CENTRO UNIVERSITÁRIO SAGRADO CORAÇÃO - UNISAGRADO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

ANDERSON BELONI CLARINDO
KETERLY GEOVANA GOUVEIA SILVA

RELATÓRIO FINAL
DETECTANDO ATAQUES DDOS COM INTELIGÊNCIA ARTIFICIAL

Bauru
2022

RESUMO DO PROJETO
A tecnologia avança e com ela também cresce o número de usuários conectados. Neste cenário, ataques virtuais se tornam comuns, sendo direcionados para comércios eletrônicos e negócios online, além de outros serviços disponíveis na Internet. A Segurança da Informação é a área que atua com o objetivo principal de manter sistemas digitais mais seguros, garantindo a confidencialidade, integridade e disponibilidade dos dados dos usuários. Partindo deste ponto, como identificar o comportamento de um servidor durante a ocorrência de ataques como o de negação de serviço, conhecido como DDOS (*Distributed Denial of Service)*? Para responder a esta questão, desenvolveu-se um ambiente controlado para monitoramento de um Servidor Web e simulação de ataques DDOS, e um sistema inteligente para detecção de ataques de inundação ping. Foi criada uma arquitetura com três máquinas Linux, responsáveis respectivamente por representar um servidor Apache, o sistema de monitoramento Zabbix Server e, por fim, a máquina de um atacante com a execução do comando PING e ferramenta T50. O servidor Apache teve seus dados visualizados no servidor Zabbix, através do protocolo SNMP e no programa Wireshark, permitindo uma análise de seu funcionamento antes e após o ataque de negação de serviço. O ataque ao servidor mostrou que o envio de um elevado número de pacotes resulta na indisponibilidade de sites e permitiu que as informações fossem utilizadas no treinamento de um modelo de Regressão Logística capaz de classificar arquivos exportados do Wireshark, e identificar anomalias no tráfego de rede. Este estudo trouxe como principal resultado a visualização das consequências do ataque DDOS em um servidor e de que forma a utilização de Machine Learning pode contribuir com o avanço de ferramentas na prevenção de ciberataques.

Palavras-Chave: Inteligência Artificial; Cibersegurança; Desenvolvimento *Web*

1. INTRODUÇÃO E JUSTIFICATIVA

Todos os dias a internet é utilizada por milhões de pessoas em todo mundo. A tendência à virtualização de serviços já era visível mesmo antes da pandemia, quando bancos digitais, lojas virtuais e plataformas de cursos online impactaram de forma positiva a vida de seus usuários. Com o início da pandemia da COVID-19 essa tendência ficou ainda mais evidente, e, segundo dados da pesquisa realizada pelo Centro Regional de Estudos para o Desenvolvimento da Sociedade da Informação (2021), o uso da internet cresceu durante a pandemia e tem 81% da população com 10 anos ou mais conectada a ela.

Com o avanço das tecnologias a segurança na internet e em sistemas de computadores é cada dia mais requisitada, buscando sempre a integridade dos dados de todos aqueles que usam e se beneficiam desses recursos digitais. Como explica o Portal da Indústria (2021), cibersegurança é a prática de proteger ativos de informações tais como sistemas, computadores e servidores entre outras contra ameaças cibernéticas ou ataques maliciosos, seguindo os pilares da segurança da informação sendo eles a

confidencialidade, integridade e disponibilidade das informações.

A empresa de segurança cibernética Fortinet (2021) realizou a publicação de um levantou dados de 2020 referente a ameaças cibernéticas no Brasil, onde foi constatado que ela sofreu mais de 8,4 bilhões de tentativas de ataques cibernéticos, sendo um número que representa cerca de um pouco mais de 20% dos casos registrados em toda América Latina, somados em 41 bilhões de casos. Destes casos de ataques digitais um dos mais praticados em todo mundo é o conhecido como ataque distribuído de negação de serviço, popularmente chamado de ataques DDOS (*Distributed Denial of Service*), que tem como objetivo o envio em massa de solicitações a um servidor ou serviço *web*, impedindo sua capacidade de processamento e impossibilitar que usuários o acessem. O ataque distribuído provém de várias origens que dificultam as tentativas de bloqueá-lo e aumenta a eficácia.

Esses ataques são gerados de diversas maneiras sendo uma das mais comuns a por meio das chamadas *botnets*, um programa malicioso que pode ser controlado remotamente por um criminoso digital, uma particularidade deste modelo de ataque é permitir que um atacante execute instruções em muitos computadores infectados com o malware simultaneamente.

Em contrapartida, o avanço da tecnologia também permitiu o estudo e propostas de soluções para detecção e mitigação de ataques como os de Negação de Serviço utilizando Aprendizagem de Máquina, um ramo da Inteligência Artificial.

Segundo Géron (2017, apud LIMA FILHO, 2019) o aprendizado de máquina é uma forma de fazer com que o computador aprenda com os dados de forma contínua, permitindo, por exemplo, a classificação supervisionada através do treinamento com uma base rotulada e da identificação de padrões, de sua validação a partir de métricas e da aplicação do modelo em novos conjuntos de dados. Uma classificação pode dizer se um tráfego é do tipo Normal ou Anormal através da análise de amostras contendo informações da rede e da identificação de uma mudança no padrão desses dados.

Partindo dessa exposição, este trabalho define um problema: Como identificar o comportamento de um servidor durante a ocorrência de ataques DDOS e utilizar Inteligência Artificial para detectar anomalias em requisições de sistema?

Ataques hackers no geral causam prejuízos a internautas e empresas do mundo todo. A exemplo do Brasil, o ataque ao Ministério da Saúde que deixou servidores sem acesso e seus sistemas indisponíveis impactou diretamente a vida de milhões de brasileiros que necessitavam utilizar informações do sistema público de saúde, como,

por exemplo, certificado de vacinação da COVID 19.

A utilização de Inteligência Artificial para identificar mudanças no padrão de requisições a rede é importante pois visa detectar o ataque DDOS e notificar especialistas de segurança para que seja mitigado, evitando prejuízos financeiros e danos às imagens das marcas, que, após demonstrarem suas vulnerabilidades, tendem a perder a credibilidade no mercado em que atuam e afeta a confiança de seus clientes.

2. OBJETIVOS

Geral: Desenvolver um sistema inteligente para classificação de ataques DDOS em servidores Linux

Específicos:

- Realizar ataques em ambiente controlado
- Desenvolver um modelo de aprendizado de máquina para a classificação de ataques
- Implementar um sistema para visualização e classificação de novos dados

3. METODOLOGIA

Este projeto foi desenvolvido em duas etapas: uma etapa teórica para escolha das melhores tecnologias e uma etapa prática para implementação da solução.

A etapa teórica consistiu na pesquisa de referências bibliográficas para o embasamento do projeto e da escolha de *softwares*, linguagem de programação e outras tecnologias a serem utilizadas na prática. Por ser um trabalho com dedicação de dois semestres para sua conclusão, a fase de planejamento é extremamente necessária para que seja possível a sua continuidade. Definiu-se que, para o primeiro semestre, seria criado o ambiente controlado para ataque e monitoramento, desenvolvido um site para receber o ataque DDOS e um *dataset* para o treinamento da inteligência artificial.

Para a criação do ambiente controlado, foi escolhido o *software* VirtualBox, o qual permite a criação de máquinas virtuais em sistemas operacionais como Windows, Linux e suas distribuições. Neste projeto foram utilizadas três máquinas virtuais:

1. Máquina Virtual Ubuntu com servidor Apache configurado para hospedagem do site que receberá o ataque. A máquina também teve o programa Wireshark instalado para análise do tráfego da rede.
2. Máquina Virtual com servidor Zabbix configurado para monitoramento do servidor Apache através do protocolo SNMP (*Simple Network Management Protocol*).

3. Máquina Virtual Kali e ferramenta t50 para a simulação dos ataques DDOS.

HTML, *CSS* e *Javascript* foram escolhidas para o desenvolvimento do site. As linguagens de marcação, estilo e programação foram utilizadas para transformar o *Layout* criado na ferramenta Figma, em um site possível de ser hospedado. A programação foi feita no *Visual Studio Code*, um editor de código-fonte.

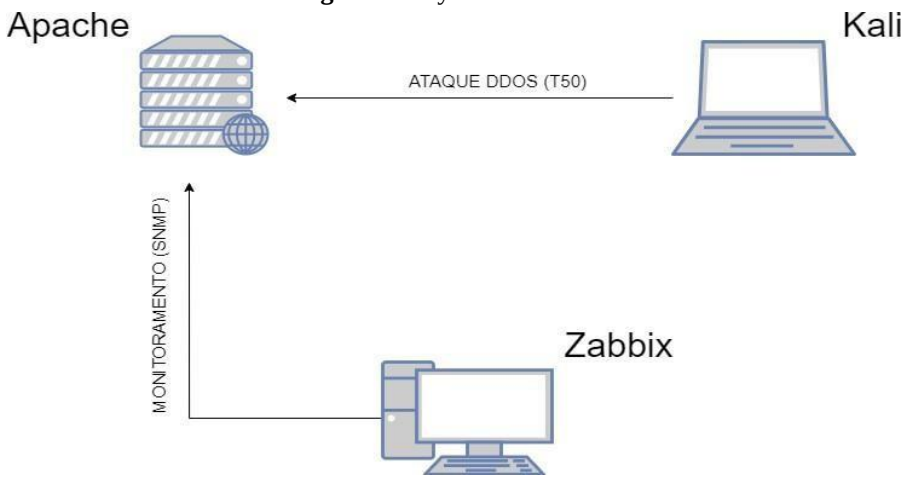
Como primeira alternativa para a criação do *dataset* de treinamento do modelo de detecção de ataques, as análises feitas pelo Wireshark foram exportadas como um arquivo CSV (Valores Separados por Vírgula).

Nesta fase teórica e de planejamento também ocorreu a definição das tecnologias, linguagens e ferramentas para o próximo semestre

1. Jupyter Notebook para compilar trechos de códigos
2. Python, linguagem para o desenvolvimento de todo o algoritmo de detecção
3. Pandas, Numpy, Scikit-learn, Tensorflow, matplotlib, entre outras bibliotecas Python, APIs e algoritmos dessas bibliotecas para seja possível desenvolver, treinar, comparar resultados, avaliar e salvar o modelo de detecção
4. *Framework* Flask para o deploy do modelo em produção
5. SQLAlchemy para a conexão com o banco de dados e realização do login no sistema que mostrará as classificações do modelo
6. MySQL como banco de dados relacional

A etapa de implementação teve início com a criação do layout da arquitetura do ambiente controlado do projeto. A figura 1 mostra as máquinas virtuais necessárias, como se dá a conexão entre elas.

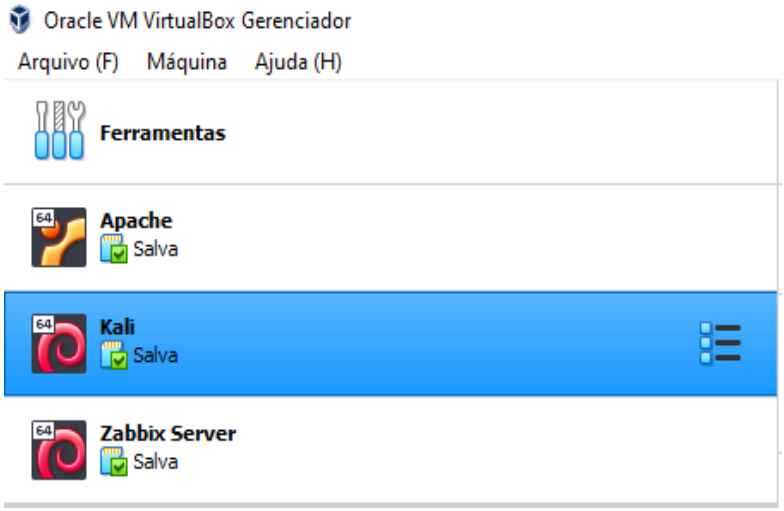
Figura 1: Layout do ambiente



Fonte: Imagem dos autores, 2022, 2022

Com o layout em mãos, as máquinas virtuais começaram a ser criadas. Antes da configuração específica para cada uma delas, todas foram criadas adicionando a *ISO* de suas distribuições Linux: Ubuntu para o servidor Apache, Kali para realizar o ataque e Debian para o servidor Zabbix. A figura 2 mostra a *interface* do VirtualBox com as máquinas virtuais criadas e nomeadas inicialmente.

Figura 2: Máquinas do VirtualBox



Fonte: Imagem dos autores, 2022

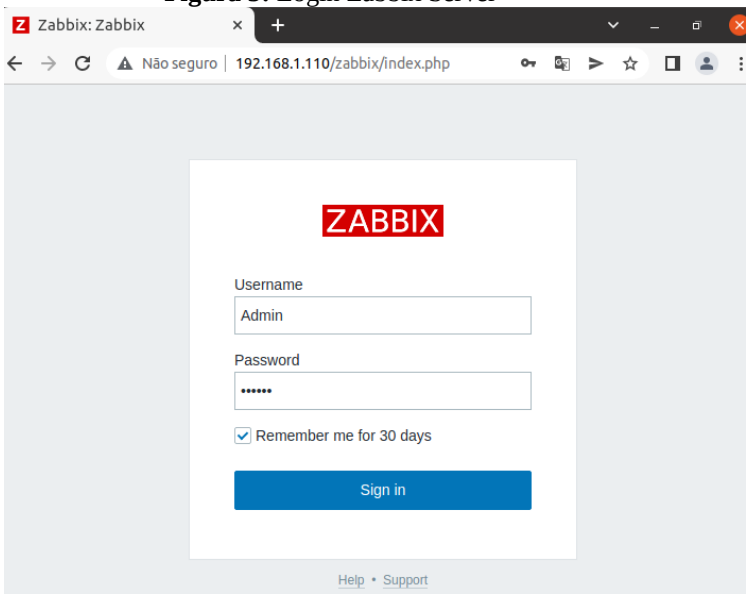
Na máquina Apache foram instalados o servidor Apache e o programa Wireshark, responsáveis respectivamente pela hospedagem do site e pela análise do tráfego de rede. Para o teste inicial da configuração do servidor foi hospedado um arquivo simples contendo poucas palavras, o qual pode ser acessado pelo IP*(Internet Protocol)* correspondente ao *localhost* (127.0.0.1). Outras configurações foram feitas nesta máquina, como a instalação do serviço SNMP e definição de um IP fixo para o monitoramento do *host(*hospedeiro*)* através do Zabbix Server.

A máquina Zabbix começou a ser trabalhada com a instalação do Zabbix Server via pacote, bem como as suas configurações específicas e indicadas na documentação oficial da ferramenta. Durante este processo de configuração do ambiente, alguns erros foram apresentados referentes ao MySQL Server, banco de dados integrado ao Zabbix e necessário para a sua utilização. Por conta disso, uma pesquisa para a resolução deste problema foi realizada e identificou-se que a troca da distribuição Linux era uma solução viável para a correção do erro.

O servidor passou a ser instalado em uma máquina Ubuntu e após configurações especificas como criação de uma base de dados, usuário, senha, mudança do time zone e a escolha de um IP fixo inserido manualmente junto a sua máscara de rede e *Gateway*,

foi possível navegar até a interface Zabbix utilizando a *URL* (Uniform Resource Locator) padronizada como http://IP_ou_nome/zabbix e finalizar definitivamente as configurações. Isso possibilitou o acesso a tela de login conforme mostra a figura 3

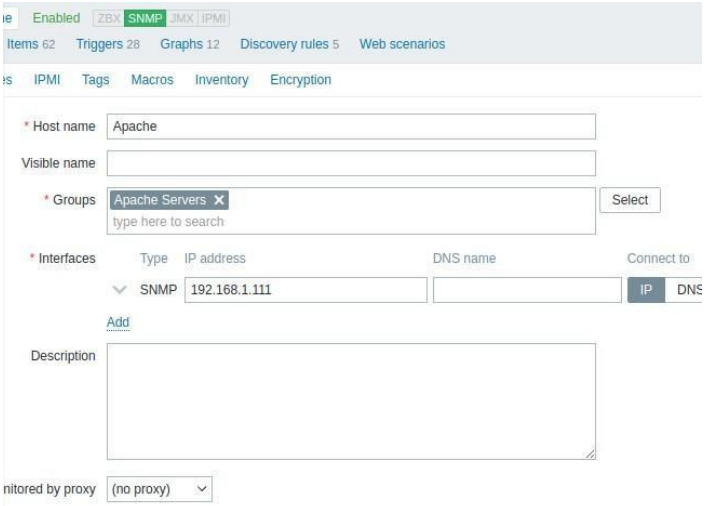
Figura 3: Login Zabbix Server



Fonte: Imagem dos autores, 2022

Com o servidor funcionando e pronto para monitorar, foi necessário fazer novas configurações na máquina Apache: escolha de um IP fixo, instalação do agente SNMP, mudança no arquivo de configuração do agente com a inserção do IP do servidor Zabbix nos parâmetros *agentaddress (endereço do agente)* e na *community*. Além disso, a máquina Zabbix também precisou que o SNMP fosse instalado. Tais alterações permitiram que um *host* fosse criado no Zabbix Server e as informações do servidor Apache pudessem ser coletadas, conforme mostram as figuras 4 e 5

Figura 4: Criação do host Apache



Fonte: Imagem dos autores, 2022

Figura 5: Monitoramento Apache com dados mais recentes

Name ▲	Last check	Last value
CPU (17 Items)		
Context switches per second	2022-03-28 09:12:02	237.5255
CPU guest nice time	2022-03-28 09:12:02	0 %
CPU guest time	2022-03-28 09:12:02	0 %
CPU idle time	2022-03-28 09:12:02	97.7449 %
CPU interrupt time	2022-03-28 09:12:02	0 %
CPU iowait time	2022-03-28 09:12:02	0.01665 %
CPU nice time	2022-03-28 09:12:02	0 %
CPU softirq time	2022-03-28 09:12:02	0 %
CPU steal time	2022-03-28 09:12:02	0 %
CPU system time	2022-03-28 09:12:02	0.03331 %
CPU user time	2022-03-28 09:12:02	0.06662 %
CPU utilization	2022-03-28 09:12:02	2.2551 %
Interrupts per second	2022-03-28 09:12:02	165.595
Load average (1m avg)	2022-03-28 09:12:02	0.01

Fonte: Imagem dos autores, 2022

Por fim, ocorreu uma primeira tentativa de configuração da máquina Kali para a simulação do ataque DDOS, que necessita da ferramenta t50. A ferramenta apresentou erros que não permitiram a conclusão de sua instalação e, para solucioná-lo, uma versão mais antiga da distribuição foi baixada para a criação de uma nova máquina virtual. Não foi mais necessário a instalação do t50 pois, nesta versão, a ferramenta veio por padrão.

Foram simulados alguns ataques utilizando a linha de comando do Kali e foi observado que estes ataques, mesmo que direcionados ao IP da máquina virtual, também atingiam a máquina física/local. A figura 6 mostra como ficaram as máquinas virtuais após a correção dos erros e é possível notar que as distribuições agora são apenas Ubuntu e Kali.

Figura 6: Novas distribuições Linux

Fonte: Imagem dos autores, 2022

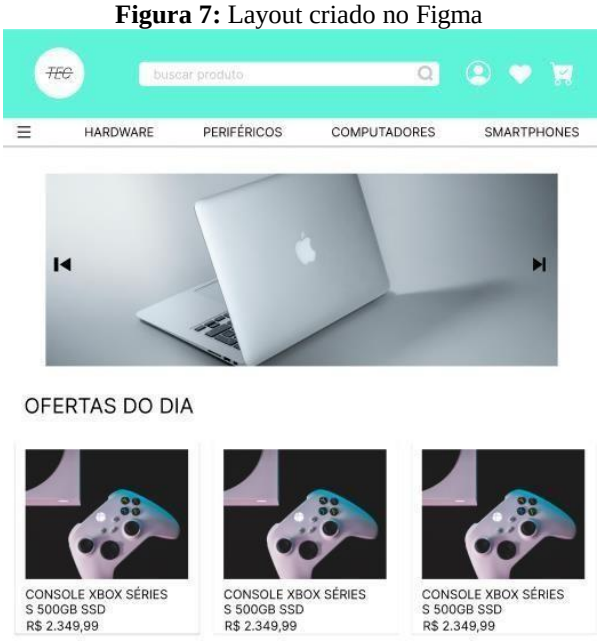
Com a identificação do problema na máquina local, decidiu-se pela utilizados dois computadores para o projeto. Um com as máquinas virtuais Zabbix e Kali, e outro

com Linux nativo para o servidor Apache. Dessa forma, passou a ser direcionado ao segundo computador (Apache).

A conexão das máquinas na rede *WIFI* do UNISAGRADO também apresentou problemas. A rede realizou alguns bloqueios, e isso impediu que os computadores se comunicassem da forma como deveriam. Para solucionar, um roteador começou a fazer parte do ambiente. Ele foi e continuará sendo responsável pela distribuição dos IPs, garantindo que todas as máquinas estejam na mesma rede.

Nesta etapa inicial de implementação também foi desenvolvido um *layout* do site que sofrerá os ataques. Na ferramenta Figma foram escolhidas cores, larguras, alturas e ícones do *website*, existindo a possibilidade de alterações conforme o desenvolvimento. Com o *layout* finalizado, a programação foi iniciada utilizando *HTML, CSS e Javascript*, algo simples para simular visualmente a tela inicial de um *e-commerce (*comércio eletrônico*)*. Por ter foco no *front-end*, o site é estático, sem lógica para login, compra de produções e armazenamento de favoritos no banco de dados. Ele foi desenvolvido no modelo página única, com botões e *links* não navegáveis. A figura 7 mostra como ficou o layout criado na ferramenta.

Figura 7: Layout criado no Figma



Fonte: Imagem dos autores, 2022

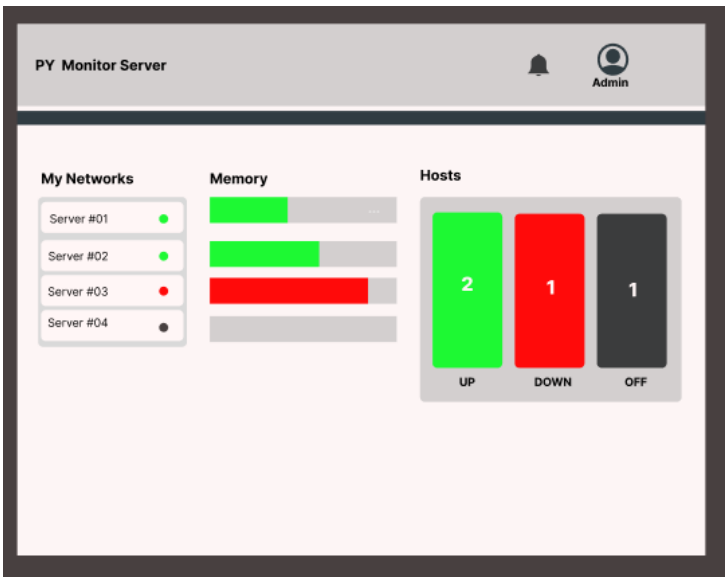
A etapa final da implementação do primeiro semestre consistiu na realização de acessos ao site hospedado. Os acessos “normais” servirão de dados para uma das classes utilizadas para o treinamento do modelo de detecção em etapas futuras. Estes acessos, analisados pelo Wireshark, foram salvos em um arquivo csv. O primeiro arquivo de teste foi exportado para a identificação de quais campos seriam pertinentes para o

modelo de IA no semestre seguinte e de que forma ele poderia ser manipulado utilizando a biblioteca Pandas. Ataques através do Kali foram feitos e direcionados ao servidor Apache para a observação do comportamento da máquina em questão de recurso de *Hardware*, como utilização de memória e CPU (Central Processing Unit*).* Tais acessos "anormais" também foram analisados e salvos em outro arquivo CSV para compor a segunda classe utilizada no treinamento. A coleta desses acessos gerou dados e foi, juntamente com a criação do ambiente, a realização dos ataques e o monitoramento do servidor Apache através do Zabbix Server, o trabalho para o primeiro semestre.

O desenvolvimento das etapas no segundo semestre iniciou-se com a divisão de duas frentes: uma para a implementação do sistema inteligente e outra para a criação do modelo de aprendizado de máquina.

Para a programação do sistema inteligente, um *layout* simples foi criado utilizando a ferramenta Figma. Diferente da utilização para o site que sofreu com os ataques anteriormente, este trouxe possíveis elementos, uma vez que a inserção das informações dependerá do sucesso no uso de uma API, que será explicada posteriormente. A figura 8 mostra o layout inicial para a tela principal

Figura 8: Layout criado no Figma



Fonte: Imagem dos autores, 2022

O sistema inteligente foi pensado para profissionais da área de segurança, com conhecimentos específicos sobre redes e mitigação de ataques e por isso contém termos técnicos como "*Host*" e a utilização do IP ao invés do nome de um servidor. Uma vez que apenas os profissionais da área tem acesso a informações do trafego de rede de um servidor em uma empresa, as funcionalidades foram escolhidas para que estes

profissionais possam utilizar o modelo de aprendizado de máquina para detectar de forma mais rápida as anomalias contidas nas informações extraidas do Wireshark.

Com o *layout* e o publico alvo definido, a programação foi iniciada. Para ser utilizada apenas uma linguagem *back-end*, optou-se pela criação do projeto em Flask, um *framework* Python para aplicações web. Além desta linguagem e do *framework*, foram aplicadas tecnologias como HTML, CSS, Javascript e Bootstrap. O banco de dados SQLITE também foi escolhido para o armazenamento dos dados dos usuários para login, cadastro, edição e exclusão de analistas. Em etapas futuras, a base contará com tabelas para armazenamento dos dados de treinamento do modelo de detecção, além dos novos dados que poderão ser classificados através do sistema de monitoramento.

O sistema foi desenvolvido com páginas para a visualização geral dos servidores monitorados, além de página para edição de dados pessoais, login e notificação. Pensando na segurança da aplicação, foi adicionada uma validação para que as páginas apenas fossem acessadas por usuários logados. Páginas específicas para cadastro e exclusão de analistas também foram criadas, mas, neste caso, acessíveis apenas para um administrador geral.

SQLITE, o banco de dados escolhido neste trabalho, inicialmente foi criado com apenas uma tabela de usuários. A tabela ficou responsável pelo armazenamento de nome, email e senha, além do identificador único de cada registro. Para a senha foi adicionada a criptografia MD5, gravando esta informação como uma *hash*.

Do lado do *back-end* o sistema inicial foi desenhado para permitir a consulta de usuários, verificação para login, cadastro de analista, exclusão e edição. Definiu-se que posteriormente também seria criada a funcionalidade de notificação. Para o sistema também foram definidas restrições, permitindo que apenas o administrador geral fosse capaz de definir uma nova senha a um cadastro existente, sendo informado ao usuário no momento do clique de “Esqueceu a senha”, que ele deve procurar o administrador da conta.

Ao fim do primeiro semestre, planejou-se que o modelo de classificação seria treinado com dados obtidos da ferramenta Wireshark, e que a API da ferramenta Zabbix também poderia ser utilizada para este fim. No entanto, Zabbix Server apresentou problemas para o monitoramento via SNMP e outros protocolos, o que impediu a coleta das informações.

A correção deste erro foi pesquisada e como alternativa, foi decidido que uma nova base de dados seria gerada através do ataque DDOS de inundação Ping. Analisando o tamanho dos pacotes enviados, a detecção de uma mudança no padrão poderá permitir que o modelo aprenda a identificar este tipo de ataque.

Foi realizada a geração de mais de 2 mil linhas de informações sobre a rede, em seu estado normal e anormal, e não apenas registros do protocolo ICMP. A figura 9 mostra a visualização da base como um *dataframe* em Python.

Figura 9: Dataframe Python

	No.	Time	Source	Destination	Protocol	Length	Info
4	5	25.522407	192.168.1.106	192.168.1.112	ICMP	1042	Echo (ping) request id=0x1777, seq=13/3328, t...
5	6	25.522452	192.168.1.112	192.168.1.106	ICMP	1042	Echo (ping) reply id=0x1777, seq=13/3328, t...
28	29	105.538407	192.168.1.106	192.168.1.112	ICMP	1042	Echo (ping) request id=0x1777, seq=93/23808, ...
29	30	105.538452	192.168.1.112	192.168.1.106	ICMP	1042	Echo (ping) reply id=0x1777, seq=93/23808, ...
87	88	229.034405	192.168.1.106	192.168.1.112	ICMP	1042	Echo (ping) request id=0x1830, seq=41/10496, ...
...	...	...	...	...	...	...	...
508	3772	494.156516	192.168.1.112	192.168.1.106	ICMP	98	Echo (ping) reply id=0x0d5d, seq=170/43520,...
509	3774	495.142099	192.168.1.112	192.168.1.106	ICMP	98	Echo (ping) request id=0x097b, seq=123/31488,...
510	3775	495.142150	192.168.1.106	192.168.1.112	ICMP	98	Echo (ping) reply id=0x097b, seq=123/31488,...
511	3776	495.156020	192.168.1.106	192.168.1.112	ICMP	98	Echo (ping) request id=0x0d5d, seq=171/43776,...
512	3777	495.159625	192.168.1.112	192.168.1.106	ICMP	98	Echo (ping) reply id=0x0d5d, seq=171/43776,...

34433 rows × 7 columns

Fonte: Imagem dos autores, 2022

Tendo Python, Pandas e Sklearn como escolhas de tecnologia, optou-se pelo uso de modelos de regressão para a criação do modelo responsável pela detecção dos ataques DDOS. Os dados gerados foram divididos em dados de treino e teste, sendo 30% destas informações dedicadas para o teste do modelo.

A programação do modelo se deu através do Google Colab, pela possibilidade de compartilhamento e armazenamento em nuvem. As informações da rede foram visualizadas como um dataframe e, para dar início ao seu desenvolvimento, decidiu-se que apenas os dados referentes ao protocolo ICMP, ICMPv6 e IPv4. Após essa primeira etapa, o *dataframe* apresentou 1688 linhas e 7 colunas de informações relevantes para o treinamento do modelo de regressão.

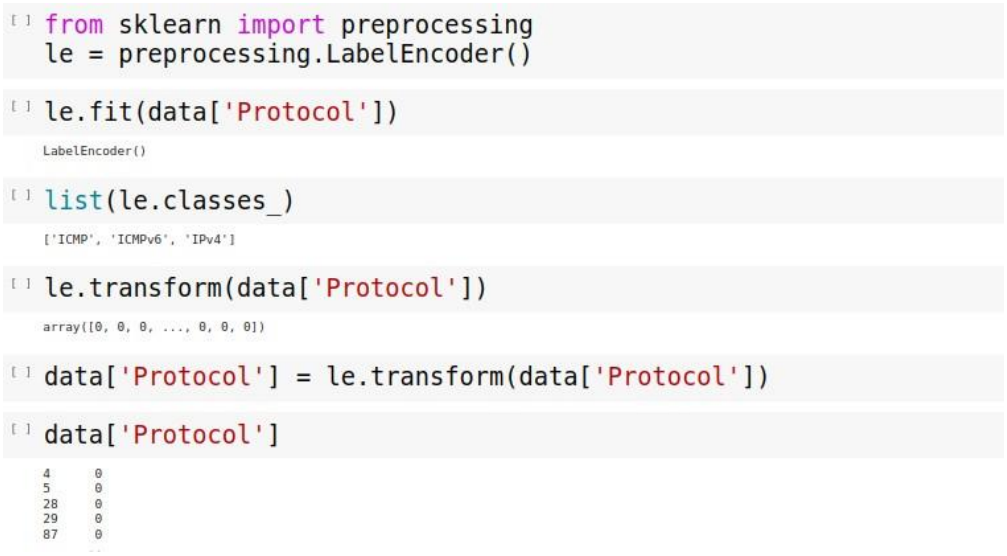
Posteriormente, foi necessário classificar todas as linhas presentes no dataframe. No aprendizado de máquina supervisionado, a classificação supervisionada pelo próprio desenvolvedor também faz parte de uma etapa inicial. A partir dela o modelo é capaz de realizar comparações e aprender a qual classe uma determinada informação pertence.

Para este conjunto de dados, duas classes foram definidas. Na programação, "*True*" representa dados com a classe "Ataque" e com um envio de pacotes com tamanho maior do que 100, e "*False*" para "Normal" quando pacotes de tamanho menor

do que 100 são enviados através dos protocolos escolhidos.

Após a conclusão da rotulação, a etapa de pré-processamento se iniciou e resultou na conversão dos nomes dos protocolos. Para treinar o modelo, os protocolos passaram a ser representados por números. Isso foi feito com a aplicação do *LabelEncoder*, uma função presente em um dos módulos da biblioteca Sklearn. A figura 10 demonstra a execução desta etapa.

Figura 10: *Encoding* dos protocolos



Fonte: Imagem dos autores, 2022

Com os dados pré-processados e separados para treinamento e teste, um modelo de regressão logística foi criado por meio da função *LogisticRegression()*, também presente na biblioteca Sklearn. Alguns parâmetros como *penalty*, *class_weight* e *random_state* foram definidos e permitiram que o modelo aprendesse e fosse capaz de prever novos dados. A figura 11 apresenta um trecho do código em Python para a criação do modelo

Figura 11: Criação do modelo de classificação

Fonte: Imagem dos autores, 2022

Após o modelo ser treinado, testado e avaliado, foi realizado o *deploy* para a aplicação web desenvolvida em Flask. Com essa adição, novas tabelas foram criadas no banco de dados, sendo responsáveis por armazenar as novas classificações e as

notificações exibidas no sistema.

A ideia inicial para a classificação de novos dados era a realização do acompanhamento da rede em tempo real. No entanto, optou-se pela classificação de arquivos CSV. A ferramenta Wireshark, por permitir a exportação das informações, possibilita que estes dados sejam salvos e anexados posteriormente no sistema pelo analista de segurança. O *back-end* foi construído para ler o arquivo, obter os dados referentes aos três protocolos analisados e retornar, após consumir o modelo de regressão, uma classificação.

Cada linha do dataframe é classificada e, para definir se o arquivo lido indica um possível ataque ou não, foi definido que o retorno seria positivo para arquivos com pelo menos 3 linhas classificadas como ataque e com *score* de no mínimo 0,8, o que indica que o modelo tem 80% de certeza de que o dado pertence a classe. Após a classificação, o sistema foi capaz de retornar para o usuário se um ataque foi detectado ou não.

Para ataques detectados, o sistema apresentou uma tabela na tela de notificações com o IP do *host* e a hora em que a notificação foi criada. Além da tela, um *bot* no Telegram foi configurado para enviar automaticamente uma mensagem informando a detecção, a data, horário e IP do *host*. Uma mensagem indicando que o analista confirmou o ataque após a classificação também pode ser enviada.

Devido ao problema encontrado no servidor Zabbix e que não permitiu que a ferramenta fosse utilizada para apresentar os gráficos da tela inicial, foi definida uma nova forma e popular esta área. A tela inicial passou a apresentar uma lista com os últimos *host*s analisados, contendo o IP e o Status, e um gráfico indicando quantos ataques foram confirmados ou não confirmados pelo analista de segurança após a detecção. Também foi indicado o número de arquivos classificados como “Ataque” e “Normal”, sendo todas essas informações referentes ao mês atual.

O sistema, nomeado como “Py Monitor Server” foi hospedado de forma gratuita na plataforma Render, dando a possibilidade de ser acessado de qualquer lugar.

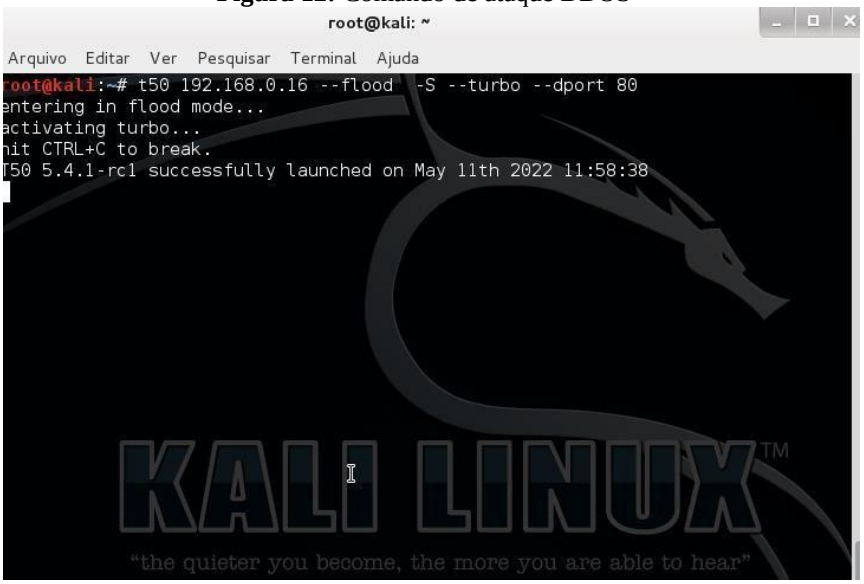
4. RESULTADOS E DISCUSSÕES

A criação de uma arquitetura do ambiente controlado foi o primeiro resultado a ser alcançado, mas que precisou ser remodelado à medida que algumas dificuldades surgiram. Com a mudança que permitiu que todas as máquinas se comunicassem em uma mesma rede, um novo computador e um roteador passaram a fazer parte do ambiente. O roteador distribui IPs, o primeiro computador manteve as máquinas Kali e

Zabbix Server e o segundo computador hospedou o servidor Apache. O Kali Linux possui uma gama de ferramentas já implementadas para análise e desenvolvimento de técnicas de *Pentest* e com a simulação de uma máquina atacante foi possível direcionar pacotes de rede para o Servidor *Web*, utilizando a ferramenta específica T50.

A figuras 12 mostra o comando executado na ferramenta para atingir a máquina de IP 192.168.0.16 neste exemplo. A porta 80 indica o protocolo *HTTP*.

Figura 12: Comando de ataque DDOS



Fonte: Imagem dos autores, 2022

O ataque feito modificou informações do *Hardware* da máquina Apache e, antes mesmo do uso de uma ferramenta própria para monitoramento, foram observadas algumas alterações. As figuras 13 e 14 mostram respectivamente o uso de alguns recursos da máquina antes do ataque DDOS e suas informações após o ataque. É possível notar que os valores de utilização se alteram, saindo de 3% para 49,5%.

Figura 13: Informações antes do ataque DDOS

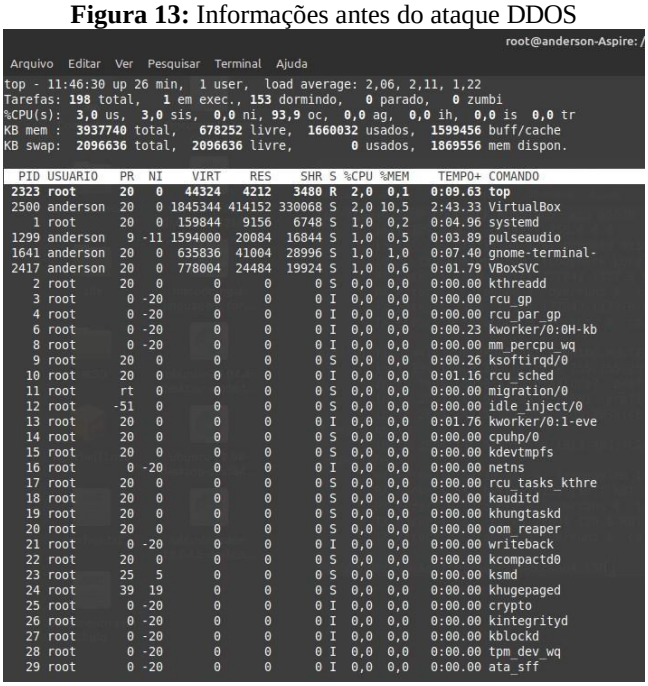
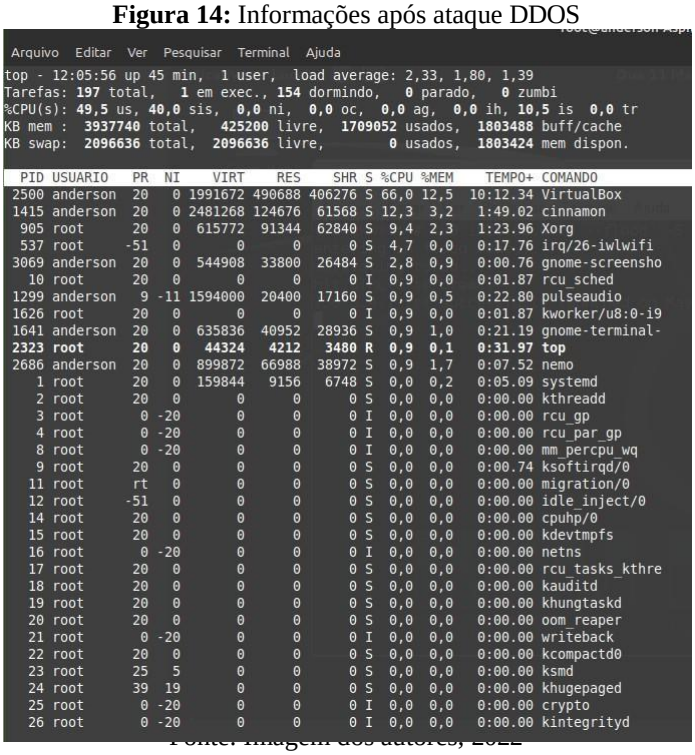


Figura 14: Informações após ataque DDOS



Fonte: Imagem dos autores, 2022

Pouco tempo depois do início do ataque DDOS direcionado ao servidor Apache, onde se encontrava a página *web* hospedada, não foi possível obter respostas. A figura 15 mostra que a página se tornou inacessível devido ao número de pacotes enviados ao servidor por meio da execução do comando na ferramenta t50.

Figura 15: Página web após ataque DDOS

Fonte: Imagem dos autores, 2022

Sendo o monitoramento um outro objetivo, a máquina Zabbix Server foi inserida A figura 16 ilustra a interface Zabbix que apresenta os últimos valores coletados de um *host*.

Figura 16: Últimos dados coletados do host Apache

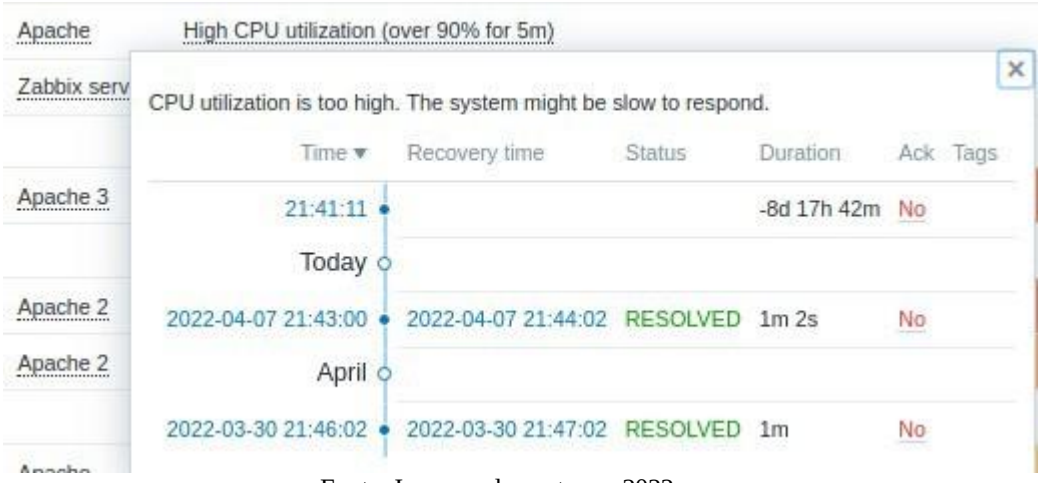
Interface enp1s0(): Outbound packets discarded	2022-04-21 03:48:10	0
Interface enp1s0(): Outbound packets with errors	2022-04-21 03:48:10	0
Interface enp1s0(): Speed	2022-04-21 03:15:04	100 Mbps
Memory (9 Items)		
Available memory	2022-04-21 03:54:35	2.43 GB
Free memory	2022-04-21 03:49:04	642.33 MB
Free swap space	2022-04-21 03:49:04	2 GB
Free swap space in %	2022-04-21 03:54:32	99.9872 %
Memory (buffers)	2022-04-21 03:49:04	66.64 MB
Memory (cached)	2022-04-21 03:49:04	1.73 GB
Memory utilization	2022-04-21 03:54:40	35.3753 %
Total memory	2022-04-21 03:49:04	3.76 GB
Total swap space	2022-04-21 03:49:04	2 GB
Status (displaying 1 to 2 of 5 Items)		
ICMP loss	2022-04-21 03:54:06	100 %
ICMP ping	2022-04-21 03:54:06	Down (0)

Fonte: Imagem dos autores, 2022

A ferramenta de monitoramento identifica dados de memória, CPU, status de disponibilidade e outras informações que são recolhidas através do protocolo SNMP. A configuração correta do Zabbix Server permitiu que alguns alertas fossem gerados e que, quando a máquina não estivesse acessível – o caso da imagem acima – o status *ping* apresentasse nomenclatura '*down*'.

Também foi possível observar os alertas gerados pelo pico de utilização da CPU durante o ataque DDOS. A figura 17 exibe um desses alertas, indicando que a utilização chegou em 90% e durou 5min. O status de resolvido representa o término do ataque, quando os valores voltam a sua normalidade.

Figura 17: Alerta de utilização de CPU

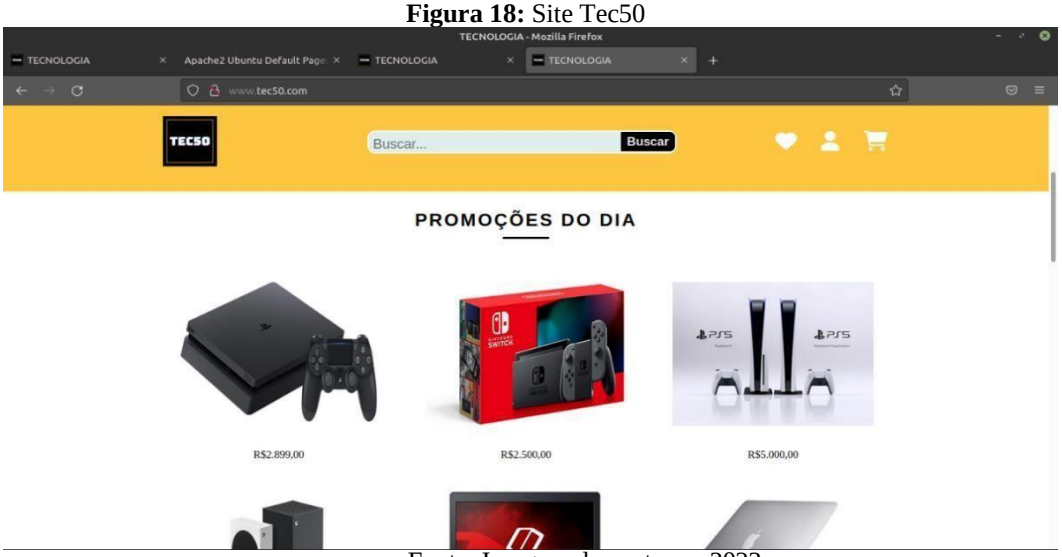


Fonte: Imagem dos autores, 2022

Buscando a criação de um site para simular visualmente uma loja virtual, a interface *web* foi criada seguindo parcialmente o layout inicial, com mudança nas cores e remoção de estilos devido a problemas na configuração do servidor Apache, o qual fez

com que definições de *CSS* não fossem ao ar. O site pode ser hospedado e acessado através da *URL* www.tec50.com, a qual apontava para o endereço local 127.0.0.1. A figura 18 mostra como ficou a versão final do site após hospedagem, sendo a conclusão de mais um dos objetivos definidos para este trabalho.

Figura 18: Site Tec50



Fonte: Imagem dos autores, 2022

Por fim, a utilização do Wireshark permitiu a coleta de dados do tráfego de rede. Foram obtidas informações como número de IP, protocolo e tamanho do pacote como demonstra a figura 19.

Figura 19: Wireshark

No.	Time	Source	Destination	Protocol	Length	Info
139	70.437203786	192.168.1.112	186.224.0.18	DNS	85	Standard query 0xc2e2 AAAA www.google.com OPT
140	70.437411356	192.168.1.112	186.224.0.18	DNS	90	Standard query 0x7ab2 AAAA support.mozilla.org C
141	71.334189134	192.168.1.105	224.0.0.251	MDNS	103	Standard query 0x0005 PTR _233637DE._sub._google
142	71.663349605	SamsungE_c9:c9:57	Broadcast	ARP	60	Who has 192.168.1.254? Tell 192.168.1.100
143	71.740739943	192.168.1.112	186.224.0.20	DNS	85	Standard query 0xa80d A www.google.com OPT
144	74.633086712	SamsungE_c9:c9:57	Broadcast	ARP	60	Who has 192.168.1.254? Tell 192.168.1.100
145	75.201138504	192.168.1.112	186.224.0.18	DNS	86	Standard query 0x0150 A www.gstatic.com OPT
146	75.201262962	192.168.1.112	186.224.0.20	DNS	86	Standard query 0x19a9 AAAA www.gstatic.com OPT
147	75.444353098	192.168.1.112	186.224.0.18	DNS	90	Standard query 0x7ab2 AAAA support.mozilla.org C
148	75.444522184	192.168.1.112	186.224.0.20	DNS	85	Standard query 0xc2e2 AAAA www.google.com OPT
149	75.444625830	192.168.1.112	186.224.0.18	DNS	90	Standard query 0x20a3 A support.mozilla.org OPT
150	76.544002035	192.168.1.112	172.217.28.4	TCP	74	[TCP Retransmission] 52648 → 443 [SYN] Seq=0 Wir
151	76.990731659	192.168.1.112	186.224.0.20	DNS	85	Standard query 0xa80d A www.google.com OPT
152	77.602582652	SamsungE_c9:c9:57	Broadcast	ARP	60	Who has 192.168.1.254? Tell 192.168.1.100
153	80.209628907	192.168.1.112	186.224.0.18	DNS	86	Standard query 0x19a9 AAAA www.gstatic.com OPT
154	80.209755600	192.168.1.112	186.224.0.20	DNS	86	Standard query 0x0150 A www.gstatic.com OPT
155	80.384015057	192.168.1.112	142.250.218.3	TCP	168	[TCP Retransmission] 42674 → 443 [FIN, PSH, ACK]
156	80.448779551	192.168.1.112	186.224.0.18	DNS	90	Standard query 0x20a3 A support.mozilla.org OPT
157	80.448923774	192.168.1.112	186.224.0.20	DNS	90	Standard query 0x7ab2 AAAA support.mozilla.org C
158	80.449026930	192.168.1.112	186.224.0.18	DNS	85	Standard query 0xc2e2 AAAA www.google.com OPT

Fonte: Imagem dos autores, 2022

Problemas de conexão de rede e poder computacional foram algumas dificuldades encontradas para o alcance dos objetivos. A ocorrência de erros fez com que os resultados demorassem a aparecer inicialmente, e à medida que foram sendo resolvidos, outros apareceram. Tratar cada um dos erros analisando os impactos na arquitetura como um todo foram essenciais para que, no final, os objetivos fossem concluídos.

Os resultados descritos, alcançados no primeiro semestre, deveriam integrar o

sistema inteligente que está sendo desenvolvido como parte finalizadora do projeto. No entanto, a ferramenta Zabbix apresentou um problema que impediu a conexão destas duas pontas, uma vez que o sistema também apresentaria dados dos *host*s monitorados, como informações de utilização de memória e CPU. Este problema também impactou na escolha do tipo de ataque a ser realizado.

Atacar máquinas em ambiente controlado foi um objetivo alcançado no desenvolvimento da segunda parte do projeto. O ataque passou a ser realizado através do protocolo ICMP, com a técnica conhecida como ping da morte. A conclusão deste objetivo permitiu que, utilizando a ferramenta Wireshark, dados fossem exportados com todas as informações necessárias para a criação de um modelo de aprendizado de máquina. A figura 20 mostra o resultado da execução deste ataque.

Figura 20: Ataque DDOS com ping da morte

```
keterly@keterly-Aspire-A315-23:~$ sudo ping 192.168.1.112 -s 65500 -l 65500
PING 192.168.1.112 (192.168.1.112) 65500(65528) bytes of data.
65508 bytes from 192.168.1.112: icmp_seq=7 ttl=64 time=950 ms
65508 bytes from 192.168.1.112: icmp_seq=8 ttl=64 time=2414 ms
65508 bytes from 192.168.1.112: icmp_seq=10 ttl=64 time=1727 ms
65508 bytes from 192.168.1.112: icmp_seq=11 ttl=64 time=3879 ms
65508 bytes from 192.168.1.112: icmp_seq=12 ttl=64 time=5662 ms
65508 bytes from 192.168.1.112: icmp_seq=14 ttl=64 time=6150 ms
65508 bytes from 192.168.1.112: icmp_seq=19 ttl=64 time=4328 ms
From 192.168.1.107 icmp_seq=49 Destination Host Unreachable
From 192.168.1.107 icmp_seq=50 Destination Host Unreachable
From 192.168.1.107 icmp_seq=53 Destination Host Unreachable
From 192.168.1.107 icmp_seq=54 Destination Host Unreachable
```

Fonte: Imagem dos autores, 2022

Sendo o desenvolvimento do modelo de classificação outro objetivo, a biblioteca Sklearn foi utilizada para a criação do modelo capaz de receber informações sobre o tráfego da rede e determinar se, para o ataque de inundação ping, os dados representavam um possível ataque ou não. Optou-se por um modelo de regressão logística e informações sobre o tamanho do pacote e o nome do protocolo para compor a variável “X”, onde, por consequência, “Y” representou a classe que se desejava prever. A figura 21 mostra como ficaram as métricas após o treinamento, para um *score* de 0.8

Figura 21: Métricas da predição

	precision	recall	f1-score	support
False	1.00	1.00	1.00	160
True	1.00	1.00	1.00	10170
accuracy			1.00	10330
macro avg	1.00	1.00	1.00	10330
weighted avg	1.00	1.00	1.00	10330

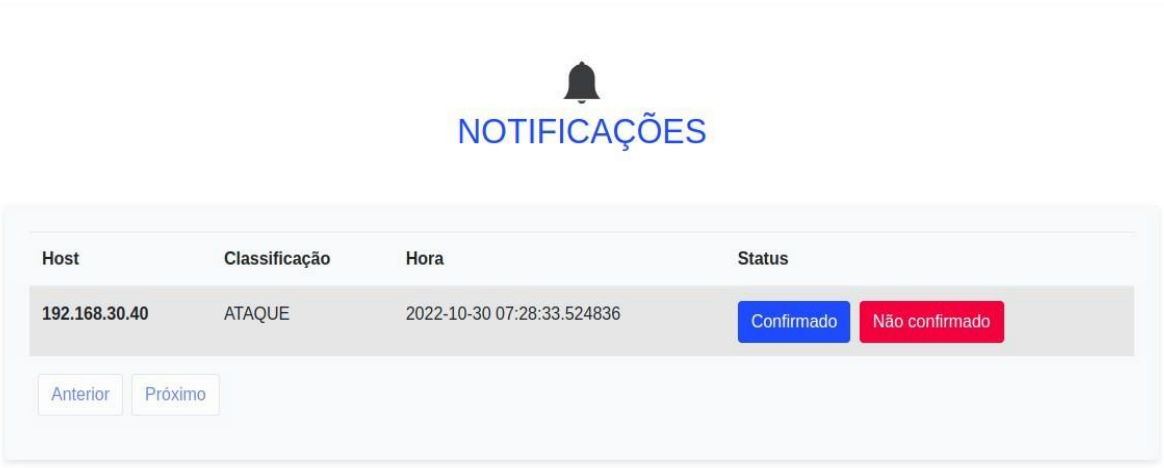
Fonte: Imagem dos autores, 2022

No aprendizado de máquina, o score representa um limiar para a classificação. O modelo, ao ler os dados e classificá-los como verdadeiro ou falso, obtém uma porcentagem. A porcentagem diz respeito a probabilidade daquela informação pertencer a classe “verdadeiro”, que para este trabalho, significa “Ataque”. 0,8 neste exemplo, significa que, para dados com um score com este valor ou mais, os dados são classificados como “Ataque” e apresentam uma precisão de 1.0

Para atingir o objetivo de visualizar e classificar novos dados, foi desenvolvida uma aplicação web em Flask Framework, que pode ser acessada através da URL **https://monitorserver.onrender.com/**, com usuário **“admin@monitor.com”** e senha **“123456”**.

A aplicação permitiu que o analista de segurança realizesse seu login e visualizasse todos os *host*s onde o modelo detectou um possível ataque, além de confirmar ou não a detecção. Esta informação, além de trazer IP e classificação, também mostrou o horário da detecção. Para classificar novos *host*s, o analista realiza o upload de um arquivo csv exportado da ferramenta Wireshark. Caso detectado um ataque, ele é notificado. As notificações são salvas e consideradas como “Lidas” apenas se o usuário clicar em “Confirmado” ou “Não Confirmado”, indicando se a detecção esta correta ou não. As figuras 22 e 23 mostram respectivamente a tela de notificação e de classificação, e a figura 24 mostra uma mensagem no grupo do telegram, enviada pelo *bot* desenvolvido neste projeto.

Figura 22: Página de notificações



Fonte: Imagem dos autores, 2022

Figura 23: Página de classificação

Fonte: Imagem dos autores, 2022

Figura 24: Bot de alerta no telegram

Fonte: Imagem dos autores, 2022

Mesmo com as dificuldades e mudanças na ideia inicial do projeto, estes resultados tornam-se relevantes para a área da pesquisa em Segurança da Informação e Inteligência Artificial, uma vez que possibilita a análise do comportamento dos dados em cenários opostos e permite uma melhor definição de quais características possuem um maior peso para a classificação de um possível ataque DDOS ou um tráfego de rede normal. Com o monitoramento do servidor utilizando a ferramenta Zabbix e a classificação dos dados através da leitura de um arquivo CSV exportado do Wireshark mostram que, em conjunto, auxiliam o trabalho de analistas de segurança. O sistema de monitoramento resultante deste trabalho pode ser visto como uma segunda opinião, ou como uma forma de analisar mais rapidamente um elevado número de informações do tráfego de rede.

4. CONSIDERAÇÕES FINAIS

Este projeto buscou desenvolver um ambiente para monitoramento e simulação de ataques DDOS a um Servidor Apache e implementar um sistema inteligente para a classificação de informações de rede, sistema que foi pensado para profissionais da área de segurança. Os objetivos foram cumpridos por meio da criação de máquinas virtuais Linux para compor a arquitetura de um ambiente controlado e por meio do desenvolvimento de uma aplicação web em Flask Framework para a visualização e classificação de novos arquivos contendo informações do tráfego de rede de um *host*. Após a finalização de toda metodologia foi possível derrubar o servidor por meio do ataque de negação de serviço, verificar as mudanças na porcentagem de utilização de recursos de Hardware e, a disponibilidade da máquina. Também foi possível, com a escolha do ataque de inundação Ping, gerar dados que serviram de insumo para o modelo de aprendizado de máquina. O modelo, depois de feito seu deploy, foi consumido pela aplicação web e permitiu que usuários logados enviassem arquivos e recebessem a classificação do estado da rede após a predição do modelo de detecção de DDOS.

Estes resultados e todos os processos desenvolvidos configuram uma etapa de grande importância para trabalhos a serem feitos no futuro. O modelo de regressão implementado neste projeto é limitado a três tipos de protocolo, no entanto, com a visualização de como um ataque funciona e quais dados mudam durante sua ocorrência, em trabalhos futuros novas abordagens podem ser escolhidas e estudadas, além da possibilidade de melhoria do modelo de classificação atual, permitindo que mais tipos de protocolos sejam classificados.

REFERÊNCIAS BIBLIOGRÁFICAS

CIBERSEGURANÇA: o que é, importância e como se qualificar. **Portal da Indústria**, 2021. Disponível em: https://www.portaldaindustria.com.br/industria-de-a-z/ciberseguranca/. Acesso em: 27 mar. 2022.

CRESCE o uso de Internet durante a pandemia e número de usuários no Brasil chega a 152 milhões, é o que aponta pesquisa do Cetic.br. **CETIC.BR**, 2021. Disponível em: https://cetic.br/pt/noticia/cresce-o-uso-de-internet-durante-a-pandemia-e-numero-de-usuarios-no-brasil-chega-a-152-milhoes-e-o-que-aponta-pesquisa-do-cetic-br/. Acesso em: 26 mar. 2022.

A AMÉRICA Latina sofreu mais de 41 bilhões de tentativas de ataques cibernéticos em 2020. **Fortinet**. Sunrise, Florida, 2021. Disponível em: https://www.fortinet.com/br/corporate/about-us/newsroom/press-releases/2021/latin-

america-suffered-more-than-41-billion-cyberattack-attempts-in-2020. Acesso em: 26 mar 2022

LIMA FILHO, F. S. de. **Smart Defender**: Um sistema de detecção e mitigação de ataques DoS/DDOS usando aprendizagem de máquina. 2019. 103 f. Tese (Doutorado em Engenharia Elétrica e de Computação) - Centro de Tecnologia, Universidade Federal do Rio Grande do Norte, Natal, 2019. Disponível em: https://repositorio.ufrn.br/bitstream/123456789/28470/1/SmartDefendersistema_LimaFi lho_2019.pdf. Acesso em: 26 mar. 2022.

O que são ataques DDOS e quais podem ser suas consequências. **Perallis Security**. Disponível em: https://www.perallis.com/news/o-que-sao-ataques-DDOS-e-quais-podem-ser-suas-consequencias. Acesso em: 27 mar. 2022.