Seleção *Online* de *Features* em *Streaming* Baseada em Alpha-Investing$^+$ para Dados de Ataques DDoS

Alissar Ali Moussa[1], **Michele Nogueira**[1], **André Luiz P. Guedes**[1]

[1]Centro de Ciência de Segurança Computacional (CCSC)
Universidade Federal do Paraná (UFPR)

{aamoussa,michele,andre}@inf.ufpr.br

Abstract. *Distributed Denial of Service (DDoS) attacks are a significant security threat, because they are financially harmful to companies and organizations as they affect services' availability. As attacks become more complex, traditional intrusion detection systems are limited to attacks in advanced stage or they can only raise an alarm when target is already damaged. However, the constant search for efficient detection systems includes selecting the most relevant features, i.e., the characteristics that will be used for detection. The more associated to the attacks' operation mode, the more relevant is the feature. Nevertheless, as attacks get more soffisticated, it is important to consider an online approach to feature selection. Thus, the feature selection algorithm must be able to work with streaming data and features. In this work, we present a feature selection method for streaming DDoS attacks data. The method is based on the $alpha$-investing algorithm and we name it $alpha$-investing$^+$, which is able to sequentially add and test features. CTU-13 botnet dataset and CICIDS2017 attacks dataset are used to verify the efficiency of our method. A qualitative analysis is present and it suggests that this technique can rapidly select features in an ongoing attack.*

Resumo. *O ataque de negação de serviço distribuído (DDoS) é uma ameaça significativa de segurança, pois compromete financeiramente empresas e organizações ao afetar a disponibilidade de seus serviços. Com a sofisticação das técnicas utilizadas para gerar esses ataques, os sistemas de detecção tradicionais se limitam a ataques DDoS em estágios avançados ou quando o alvo está comprometido. Entretanto, a busca por maior eficiência nesses sistemas de detecção é uma constante e passa pelo problema de selecionar as features mais relevantes, ou seja, as características a serem utilizadas como base para a detecção. Quanto mais associada com o modo de operação do ataque, mais relevante é a feature. No contexto de ataques DDoS, é interessante que a seleção de features consiga funcionar sob dados e/ou features em streaming. Desta forma, apresentamos um método para seleção de features em streaming na detecção de ataques DDoS. Ele está embasado na nossa adaptação do algoritmo α-investing, chamado de α-investing$^+$, o qual atua de forma sequencial na adição e nos testes das features. O método é aplicado nas bases de dados CTU-13 e CICIDS2017 e através de uma análise qualitativa, sugere-se que a técnica pode selecionar rapidamente features relevantes de um ataque em curso.*

1. Introdução

Um ataque volumétrico de negação de serviço distribuído (DDoS - *Distributed Denial of Service*) ocorre tradicionalmente quando vários dispositivos finais infectados (*a.k.a.*, bots)

realizam ataques coordenados por uma máquina mestre contra um alvo a fim de afetar a disponibilidade de seus recursos. Esse tipo de ataque representa uma ameaça significativa às empresas e às organizações, pois estas são afetadas financeiramente. Além do impacto financeiro, os ataques DDoS também podem ter motivações políticas, como foi o caso do ataque DDoS executado pelos ciberativistas da Anonymous contra o Banco Original em maio de 2017, o qual tornou indisponível o site da instituição. No primeiro trimestre de 2018, houve 1,4 milhão de ataques DDoS no mundo inteiro, com duração média de 14 horas e pico de tráfego de dados registrado em 153 Gbps [Borini 2018].

Devido à sofisticação das técnicas utilizadas para gerar ataques DDoS, as ferramentas tradicionais de detecção só conseguem reportá-los quando atingem estágios avançados e o alvo já está comprometido, sendo tarde demais para uma ação eficiente contra esses ataques, conforme demonstrado através do modelo em [Santos et al. 2017]. Assim, é cada vez mais necessário que as técnicas de detecção sejam capazes de identificar comportamentos suspeitos antes da sobrecarga do alvo, para que um ataque em potencial possa ser impedido ou que seja possível mitigar eficientemente os seus efeitos. Além disso, os ataques DDoS volumétricos estão relacionados a um volume de dados muito grande, com alta dimensionalidade, que dificulta a análise em tempo real. A Maldição da Dimensionalidade, descrita por Bellman em 1957, é um problema causado pelo aumento exponencial de volume associado com a adição de dimensões extras em um espaço Euclidiano [Bellman 1966]. Um dos meios empregados com frequência para solucionar ou mitigar esse problema, tratando-se de dados reais, é a seleção de *features*. *Feature* é sinônimo de dado de entrada ou atributo [Guyon and Elisseeff 2008]. Em um conjunto de dados de diagnósticos médicos, por exemplo, as *features* podem ser os sintomas de cada doença, isto é, o conjunto de atributos que descrevem uma determinada enfermidade. Em nosso trabalho, as *features* são os atributos presentes em dados de captura de pacotes, tais como endereço IP de origem, protocolo, porta de origem, entre outros.

Tradicionalmente, os algoritmos de seleção de *features* consideram que o espaço de dados e de *features* já são conhecidos [Li et al. 2017]. Porém, tratando-se de aplicações do mundo real nem sempre os conjuntos completos de dados e/ou de *features* estão disponíveis, impossibilitando o uso das técnicas tradicionais. Assim, é preciso que um algoritmo de seleção de *features* para dados dessa natureza consiga executar a tarefa de seleção sem conhecer o espaço total de *features* ou de dados [Hu et al. 2018], o que é um grande desafio e um relevante problema de pesquisa. Nos últimos anos, o interesse pela seleção de *features* em *streaming* tem aumentado devido ao crescimento das aplicações de *big data* e que geram grande volume de dados. Diversos trabalhos propõem técnicas de seleção de *features* em *streaming* fundamentados em conceitos como a regularização de esparcidade [Jialei Wang et al. 2014] e a probabilidade condicional [Wu et al. 2013]. Entretanto, as publicações referentes aos cenários de ataques DDoS em sua maioria abordam o problema da seleção de *features* de maneira *offline*, utilizando técnicas tradicionais, como os métodos de filtro [Osanaiye et al. 2016] e de *clustering* [Idhammad et al. 2018].

Neste trabalho, apresentamos um método de detecção online de *features* para dar suporte à detecção ou predição de ataques DDoS. Este método está fundamentado na nossa adaptação do algoritmo α-investing [Zhou et al. 2005], chamada de α-investing^{+}, para seleção de *features* em *streaming*. O método é *online* e constrói um conjunto de *features* selecionadas no momento de sua descoberta, após uma avaliação de independência

entre a *feature* recém-descoberta e as já existentes no conjunto. Cada variável candidata a *feature* selecionada é examinada apenas uma vez, e a verificação constante de independência entre as variáveis reduz a possibilidade de haver *features* redundantes no conjunto selecionado. O algoritmo foi adaptado para lidar com dados de ataques de DDoS, que são não-paramétricos e categóricos, sugerindo a necessidade de testes que atendam a essas características. Por isso, foi utilizado o Teste de Qui-Quadrado para Independência entre as variáveis com o valor-p estimado a partir de uma simulação de Monte Carlo.

O método proposto foi avaliado sob as bases de dados CTU-13 [García et al. 2014] e CICIDS2017 [Sharafaldin et al. 2018]. A primeira contém dados de tráfego de *botnets* testadas na Czech Technical University (CTU). Os dados estão divididos em 13 capturas, referidas pelos autores como cenários, e neste trabalho são utilizados os cenários cujos bots se coordenam e geram ataques DDoS. Já a segunda base foi construída a partir de uma série de diversos ataques, em um cenário experimental criado na University of New Brunswick, e entre esses ataques está um DDoS. Uma análise qualitativa é apresentada, confrontando os resultados apresentados pelo método com o conhecimento dos autores sobre a base. Os resultados sugerem que o método consegue selecionar as *features* de maneira rápida, mas que pode apresentar resultados inconclusivos dependendo do cenário ao qual o algoritmo é submetido. Em um cenário mais diverso, com tráfego de dados variando entre períodos com mais ou menos volume de dados, as *features* selecionadas são mais fieis ao contexto do ataque analisado.

Este trabalho está organizado da seguinte forma. A Seção 2 discute os trabalhos relacionados e o estado da arte. A Seção 3 descreve o método proposto fundamentado no algoritmo α-investing adaptado para dados de ataques DDoS. A Seção 4 expõe o método, a sua implementação e discute os resultados obtidos. A Seção 5 conclui o trabalho.

2. Trabalhos relacionados

Os métodos tradicionais de seleção de *feature* podem ser classificados em três categorias: filtro, *wrapper* e *embedded* [Guyon and Elisseeff 2003]. As técnicas de filtro selecionam subconjuntos de *features* na etapa de pré-processamento, sem auxílio de um classificador. A avaliação para cada subconjunto envolve diversas métricas como ganho de informação, consistência e relevância [Eskandari and Javidi 2016]. Os métodos *wrapper* utilizam classificadores para avaliar os subconjuntos de *features* gerados. Por fim, a técnica *embedded* atrela o processo de seleção de *feature* ao treinamento do modelo.

Como os métodos citados acima consideram que ambos os espaços de *feature* e de dados já são conhecidos previamente, há a necessidade de técnicas que consigam tratar dados gerados em tempo real (*online*) e de maneira contínua. A literatura menciona técnicas *online* seleção de *features* referentes a dados em streaming e a *features* em *streaming*. No primeiro caso, considera-se que o conjunto de *features* é conhecido previamente e o espaço de dados é atualizado com a chegada de novas entradas. Já no segundo caso, o tamanho do conjunto de *features* é desconhecido e potencialmente infinito [Li et al. 2017]. Neste trabalho, o foco será nos algoritmos que tratam de *features* em *streaming*.

Em [Perkins et al. 2003] é proposta uma técnica de seleção incremental de *features* chamada *Grafting* que, embora não pertença a mesma categoria dos algoritmos discutidos neste trabalho, é precursora dos métodos que lidam com dados e *features* em

streaming. Segundo os autores, o *Grafting* constrói gradativamente do conjunto de *features* enquanto treina um modelo de predição usando gradiente descendente. Contudo, essa técnica necessita que um parâmetro de regularização λ seja definido previamente para determinar quais *features* serão selecionadas a cada iteração. Por consequência, requer-se um conhecimento prévio de todo o espaço de *features*, impossibilitando o uso desse método para selecionar *features* em *streaming*.

[Wu et al. 2013] apresentam o *Fast-OSFS*, método que analisa a relevância das *features* e remove *features* redundantes dentre as já selecionadas. Baseando-se no conceito de probabilidade condicional, o algoritmo determina se duas *features* x_1 e x_2 têm forte relevância, fraca relevância ou irrelevância entre si. Seus resultados são comparados com o α-investing, técnica na qual se baseia o método descrito neste trabalho. Os autores argumentam que o α-investing somente tem desempenho melhor que o *Fast-OSFS* quando existe conhecimento sobre a estrutura do espaço de *features*. Além disso, nos resultados apresentados não estão inclusas bases com dados de ataques DDoS.

[Eskandari and Javidi 2016] afirmam que as *features* em *streaming* são partes fundamentais da maiorias das aplicações de mundo real. Além disso, apontam que o uso de um algoritmo de seleção de *features* em *streaming* pode ser utilizado em dois casos. O primeiro consiste no cenário em que não se conhece o espaço de *features*, e o segundo caso se refere ao cenário quando se conhece o espaço de *features*, mas a seleção pode melhorar o desempenho e a acurácia na classificação. Eles propõem um algoritmo fundamentado no conceito de conjuntos aproximados, que expressam a imprecisão em regiões limítrofes de um conjunto. As *features* são selecionadas a partir de uma análise de significância.

Os trabalhos recentes que abordam especificamente os ataques DDoS tendem a propor soluções que se aproximam das técnicas tradicionais, que é caso de [Osanaiye et al. 2016] e [Idhammad et al. 2018]. [Osanaiye et al. 2016] propõem um método que baseado em filtro para seleção de *features* em dados de ataques DDoS em nuvem. Nesse trabalho, quatro filtros são combinados (*Information Gain*, *Gain Ratio*, Qui-Quadrado e *ReliefF*) para obter uma seleção ótima de atributos em dados históricos de ataques DDoS. Para que um filtro consiga retornar um vetor de características, ou seja, o conjunto de *features* selecionadas, é necessário que o conjunto de dados de entrada seja rotulado. Isso impede a análise de outros dados que não tenham rótulo associado, como é o caso das bases utilizadas em nosso trabalho. Estas foram geradas a partir de ferramentas de monitoramento de tráfego, contendo informações sobre os pacotes da rede, sem qualquer tipo de distinção entre dados que são parte de um ataque e entre os que não são.

A proposta presente no trabalho de [Idhammad et al. 2018] segue uma abordagem semi-supervisionada, onde é aplicado um algoritmo de *clustering* para reduzir o espaço de *features* e encontrar tráfego anômalo. Assume-se que são produzidos três *clusters*, um que contém apenas tráfego normal, outro apenas com tráfego de ataque DDoS e um último com dados de ataques DDoS e tráfego normal. Estes *clusters* são gerados a partir de *NetFlows* oriundos de um histórico e são utilizados na classificação de ataques DDoS. Assim como [Osanaiye et al. 2016], o método de [Idhammad et al. 2018] é *offline*, ou seja, não é em tempo real, e não lida com dados em *streaming*, portanto não são aplicáveis em situações onde a análise tenha que ser executada sob demanda.

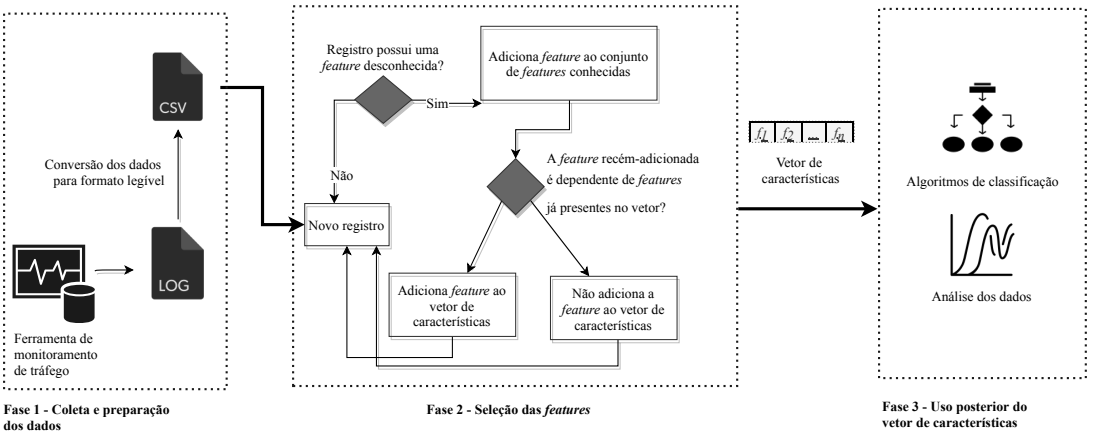

Figura 1. Visão geral das fases do método proposto.

3. Método de seleção de *features* em *streaming* para detecção de ataques

Nesta seção apresentamos o método de seleção de *features* em streaming para detecção de ataques volumétricos DDoS. Como ilustra a Figura 1, nosso método é composto por três fases: tratamento dos dados, análise dos dados pela versão adaptada do algoritmo α-*investing* e uso das *features* selecionadas. A primeira fase consiste em coletar e processar os dados em tempo real, a fim de prepará-los para a análise. A segunda fase aplica a seleção de *features* nos dados coletados até um dado momento. A terceira fase é responsável por obter um vetor de características gerado a partir da seleção executada na segunda fase, destinando-o a algoritmos de classificação ou outras atividades.

3.1. Tratamento dos dados

Na primeira etapa, assume-se que alguma ferramenta de captura e monitoramento de tráfego, como o tcpdump, Wireshark ou Netflow, estejam monitorando e coletando dados do tráfego da rede. Nesses dados, disponíveis em *logs*, estão o conteúdo de cada pacote ou fluxo de rede capturado e estes passam por um processo de *parsing*, para obter uma *string* ou um conjunto de *strings* em um formato legível para alimentar a segunda fase. A fase de tratamento de dados pode estar integrada à segunda fase se cada entrada no *log* for analisada separadamente, considerando a situação de uma análise *online*, feita sob demanda. Nesse caso, o algoritmo é alimentado com um registro do *log* por vez, que é o processo apresentado neste trabalho. Um registro é uma entrada no *log*, que contém as informações sobre o pacote capturado na ferramenta de análise de tráfego. Outra possibilidade de entrada para o algoritmo é um conjunto de registros, para que se tenha um pseudo-histórico de dados. Essa situação seria possível se for observado algum comportamento estranho no *log* que deva ser considerado para a análise aplicada aos dados conseguintes.

3.2. Seleção de *features* pelo algoritmo α-*investing*$^+$

O α-*investing* é um algoritmo de seleção de *features* em *streaming*, que visa diminuir a taxa de falsas descobertas (FDR - *False Discovery Rate*) ao ajustar dinamicamente um limiar α_i. Este limiar determina se uma *feature* recém-descoberta pode ser adicionada ao conjunto de *features* selecionadas [Zhou et al. 2005]. Neste trabalho, seguimos o algoritmo α-*investing*$^+$, uma adaptação nossa do algoritmo α-*investing* original, que não

Algoritmo 1: Algoritmo α-investing$^+$

Dados: Logs de rede
Resultado: Conjunto de *features* selecionadas

1 $W \leftarrow [0.5]$; // Conjunto de pesos
2 $w_0 \leftarrow W_0$; // Peso inicial
3 $F \leftarrow \{\}$; // Conjunto de *features* detectadas
4 $S \leftarrow \{\}$; // Conjunto de *features* selecionadas
5 $\alpha_\Delta \leftarrow 0.5$;
6 i $\leftarrow 1$;
7 **enquanto** *novas features forem detectadas* **faça**
8 adicionarFeature(f_i, F);
9 $\alpha_i \leftarrow w_i/2 * i$;
10 **para** *cada* $s \in S$ *tal que* $s \neq f_i$ **faça**
11 tabela $\leftarrow$ gerarTabelaContingencia(f_i, m);
12 resultado $\leftarrow$ chiQuadrado(tabela);
13 **se** *resultado[valorP]* $\leq \alpha_i$ **então**
14 $w_{i+1} \leftarrow w_i - \alpha_i$;
15 **senão**
16 adicionarFeature(f_i, S);
17 $w_{i+1} \leftarrow w_i + \alpha_\Delta - \alpha_i$;
18 **fim**
19 **fim**
20 i $\leftarrow$ i + 1;
21 **fim**

requer um tamanho fixo para o conjunto de *features*, admitindo a adição de novas *features* geradas dinamicamente ao longo do tempo ou a partir de outras já existentes. Além disso, este algoritmo funciona independentemente de como as *features* são geradas. O algoritmo α-*investing*$^+$ foi criado para se adequar à natureza dos dados tratados diante de cenários de ataques DDoS volumétricos, sendo estes não-paramétricos e categóricos. O α-*investing*$^+$ é apresentado em detalhes a seguir e seus passos são ilustrados no Algoritmo 1.

Seja $D = \{d_1, d_2, ..., d_n\}$ um conjunto de dados gerados até um momento t, tal que $n > 0$ e cada $d_i \in D$, sendo $0 < i \leq n$ e d_i uma j-tupla $< f_1, ..., f_j >$ (também chamado de **registro** neste artigo). Temos que para $j > 0$, f_k é um campo de texto que descreve uma característica de um d_i e $0 < k \leq j$. Cada f_j é denotado como ***feature***. Sejam $F = \{f_l$: uma feature detectada $\mid l > 0\}$ e $S = \{s_m$: uma feature selecionada $\mid m > 0\}$. Ambos os conjuntos iniciam-se vazios e não admitem elementos repetidos. É importante notar que cada *feature* única presente nos registros do conjunto D são pertencentes ao conjunto F e $S \subseteq F$. O algoritmo, ao ser iniciado, recebe um dado registro d_i (linha 7 em Algoritmo 1), que também pode ser representado como d_{t_s}, tal que t_s é o momento inicial da análise e $t_s > t$. Para que uma *feature* seja adicionada ao conjunto F (linha 8 em Algoritmo 1), são executados os seguintes passos:

1. Lê-se um novo registro;
2. Verificam-se quais *features* têm seus valores preenchidos naquele registro;
3. Descartam-se as *features* com valores não-preenchidos;

4. Para cada *feature* que não foi descartada, é verificado se ela já pertence a F;
   (a) Se a *feature* está em F, a execução é interrompida e o passo 4 é executado para a próxima *feature*;
   (b) Se *feature* não está em F, ela é adicionada ao conjunto.

Uma *feature* f_i é adicionada a S se ela não tiver relação de dependência com algum s_m em S (linha 16 em Algoritmo 1). O limiar α_i, atualizado na linha 9 do Algoritmo 1, determina a probabilidade de se incluir uma *feature* irrelevante ao modelo ao i-ésimo passo do algoritmo. O peso w_i representa o número de possíveis falsos positivos a serem incluídos ao modelo nos passos seguintes. As linhas 14 e 17 do Algoritmo 1 mostram a atualização de w_i quando f_i é dependente de algum elemento em S ou independente de algum elemento em S, respectivamente. Um falso positivo, no contexto deste trabalho, é uma *feature* $x \in S$ tal que existe pelo menos um $y \in S$ que possui uma relação de dependência com x e $x \neq y$. α_i é ajustado a partir do peso w_i. O valor-p corresponde à probabilidade das *features* f_1 e f_2 serem variáveis independentes e pode ser obtido a partir de um teste estatístico.

Como mencionado anteriormente, os dados de ataques DDoS são, frequentemente, não-paramétricos e categóricos. Assim, é necessário que o teste estatístico utilizado seja adequado a esse tipo de dados. Por isso, neste trabalho optou-se por utilizar o Teste do Chi-Quadrado para Independência de Variáveis, que basicamente é um teste de hipóteses. Neste trabalho, o valor-p é obtido a partir de uma simulação de Monte Carlo utilizada para o Teste Chi-Quadrado para Independência de Variáveis (linha 12 em Algoritmo 1).

O teste Chi-Quadrado pode ser calculado a partir de uma tabela de contingência que determina a frequência dos valores observados entre as categorias de duas variáveis v_1 e v_2 [Freedman et al. 1998]. Cada célula nessa tabela aponta a frequência que cada um dos valores de v_1 ocorre com cada um dos valores de v_2 em uma mesma observação. Em relação às hipóteses que serão testadas, a hipótese nula, denotada como H_0, e a hipótese alternativa, denotada como H_1, são definidas da seguinte forma:

H_0: v_1 e v_2 são independentes entre si

H_1: v_1 e v_2 são dependentes entre si

Tratando-se de dados de rede, uma tabela de contingência pode facilmente tornar-se esparsa, devido à grande variedade de informações contidas em cada variável categórica. Quando o teste Chi-Quadrado é aplicado em tabelas que contém frequências observadas menores do que 1, os resultados retornados não são confiáveis, incluindo o valor-p. Nesse caso, uma opção viável é estimar o valor-p através de uma simulação de Monte Carlo [Kim et al. 2006]. A partir de uma simulação de Monte Carlo é possível obter um valor empírico do erro de Tipo I, que ocorre quando H_0 é rejeitada quando na realidade ela é verdadeira. O teste é replicado diversas vezes assumindo a H_0 como verdadeira e o erro Tipo I empírico é a amostra proporcional de estatísticas significantes entre os testes replicados [Rizzo 2007].

O critério de parada do α-*investing*$^+$ é arbitrário, já que enquanto o algoritmo estiver sendo executado, serão esperadas detecções de novas *features*. Ao fim da execução, o conjunto S é denominado vetor de características e assume-se que estas características são as que melhor representam o conjunto de dados analisados do momento t_s até a parada.

3.3. Uso posterior das *features* selecionadas

A fase dois retorna um vetor de características que pode ser aplicado para pré-processamento de dados volumétricos, a fim de reduzir as dimensões e compactar o tamanho da base. Além disso, o vetor pode ser utilizado para extrair representações de bases de ataques DDoS que são usadas em classificadores ou em algoritmos de *clustering* para distinguir os diferentes tipos de tráfego. As características também podem ser aplicadas na gerência de segurança, para que saiba quais variáveis monitorar quando o objetivo é identificar um ataque, assim como para ajustar sistemas de detecção de ataques. Neste último, a seleção de *features* pode ser útil para refinar os padrões aos quais os dados são testados para que se verifique se há um ataque em curso, como é o caso das ferramentas de detecção de intrusão.

4. Metodologia

Esta seção detalha a implementação da Fase 1 e Fase 2 do método de seleção de *features* em *streaming* apresentado neste trabalho, assim como os resultados obtidos. Para isso, são utilizadas duas bases de dados públicas: CTU-13 e CICIDS2017, que contém dados de tráfego normal e de tráfego de ataques DDoS. Em ambas as bases são registrados diferentes tipos de ataques DDoS, proporcionando uma diversificação da análise e dos resultados apresentados.

4.1. Preparação dos dados

As duas bases são disponibilizadas no formato pcap, que contém a captura de cada pacote da rede durante o período de monitoramento. Para que a análise fosse facilitada, em termos de diminuição do tamanho dos arquivos a serem processados e também de legibilidade das informações, cada arquivo foi convertido para o formato CSV.

4.2. Detalhes técnicos da implementação do α-*investing*$^+$

O algoritmo foi implementado nas linguagens Python[1] na versão 3.7 e R na versão 3.5, com auxílio das bibliotecas Pandas[2] e rpy2[3], ambas disponíveis para a linguagem Python. Pandas é uma biblioteca que oferece ferramentas para análise de dados, estatística e gráficos. rpy2 é uma biblioteca que realiza a interface entre o Python e o R. A princípio, apenas o Python estava sendo utilizado, porém o Pandas não possui uma função que calcula o valor-p a partir de uma simulação de Monte Carlo para o teste do Chi-Quadrado. Por isso, optou-se por executar uma instância do R a partir do *script* escrito em Python para fazer a chamada da função do R `chisq.test()`, que atende aos requisitos para os dados utilizados neste trabalho.

4.3. CTU-13

CTU-13 é uma base pública de dados de tráfego de *botnets*, capturada pela CTU University na República Tcheca, em 2011. A topologia empregada para criar a base consiste em um conjunto de máquinas virtuais criadas sob um *host* Debian, sendo que as

[1]Documentação da versão 3.7: `https://docs.python.org/3.7/`

[2]Documentação da biblioteca Pandas: `http://pandas.pydata.org/pandas-docs/stable/`

[3]Documentação da biblioteca Pandas: `https://rpy2.readthedocs.io/en/version_2.8.x/index.html`

virtualizações executam Windows XP SP2. A base foi composta a partir da execução de 13 *malwares* e contém dados de tráfegos normais, de *botnet* e de *background*. Para cada *malware*, [García et al. 2014] descrevem que um *cenário de botnet* é a infecção das máquinas virtuais por um *malware* específico. Entre as características interessantes dessa base, destacam-se o fato de que nela estão descritos apenas ataques reais de *botnets* de diferentes tipos e que há tráfego desconhecido/não-classificado obtido de uma rede de grande porte.

Cada cenário foi projetado para representar o comportamento de algum *malware*. O tráfego de dados capturado pela ferramenta `tcmpdump`, que gerou arquivos pcap e *NetFlow* para cada cenário. Neste trabalho, serão utilizados apenas os arquivos pcap, por conterem as informações sobre os pacotes capturados individualmente e por os dados não serem rotulados. É importante notar que os arquivos pcap disponibilizados pelos autores da base contém dados somente do tráfego de *botnets*, excluindo os tráfegos normais e de *background* para preservar a privacidade da rede. Para emular uma execução em tempo real, em cada experimento é fornecido apenas um pacote por vez para análise, dentro do período estipulado.

Dos 13 cenários criados, três deles são referentes a ataques DDoS volumétricos: cenários 4, 9 e 10. No cenário 4 houve um ataque DDoS de ICMP e UDP *Flood*, teve duração maior que 4 horas, o volume de dados capturado foi de 55 GB. No cenário 9 foi executado um ataque DDoS de UDP *Flood*, com duração de pouco mais de 5 horas, gerando um volume de dados de 94 GB, o maior entre os três cenários. Por fim, o cenário 10 descreve um ataque DDoS de ICMP *Flood*, com duração de quase 5 horas e com volume de dados de 73GB. Neste trabalho, foi utilizado o cenário 4, por se tratar de uma captura envolvendo ataque DDoS com *flooding* de dois protocolos distintos.

A topologia de rede construída para realizar as capturas de cada cenário tinham controle de largura de banda e o tráfego de saída para Internet não estava sendo filtrado, a fim de que pudessem a rede pudesse receber ataques reais. Tanto o tráfego normal e de *background* quando o tráfego de ataques DDoS estavam sendo capturados pela mesma ferramenta. No cenário 4, as *botnets* utilizaram protocolo IRC e, como já descrito antes, executaram ataques DDoS de ICMP e UDP *Flood*.

4.3.1. Experimentos no Cenário 4

Nesse cenário, foram executados três experimentos. O experimento 1 compreende o período das 10:32 até às 10:52, ao qual foi registrado um intenso fluxo de dados. O experimento 2 abrange o início do *log* até o início do período utilizado no experimento 2. Finalmente, no experimento 3, o algoritmo foi executado sobre o *log* completo. O intuito é analisar o comportamento do algoritmo em três situações: durante o momento mais crítico do ataque, o momento que antecede o período mais crítico e o cenário completo, pré e pós-período mais crítico.

As *features* selecionadas em cada experimento são apresentadas na Tabela 1. A Tabela 2 mostra as *features* que apresentaram relação de dependência em cada experimento e, portanto, não foram selecionadas. Em todos os experimentos, foram detectadas 18 *features*, sendo elas: [‘SourceAddr’, ‘SourcePort’, ‘DestAddr’, ‘Protocol’, ‘Length’,

	Features selecionadas
Experimento 1	[‘SourceAddr’, ‘Protocol’, ‘DestPort’, ‘TcpAck’, ‘TcpFlagAck’, ‘TcpFlagRst’]
Experimento 2	[‘SourceAddr’, ‘Protocol’, ‘Length’, ‘FrameLen’, ‘IpTtl’, ‘IpOffset’, ‘Severity’]
Experimento 3	[‘SourceAddr’, ‘Protocol’, ‘Length’, ‘FrameLen’, ‘IpTtl’, ‘IpOffset’, ‘Severity’]

Tabela 1. Resultados dos experimentos no cenário 4

	Pares de features que são dependentes entre si
Experimento 1	[(’Length’, ’SourceAddr’), (’FrameLen’, ’SourceAddr’), (’IpTtl’, ’SourceAddr’), (’IpOffset’, ’Protocol’), (’TcpFlagPsh’, ’SourceAddr’), (’TcpFlagSyn’, ’SourceAddr’), (’TcpFlagFin’, ’SourceAddr’), (’TcpStream’, ’SourceAddr’), (’TcpWindowSize’, ’SourceAddr’), (’Severity’, ’SourceAddr’)]
Experimento 2	[(’DestPort’, ’Protocol’), (’TcpAck’, ’Length’), (’TcpFlagAck’, ’Length’), (’TcpFlagRst’, ’SourceAddr’), (’TcpFlagPsh’, ’SourceAddr’), (’TcpFlagSyn’, ’SourceAddr’), (’TcpFlagFin’, ’SourceAddr’), (’TcpStream’, ’Length’), (’TcpWindowSize’, ’SourceAddr’)]
Experimento 3	[(’DestPort’, ’Protocol’), (’TcpAck’, ’Length’), (’TcpFlagAck’, ’Length’), (’TcpFlagRst’, ’SourceAddr’), (’TcpFlagPsh’, ’SourceAddr’), (’TcpFlagSyn’, ’SourceAddr’), (’TcpFlagFin’, ’SourceAddr’), (’TcpStream’, ’Length’), (’TcpWindowSize’, ’SourceAddr’)]

Tabela 2. Dependência entre as *features* observadas no cenário 4

‘FrameLen’, ‘DestPort’, ‘IpTtl’, ‘IpOffset’, ‘Severity’, ‘TcpAck’, ‘TcpFlagAck’, ‘TcpFlagRst’, ‘TcpFlagPsh’, ‘TcpFlagSyn’, ‘TcpFlagFin’, ‘TcpStream’, ‘TcpWindowSize’].

No **experimento 1**, é interessante observar que todas as *features* que não foram selecionadas têm relação de dependência com a *feature* SourceAddr (endereço IP de origem) (Tabela 2). Dentre essas relações, destaca-se a *feature* Length (tamanho do pacote), que nos outros experimentos foi dada como independente de todas as *features* já existentes no conjunto M. Esse resultado sugere que durante o pico deste ataque cada *host* atacante estava enviando muitos pacotes do mesmo tamanho ou de tamanhos parecidos, a fim de sobrecarregar a rede.

Porém, a variação do tamanho do pacote é um sinal importante de que um ataque DDoS possa estar em curso, o que nos leva a acreditar que essa *feature* deveria estar presente em um conjunto que seleciona as *features* mais representativas do tráfego de um ataque DDoS. Além disso, nota-se que três *features*, TcpAck, TcpFlagAck e TcpFlagRst, que referentes ao protocolo TCP, foram selecionadas. Considerando o fato de que a rede sofria ataques de ICMP e UDP *Flood*, essas *features* não são muito descritivas ou representativas em relação a esse tipo de ataque. Portanto, esse experimento que ana-

lisa exclusivamente o momento de tráfego mais intenso do ataque apresentou resultados inconclusivos.

O **experimento 2** e o **experimento 3** apresentaram o mesmo conjunto de *features* selecionadas e os mesmos pares de *features* que têm relação de dependência. Isso sugere que as *features* detectadas e selecionadas no estágio inicial da análise influenciam no conjunto final, já que as *features* não passam por uma reavaliação de relevância ao decorrer da análise. Entretanto, as *features* selecionadas nos dois experimentos se mostram mais relevantes ao contexto dos ataques de ICMP e UDP *Flood* se comparadas ao experimento 1. A *feature* Length está presente no conjunto, indicando que houve maior variação no tamanho do pacote em relação tanto no cenário completo quando no período pré-ataque.

Ainda referente aos experimentos 2 e 3, nota-se que foi selecionada a *feature* IpOffset, relativa à fragmentação ICMP, indicando que o tamanho máximo de um datagrama IP foi extrapolado, sendo necessária a sua divisão em pacotes menores. Esse acontecimento é comum em ataques DDoS relacionados ao protocolo ICMP e, de fato, pode ser considerado representativo no contexto desse cenário. Outra *feature* selecionada interessante é o IpTtl, que indica o tempo que o pacote é mantido pelo *host* antes de ser descartado, já que valores anormais de TTL (*Time To Live*) pode denunciar um ataque em curso [Yamada and Goto 2012]. Essa *feature* então pode ser considerada relevante para ambos os ataques de ICMP e UDP *Flood*.

4.4. CICIDS2017

O CICIDS2017[Sharafaldin et al. 2018] é uma base pública de 2017 que contém dados de ataques recentes, disponibilizada pela *University of New Brunswick*. Na base estão armazenados registros gerados por Sistemas de Detecção de Intrusão (IDS)e por Sistemas de Prevenção de Intrusão (IPS), incluindo dados de ataques de força-bruta, ataques *heartbleed*, *botnets*, ataques DoS e DDoS, ataques Web e ataques de infiltração. O período de captura iniciou-se às 09:00 de uma segunda-feira, 3 de julho de 2017 e terminou às 17:00 de uma sexta-feira, 7 de julho de 2017, durando cinco dias. Em cada dia, houve um conjunto de ataques diferentes:

- Segunda-feira: dados benignos, sem ataques;
- Terça-feira: Ataques de força-bruta sob os serviços STFP e SSH;
- Quarta-feira: Ataques DoS e de *Heartbleed*
- Quinta-feira: Ataques Web e de infiltração;
- Sexta-feira: Ataques DDoS, *botnets* e *PortScans*

Neste trabalho, serão avaliados os resultados do nosso método sob os dados capturados na sexta-feira. Nesse dia pela manhã foi executado o ataque de botnets e durante a tarde houve o ataque DDoS. O ataque DDoS foi executado através da ferramenta *Low Orbit Ion Canon* (HOIC), que enviava requisições UDP, TCP ou HTTP às vitimas. Os atacantes eram máquinas Windows 8.1 e a vítima era um servidor *web* sob o Ubuntu 16.

Nesta base foi executado um experimento apenas, sob todo o conjuntos de dados registrados durante a tarde de sexta-feira, período ao qual o ataque DDoS foi desempenhado. No total, foram capturados pouco mais de 8,3 milhões de pacotes. Foram detectadas 16 *features* neste conjunto de dados e o vetor de características foi retornado com 5 elementos, demonstrando uma redução importante no espaço de *features*. Na Tabela 3 é

Vetor de características	[‘DestPort’, ‘SourcePort’, ‘Protocol’, ‘IpFlags’, ‘IpLen’]
Features dependentes entre si	[(‘SrcAddr’, ‘DestPort’), (‘IpTtl’, ‘DestPort’), (‘FrameLen’, ‘DestPort’), (‘FrameProtocols’, ‘DestPort’), (‘TcpAck’, ‘DestPort’), (‘TcpFlags’, ‘DestPort’), (‘TcpFlagsAck’, ‘DestPort’), (‘TcpLen’, ‘DestPort’), (‘TcpWindowSize’, ‘DestPort’)]
Features detectadas	[‘DestPort’, ‘DestAddr’, ‘SourcePort’, ‘SrcAddr’, ‘Protocol’, ‘IpFlags’, ‘IpLen’, ‘IpTtl’, ‘FrameLen’, ‘FrameProtocols’, ‘TcpAck’, ‘TcpFlags’, ‘TcpFlagsAck’, ‘TcpLen’, ‘TcpWindowSize’, ‘IpFragment’]

Tabela 3. Resultados dos experimento executado sob os dados observados durante o ataque DDoS na tarde de sexta-feira

possível visualizar as *features* selecionadas e detectadas, assim como as *features* que são dependentes entre si.

Nota-se que uma característica importante presente no vetor é o DestPort, já que esta *feature* é relevante nos ataques de TCP e HTTP *Flooding*. Este último ataque é especialmente mais díficil de ser detectado, já que não são utilizados pacotes mal-formados como nos outros tipos de *Flood*, atuando na camada de aplicação. O atacante pode sobrecarregar um servidor *web* ao criar múltiplas requisições GET e POST, cujos dados de monitoramento refletem na porta 80 ou 8080. Outra *feature* que se destaca é a do tamanho dos datagramas IP, que podem ser oriundos de um ataque de UDP *Flooding* e que indicam possíveis ataques em curso, em especial se o tamanho é crescente ao longo de tempo.

Um fato que chama atenção é que não há *features* no vetor de características que são exclusivamente relacionadas ao ataque de TCP *Flooding*, como aconteceu nos experimentos com a base do CTU-13. O que se nota, aliás, é que as características relativas ao protocolo TCP na verdade foram percebidas pelo método como dependentes de outras *features*. TcpAck e TcpFlagsAck, por exemplo, são dependentes de DestPort, provavelmente por não apresentarem variação de porta de destino nos pacotes TCP que continham informações sobre a *flag* ACK.

Outra possibilidade seria que nesse experimento essas *features* têm poder discriminante muito similar, o que permite a exclusão das que se referem ao protocolo TCP. Contudo, como o cenário proposto era sintético, com apenas uma máquina atuando como alvo e com um pequeno grupo de atacantes agindo, a variação de valores nas *features* não é tão expressiva. Isso pode ocasionar a inferência incorreta de algumas dependências, devido às poucas categorias (ou sejam, valores distintos) existentes para cada *feature*. Neste caso, encontra-se uma falha em nosso método, já que nele ainda não está presente um mecanismo que consiga distinguir essas duas situações apresentadas.

5. Conclusão

Os ataques DDoS volumétricos, por apresentarem ameaça a empresas e organizações, requerem métodos que permitam a sua detecção o mais rápido possível. Como é necessário

que a análise seja em tempo real, sob dados gerados de maneira contínua, as técnicas que somente trabalham com dados históricos não são as mais adequadas. Além disso, a partir do estudo dos trabalhos presentes na literatura, percebe-se a dificuldade existente em tratar de dados de ataques DDoS em algoritmos de seleção de *feature*, tendo em vista que muitas vezes não é possível ou viável utilizar as ferramentas mais tradicionais. Por isso, o método proposto neste trabalho visa oferecer um meio de realizar a análise de maneira *online*, sob dados em *streaming*.

Uma das maiores preocupações em lidar com esse tipo de dados é que a técnica precisa se adaptar à natureza dos mesmos, que são em sua maioria não-paramétricos e, frequentemente, esparsos. O nosso método mostra-se como uma opção interessante para análise rápida e em tempo real em casos que o volume do tráfego do ataque DDoS é bastante alto e o conteúdo das *features* variam dentro da janela de análise. Uma das principais vantagens do método é que é utilizado um critério simples e eficaz, que é a independência de variáveis, para selecionar as *features* mais representativas. Ademais, destaca-se a capacidade de selecionar *features* discriminativas especialmente nos conjuntos cujos dados possuem grande variação de valores, que é o caso da maioria das situações no mundo real.

Para os trabalhos futuros, procura-se explorar a maior falha do método percebida até então que impede a apresentação de resultados totalmente satisfatórios em bases de caráter sintético, com poucos atacantes e/ou poucos alvos do ataque. E há intenção de verificar se o vetor de características retornado pelo método é capaz de reduzir o erro de classificação, nas técnicas baseadas em aprendizagem de máquina.

Referências

[Bellman 1966] Bellman, R. (1966). Dynamic Programming. *Science*, 153(3731):34–37.

[Borini 2018] Borini, G. (2018). Primeiro trimestre tem 1,4 milhão de ataques DDoS, aponta levantamento. https://computerworld.com.br/2018/05/08/primeiro-trimestre-tem-14-milhao-de-ataques-ddos-aponta-levantamento/.

[Eskandari and Javidi 2016] Eskandari, S. and Javidi, M. M. (2016). Online streaming feature selection using rough sets. *International Journal of Approximate Reasoning*, 69:35–57.

[Freedman et al. 1998] Freedman, D., Pisani, R., and Purves, R. (1998). *Statistics*. W.W. Norton.

[García et al. 2014] García, S., Grill, M., Stiborek, J., and Zunino, A. (2014). An empirical comparison of botnet detection methods. *Computers & Security*, 45:100–123.

[Guyon and Elisseeff 2003] Guyon, I. and Elisseeff, A. (2003). An Introduction to Variable and Feature Selection 1 Introduction. *An Introduction to Variable and Feature Selection*, 3(Mar):1157–1182.

[Guyon and Elisseeff 2008] Guyon, I. and Elisseeff, A. (2008). An Introduction to Feature Extraction. In *Feature Extraction*, pages 1–25.

[Hu et al. 2018] Hu, X., Zhou, P., Li, P., Wang, J., and Wu, X. (2018). A survey on online feature selection with streaming features.

[Idhammad et al. 2018] Idhammad, M., Afdel, K., and Belouch, M. (2018). Semi-supervised machine learning approach for DDoS detection. *Applied Intelligence*, 48(10):3193–3208.

[Jialei Wang et al. 2014] Jialei Wang, Peilin Zhao, Hoi, S. C. H., and Rong Jin (2014). Online Feature Selection and Its Applications. *IEEE Transactions on Knowledge and Data Engineering*, 26(3):698–710.

[Kim et al. 2006] Kim, H., Robert, C. P., and Casella, G. (2006). *Monte Carlo Statistical Methods*. Number 4.

[Li et al. 2017] Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., and Liu, H. (2017). Feature selection: A data perspective. *ACM Comput. Surv.*, 50(6):94:1–94:45.

[Osanaiye et al. 2016] Osanaiye, O., Cai, H., Choo, K. K. R., Dehghantanha, A., Xu, Z., and Dlodlo, M. (2016). Ensemble-based multi-filter feature selection method for DDoS detection in cloud computing. *Eurasip Journal on Wireless Communications and Networking*, 2016(1):130.

[Perkins et al. 2003] Perkins, S., Lacker, K., and Theiler, J. (2003). 10.1162/153244303322753698. *The Journal of Machine Learning Research*, 1(Mar):1333–1356.

[Rizzo 2007] Rizzo, M. (2007). *Statistical Computing with R*. Chapman & Hall/CRC The R Series. Taylor & Francis.

[Santos et al. 2017] Santos, A. A., Nogueira, M., and Moura, J. M. F. (2017). A stochastic adaptive model to explore mobile botnet dynamics. *IEEE Communications Letters*, 21(4):753–756.

[Sharafaldin et al. 2018] Sharafaldin, I., Habibi Lashkari, A., and Ghorbani, A. A. (2018). Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. pages 108–116.

[Wu et al. 2013] Wu, X., Yu, K., Ding, W., Wang, H., and Zhu, X. (2013). Online feature selection with streaming features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(5):1178–1192.

[Yamada and Goto 2012] Yamada, R. and Goto, S. (2012). Using abnormal TTL values to detect malicious IP packets. *Proceedings of the Asia-Pacific Advanced Network*, 34(0):27.

[Zhou et al. 2005] Zhou, J., Foster, D., Stine, R., and Ungar, L. (2005). Streaming feature selection using alpha-investing. In *Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining - KDD '05*, page 384, New York, New York, USA. ACM Press.