EB-CyberDef: Um Ambiente Integrado de Defesa Cibernética para Apoio à Detecção e ao Combate de Comportamentos Maliciosos no Tráfego de Redes de Computadores

Gustavo A. Testoni[1], **Marcos Paulo L. A. Pereira**[1],
Sérgio S. Cardoso[1], **Clayton E. das Chagas**[1], **Ronaldo R. Goldschimidt**[1]

[1]Seção de Engenharia de Computação – Instituto Militar de Engenharia (IME)
Rio de Janeiro – RJ – Brazil

{gustavo.testoni,marcos.pereira,scardoso,escouper,ronaldo.rgold}@ime.eb.br

Abstract. *The growing number of malicious attacks on the internet has ramped up the detection and combat of malware, two major problems in the Information Security area. Despite the development of many solutions to overcome these problems, most of them do not deal with both problemas simultaneously. Moreover, they are usually restricted to one type of malware. In this scenario, the present article proposes EB-CyberDef, an integrated cybernetic defense environment that supports network malicious behavior detection and combat. The proposed environment is configurable and can be used to identify and mitigate the effects of different types of malware. Results from preliminary experiments with EB-CyberDef's developing prototype illustrate its potential to detect different botnet families.*

Resumo. *Diante do crescente volume de ataques maliciosos ocorridos na Internet, dois dos principais problemas enfrentados pela área de Segurança da Informação atualmente são a detecção e o combate de* malwares. *Embora diversas soluções estejam sendo desenvolvidas para lidar com esses problemas, a maioria delas trata de forma mutuamente exclusiva cada um desses dois problemas ou se concentra apenas no tratamento de um único tipo de* malware. *Neste contexto, o presente artigo tem por objetivo propor o EB-CyberDef, um ambiente integrado de defesa cibernética para apoiar a detecção e o combate de comportamentos maliciosos no tráfego de redes de computadores. O ambiente proposto é configurável e pode ser usado na identificação e na mitigação dos efeitos de diferentes tipos de* malware. *Resultados de experimentos preliminares sobre a capacidade do EB-CyberDef em detectar diferentes famílias de* botnets *ilustram o funcionamento e o potencial do protótipo em desenvolvimento.*

1. Introdução

Impulsionados a partir do surgimento da *World Wide Web* em 1982, diversos avanços na área da Tecnologia da Informação ocorridos nas últimas décadas têm permitido que um número cada vez maior de dispositivos e pessoas se conectem pela Internet ao redor do mundo a fim de realizar diferentes atividades tais como divulgação e consulta de notícias e informações em geral, compra e venda de produtos por meio de comércio eletrônico, interações em redes sociais virtuais, dentre outras [Berners-Lee 2019]. Diante

deste cenário, tem crescido também o número de atividades maliciosas ocorridas na Internet, como, por exemplo, espionagem industrial, manipulação de pesquisas e opiniões, ataques a infraestruturas críticas, sequestros de dados e o surgimento da guerra cibernética [Clarke and Knake 2010]. Estima-se que os prejuízos econômicos anuais causados por este tipo de atividade esteja na ordem de bilhões de dólares [Farahbod et al. 2020].

Grande parte das atividades maliciosas ocorridas atualmente na Internet são executadas por meio de softwares maliciosos (*malwares*). Um exemplo representativo de *malware* são os *bots* que infectam máquinas e formam redes denominadas *botnets* cujo controle é feito remotamente por uma ou mais pessoas (*botmasters*) [Silva et al. 2013].

Entre os principais problemas enfrentados pela área de Segurança da Informação estão a detecção e o combate de *malwares*. Diversas soluções vêm sendo desenvolvidas para lidar com esses problemas [Veeramachaneni et al. 2016]. No entanto, a maioria delas trata de forma mutuamente exclusiva um desses dois problemas ou se concentra apenas no tratamento de um único tipo ou família de *malware* [Veeramachaneni et al. 2016].

Assim sendo, este artigo tem por objetivo apresentar o EB-CyberDef, um ambiente integrado de defesa cibernética para apoio à detecção e ao combate de comportamentos maliciosos no tráfego de redes de computadores no contexto do Exército Brasileiro (EB). Estendido a partir da arquitetura descrita em testo[Silva and Salles 2012], o EB-CyberDef supre uma importante lacuna existente no EB por um *software* com as funcionalidades oferecidas pelo ambiente proposto. Tal ambiente pode ser configurado para apoiar simultaneamente à detecção e à mitigação dos efeitos de diferentes tipos de *malware*. Para tanto, o EB-CyberDef encontra-se dividido em módulos que abrangem a coleta de dados sobre o tráfego de rede, a análise de legitimidade do tráfego por meio de técnicas de inteligência artificial para filtragem de casos suspeitos, o acionamento de mecanismos de defesa compatíveis com o tipo de ameaça detectada e um painel de apoio à supervisão humana do processo. Resultados de experimentos preliminares sobre a capacidade do módulo de legitimidade em detectar diferentes famílias de *botnets* ilustram o funcionamento do protótipo em desenvolvimento.

O presente texto possui mais três seções. A Seção 2 apresenta o EB-CyberDef, descrevendo as principais funcionalidades de cada um dos módulos que compõem o ambiente proposto. A Seção 3 resume o protótipo desenvolvido até o presente momento, especificando as funções já implementadas. A Seção 4 reporta os resultados de experimentos preliminares realizados com o módulo de análise de legitimidade do tráfego de rede, ilustrando o potencial de aplicação do EB-CyberDef. Por fim, a Seção 5 destaca as contribuições do trabalho e aponta para as iniciativas de trabalhos futuros, algumas delas já em andamento.

2. Ambiente Proposto

Denominado EB-CyberDef, o ambiente proposto neste artigo é uma extensão da arquitetura proposta em [Daisy C. A. Silva and Salles 2017]. e tem como objetivo oferecer apoio informatizado e integrado ao processo de detecção e combate de comportamentos maliciosos no tráfego de redes de computadores. A Figura 1 apresenta uma visão geral do processo executado pelo EB-CyberDef em um cenário de operação em produção. A explicação de cada módulo funcional encontra-se a seguir.

O módulo de *Coleta de Dados* deve ser capaz de extrair informações de diferentes

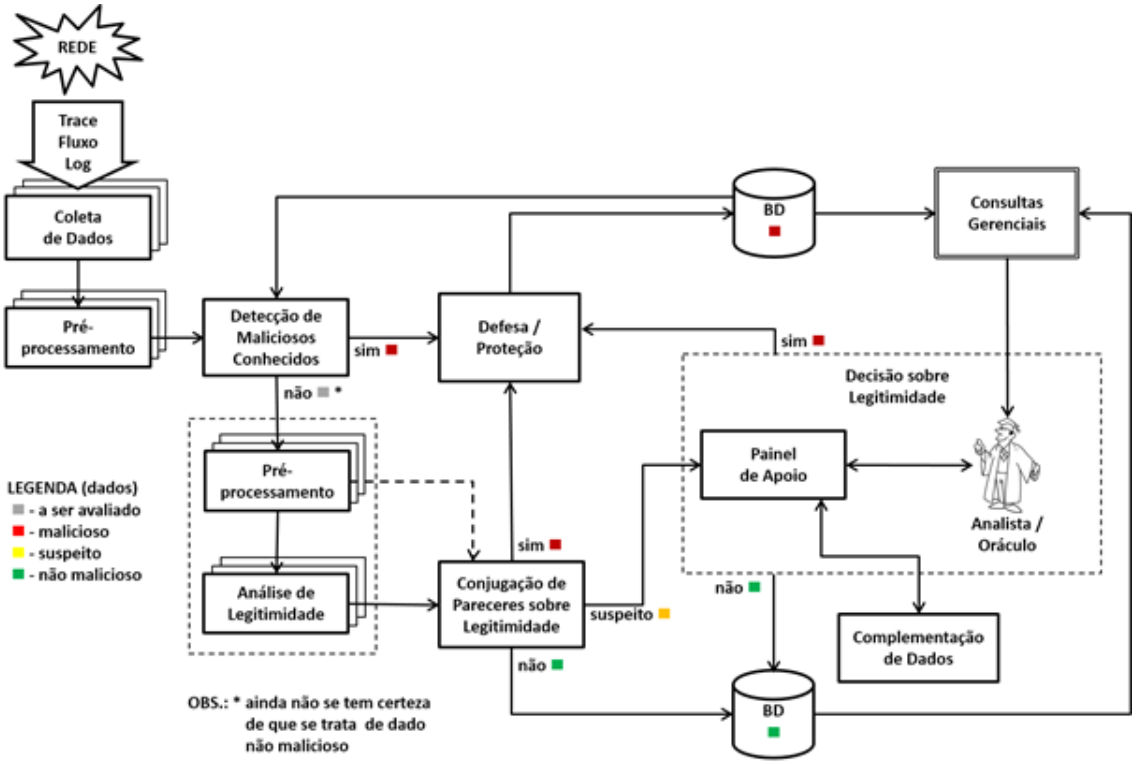

Figura 1. Visão Macro-Funcional do EB-CyberDef: Processo em Produção.

fontes de dados (*traces*, *logs* e fluxos) em diferentes formatos para serem processadas pelo ambiente.

Em seguida, o módulo de *Pré-processamento* é responsável por transformar as informações coletadas anteriormente em registros de dados cujos atributos descrevam de forma sumarizada o conteúdo coletado. Cada registro de dados r é representado por uma lista ordenada de valores $(v_1, v_2, ..., v_n)$ onde cada v_i corresponde ao valor de um atributo descritor dos dados coletados. Por exemplo, sobre um registro de fluxo de redes devem ser armazenados os valores de atributos como os endereços de origem e destino, portas empregadas, protocolos, direção, volume de tráfego, entre outras propriedades que podem ser necessárias ao processamento.

Já o módulo *Detecção de Maliciosos Conhecidos* é responsável pela identificação do tráfego malicioso (vermelhos) através de sua assinatura, isto é, características que podem ser identificadas no registro de dados sem a necessidade de emprego de mecanismos baseados em Inteligência Artificial e Aprendizado de Máquina. Para isso é mantida uma lista L com os registros dos padrões maliciosos conhecidos de forma que este módulo verifique, para cada registro de dados r a ser analisado pelo sistema, se $r \in L$.

Confirmado que determinado fluxo é malicioso, o módulo *Defesa e Proteção* aciona medidas de defesa e/ou proteção para mitigar o problema. Tais medidas são denominadas Funções de Proteção. Estas funções podem ser especializadas para *firewalls*, servidores e roteadores. Um exemplo de medida de proteção que pode ser tomada é que após a confirmação que determinado tráfego é malicioso, seja adicionada uma regra em um *firewall* com o objetivo de bloquear, exclusivamente, o fluxo malicioso.

O módulo *Pré-processamento e Análise de Legitimidade* é composto de um conjunto de algoritmos de Aprendizado de Máquina $C = \{c_1, c_2, ..., c_m\}$ voltados à classificação de dados. Para cada registro de dados r recebido por este módulo, r é pré-processado e submetido a cada $c_i \in C$. O conjunto de operações de pré-processamento pode variar em função de c_i e abrange algoritmos como codificação, normalização, imputação, dentre outros. Por exemplo, se c_i é uma rede neural, todos os valores de r devem ser quantitativos. Assim, caso r possua atributos qualitativos, uma operação de codificação qualitativa-quantitativa deve ser aplicada em r antes que r seja submetido à c_i. Após o pré-processamento, r é submetido a c_i que produz uma saída que pode ter valor *malicioso* ou *legítimo* e corresponde ao parecer de c_i sobre a legitimidade de r. Ao final da execução deste módulo, são produzidas m saídas (ie., os pareceres dos algoritmos de C sobre a legitimidade de r) que são enviadas para o módulo *Conjugação de Pareceres de Legitimidade*.

A partir dos m pareceres sobre r produzidos pelo módulo anterior, o módulo *Conjugação de Pareceres sobre Legitimidade* possui um árbitro A que atua como uma espécie de juiz que analisa esses pareceres a fim de emitir um único parecer que reflita a impressão final do sistema quanto à classificação de r. Desta forma, A classifica r em uma de três opções: (i) certamente não malicioso (verde), (ii) certamente malicioso (vermelho) e (iii) suspeito (amarelo). A última classificação reflete situações em que o sistema não consiga identificar com clareza se r é malicioso ou não. Tais situações podem ocorrer devido à falta de consenso entre os pareceres dos algoritmos de classificação sobretudo diante de ataques *zero day* (ie., novos *malwares* e/ou atividades maliciosas).

Sempre que r é considerado malicioso pelo módulo *Conjugação de Pareceres sobre Legitimidade*, o módulo *Defesa e Proteção* é acionado a fim de tomar as providências cabíveis. Por exemplo, caso seja verificado que se trata de uma tentativa de invasão a um serviço, é gerado um alerta ao administrador do serviço e, em paralelo, realizado o bloqueio do acesso ao servidor. Por outro lado, quando r é considerado não malicioso, seus dados são simplesmente armazenados em um banco de dados e nenhuma providência adicional é necessária. No entanto, quando r é considerado suspeito, este é enviado para o módulo *Decisão de Legitimidade*, a fim de que seja analisado por um analista humano (oráculo).

Diante dos dados apresentados, o oráculo deve opinar sobre a situação de r e indicar como r deverá ser tratado pelo sistema. Caso o oráculo deseje, o módulo *Painel de Apoio* pode apresentar dados externos ao sistema que auxiliem no processo decisório. Para tanto, o módulo *Complementação de Dados* realiza buscas adicionais sobre r. Por exemplo, se r estiver relacionado a um tráfego do serviço DNS, são obtidas informações adicionais por meio do serviço *whois* para que seja identificado quem é o responsável pelo registro do nome de domínio, data de criação, país de registro, entre outras informações que irão auxiliar na decisão de legitimidade. Após a decisão final do oráculo, as ações a serem tomadas são semelhantes às executadas a partir da saída do módulo *Conjugação de Pareceres sobre Legitimidade*.

É importante enfatizar que todas as ações e identificações realizadas ao longo do processamento do EB-CyberDef são armazenadas em bancos de dados para fim de serem utilizadas posteriormente para geração de relatórios, estatísticas e até mesmo permitir que sejam realizados eventuais ajustes dos parâmetros do sistema. Tais relatórios são

relevantes também para identificar informações úteis para serem apresentadas no *Painel de Apoio*.

3. Protótipo

Com o intuito de fornecer uma prova de conceito acerca da viabilidade de funcionamento do EB-CyberDef foi desenvolvido um protótipo que integra implementações dos módulos do ambiente produzidas por meio de diversas iniciativas de pesquisa e frentes de trabalho tais como dissertações de mestrado, projetos de fim de curso de graduação e projetos de iniciação científica. É importante mencionar que os módulos do EB-CyberDef foram inicialmente desenvolvidos de forma independente, sem uma preocupação explícita quanto à forma de integração, sendo cada projeto focado em resolver os problemas inerentes ao seu objeto de estudo. A seguir, encontram-se comentadas, em ordem cronológica de desenvolvimento, as principais características das implementações dos módulos do EB-CyberDef que compõem a prova de conceito do ambiente. Por fim, são apresentadas as providências empregadas de forma a viabilizar a integração dos referidos módulos.

O primeiro projeto de fim de curso voltado ao desenvolvimento do EB-CyberDef buscou desenvolver uma ferramenta para criação de comitês de classificadores capazes de rotular fluxos gerados por Botnets[de Brito and Bezerra 2018]. A ideia do projeto foi desenvolver modelos de classificação que implementassem os módulos *Pré-processamento e Análise de Legitimidade* e *Conjugação de Pareceres sobre Legitimidade*. Para cumprir esse objetivo, algoritmos de classificação como Decision Tree, KNN, SVM, Naive Bayes e MLP foram implementados utilizando-se a linguagem *Python*[1] por meio das bibliotecas *Scikit-learn*[2], *Pandas*[3] e *Numpy*[4]. A definição dos valores dos hiperparâmetros de cada algoritmo pode ser feita de forma empírica ou por meio de mecanismos de busca como a função *GridSearch* da *Scikit-learn*. Por fim, com a finalidade de avaliar os modelos, foi implementada a técnica de validação cruzada com K conjuntos [Lunardi 2014].

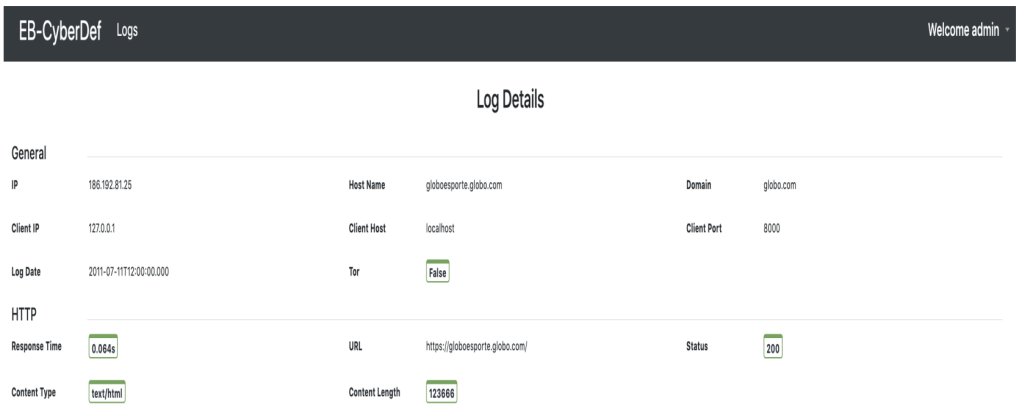

Figura 2. Tela de fluxos classificados como suspeitos pelo módulo de Análise de Legitimidade e Conjugação de Pareceres no módulo Painel de Apoio.

Em um projeto de fim de curso realizado no ano seguinte foi desenvolvido o *Pai-*

[1]https://www.python.org/
[2]https://scikit-learn.org/stable/
[3]https://pandas.pydata.org/
[4]https://numpy.org/

nel de Apoio[de Araujo and Ventura 2019]. Para este módulo foi implementada uma interface utilizando-se a biblioteca *React*[5] escrita em *Javascript*[6], haja vista a natureza visual do módulo. Nela são apresentadas características relevantes dos fluxos e, assim, um analista humano (no papel de oráculo) pode conferir o juízo acerca da legitimidade dos fluxos classificados como suspeitos pelo módulo de Conjugação de Pareceres. A Figura 2 apresenta um exemplo por meio da referida interface. Além disso, com a finalidade de persistir os fluxos e realizar posteriormente o processo de manutenção desses dados foi utilizado o MongoDB[7], que é um sistema de gerenciamento de banco de dados não relacional que permite flexibilidade na representação dos dados, simplificando a construção da aplicação. Por fim, para acessar esses dados e realizar lógicas como uma verificação inicial de assinaturas e classificar o fluxo, foi implementada uma API escrita em *Python* utilizando-se a biblioteca HTTP *Sanic*[8], que tem por características simplicidade no desenvolvimento de API's e um bom desempenho computacional.

Dando continuidade ao desenvolvimento dos módulos, no ano subsequente, foi desenvolvido um protótipo do módulo de *Defesa e Proteção*[Conterno and de Oliveira 2020]. Ao receber informações acerca de um ataque, tal módulo invoca de forma automática as funções de defesa pré-configuradas, utilizando-se protocolos e ferramentas consolidadas, como o RDAP (Protocolo de Acesso de Dados de Registro) por meio da API fornecida pelo site https://registro.br/rdap/ e o IPTables com a finalidade de criar regras de tráfego de rede, entre outros. Para atingir esse objetivo, um servidor BackEnd escrito em *Javascript* por meio da plataforma *NodeJS*[9] foi implementado. Além da implementação do protótipo automatizado, foi disponibilizada uma integração com o serviço de mensageria Telegram para notificações e configurações por parte do usuário. As Figuras 3, 4, 5 e 6 apresentam algumas destas funcionalidades aplicadas a exemplos ilustrativos.

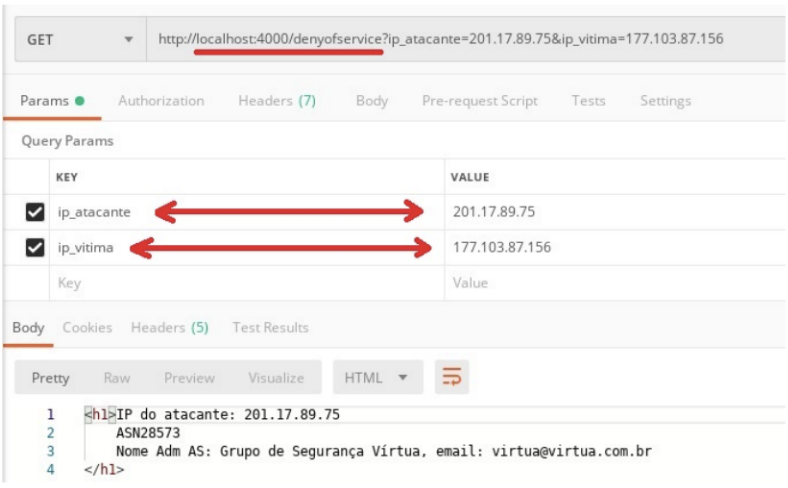

Figura 3. Notificação de um ataque DoS em curso enviada ao Módulo de *Defesa e Proteção*.

[5]https://pt-br.reactjs.org/
[6]https://www.javascript.com/
[7]https://www.mongodb.com/pt-br
[8]https://sanic.dev/en/
[9]https://nodejs.org/en/

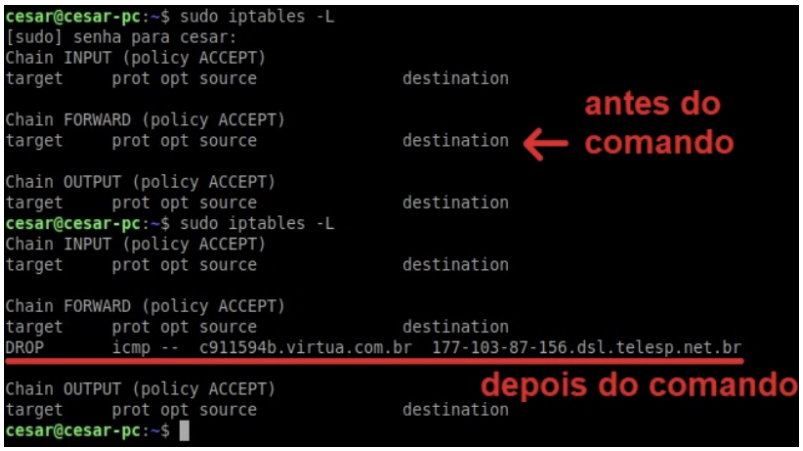

Figura 4. Inclusão na Tabela de Roteamento de instrução para bloquear IP do Atacante - antes e após a execução do comando correspondente.

Figura 5. Notificação de um ataque DoS enviada ao administrador da rede da vítima.

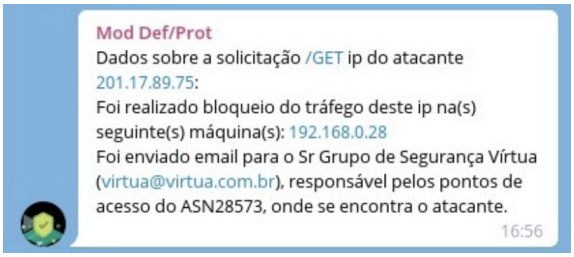

Figura 6. Notificação de ataque DoS tratado pelo Módulo de Defesa/Proteção enviada ao Telegram para log.

Após o desenvolvimento dos módulos mencionados acima e tendo em vista que o objetivo do presente trabalho foi implementar um ambiente de defesa integrado, foi necessário elaborar uma solução para o problema da integração entre os componentes já desenvolvidos. Para isso, considerando que na concepção do Módulo de Defesa e Proteção e do Módulo de Painel de Apoio foram utilizadas API's HTTP, decidiu-se por continuar a utilizar esse mesmo protocolo, com adição do protocolo WebSocket, que apareceu como solução para a necessidade de exibir informações de novos fluxos suspeitos em tempo real para o Oráculo por meio do Painel de Apoio.

Além disso, como os projetos anteriores foram construídos em contextos diferentes, algumas modificações de padronização nas API's tiveram que ser realizadas, além da implementação de uma API HTTP para o módulo de Conjugação de Pareceres, que antes somente funcionava por linha de comando ou por uma interface gráfica. Ainda entre as modificações, surgiu a necessidade também de que fosse adicionada uma nova classe denominada "suspeito" como resultado da classificação do Módulo de Conjugação de Pareceres.

Conforme previsto na Figura 1, a solução compreende um módulo de Detecção de Maliciosos Conhecidos, o qual tem por responsabilidade realizar uma filtragem inicial nos dados. Assim sendo, também foi realizado o desenvolvimento desse módulo, utilizando as mesmas tecnologias e protocolos dos módulos anteriores, além de uma integração por API com o software *WireShark* para captura de fluxo da rede local.

Ainda como necessidade de aprimorar o módulo Painel de Apoio, a tela que exibe os fluxos classificados como suspeitos pelo módulo sofreu uma modificação para que essa exibição fosse feita em tempo real. Para tanto, foi implementado o protocolo *WebSocket* tanto do lado Cliente do Painel quanto do lado Servidor. Além disso, foi adicionada uma nova tela que tem por finalidade exibir as ações realizadas pelo módulo de Defesa e Proteção, servindo como possível fonte estatística para o operador do software.

Por fim, como conclusão da integração dos módulos, o então *BackEnd* do módulo Painel de Apoio recebeu a denominação de módulo Integrador, passando a ter papel fundamental na execução das consultas às fontes de dados e na posterior exibição de resultados na interface do software.

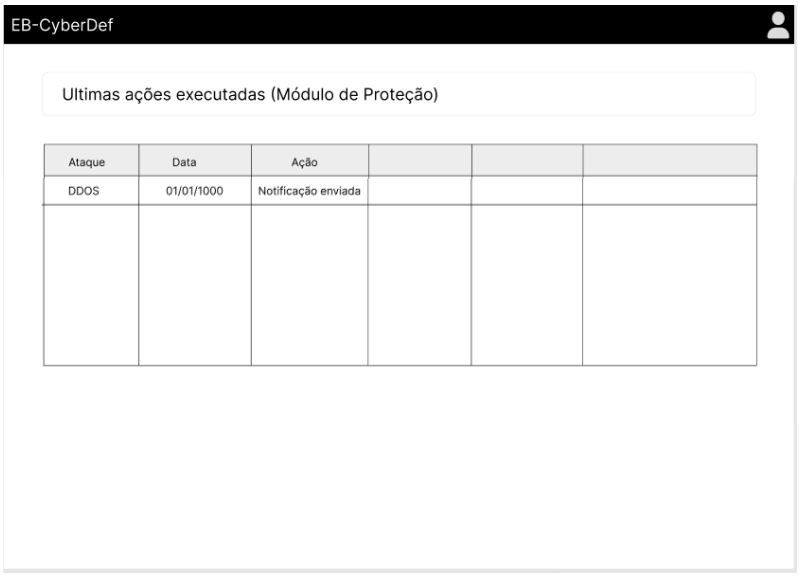

Figura 7. Tela de ações executadas pelo módulo de *Defesa e Proteção*.

4. Experimentos Preliminares

Com o intuito de verificar a viabilidade do modelo proposto na detecção de *botnets*, foram realizados experimentos no contexto dos módulos de *Pré-Processamento*

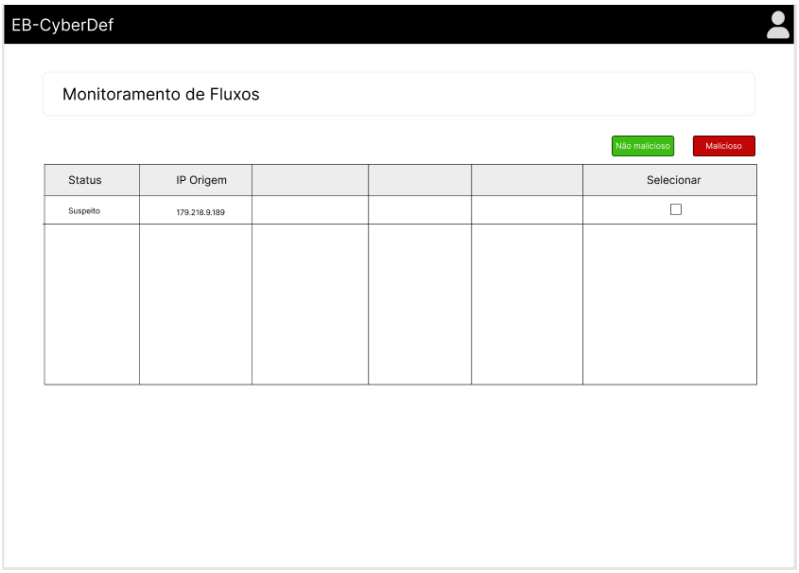

Figura 8. Tela de monitoramento de fluxos em tempo real utilizada pelo Oráculo (Especialista).

e de Análise de Legitimidade e *Conjugação de Pareceres*. Esta seção descreve resumidamente dois experimentos realizados: o primeiro que foi apresentado em [Daisy C. A. Silva and Salles 2017] e o segundo pelo projeto de fim de curso [de Brito and Bezerra 2018]. O primeiro experimento teve como objetivo avaliar individualmente os classificadores na identificação de *malware*, e o segundo utilizou árbitros para avaliar a melhora no desempenho de classificação.

Os conjuntos de algoritmos de classificação adotados no primeiro e no segundo experimentos foram, respectivamente, $C_1 = \{$*Naive Bayes, J48, SVM, k-NN*$\}$ e $C_2 = \{$*Decision Tree, SVM, k-NN, MLP, Naive Bayes*$\}$. Além disso, no segundo experimento foram utilizados como árbitros os algoritmos do conjunto $A = \{$*Votos, Decision Tree, SVM, k-NN, MLP, Naive Bayes*$\}$. Em ambos os experimentos, a escolha dos algoritmos foi norteada por dois critérios básicos: (1) a popularidade dos algoritmos em diversas pesquisas na área de aprendizado de máquina [Daisy C. A. Silva and Salles 2017]; (2) a busca pela diversidade de vieses de aprendizado. Com a escolha realizada, os vieses estatístico (*Naive Bayes*), simbólico (*J48*) e indutivo (*k-NN* e *SVM*) foram representados. Detalhes sobre os algoritmos e seus vieses de aprendizado podem ser obtidos em [Faceli et al. 2011].

A base de dados utilizada em ambos os experimentos foi a *Czech Technical University* (CTU) [Garcia et al. 2014]. Esta base foi escolhida por apresentar tráfegos associados a diferentes famílias de *bots* e por sua popularidade em experimentos na área. Nela, cada registro de dados r encontra-se classificado em uma dentre as seguintes opções: tráfego normal, *botnet* e *background*. Nestas classes estão enquadrados os registros de dados não maliciosos, maliciosos e indeterminados, respectivamente. Outra característica desta base é que ela está dividida em treze cenários, e cada um deles contém algum comportamento de *malware*, totalizando, no mínimo, treze tipos de *bots*.

Inicialmente, a base foi pré-processada com o objetivo de reduzir sua dimensionalidade ao remover atributos irrelevantes tais como IP de origem, porta de origem, IP

de destino e porta de destino. Além disso, por ser uma base de dados desbalanceada e os algoritmos de classificação serem sensíveis a desbalanceamentos, foi adotado o processo de validação cruzada com $K = 10$ conjuntos. Cada modelo de classificação gerado ao longo do processo foi armazenado a fim de identificar o melhor deles para cada algoritmo em cada cenário. A acurácia foi utilizada como métrica de desempenho dos algoritmos de aprendizado.

A Figura 9 apresenta os resultados do primeiro experimento, que envolveu a aplicação do processo de validação cruzada com os algoritmos de C_1 em cada cenário da base de dados. Tais resultados mostram que o algoritmo *Naive Bayes* obteve os piores resultados, como já era esperado, uma vez que, na maioria dos problemas, as variáveis não são independentes entre si, o que contraria a premissa em que o algoritmo se baseia. O algoritmo *J48* não obteve bons resultados nos cenários desbalanceados, com uma baixa taxa de acerto na classe *botnet*. O algoritmo *SVM* obteve bons resultados em cenários desbalanceados, porém com um alto tempo de processamento na fase de treinamento, aumentando assim os custos computacionais do experimento. O algoritmo *k-NN* obteve destaque no tempo de processamento, sendo o algoritmo mais rápido e com bons resultados mesmo em cenários desbalanceados.

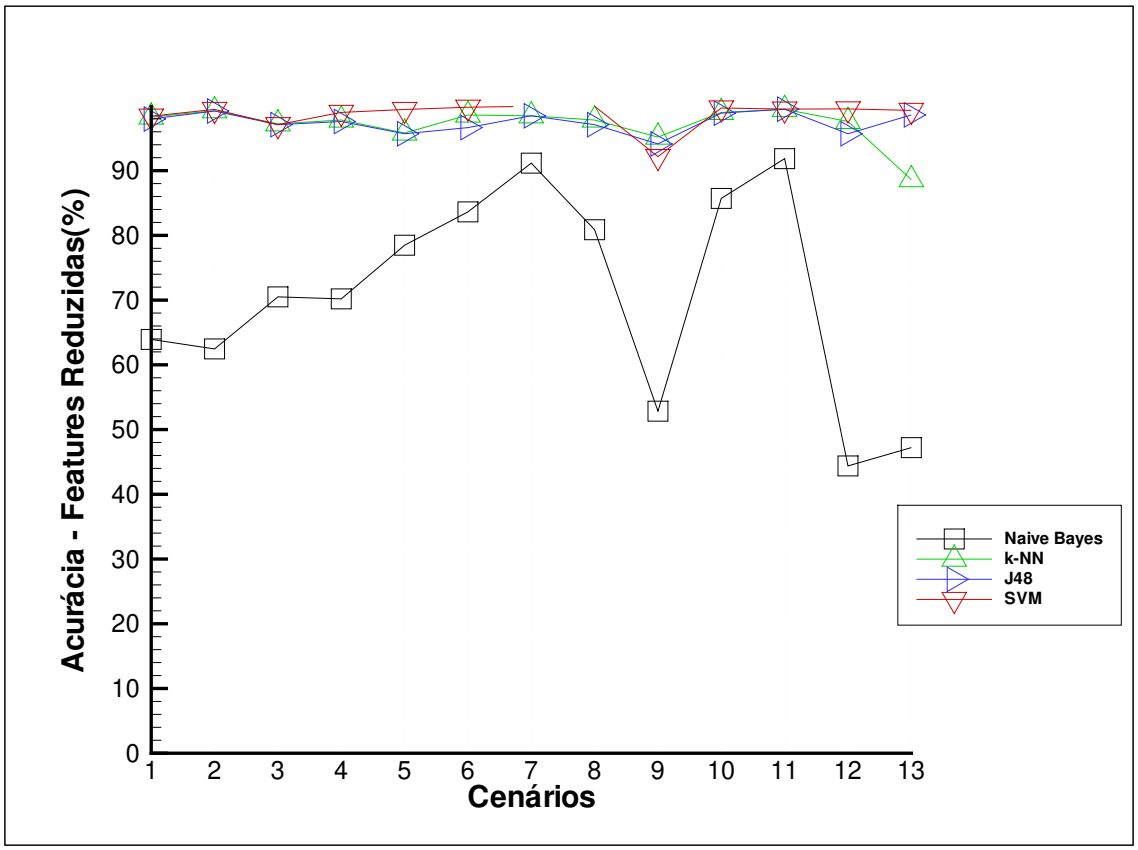

Figura 9. Desempenho dos algoritmos no primeiro experimento - Métrica de Avaliação: Acurácia

Já as Tabelas 1 e 2 apresentam os resultados do segundo experimento. Com o intuito de obter os melhores resultados e evitar *underfitting*, neste experimento, os algoritmos de C_2 tiveram seus hiperparâmetros escolhidos de forma empírica: criando con-

juntos de possíveis combinações destes, e verificando qual gerava melhor resultado para cada classificador. Na Tabela 1 estão os resultados do processo de validação cruzada (com $k = 10$) aplicado aos algoritmos de C_2 sobre um subconjunto S da base de dados. Esse subconjunto foi gerado concatenando os 13 cenários e considerando apenas as classes *botnet* e normal. Tal seleção foi feita tendo em vista que o objetivo do segundo experimento era avaliar a performance dos classificadores em diferenciar a classe *botnet* e a classe normal, e os registros com rótulo *background* não contém informações relevantes sobre a rotulação dessas instâncias. A Tabela 2, por outro lado, contém os desempenhos dos árbitros do conjunto A diante do processo de validação cruzada (com $k = 10$ conjuntos) aplicado sobre S enriquecido com os pareceres dos algoritmos de C_2 sobre cada registro de dados $r \in S$. Os resultados dos testes mostram que, tanto como classificador, quanto como árbitro, o algoritmo *MLP* obteve os piores resultados. Além disso, percebe-se que existe pouca diferença entre os árbitros classificadores e os classificadores do mesmo algoritmo. Tal fato sugere que os dados disponíveis em S assumiram um peso maior que os pareceres dos outros classificadores durante a fase de treinamento dos árbitros.

Tabela 1. Avaliação individual dos algoritmos no segundo experimento - Métrica de Avaliação: Acurácia

Classificador	Acurácia	Botnet	Normal
Decision Tree	93.89%	96.24%	83.07%
SVM	85.11%	98.79%	22.30%
KNN	91.17%	95.96%	69.17%
MLP	69.53%	79.97%	21.55%
Naive Bayes	89.23%	98.96%	44.52%

Tabela 2. Avaliação individual dos árbitros no segundo experimento - Métrica de Avaliação: Acurácia

Árbitro	Acurácia	Botnet	Normal
Votos	91.80%	99.75%	55.32%
Classificador (Decision Tree)	91.46%	96.74%	57.23%
Classificador (SVM)	86.75%	98.19%	34.24%
Classificador (KNN)	91.52%	95.96%	69.17%
Classificador (MLP)	69.53%	79.99%	20.20%
Classificador (Naive Bayes)	89.23%	98.96%	44.52%

Por fim, cabe ressaltar que, como os modelos de classificação gerados por cada algoritmo de aprendizado durante o processo de validação cruzada nos cenários foram armazenados durante os experimentos, os melhores modelos podem ser aproveitados e utilizados para analisar os novos registros de dados monitorados pelo protótipo funcional do EB-CyberDef quando o mesmo estiver operando em ambiente de produção.

5. Considerações Finais

Em um cenário de crescimento do número de ataques perpetrados por diversos tipos de *malware* em todo o mundo, o presente artigo teve por objetivo propor o EB-CyberDef, um ambiente de apoio à defesa cibernética que integra as funcionalidades de detecção e de combate aos comportamentos maliciosos no tráfego de redes de computadores. Além da descrição conceitual dos módulos do EB-CyberDef, este trabalho também apresentou como contribuições a especificação do protótipo implementado e os resultados preliminares obtidos pelos módulos de *Pré-processamento e Análise de Legitimidade* e de *Conjugação de Pareceres*, assim como os melhores modelos de classificação gerados durante os experimentos realizados com dados de uma base sobre *botnets* popularmente utilizada em pesquisas na área. Entre as iniciativas de trabalhos futuros (já em andamento) estão os testes de integração reunindo todos os módulos do EB-CyberDef, assim como testes com dados reais, a fim de mostrar a viabilidade de funcionamento integrado de todos os módulos do protótipo em situações práticas. Também estão em andamento, pesquisas voltadas à compreensão das causas de ocorrência de falsos positivos, a fim de elaborar soluções que possam contribuir para reduzir este tipo de ocorrência.

Referências

Berners-Lee, T. (2019). 30 years on, what's next #fortheweb? Web Foundation.

Clarke, R. and Knake, R. (2010). *Cyber War: The Next Threat to National Security and What to Do About It*. HarperCollins e-books.

Conterno, C. M. and de Oliveira, N. S. F. (2020). *Funções de Proteção Controladas por uma Central de Detecção de Padrões Maliciosos*. Graduação em engenharia de computação computação, Instituto Militar de Engenharia, Rio de Janeiro.

Daisy C. A. Silva, S. S. and Salles, R. (2017). Metodologia de detecção de botnets utilizando aprendizado de máquina. In *Anais do XXXV Simpósio Brasileiro de Telecomunicações e Processamento de Sinais*.

de Araujo, R. T. A. and Ventura, T. B. (2019). *Painel de Apoio para uma Central de Detecção de Padrões Maliciosos*. Graduação em engenharia de computação computação, Instituto Militar de Engenharia, Rio de Janeiro.

de Brito, R. C. and Bezerra, Y. F. (2018). *Comitês de Classificadores Aplicados à Detecção de Padrões Maliciosos no Tráfego de Redes de Computadores*. Graduação em engenharia de computação computação, Instituto Militar de Engenharia, Rio de Janeiro.

Faceli, K., Lorena, A. C., Gama, J., and Carvalho, A. C. P. d. L. F. d. (2011). *Inteligência artificial: uma abordagem de aprendizado de máquina*. LTC.

Farahbod, K., Shayo, C., and Varzandeh, J. (2020). Cybersecurity indices and cybercrime annual loss and economic impacts. *Journal of Business and Behavioral Sciences*, 32(1):63–71.

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

Lunardi, G. (2014). *Mineração de dados no curso pró-conselho/UFSM para a identificação do perfil dos conselhos municipais de educação do RS*. Monography.

Silva, S. and Salles, R. (2012). Arquitetura de um sistema integrado de defesa cibernética para detecção de botnets. In *Anais do XII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, pages 303–309, Porto Alegre, RS, Brasil. SBC.

Silva, S. S., Silva, R. M., Pinto, R. C., and Salles, R. M. (2013). Botnets: A survey. *Computer Networks*, 57(2):378 – 403. Botnet Activity: Analysis, Detection and Shutdown.

Veeramachaneni, K., Arnaldo, I., Korrapati, V., Bassias, C., and Li, K. (2016). Ai2: Training a big data machine to defend. In *2016 IEEE 2nd IEEE Int. Conf. on Intelligent Data and Security (IDS)*, pages 49–54.