Reprodutibilidade e Extensibilidade de *Datasets* de Rede: um estudo da replicação de *traces* de pacotes.

Luciano B. Fiorino, Maxwell E. Monteiro,
Cristina K. Dominicini e Gilmar L. Vassoler
Programa de Pós-graduação em Computação Aplicada (PPComp)
Campus Serra do Instituto Federal do Espírito Santo (IFES)
Serra, Brasil
{lbiancardi,maxmonte,cristina.dominicini,gilmarvassoler}@ifes.edu.br

João H. Corrêa e
Rodolfo S. Villaça
Programa de Pós-Graduação em Informática (PPGI)
Universidade Federal do Espírito Santo (UFES)
Vitória, Brasil
jhenrique@gmail.com, rodolfo.villaca@ufes.br

***Resumo*—Em geral, a aplicação de algoritmos de aprendizado de máquina em problemas de redes utilizam *datasets* gerados a partir de *traces* de pacotes. Entretanto, o processo de geração dos *datasets* atuais não segue critérios que permitam identificar as informações necessárias para a reprodução e extensão dos mesmos. Dessa forma, este trabalho realiza um estudo detalhado sobre formas e ferramentas para reprodução dos tráfegos de rede dos *datasets*. Diante dos problemas de reprodutibilidade identificados, propomos uma metodologia para geração de *datasets* de *traces* de pacotes, de forma a minimizar esses problemas, possibilitando a reprodução dos seus tráfegos de rede e estendê-los com novos dados.**

***Index Terms*—replicação de tráfego, *traces* de pacotes, computação em nuvem, *datasets*, aprendizado de máquina.**

I. Introdução

Nos últimos anos, técnicas de aprendizado de máquina têm sido muito utilizadas para solução de problemas complexos em redes, como previsão de tráfego, roteamento, classificação, controle de congestionamento, gerenciamento de recursos e segurança de rede [1]. Em geral, o aprendizado de máquina é aplicado para encontrar padrões ou tendências em um conjunto grande e representativo de dados.

Entretanto, o desenvolvimento de soluções que aplicam as técnicas de aprendizado de máquina à área de redes sofre com problemas de precisão na sua avaliação, comparação e implantação, devido à escassez de conjuntos de dados adequados [2]. Isso acontece porque não se conhece uma metodologia que forneça critérios para descrever o ambiente, planejar e gerar o tráfego, coletar métricas e finalmente gerar as amostras dos *datasets*, tornando difícil a comparação de diferentes conjuntos de dados e avaliar sua adequação para diferentes problemas de rede [1].

Em particular, o processo de criação de *datasets* na área de redes, tipicamente, utiliza métricas referentes aos fluxos de rede, obtidas, a exemplo, da análise de *traces* de pacotes em arquivos do tipo PCAP[1] [3].

No entanto, ao analisar os *datasets* disponibilizados na literatura, observa-se a falta de informações importantes no processo de criação, tais como: o cenário do experimento detalhado, os *softwares* e métodos utilizados na geração dos tráfegos de rede. A falta de informações torna difícil, e por vezes impossível, a reprodução dos *traces* de pacotes para verificação de resultados ou coleta de novas métricas, que permitiriam estender o conjunto de características do *dataset*. Como consequência, quando novas pesquisas necessitam de métricas não presentes nos *datasets* existentes, é necessário gerar suas próprias bases de dados para os experimentos, pois existem dificuldades em reproduzir *datasets* existentes.

Em se tratando de algoritmos de aprendizado de máquina, gerar a própria base de dados requer muito cuidado para que existam dados representativos para todos os casos, de modo a evitar erros de generalização nos modelos de aprendizado [4]. Assim, a reprodução de *datasets* confiáveis, gerando novas bases e/ou estendendo os mesmos com novas métricas, pode reduzir o risco de modelos de aprendizado tendenciosos. A reprodução de *datasets* também pode contribuir para comparações entre técnicas de aprendizado de máquina, além da aplicação dos tráfegos gerados originalmente em novos ambientes de interesse.

Para abordar o problema de reprodutibilidade e extensibilidade dos *datasets* de redes existentes, este trabalho propõe uma metodologia para geração de *datasets* de *traces* de pacotes, com critérios mínimos e um conjunto de etapas, que permite a reprodução e extensão dos mesmos em novas pesquisas. O ponto de partida para elaborar a metodologia será a avaliação de *datasets* disponíveis na literatura e de ferramentas abertas para replicação de *traces* de pacotes, com o objetivo de analisar a reprodutibilidade dos *datasets* avaliados para coletar novas métricas.

Como principais contribuições deste trabalho, pode-se destacar: (i) identificação das deficiências dos *datasets* em termos de reprodutibilidade e extensibilidade; (ii) identificação das deficiências das ferramentas de replicação de tráfegos de rede capturados em arquivos do tipo PCAP; (iii) proposta de metodologia com critérios e melhores práticas para geração de um *dataset* com informações suficientes para garantir sua reprodutibilidade e extensibilidade; e (iv) avaliação da replicação de *traces* de pacotes contidos em *datasets* da literatura com ferramentas de replicação abertas, de modo a coletar novas métricas de interesse, gerando novos conjuntos de dados e/ou estendendo os atuais.

[1] https://www.tcpdump.org/

Para avaliar a replicação *traces* de pacotes, este artigo apresenta um experimento baseado na arquitetura de nuvem *OpenStack* [5]. Este cenário foi escolhido como prova de conceito, pois as arquiteturas de nuvem possuem mecanismos nativos de monitoramento, os quais fornecem novas métricas não exploradas nos *datasets* de redes tradicionais (e.g., uso de CPU, consumo de memória e uso de disco). O experimento mostrou problemas de reprodutibilidade a partir da replicação de *traces* de pacotes, os quais enfatizam a necessidade da proposta de geração de *datasets* apresentada.

Este artigo está organizado da seguinte forma. A Seção 2 mostra os trabalhos relacionados. A Seção 3 apresenta a proposta de geração de *datasets*. A Seção 4 apresenta um estudo das capacidades das ferramentas de replicação disponíveis. A Seção 5 apresenta a avaliação da replicação de *traces* de pacotes. Por fim, a Seção 6 apresenta as conclusões.

II. Trabalhos Relacionados

Esta seção compara oito *datasets* da literatura em relação a um conjunto de características, importantes para garantir a reprodutibilidade e extensibilidade de *datasets* de redes. A Tabela I apresenta uma síntese da análise dos *datasets*, destacando suas características para extração dos tráfegos nos arquivos de captura e a replicação destes tráfegos.

O **detalhamento da topologia de rede** permite identificar os elementos participantes do cenário e a matriz de tráfego. Essas informações são essenciais para identificação da origem e destino dos *traces* de pacotes do *dataset*.

Arquivos PCAP são gerados a partir de capturas de *traces* de pacotes com ferramentas como o *Tcpdump* e *Wireshark*. Os arquivos PCAP são indispensáveis para a reprodutibilidade dos *traces* de pacotes de rede. Estes arquivos podem chegar a centenas de *gigabytes*, dependendo da duração do tráfego.

Em geral, o **MTU padrão** das redes é de 1500 *bytes*[2]. A existência de pacotes com tamanho maior que o MTU da rede utilizada é reflexo da captura ter sido feita com recursos de *Offloads*[3] habilitados na interface de rede. Esses pacotes tipicamente apresentam problemas para a replicação, pois as ferramentas de replicação devem ser capazes de fragmentar os pacotes para se adequar ao MTU, caso contrário haverá descarte de pacotes e o tráfego não será replicado fielmente. É importante enfatizar que nos tráfegos capturados com *Offloads* habilitados, o número de pacotes no arquivo PCAP difere do número real transmitidos na rede, impactando em métricas como o número de pacotes recebidos/enviados.

Os ***timestamps* de início e fim dos tráfegos** nas amostras do *dataset* informam o intervalo de tempo do respectivo tráfego, para o qual a amostra foi gerada. Estas são informações essenciais para extrair com exatidão o *trace* de pacotes da tupla TCP/UDP em cada amostra. Idealmente, a precisão dos *timestamps* de início e fim dos tráfegos deve ser igual à precisão dos *timestamps* dos pacotes no arquivo PCAP.

[2]https://www.rfc-editor.org/rfc/rfc894.txt

[3]https://www.kernel.org/doc/html/latest/networking/segmentation-offloads.html

TABELA I
Características dos Datasets Avaliados

Dataset	Topologia[1]	Arquivo PCAP	Pacotes MTU padrão	*Timestamp*[2] de início e fim	Dados de treino e teste	Dados rotulados
CIC-DDoS2019 [6]	✓	✓	✗	✗	✓	✓
CIC-IDS2017 [7]	✓	✓	✗	✗	✗	✓
CTU-13 [8]	✗	✓	✗	✓[3]	✓	✓
ISCX-IDS-2012 [2]	✓	✓	✓	✗[4]	✗	✓
UNSW-NB15 [9]	✓	✓[5]	✓	✗	✗[6]	✓
NDSec-1 [10]	(✓)	✓	✗	(✓)[7]	✗	✓
KDDCup'99 [11]	✗	✗	–	–	✓	✓
NSLKDD [12]	✗	✗	–	–	✓	✓

✓=atende, (✓)=atende parcialmente, ✗=não atende, "–"=não disponível

[1] Topologia e matriz de tráfego detalhada;
[2] *Timestamp* de início e fim dos tráfegos nas amostras;
[3] *Timestamp* de início e duração; [4] Não contém casas decimais;
[5] *Linux cooked-mode capture (SLL)*, não contém *Ethernet header;*
[6] A partir de separações do dados;
[7] Possui diferença de 03 casas decimais.

Dados de treino e teste são as partes do *dataset* utilizadas para o treinamento e, posteriormente, o teste dos algoritmos de classificação. Um *dataset* que não tenha dados distintos de treino e teste, necessita ser dividido manualmente, podendo gerar problemas no modelo devido a falta de aleatoriedade nos conjuntos de dados. Dessa forma o conjunto de treino pode conter dados não presentes no conjunto de teste e vice-versa, impactando no desempenho dos modelos de aprendizado [4].

Ter os **dados rotulados** significa que cada amostra do *dataset* foi classificada para cada tipo de tráfego, uma característica fundamental para o treinamento dos algoritmos de classificação. O rótulo possibilita filtrar as amostras de cada tipo de tráfego e juntamente com as características anteriores, extrair dos arquivos PCAP os respectivos tráfegos de interesse.

III. Proposta de Geração de Datasets Reproduzíveis e Extensíveis

Esta seção apresenta uma proposta para geração de *datasets* baseados em *traces* de pacotes, elaborada a partir da análise dos *datasets* e das ferramentas de replicação de *traces* de pacotes, de forma a assegurar a reprodutibilidade dos *traces* de pacotes. A Fig. 1 ilustra as etapas da metodologia.

1. Elaboração do Cenário: Alguns *datasets* não detalham a ***topologia*** de rede, matriz de tráfego, MTU e a taxa de transferência da rede. Dessa forma não é possível, em alguns casos, identificar com precisão os tráfegos a partir dos arquivos de *traces* de pacotes, o que inviabiliza a reprodução do cenário original. É necessário que a topologia e a matriz de tráfego sejam detalhadas para identificação dos *hosts* e reprodução dos tráfegos. A informação do ***MTU*** da rede é importante para reprodução do cenário, uma vez que as redes podem ter diferentes valores de MTU dependendo do protocolo de encapsulamento utilizado. A ***taxa de transferência*** da rede é necessária para evitar o problema de reproduzir tráfegos em redes de menor capacidade, o que gera padrões diferentes para estes mesmos tráfegos. Outro ponto muito importante são os ***hosts participantes do cenário***, em especial os de destino dos fluxos. Muitos *datasets* não fornecem informações dos servidores alvos dos tráfegos. Há casos, como no trabalho [13], em que é utilizado um ambiente privado, gerando problemas legais

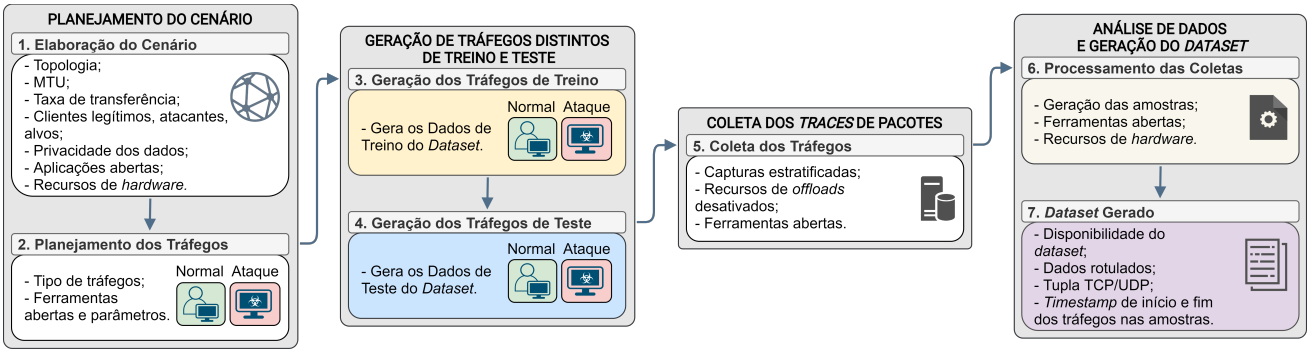

Fig. 1. Proposta de geração de *Dataset*.

quanto a ***privacidade dos dados*** trafegados e sua disponibilidade. A indisponibilidade das aplicações hospedadas inviabiliza a reprodução do cenário original e a replicação dos tráfegos. Para solucionar estes problemas, é recomendado o uso de ***aplicações abertas*** e dados anonimizados/fictícios em um cenário de teste para eliminar o problema de privacidade e permitir a reprodução do servidor alvo. As ***informações de hardware*** possibilitam reproduzir fielmente os servidores, o que permite comparar as métricas originalmente coletadas com outras não exploradas para um tráfego equivalente reproduzido e com a mesma carga de processamento.

2. Planejamento dos Tráfegos: Observa-se nos *datasets* pouco detalhamento nos métodos utilizados na geração dos tráfegos, onde muitas vezes apenas a ferramenta utilizada é citada. ***Detalhar a forma de geração dos tráfegos***, com informações do tipo do tráfego, da duração, do número de clientes e/ou atacantes do cenário, bem como das ***ferramentas e seus parâmetros de configuração***, permitem reproduzir um tráfego equivalente ao original, uma vez que não há ferramentas abertas capazes de replicar tráfegos capturados em modo *stateful* igual ao original. Dessa forma, encoraja-se que as ferramentas e os parâmetros de configuração utilizados para geração de tráfego devem ser abertos.

3. Tráfegos de Treino / 4. Tráfegos de Teste: De acordo com [4], a melhor forma de testar um modelo de aprendizado de máquina é a separação dos dados em conjuntos de treino e teste, de forma aleatória para evitar problemas na classificação devido à falta ou existência de poucas amostras de determinada classe no conjunto de treinamento. Ter ***tráfegos normal e de ataque distintos*** para geração de cada conjunto de treino e teste elimina este problema, pois haverá amostras representando todos os tipos de tráfegos em ambos os conjuntos.

5. Coleta dos Tráfegos: Os *datasets* da literatura são gerados após análise e processamento dos arquivos de *traces* de pacotes e, normalmente, disponibilizados como um único arquivo do tipo PCAP. Nos testes realizados, observou-se que um arquivo de captura com alguns *gigabytes* requer alto poder computacional para processar a separação de tráfegos, devido à grande quantidade de pacotes que precisam ser analisados. Uma solução para esse problema é a ***captura de forma estratificada***, onde, para cada tráfego, um arquivo de captura é gerado. Outro ponto está relacionado com os ***recursos de Offloads***, que devem estar desativados na interface de rede para que o arquivo de captura contenha os pacotes com tamanho dentro do MTU da rede utilizada no experimento, tipicamente no padrão de 1500 *bytes*. Também foi observado que algumas ferramentas, como em [14] e [15], não utilizam o formato aberto PCAP para os *traces* de pacotes, dessa forma é recomendada a utilização de ***ferramentas e padrões abertos*** para permitir a reprodução do experimento original.

6. Processamento das Coletas: Para ser possível reproduzir as amostras de um *dataset*, é necessário que o ***método de geração dessas amostras seja detalhado*** e as métricas de rede (características) as quais compõem o *dataset* sejam identificadas. Além do método de geração, os seguintes pontos devem ser considerados: as ***ferramentas de análise***, que devem estar disponíveis para utilização; e os ***recursos de hardware*** que foram necessários para processamento, uma vez que processar grandes arquivos PCAP requer maior poder computacional.

7. *Dataset* Gerado: O conjunto de dados gerado ***deve estar disponível***, e além de ter seus ***dados devidamente rotulados***, deve conter: i) conjuntos distintos de treino e teste para avaliação dos modelos de aprendizado de máquina, conjuntos estes gerados a partir dos tráfegos de treino e teste; ii) a ***tupla TCP/UDP*** nas amostras, facilitando a identificação da matriz de tráfego; iii) os ***timestamps*** de início de fim do tráfego nas amostras, com a mesma precisão dos pacotes capturados, para se extrair com exatidão os *traces* de pacotes referentes às amostras nos arquivos de captura.

A hipótese deste trabalho é que *datasets* gerados levando em consideração as informações descritas nas etapas acima, facilitariam a reprodutibilidade de seus *traces* de pacotes, consequentemente permitindo a extensão dos *datasets* por meio de coletas de novas características.

IV. Estudo das Capacidades das Ferramentas de Replicação Disponíveis

Esta seção avalia doze ferramentas de código aberto, identificadas na literatura, capazes de replicar tráfegos de rede a partir de *trace* de pacotes capturados. A Tabela II sumariza a avaliação realizada, considerando as seguintes características: modo de tráfego, suporte a arquivos PCAP, suporte a pacotes maiores que o MTU padrão, replicação com preservação do

timestamp, grau de maturidade e sua disponibilidade. Algumas ferramentas não foram testadas devido à indisponibilidade para *download*, exceto a TRex, que não permite replicar tráfegos respeitando os intervalos dos pacotes. Vale ressaltar que este trabalho não teve acesso a soluções comerciais com suporte a arquivos PCAP (e.g., Xena Networks Vulcan, Burp Suite).

Tcpreplay/Tcpliveplay: Uma suíte de ferramentas abertas composta por vários utilitários para manipular e replicar tráfegos de rede contidos em arquivos PCAP. O utilitário ***Tcpreplay*** funciona na Camada 2 da pilha TCP/IP e possui um parâmetro multiplicador, que permite enviar os pacotes respeitando os intervalos de tempo entre eles. O tamanho do pacote é limitado ao MTU da interface de rede. Já o utilitário ***Tcpliveplay*** permite replicar o tráfego de um arquivo PCAP em modo *stateful*, porém este utilitário não suporta os ajustes do Tcpreplay, e com a limitação de enviar todos os pacotes na mesma conexão e porta de origem.

GopherCap: Uma recente ferramenta de código aberto para replicação de tráfego em arquivos PCAP baseada na biblioteca GoPacket. Ela permite reproduzir o tráfego preservando o *timestamp* entre cada pacote. Também é capaz de reproduzir vários arquivos PCAP em uma única execução. Assim como o Tcpreplay, o tamanho do pacote é limitado ao MTU da interface de rede da máquina que está realizando a replicação.

Moogen: Um gerador de pacotes programável baseado no *framework libmoon*, que permite desenvolver aplicações DPDK[4] e gerar tráfegos acima dos 100Gbps. O controle do gerador é feito com *scripts* na linguagem Lua e a ferramenta disponibiliza o código para replicação de pacotes em arquivos PCAP respeitando os *timestamps* entre os pacotes. Apresenta limitações no suporte a pacotes maiores que o MTU padrão.

Pktgen-DPDK: Um gerador de tráfego de código aberto baseado na biblioteca DPDK e capaz de gerar tráfegos de 10Gbps. Com o acesso direto a interface de rede, a ferramenta é capaz de enviar pacotes acima do MTU padrão. Também é capaz de enviar pacotes contidos em arquivos PCAP, permitindo controlar a taxa de envio como um percentual da taxa da interface de rede. Sua desvantagem é não suportar o envio dos pacotes respeitando o *timestamp* de captura.

DPDK-burst-replay: Outra ferramenta baseada nas bibliotecas DPDK para replicação de tráfegos contidos em arquivos PCAP, com foco no envio de pacotes em interfaces de alto desempenho. Como desvantagem, não permite o envio de pacotes respeitando os intervalos de tempo entre eles.

GoReplay: Uma ferramenta com versões aberta e comercial para testes de aplicações *Web*. A ferramenta permite capturar e replicar tráfegos HTTP de modo *stateful*, extraindo as requisições HTTP do tráfego original e as replicando em novas conexões respeitando o intervalo de tempo das requisições. Possui suporte a arquivos PCAP e, em sua versão comercial, a captura e replicação das sessões TCP é precisa. Limita-se a extrair requisições HTTP de arquivos PCAP.

[4]https://www.dpdk.org/

TABELA II
CARACTERÍSTICAS DAS FERRAMENTAS DE REPLICAÇÃO DE TRÁFEGO

Ferramenta	Modo de tráfego	Arquivo PCAP	Pacotes maiores MTU padrão[1]	Preserva *timestamp* do pacote	Maturidade	Testada
Tcpreplay [16]	*stateless*	✓	(✓)[2]	✓	estável	sim
Tcpliveplay [16]	*stateful* [3]	✓	(✓)[2]	✗	estável	sim
GopherCap [17]	*stateless*	✓	(✓)[2]	✓	estável	sim
Moogen [18]	*stateless*	✓	(✓)[4,8]	✓[5]	*dev*	sim
Pktgen-DPDK [19]	*stateless*	✓	✓[8]	✗	estável	sim
DPDK-burst-replay [20]	*stateless*	✓	–[8]	✗	*dev*	sim
GoReplay [21]	*stateful* [6]	✓	–	✗	estável	sim
Socketreplay [22]	*stateful*	✓	–	–	*dev*	não
TCPOpera [23]	*stateful*	✓	–	–	–	não
TCPTivo [14]	*stateful*	✗[7]	✗	–	*dev*	não
DETER [15]	*stateful*	–	–	✓	–	não
TRex [24]	*stateless*/ *stateful*	✓[9]	✓[8]	✗	estável	não

✓=suporta, (✓)=suporta parcialmente, ✗=não suporta, "–"=não disponível, *dev*=em desenvolvimento

[1] Pode gerar problemas de fragmentação e descarte de pacotes;
[2] Limitado ao MTU da interface; [3] Com limitações, usa a mesma porta de origem;
[4] Com limitações, truncou pacotes; [5] Disponibiliza código de replicação;
[6] Somente tráfego HTTP; [7] Necessita de conversão para para formato DAG;
[8] DPDK requer *hardware* compatível e as ferramentas precisam suportar *jumbo frame*;
[9] Replicação de PCAPs em modo *stateless*.

V. AVALIAÇÃO DA REPLICAÇÃO DE TRACES DE PACOTES

O experimento descrito nesta seção tem como objetivo demonstrar as limitações existentes nos *datasets* da literatura e nas ferramentas abertas para replicação de *traces* de pacotes. O intuito do experimento é reproduzir o trabalho de [25], porém utilizando tráfegos de rede de um *dataset* de *trace* de pacotes disponível na literatura, e assim gerar um novo conjunto de dados a partir da telemetria de nuvem, para aplicação em algoritmos de aprendizado de máquina.

A. Descrição do Cenário do Experimento

O experimento realizado replicou os *traces* de pacotes, descritos na Seção V-C, em um ambiente de nuvem OpenStack [5]. Durante a replicação dos tráfegos, foram coletadas várias métricas pelo sistema de telemetria da nuvem. A Fig. 2 ilustra o cenário do experimento.

Conforme mostrado na Fig. 2, no plano *HOSTS VIRTUAIS* encontram-se a *VM Replay*, responsável por replicar os *traces* de pacotes, e o *Web Server Apache*, que representa a máquina alvo do tráfegos, estes dois *hosts* estão conectados por uma rede virtual denominada *Cloud Network*. O plano *INFRAESTRUTURA OPENSTACK* ilustra os componentes utilizados para coletar as métricas da nuvem. O módulo Ceilometer[5] coleta automaticamente diferentes tipos de métricas, que podem ser configuradas, armazenando-as no banco de dados de séries temporais Gnocchi[6]. Por fim, o plano *HARDWARE* representa os equipamentos físicos utilizados para suportar o ambiente de nuvem. Um dos equipamentos representa o *Nó de Controle da Nuvem* e o outro o *Nó de Computação*, onde máquinas virtuais são executadas. O cenário é composto por dois servidores PowerEdge T640 com 96 GB RAM e CPU Intel Xeon Silver 4114 de 2.2 GHz.

[5]https://docs.openstack.org/ceilometer/victoria/
[6]https://gnocchi.xyz/

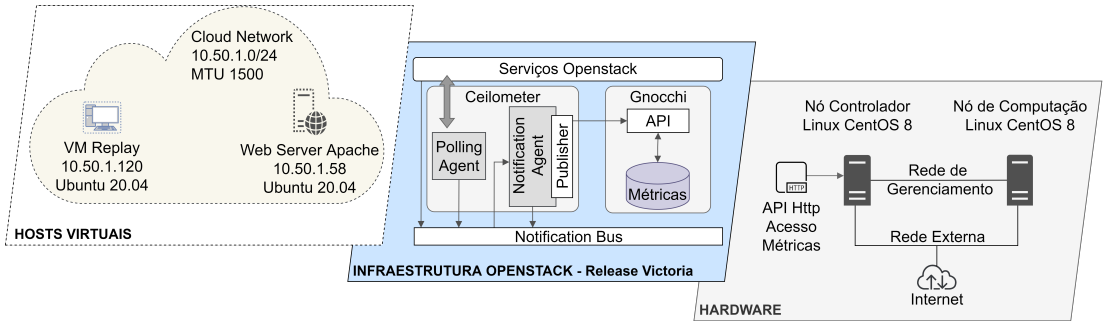

Fig. 2. Cenário do experimento.

O experimento teve como premissa preservar os *timestamps* dos pacotes durante a replicação, de modo a enviar os pacotes nos mesmos intervalos de tempo, tornando o tráfego replicado o mais próximo do original. Para isso foram escolhidas as ferramentas Tcpreplay, Moongen, GopherCap e GoReplay.

Para fins de comprovação, os códigos utilizados, os arquivos PCAP com os tráfegos de rede selecionados e preparados para replicação, e o *dataset* gerado no experimento deste trabalho estão disponíveis em: https://lbfiorino.github.io/.

B. Novas Métricas para os Tráfegos de Interesse do Dataset

Como dados de interesse para o experimento, foram escolhidas métricas que variam de acordo com a carga de utilização das instâncias da nuvem em função da replicação do tráfego. Ao todo, foram selecionadas doze métricas relacionadas às instâncias, sendo elas: *cpu usage, memory usage, memory swap in, memory swap out, disk device read requests, disk device write requests, disk device read bytes, disk device write bytes, network incoming bytes, network outgoing bytes, network incoming packets, network outgoing packets*.

A coleta dos dados foi realizada por uma aplicação Python através da API do serviço Gnocchi e de forma *offline*, após a replicação dos tráfegos. Para isso, os relógios dos *hosts* foram sincronizados para a coleta ser realizada no intervalo de tempo entre o início e o fim da replicação. A nuvem utilizada gerava as métricas a cada 5 segundos. Assim, cada amostra do *dataset* gerado corresponde a coletas das métricas a cada 5 segundos.

C. Seleção e Preparação dos Tráfegos para Replicação

Dentre os *datasets* avaliados, o *NDSec-1* [10] foi escolhido por disponibilizar as seguintes informações, as quais permitem extrair tráfegos de interesse do arquivo PCAP satisfatoriamente: i) detalhes da topologia de rede; ii) amostras rotuladas em tráfego normal e de ataque; iii) tráfego normal e de ataque para um mesmo destino, identificado com análise da tupla TCP nas amostras; iv) intervalos de tempo dos tráfegos nas amostras; v) arquivos PCAP menores, que reduziram o tempo de processamento. Apesar de não disponibilizada, foi identificado nos *traces* o uso da aplicação *web* aberta DVWA[7], possibilitando a reprodução do servidor *web*.

[7]https://dvwa.co.uk/

O vetor de ataque SYN-Flood é um dos dominantes na Internet [26]. Diante disso, foram escolhidos os tráfegos de ataque SYN-Flood e o tráfego normal HTTP do *dataset*, ambos para o mesmo destino. A escolha do cenário de tráfegos contidos no *dataset NDSec-1* foi baseada nos seguintes fatores:

1) ***Menor custo computacional para processamento do arquivo PCAP***. Para cada amostra do *dataset*, é necessário um processo para extrair do arquivo PCAP os pacotes do fluxo referente à amostra, pois a informação de início e fim do fluxo está na amostra. Para isso, foi utilizada a ferramenta *Tshark*[8], que não suporta *multithreading*. A diferença de três casas decimais na precisão do *timestamp* entre as amostras e os pacotes capturados agrava o custo computacional devido a passos adicionais no processo de extração, que foi possível combinando a informação do *timestamp* com o número do *frame* do pacote no arquivo PCAP. Por estes motivos, foi escolhido o arquivo PCAP do cenário *botnet* do *dataset*.

2) ***Pacotes adequados ao MTU padrão***. Após análise dos *traces* de pacotes do cenário *botnet*, foram encontrados tráfegos de ataque SYN-Flood e tráfego normal HTTP adequados ao MTU padrão, os quais puderam ser utilizados no experimento.

Uma vez escolhidos os tráfegos, os mesmos foram extraídos do arquivo PCAP original, gerando um arquivo PCAP para cada tráfego de forma estratificada. A extração dos pacotes de cada tráfego foi feita de acordo com a tupla TCP, o *timestamp* de início e fim, e o rótulo contido em cada amostra.

Por fim, os pacotes extraídos precisaram ser modificados para se adequar à arquitetura de replicação. As seguintes alterações foram realizadas nos pacotes: (i) alteração da tupla TCP (endereço MAC e IP de origem e destino) para ambos os tráfegos, normal e ataque; e (ii) no tráfego normal, os campos *host* e *referer* do protocolo HTTP foram alterados para o *host* do servidor *web* do novo cenário. Por fim, os *checksums* IP/TCP foram recalculados. Para este processo foram utilizadas as ferramentas *scapy*[9] e *tcprewrite* [16].

D. Replicação dos tráfegos

Os tráfegos de interesse foram replicados em momentos diferentes, portanto, sem concorrência dos tráfegos. As cap-

[8]https://www.wireshark.org/
[9]https://scapy.net/

turas dos tráfegos replicados foram feitas no *host Nó de Computação* apresentado na Fig. 2.

A ferramenta GoReplay foi utilizada para replicar o tráfego *stateful* HTTP. As requisições foram extraídas do arquivo PCAP e salvas em um novo arquivo, no padrão da ferramenta. A replicação foi realizada de forma esperada, com duração de 0,014% superior ao original. Enfatiza-se que o tráfego replicado não envolvia processos de autenticação sofisticada.

O tráfego SYN-Flood foi replicado utilizando as ferramentas Tcpreplay, MoonGen e GopherCap, devido a este tipo de ataque não requerer o estabelecimento da conexão, sendo necessário apenas o envio do pacote SYN ao *host* de destino. As ferramentas apresentaram bons resultados, onde os tráfegos replicados tiveram durações superiores ao original de 0,013%, 0,351% e 0,672% respectivamente.

Apesar dos tráfegos selecionados terem sido replicados com sucesso, o volume pequeno de tráfego não alterou consideravelmente as métricas escolhidas. Dessa forma, é necessário tráfegos massivos para possibilitar uma análise comparativa entre os *datasets* nos algoritmos de aprendizado de máquina.

Importante notar que o experimento demonstrou a possibilidade de replicação de tráfegos específicos, tais como SYN-Flood e HTTP simples. No entanto, há carência de ferramentas abertas para replicação satisfatória de outros tipos de tráfegos, bem como a necessidade de uma metodologia para gerar novos *datasets*, com as informações para facilitar a extração e reprodução dos fluxos de interesses.

VI. Conclusão

Este trabalho analisou as deficiências, tanto nos *datasets* da literatura quanto nas ferramentas de replicação de *traces* de pacotes, para gerar novas bases de dados e/ou estender as atuais por meio de coleta de métricas de nuvem. A avaliação identificou problemas desde a falta de informações a respeito dos experimentos dos *datasets* quanto deficiências das ferramentas abertas para replicação de tráfego *stateful*. A indisponibilidade das ferramentas e parâmetros utilizados na geração dos tráfegos originais gera a necessidade de novas ferramentas de replicação. Diante dessas limitações, concluise que não é possível reproduzir e estender satisfatoriamente os *datasets* de redes analisados, mas que a metodologia proposta avança em identificar os passos para permitir que novos *datasets* sejam gerados para preencher esta lacuna.

Como trabalhos futuros, pretendemos gerar um *dataset* de rede próprio, utilizando a metodologia proposta, com tráfegos massivos, os quais possam ser replicados ou reproduzidos com as ferramentas atuais de acordo com suas especificidades. Isso permitirá realizar o ciclo completo da metodologia para gerar o *dataset*, reproduzi-lo e estendê-lo com novas métricas. Por fim, esperamos avaliar a relevância de novas métricas na solução de problemas usando técnicas de aprendizado de máquina.

Agradecimentos

Agradecemos ao IFES pelo apoio financeiro via Programa Institucional de Apoio à Pós-graduação Stricto Sensu (Propós) e à FAPES pelo suporte ao laboratório CIDIG do Centro de Pesquisa, Inovação e Desenvolvimento do Espírito Santo.

Referências

[1] R. Boutaba *et al.*, "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities," *Journal of Internet Services and Applications*, vol. 9, no. 1, p. 16, 2018.

[2] A. Shiravi *et al.*, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Computers & Security*, vol. 31, no. 3, pp. 357–374, 2012.

[3] M. A. Ferrag *et al.*, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *Journal of Information Security and Applications*, vol. 50, p. 102419, 2020.

[4] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, 2019, ch. 1, pp. 24–31.

[5] O. I. F. OpenStack, "Openstack - open source cloud computing platform software," https://www.openstack.org/software/, acesso em: 25 fev. 2021.

[6] I. Sharafaldin *et al.*, "Developing realistic distributed denial of service (ddos) attack dataset and taxonomy," in *ICCST*, 2019, pp. 1–8.

[7] ——, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *ICISSP*, 2018, pp. 108–116.

[8] S. García *et al.*, "An empirical comparison of botnet detection methods," *Computers & Security*, vol. 45, pp. 100–123, 2014.

[9] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *MilCIS*, 2015, pp. 1–6.

[10] F. Beer *et al.*, "A new attack composition for network security," in *10. DFN-Forum Kommunikationstechnologien*, P. Müller, B. Neumair, H. Raiser, and G. Dreo Rodosek, Eds. Bonn: Gesellschaft für Informatik e.V., 2017, pp. 11–20.

[11] I. University of California, "Kdd cup 1999 data," http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, 1999, acesso em: 20 fev. 2021.

[12] M. Tavallaee *et al.*, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009.

[13] F. Beer and U. Bühler, "Feature selection for flow-based intrusion detection using rough set theory," in *2017 IEEE 14th ICNSC*, May 2017, pp. 617–624.

[14] W.-c. Feng *et al.*, "Tcpivo: A high-performance packet replay engine," in *Proceedings of the ACM SIGCOMM Workshop on Models, Methods and Tools for Reproducible Network Research*. New York, NY, USA: ACM, 2003, p. 57–64.

[15] Y. Li, R. Miao, M. Alizadeh, and M. Yu, "DETER: Deterministic TCP replay for performance diagnosis," in *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*. Boston, MA: USENIX Association, feb 2019, pp. 437–452.

[16] AppNeta, "Tcpreplay - pcap editing and replaying utilities," https://tcpreplay.appneta.com/, 2020, acesso em: 20 jan. 2021.

[17] Stamus Networks, "Gophercap: Accurate, modular, scalable pcap manipulation tool written in go." https://github.com/StamusNetworks/gophercap, 2020, acesso em: 08 fev. 2021.

[18] P. Emmerich *et al.*, "Moongen: A scriptable high-speed packet generator," *Proceedings of the 2015 Internet Measurement Conference*, Oct 2015.

[19] K. Wiles, "Pktgen - traffic generator powered by dpdk," https://github.com/pktgen/Pktgen-DPDK, 2020, acesso em: 01 fev. 2021.

[20] J. Ribas, "Dpdk burst replay tool," https://github.com/FraudBuster/dpdk-burst-replay, 2019, acesso em: 01 fev. 2021.

[21] Leonid Bugaev, "Goreplay," https://goreplay.org/, 2020, acesso em: 26 mai. 2021.

[22] Y.-D. Lin *et al.*, "Low-storage capture and loss recovery selective replay of real flows," *IEEE Communications Magazine*, vol. 50, no. 4, pp. 114–121, 2012.

[23] S.-S. Hong and S. F. Wu, "On interactive internet traffic replay," in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2005, pp. 247–264.

[24] Cisco, "Cisco t-rex," https://trex-tgn.cisco.com/, 2021, acesso em: 11 jun. 2021.

[25] J. H. Corrêa *et al.*, "Ml-based ddos detection and identification using native cloud telemetry macroscopic monitoring," *Journal of Network and Systems Management*, vol. 29, no. 2, pp. 1–28, 2021.

[26] T. C. B. Cloudflare, "Network-layer ddos attack trends for q4 2020," https://blog.cloudflare.com/network-layer-ddos-attack-trends-for-q4-2020/, 2021, acesso em: 22 mar. 2021.