Um Framework Baseado na Pilha ELK Para Análise Pós-Intrusão de Ataques de DDoS

Camilla Alves[1], **André Monteiro**[1]

[1]Engenharia de Computação - CEFET/RJ campus Petrópolis, Petrópolis - RJ - Brasil.

camilla.silva@aluno.cefet-rj.br, andre.monteiro@cefet-rj.br

Abstract. *This work presents a framework based on the Elasticsearch, Logstash and Kibana (ELK) aimed to analyze the logs of a computing platform denial-of-service attacks, known as DDoS. The proposed framework enables the post-intrusion investigation, running an attack identification algorithm and performing the data storage, analysis and visualization related to the cybernetic attack. Thus, the log analyzing can be performed in a friendly interface, since in a normal scenario there are several system logs and raw data, leading the attack verification into a complex process. The tests were performed using two differents DDoS aproaches. The results show the framework was able to collect information only from the system network logs, identify the malicious packages, and forward them to the visualization interface.*

Resumo. *Este trabalho apresenta um framework baseado no Elasticsearch, Logstash e Kibana (pilha ELK) projetado para analisar os logs de ataques de negação de serviço (DDoS) a um ambiente computacional. O framework proposto viabiliza a investigação pós-intrusão, executando um algoritmo de identificação de ataques e realizando o armazenamento, análise e visualização das informações relacionadas ao ataque cibernético. Assim, a análise dos logs pode ser feita de forma objetiva em uma interface amigável, pois em geral os logs de ambientes computacionais apresentam um volume massivo de dados não estruturados, tornando o processo de investigação de ataques em uma tarefa complexa. Foram realizados testes com duas abordagens de ataques de DDoS, ratificando que o framework foi capaz de coletar informações diretamente dos logs da rede de dados, identificar os pacotes maliciosos e encaminha-los a uma interface visual para investigação dos administradores do ambiente alvo.*

1. Introdução

Um Sistema de Detecção de Intrusão (*Intrusion Detection System*, também conhecido como IDS) é definido como um *software* ou *hardware* usado para detectar o tráfego não autorizado ou atividades que vão contra a política de segurança implementada em um determinado ambiente computacional. Os IDS podem ser classificados como *Host-Based Intrusion Detection System* (HIDS), *Network-Based Intrusion Detection System* (NIDS) ou uma combinação de ambos. Enquanto os HIDS examinam os eventos em um computador conectado à rede, o NIDS examinam o tráfego da rede que suporta o ambiente computacional. Em ambos os sistemas de detecção mencionados dois modos de operação são disponíveis: IDS baseado em assinatura e IDS baseado em anomalia. O primeiro se baseia na identificação de uma assinatura de um evento de intrusão, buscando por eventos

que correspondam a padrões pré-definidos [Verma et al. 2022]. Já o segundo concentra-se na identificação de padrões de atividades incomuns, com base em um perfil padrão associado a normalidade do tráfego de dados [Khayam et al. 2009].

O *framework* apresentado neste trabalho é um NIDS baseado em assinatura, que utiliza as ferramentas Elasticsearch, Logstash e Kibana, também conhecidas como pilha ELK (ELK *Stack*) [ELK 2021], e foi projetado para ambientes computacionais que sofreram Ataques de Negação de Serviço Distribuído, também conhecidos como *Distributed Denial of Service*, ou DDoS. Conforme apontado em [He et al. 2021], a utilização de uma abordagem eficiente de análise das informações contidas nos *logs* de ambientes computacionais é capaz de mitigar significativamente os problemas não visíveis no dia a dia. O objetivo do *framework* proposto é viabilizar a análise pós-intrusão de um ataque cibernético por meio da identificação das principais características do evento. Os testes da solução desenvolvida consideraram dois cenários distintos de ataques de DDoS. Os resultados indicaram que o *framework* foi capaz de selecionar informações diretamente dos pacotes de dados trafegados no momento do ataque, efetuando o tratamento e apresentação dos dados por meio de uma interface de visualização apropriada para os responsáveis pela camada de segurança de um ambiente computacional.

Este trabalho encontra-se organizado da seguinte forma: na Seção 2 é realizada uma discussão dos principais trabalhos relacionados que utilizam a pilha ELK e a análise de *logs* para o gerenciamento de eventos relacionados a ataques cibernéticos. Na Seção 3 é apresentado o *framework* proposto neste trabalho. A avaliação de desempenho da solução e os resultados obtidos são discutidos na Seção 4. Por fim, na Seção 5 são apresentadas as conclusões deste trabalho e também indicados os pontos para trabalhos futuros.

2. Trabalhos Relacionados

Em [Stoleriu et al. 2021] é apresentado um trabalho baseado na pilha ELK que é integrado a um sistema de segurança cibernética. O sistema faz interface com o módulo de aprendizado de máquina do Elasticsearch, para otimizar a detecção de anomalias no tráfego de rede. De forma similar à solução apresentada neste trabalho, a proposta mencionada faz uso de *logs* no Logstach para identificação da localização geográfica dos ataques. Entretanto, problemas de discrepância e inconsistência de dados demandaram a implementação de filtros específicos, prejudicando a análise por parte do Logstash, e a acurácia final do sistema para a detecção de um ataque cibernético. Além disso, o referido trabalho não oferece um painel de controle para consolidar as informações obtidas.

No trabalho de [Kumar et al. 2018], a pilha ELK é utilizada em conjunto com um algoritmo proposto para análise de ataques. O algoritmo especifica funções para cada tipo de *log* diferente, onde a comparação é realizada através de um valor de limite definido. Caso este limite seja violado, o IP específico é marcado como malicioso e é armazenado em um arquivo de texto para uso futuro, junto com a localização do mesmo. Como o valor limite varia de sistema para sistema, é necessário recalibrar esse valor a cada implementação, prejudicando assim a portabilidade da solução. No *framework* proposto neste trabalho o algoritmo de detecção de ataques possui um alto grau de generalização, facilitando o uso da solução em cenários ambiente computacionais heterogêneos.

Em [Muhammad et al. 2023] é apresentado um sistema para análise em tempo real com IDS composto por um conjunto de elementos integrados, dentre eles a pilha ELK.

O referido trabalho descreve uma análise de desempenho detalhada do sistema proposto, observando as características de performance dos elementos do sistema sob a ótica de diversas métricas (principalmente uso de memória e CPU) do ambiente de processamento. Os resultados apontam que o Elasticsearch é o maior responsável pelo consumo de recursos computacionais, atingindo 78% de uso de CPU e consumindo 2300 MB de memória. Apesar de não apresentar uma solução de ponta a ponta para DDoS, a trabalho realiza uma discussão relevante sobre a plataforma computacional necessária para implementar sistemas baseados na pilha ELK, abordando os seus requisitos não funcionais. Essas questões mostram-se relevantes para analisar a viabilidade de implementação de NIDS baseados na pilha ELK, como o *framework* proposto neste trabalho e descrito na seção a seguir.

3. Framework Proposto

A pilha ELK oferece a capacidade de realizar análises operacionais em todos os tipos de fonte de dados de maneira confiável para pesquisar, analisar e visualizar esses dados. De acordo com [Praneeth and Sreedevi 2019] o ELK possui alguns benefícios, como: escalabilidade, tolerância a talhas e automatização. A arquitetura da solução proposta é baseada na integração da pilha ELK. Essa integração é ilustrada na Fig. 1, que demonstra o fluxo de dados entre os elementos da pilha. Como pode ser observado, os *logs* oriundos de ataques DDoS no formato *pcap* são encaminhados para uma primeira análise. Em seguida, com o auxílio do Logstash os dados são transformados em dados estruturados, e são enviados ao Elasticsearch, sendo posteriormente enviados ao Kibana para uma visualização e melhor compreensão das informações.

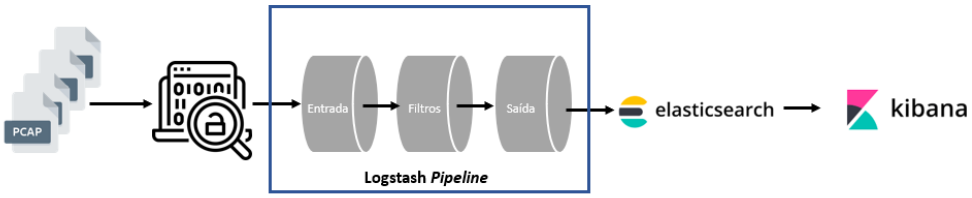

Figura 1. Arquitetura do framework proposto.

Pode-se configurar no Elasticsearch várias máquinas virtuais denominadas nós, que juntas formam um único *cluster*. Cada nó possui um conjunto de dados que ao serem somados com os dados dos demais nós do *cluster* resultam na informação completa. Estes também respondem as requisições HTTP REST do Elasticsearch, e o nó *master* é o responsável por receber cada requisição e orquestrar a consulta destes dados. Para isso, são criados documentos que são armazenados em índices, que por sua vez, possuem um ID. Tratando-se de dados com grande volume, é interessante poder segmentar esses dados para facilitar a consulta dos mesmos. Desta forma, através das *shards* o índice pode ser fragmentado e dividido entre os nós, e além disto, é possível paralelizar a leitura dos dados. À medida em que o *cluster* aumenta ou diminui, o Elasticsearch migra automaticamente os *shards* entre os nós para que o *cluster* permaneça equilibrado e escalável. Devido a limitação de hardware neste projeto, foram criados um nó *master* e um nó de dados, o que se mostrou adequado conforme resultados descritos na Seção 4.

A estrutura do Logstash é composta por três etapas: entrada, filtros e saída. Na primeira etapa de entrada, os dados chegam ao Logstash através de protocolos para transferência de arquivos, onde são armazenados em uma pasta específica para coleta dos dados. Esses dados de entrada podem ser os mais variados possíveis, sendo que neste trabalho foi escolhido o formato JSON como padrão. Em seguida, são adicionados filtros de forma que o fluxo de dados seja encaminhado para todos os filtros especificados. Por fim, a etapa de saída informa onde e como as informações de *log* serão salvas após serem filtrados, indicando nesse caso, o banco de dados Elasticsearch. Através do Kibana é possível transformar os *logs* em informações úteis através de painéis de controle devido à capacidade de realizar correlação de eventos, filtrar *logs* por origem, hospedeiros e outras formas de combinações.

Algoritmo 1: Análise de ataques DDoS

```
Entrada: Arquivo no formato pcap
Saída: Vetor de ataque no formato JSON
início
    df []← converter_pcap_para_dataframe(arquivo_pcap);
    total_pacotes ← tam(df);
    dist_ip_dest []← freq(df[], dst_ip);
    df_atualizado [] ← filtro (df[], dist_ip_dest[0]);
    df_backup [] ← df_atualizado [] ;
    quant_syn [], quant_syn_ack ← processar_pacotes(df[]) ;
    repita
        dist_prot [] ← freq(df_atualizado [] , protocolo);
        df_atualizado [] ← filtro (df[], dist_prot[0]);
        if protocolo == 'UDP' ou protocolo == 'TCP' then
            dist_porta_orig []← freq(df_atualizado [] , porta_origem);
            dist_porta_dest []← freq(df_atualizado [] , porta_destino);
            if dist_porta_orig[0].valor() ¿ dist_porta_dest[0].valor() then
                df_atualizado [] ← filtro (df[], dist_porta_orig[0]);
            else
                df_atualizado [] ← filtro (df[], dist_porta_dest[0]);
        ips_orig [] ← retorna_ip_orig(df_atualizado [])
        if tam(ips_orig) < 2 then
            break;
        else
            df_final ← df_atualizado[string_vetor_ataque]
        vetor_ataque [] ← vetor_ataque[] + [dist_prot[0], dist_porta_orig[0],
          dist_porta_dest[0], ips_orig] ;
        df_atualizado [] ← filtro (df_backup [],vetor_ataque []);
        df_backup [] ← df_atualizado []
    até df_atualizado [] > 1;
    converte_timestamp (vetor_ataque[timestamp]);
    encaminha_logstash (vetor_ataque [])
```

A proposta de identificação de ataques desenvolvida neste trabalho, utiliza um arquivo *pcap* contendo o tráfego legítimo e o tráfego de ataque e, em seguida, analisa-os com o objetivo de identificar características recorrentes em relação ao restante do tráfego, conforme abordagem descrita em [DDo 2021]. Para este fim, a solução implementada caracteriza que um ataque DDoS é uma repetição de um tráfego de rede com características semelhantes de um conjunto de endereços IP para um único endereço IP de destino, conforme especificação descrita a seguir no Algoritmo 1. Na linha 4 começa a filtragem de endereço IP de destino com maior frequência de pacotes recebidos, objetivando encontrar o alvo do ataque. Posteriormente, faz-se uma análise da quantidade de SYN e SYN/ACK recebidos, a fim de encontrar evidências da diferença de número de conexões iniciadas e estabelecidas, o que pode demonstrar indícios de um ataque por inundação de tráfego. Em seguida (linha 8), tem-se o início de uma estrutura de repetição para investigar possíveis vetores de ataque destinados ao endereço IP de destino encontrado. Nessa estrutura de repetição são realizados filtros para encontrar protocolo e porta mais utilizados no ataque.

Após esses filtros, há uma junção entre os endereços IP de origem que enviam tráfego para o endereço IP de destino, baseando-se nos protocolos e portas encontrados na filtragem que possuam o mesmo *payload*, ou seja, a parte principal dos dados. Essa junção resulta em um vetor de ataque DDoS (linha 23). Cabe ressaltar que na linha 19 é verificado se o número de endereços IP de origem é menor que dois, e em caso afirmativo a análise é interrompida por não se caracterizar um ataque DDoS. Em seguida, o *loop* ocorre novamente visando identificar outros vetores de ataques. Com o término deste *loop*, uma modificação do *timestamp* é feita (linha 27), adequando para o formato ideal a ser lido pelo Logstash. Ao final, com todos os pacotes analisados e os vetores de ataques formados, um *fingerprint* é criado contendo as características do ataque e este é exportado para um arquivo JSON. Este arquivo é encaminhado para o Logstash, dando início ao processo de indexação. Por fim, após a indexação e envio para o Elasticsearch, os pacotes agora transformados em documentos podem ser verificados no Kibana e analisados através da criação de painéis de visualizações e *dashboards*.

4. Avaliação de Desempenho

A avaliação de desempenho do *framework* proposto aborda dois cenários analisados de ataques de DDoS. O cenário 1 é baseado na abordagem descrita em [Koroniotis et al. 2019], e também relacionado ao trabalho apresentado em [Peter et al. 2021]. Essa abordagem utiliza um cenário de conjunto de dados chamado BoT-IoT, que incorpora o tráfego de rede de aplicações IoT (*Internet of Things*) legítimo e simulado, juntamente com vários tipos de ataques. O cenário 2 é baseado nos trabalhos descritos em [Santanna et al. 2015] e [Heinrich et al. 2022], e possui ataques de negação de serviço pertencentes à classe de ataques de reflexão e amplificação DNS. A motivação para escolha deste cenário foi analisar os ataques conhecidos como *Booter*, que são realizados através de portais que oferecem o chamado "DDoS-as-a-Service", onde são disponibilizados serviços que executam um ataque cibernético orquestrado mediante pagamento de um valor para tal. Para ambos os cenários são apresentadas as visualizações criadas no Kibana, onde podem ser utilizadas em conjunto formando um painel de controle. Essas visualizações representam a saída do *framework*, e foram confeccionadas a partir do fluxo de pacotes de ataque enviados para um ambiente computacional alvo.

Como no cenário 1 o ataque ocorreu dentro da própria rede, pode ser interessante

analisar quais foram os endereços MAC utilizados para realizar o ataque (em um cenário onde não foi realizado MAC *spoofing*, por exemplo), inserindo-os futuramente em uma lista de bloqueio. A Fig. 2 mostra como foi feita esta distribuição. A Fig. 3 representa os endereços IPs de origem utilizados para o ataque no cenário 2 e a localização dos mesmos baseado na base de dados. Como o cenário 2 aborda um ataque encomendado a terceiros, saber a localização de origem dos pacotes envolvidos no evento se mostra uma informação relevante para investigação dos administradores do ambiente computacional e das autoridades de segurança de cada país. Para este fim, os pacotes são separados de acordo com a posição geográfica, de forma que quanto maior a quantidade de pacotes naquela região, maior será a circunferência no mapa.

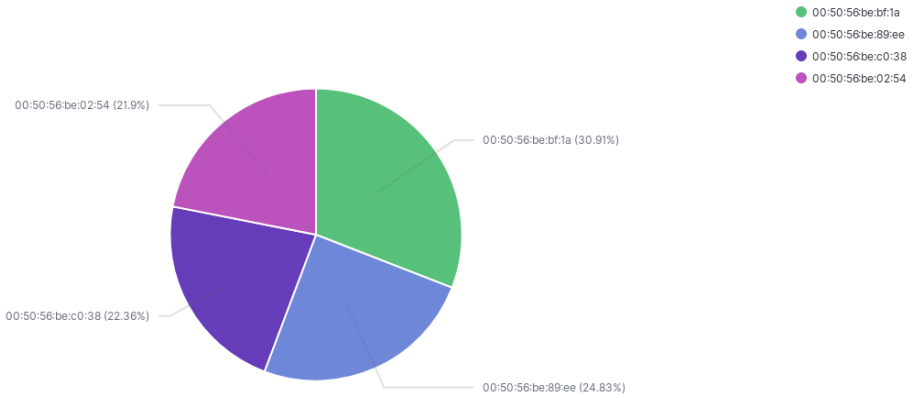

Figura 2. Distribuição do endereço MAC atacante do cenário 1.

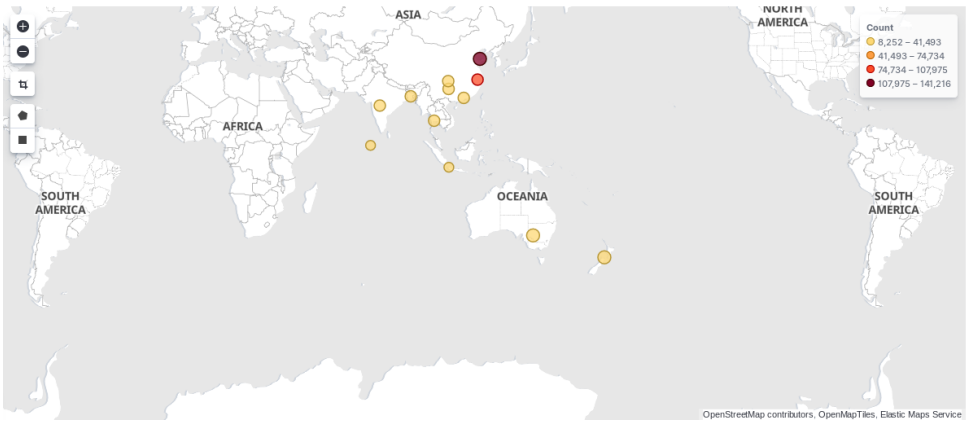

Figura 3. Geolocalização dos IPs do cenário 2.

5. Conclusão e Trabalhos Futuros

O *framework* apresentado neste trabalho é baseado na pilha ELK e realiza uma análise de conjuntos de dados oriundos de ataques de negação de serviço conhecidos como DDoS, com o foco na análise pós-intrusão. Os resultados apontam que o *framework* foi capaz de realizar o tratamento dos *logs* através do Logstech e do Elasticserach, identificar os pacotes maliciosos por meio do algoritmo proposto, e fornecer um painel de visualização no Kibana para subsidiar a análise e verificação de danos do ataque. Como trabalhos futuros pretende-se adequar o framework para a análise de ataques em tempo de real, observando as restrições de processamento impostas pela plataforma computacional.

Referências

(2021). Ddos dissector. Disponível em: `https://github.com/ddos-clearing-house/ddos_dissector`. Acesso em: 28 fev. 2021.

(2021). Elastic stack. Disponível em: `https://https://www.elastic.co/pt/elastic-stack`. Acesso em: 22 mar. 2021.

He, S., He, P., Chen, Z., Yang, T., Su, Y., and Lyu, M. R. (2021). A survey on automated log analysis for reliability engineering. *ACM computing surveys (CSUR)*, 54(6):1–37.

Heinrich, T., Will, N. C., Obelheiro, R. R., and Maziero, C. A. (2022). Um estudo de correlaçao de ataques drdos com fatores externos visando dados de honeypots. In *Anais do XXII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, pages 358–371. SBC.

Khayam, S. A., Mirza, F., et al. (2009). A survey of anomaley-based intrusion detection systems. *School of Electrical Engineering and Computer Science (SEECS), National University of Sciences & Technology (NUST)*.

Koroniotis, N., Moustafa, N., Sitnikova, E., and Turnbull, B. (2019). Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. *Future Generation Computer Systems*, 100:779–796.

Kumar, A., Bandyopadhyay, A., Bhoomika, H., Singhania, I., and Shah, K. (2018). Analysis of network traffic and security through log aggregation. *International Journal of Computer Science and Information Security (IJCSIS)*, 16(6).

Muhammad, A. R., Sukarno, P., and Wardana, A. A. (2023). Integrated security information and event management (siem) with intrusion detection system (ids) for live analysis based on machine learning. *Procedia Computer Science*, 217:1406–1415.

Peter, C. S., Oliveira, T., Monks, E. M., Motta, F. P., Barbosa, J. L., and Yamin, A. C. (2021). iota: An approach to secure over-the-air updates on the internet of things scenario. In *Proceedings of the Brazilian Symposium on Multimedia and the Web*, pages 173–176.

Praneeth, J. and Sreedevi, M. (2019). Detecting and analyzing the malicious windows events using winlogbeat and elk stack. *Int J Recent Technol Eng*, pages 156–160.

Santanna, J. J., van Rijswijk-Deij, R., Hofstede, R., Sperotto, A., Wierbosch, M., Granville, L. Z., and Pras, A. (2015). Booters—an analysis of ddos-as-a-service attacks. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 243–251. IEEE.

Stoleriu, R., Puncioiu, A., and Bica, I. (2021). Cyber attacks detection using open source elk stack. In *2021 13th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, pages 1–6. IEEE.

Verma, J., Bhandari, A., and Singh, G. (2022). inids: Swot analysis and tows inferences of state-of-the-art nids solutions for the development of intelligent network intrusion detection system. *Computer Communications*, 195:227–247.