma amostra pertencer a uma classe. É um método gráfico robusto e direto que permite estudar a variação da sensibilidade e especificidade para diferentes valores de corte. Foi constatado que os algoritmos Floresta Aleatória (FA) e *Gradient Boosted Tree* (GB) obtiveram resultados similares e superiores à Regressão Linear (RL) e *Naive Bayes* (NB). Dessa forma, FA e GB foram utilizados como modelos candidatos para o Super Aprendizado Incremental.

Avaliou-se também se o aumento na quantidade de mineradores na rede impacta no desempenho dos classificadores, então, nessa avaliação, a cada 30 minutos adicionou-se um novo minerador na rede.

A Figura 2 mostra que, à medida que mais mineradores são adicionados à rede, a precisão e a sensibilidade dos modelos não diminuem consideravelmente. Contudo, conforme o tempo passa, o desempenho diminui. Assim, os resultados demostram que aumentar o número de mineradores não interfere no desempenho, mas sim o tempo. Paralelamente, foi proposta uma modificação do *super learner* [Van der Laan et al. 2007]



(a) Precisão, sensibilidade, especificidade e precisão do modelo Floresta Aleatória para cenários com diferentes números de mineradores.



(b) As mesmas métricas foram utilizadas para o modelo *Gradient Boosted Tree* no cenário com diferentes números de mineradores.

Figura 2. Avaliação dos algoritmos de aprendizado de máquina de acordo com o crescimento do número de mineradores.

usando esses dois modelos pelos seguintes motivos: i) é desejado ter um desempenho igual ou superior ao dos algoritmos usados no *super learner*; ii) é importante o uso de algoritmos de aprendizado de máquina presentes na biblioteca MLlib [Meng et al. 2016] do Spark, pois obtêm-se melhores aproveitamentos da abstração de dados do Spark e a biblioteca não possui suporte para o aprendizado incremental. Assim, utilizou-se uma rede neural como o super modelo, pois suporta o aprendizado incremental. Para a avaliação da proposta comparou-se o resultado da floresta aleatória, do *gradient boosted tree* e do super aprendizado incremental no mesmo cenário de onze mineradores de criptomoeda e cinco *hosts*, reproduzindo tráfego regular. Foi medida a acurácia dos modelos a cada 30 minutos. Nessa avaliação a quantidade de mineradores foi estática, visto que no teste re-

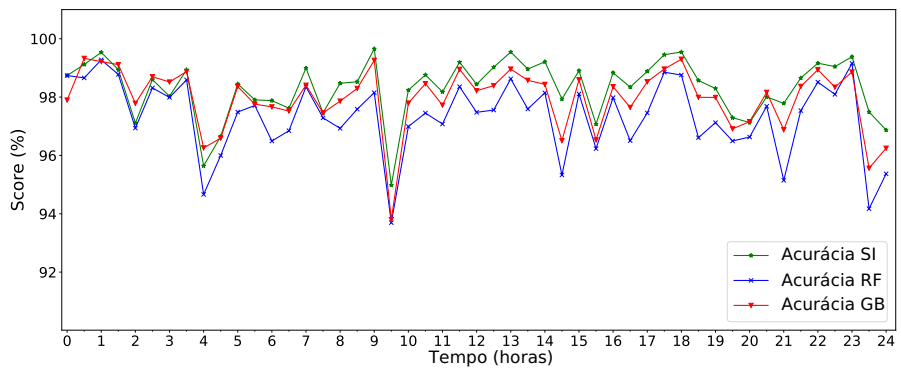


Figura 3. Acurácia do Super Aprendizado Incremental (SI), Floresta Aleatória (RF) e *Gradient Boosted Tree* (GB) em uma avaliação de um dia. A acurácia foi mensurada a cada 30 minutos.

presentado pela Figura 2, a variação no número de mineradores não altera o desempenho dos modelos.

É possível verificar na Figura 3 que em algumas fatias de tempo o modelo se comporta mal, mas melhora quando aprende com os novos dados. Ainda, em alguns momentos, acontece o desvio de conceito, mas o super modelo se adapta, pois é treinado de forma incremental com as melhores amostras selecionadas de acordo com sua probabilidade de classificação e, então, é garantido que o super modelo aprenderá com dados que possuem alta probabilidade de estarem corretos.

5. Trabalhos Relacionados

Os principais trabalhos relacionados protegem apenas um único *host* da mineração não autorizada de criptomoedas. Por outro lado, o objetivo central deste trabalho é prover essa proteção para toda a rede, utilizando *Big Data Analytics* e aprendizado de máquina. Tahir *et al.* propõem a ferramenta MineGuard para detectar em tempo real o comportamento de processos de mineração em máquinas virtuais utilizando contadores de desempenho HPC, permitindo, assim, rastrear com precisão as operações de mineração de baixo nível ou eventos dentro da CPU e GPU [Tahir et al. 2017]. O MineSweeper [Konoth et al. 2018] usa a URL do sítio *Web* como entrada para verificar se há algum *software* de mineração de criptomoedas oculto usando uma métrica para identificar funções criptográficas, medindo o número de operações executadas por um aplicativo. Wang *et al.* [Wang et al. 2018] propõem o SEISMIC (*SEcure In-lined Script Monitors for Interrupting Cryptojacks*), que é um programa que modifica automaticamente os programas binários Wasm para que eles se auto-projetem enquanto são executados, detectando atividades de mineração. Segundo os autores, os novos códigos de mineração estão utilizando WebAssembly (Wasm), que é uma nova linguagem de *bytecode* para navegadores Web que oferece computação mais rápida e eficiente do que as linguagens de *script* anteriores, como o JavaScript. Liu *et al.* [Liu et al. 2018] propõem o BMDetector, que utiliza redes neurais recorrentes para identificar mineração de criptomoedas em navegadores web. Os autores modificaram o código do kernel do Chrome[10] para implantar o detector de *cryptojacking*. Em Rodriguez *et al.* [Rodriguez e Posegga 2018], é criado um mecanismo de detecção de código de mineração em navegadores. Para isso, os autores utilizam o algoritmo de aprendizado de máquina *Support Vector Machine* (SVM). Os autores também utilizaram a lista Alexa com os TOP 330.500 sites mais visitados. Foram utilizados navegadores reais para visitar sites enquanto eram monitoradas chamadas da API relacionadas ao consumo de recursos de CPU do navegador.

6. Considerações Finais

Esse trabalho propôs o sistema MineCap [Neto et al. , Neto et al. 2019b, Neto et al. 2019a] para identificar e bloquear os fluxos em linha de mineração de criptomoedas em uma rede definida por *software*. É proposta também a técnica de super aprendizado incremental, em que os modelos de aprendizado de máquina, chamados de modelos candidatos, são treinados e as probabilidades das amostras serem normais ou mineração alimentam um super modelo que a partir dessas probabilidades realiza a classificação. Durante a avaliação dos modelos de aprendizado de máquina, constatou-se

[10]Disponível em https://www.google.com/intl/pt-BR/chrome/. Acessado em 13 de maio de 2020.

que os algoritmos de aprendizado de máquina Floresta Aleatória e *Gradient Boosted Tree* obtiveram melhores resultados em relação aos outros algoritmos e, então, foram incorporaram à técnica de super aprendizado incremental. Ambos apresentaram boa capacidade de generalização, com precisão e especificidade de cerca de 100% e sensibilidade de 66,5%. A técnica de super aprendizado incremental obteve bons resultados e demonstrou um correto funcionamento em relação a fatias de tempo com diferentes quantidades de mineradores na rede. A técnica aprende com novos dados, mantendo alto desempenho desde o início da execução e, em alguns casos, melhorando com o tempo. Os resultados da dissertação foram publicados em um congresso internacional (BRAINS 2019), um congresso nacional (SBSeg 2019), um minicurso (SBRC 2019), uma revista indexada (*Annals of Telecommunications*, Springer) e, ainda, há um artigo em processo de revisão no *Journal of Internet Services and Applications* (JISA, Springer).

7. Referências

[Andreoni Lopez et al. 2017] Andreoni Lopez, M., Sanz, I., Menezes, D., Duarte, O. e Pujolle, G. (2017). Catraca: uma ferramenta para classificação e análise tráfego escalável baseada em processamento por fluxo. *Salão de Ferramentas do XVII Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais-SBSeg*.

[Konoth et al. 2018] Konoth, R. K., Vineti, E., Moonsamy, V., Lindorfer, M., Kruegel, C., Bos, H. e Vigna, G. (2018). Minesweeper: An in-depth look into drive-by cryptocurrency mining and its defense. Em *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, páginas 1714–1730. ACM.

[Liu et al. 2018] Liu, J., Zhao, Z., Cui, X., Wang, Z. e Liu, Q. (2018). A novel approach for detecting browser-based silent miner. Em *2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*, páginas 490–497.

[Luengo et al. 2011] Luengo, J., Fernández, A., García, S. e Herrera, F. (2011). Addressing data complexity for imbalanced data sets: analysis of smote-based oversampling and evolutionary undersampling. *Soft Computing*, 15(10):1909–1936.

[Medeiros et al. 2019] Medeiros, D. S. V., Cunha Neto, H. N., Andreoni Lopez, M., Magalhães, L. C. S., Silva, E. F., Vieira, A. B., Fernandes, N. C. e Mattos, D. M. F. (2019). Análise de dados em redes sem fio de grande porte: Processamento em fluxo em tempo real, tendências e desafios. *Minicursos do Simpósio Brasileiro de Redes de Computadores-SBRC*, 2019:142–195.

[Meng et al. 2016] Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., Freeman, J., Tsai, D., Amde, M., Owen, S. et al. (2016). Mllib: Machine learning in apache spark. *The Journal of Machine Learning Research*, 17(1):1235–1241.

[Neto et al. 2019a] Neto, H. N. C., Fernandes, N. C. e Mattos, D. M. F. (2019a). Minecap: Online detection and blocking of cryptocurrency mining on software-defined networking. Em *1st Blockchain, Robotics and AI for Networking Security Conference*. DNAC.

[Neto et al.] Neto, H. N. C., Lopez, M. A., Fernandes, N. C. e Mattos, D. M. F. Minecap: super incremental learning for detecting and blocking cryptocurrency mining on software-defined networking. *Annals of Telecommunications*, páginas 1–11.

[Neto et al. 2019b] Neto, H. N. C., Lopez, M. A., Fernandes, N. C. e Mattos, D. M. F. (2019b). Um mecanismo de aprendizado incremental para detecção e bloqueio de mineração de criptomoedas em redes definidas por software. *XIX Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais-SBSeg*.

[Rodriguez e Posegga 2018] Rodriguez, J. D. P. e Posegga, J. (2018). Rapid: Resource and api-based detection against in-browser miners. Em *Proceedings of the 34th Annual Computer Security Applications Conference*, ACSAC '18, páginas 313–326, New York, NY, USA. ACM.

[Tahir et al. 2017] 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. Em *International Symposium on Research in Attacks, Intrusions, and Defenses*, páginas 287–310. Springer.

[The OpenFlow Consortium 2012] The OpenFlow Consortium (2012). *OpenFlow Switch Specification Version 1.3.0 (Wire Protocol 0x04)*. The OpenFlow Consortium.

[Van der Laan et al. 2007] Van der Laan, M. J., Polley, E. C. e Hubbard, A. E. (2007). Super learner. *Statistical applications in genetics and molecular biology*, 6(1).

[Wang et al. 2018] Wang, W., Ferrell, B., Xu, X., Hamlen, K. W. e Hao, S. (2018). Seismic: Secure in-lined script monitors for interrupting cryptojacks. Em *European Symposium on Research in Computer Security*, páginas 122–142. Springer.