Pós-Graduação em Ciência da Computação

Pablo Philipe Pessoa Do Nascimento

Uma Metodologia para Selecionar Contadores de Desempenho de Hardware para dar Suporte ao Diagnóstico não Invasivo e a Classificação de Ataques DDoS de Inundação em Servidores Web

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
http://cin.ufpe.br/~posgraduacao

Recife
2021

Pablo Philipe Pessoa Do Nascimento

Uma Metodologia para Selecionar Contadores de Desempenho de Hardware para dar Suporte ao Diagnóstico não Invasivo e a Classificação de Ataques DDoS de Inundação em Servidores Web

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Área de Concentração: Avaliação de Desempenho e Dependabilidade de Sistemas
Orientador: Dr. Paulo Romero Martins Maciel
Coorientador: Ms. Isac Fernando Ferreira Colares

Recife
2021

FICHA

Pablo Philipe Pessoa Do Nascimento

"**Uma Metodologia para Selecionar Contadores de Desempenho de Hardware para dar Suporte ao Diagnóstico não Invasivo e a Classificação de Ataques DDoS de Inundação em Servidores Web**"

Aprovado em: 06/09/2021.

Orientador: Dr. Paulo Romero Martins Maciel

BANCA EXAMINADORA

Prof. Dr. Jamilson Ramalho Dantas
Centro de Informática / UFPE

Prof. Dr. Almir Pereira Guimarães
Instituto de Computação / UFAL

Prof. Dr. Paulo Romero Martins Maciel
Centro de Informática / UFPE

Dedico este trabalho a todas as pessoas que se fizeram essenciais para o meu desenvolvimento pessoal e profissional e o desenvolvimento deste trabalho.

AGRADECIMENTOS

Gostaria de inicialmente deixar o meu agradecimento especialmente ao professor Paulo Maciel, por tornar essa realização possível, acreditando em mim e no meu potencial, me orientando, ensinando e aconselhando como um verdadeiro Sensei.

Gratidão a toda minha família, pelo suporte para que eu conseguisse chegar até aqui, em especial a minha Avó, Juvanira Honoria e minha Mãe, Maria José, mulheres fortes que me inspiraram a persistir e ao meu pai Cremildo José que me ensinou a ser determinado.

Agradeço ao professor Isac Ferreira, por me acompanhar desde a minha graduação até aqui, me aconselhando, ajudando e trabalhando junto comigo. Fazendo tudo isso ser possível, acreditando e me apoiando no que fosse possível.

Aos meus amigos que sempre acreditaram em mim e me apoiaram em todas as fases da minha vida, extremamente importantes para minha construção pessoal como ser humano. Enorme gratidão aos Defensores, Alex Gabriel, Ayrton Eusébio, Denílson Souza, Djalma Neto, Matheus Xenofonte, Lenine Rodrigues, Ragne Rodrigues e em especial a Welton Dionísio que me apoiou e auxiliou em várias etapas deste processo. A minha irmã Helena Pessoa, sendo umas das minhas principais inspiração dentro da família, e que fique registrado e sirva como exemplo e inspiração para as novas gerações, meu sobrinho e afilhado Heitor Pessoa e meu afilhado Luke Xenofonte, que sozinhos não somos nada.

Meus sinceros agradecimentos a todos os meus companheiros e amigos do grupo de pesquisa MoDCS, que colaboram para o avanço e a existência dele e ao professor Almir Guimarães. Em especial aos que estiveram ao meu lado nessa jornada e fizeram possível ela acontecer, Marco Aurélio, Paulo Pereira, Ronierison Maciel, Jamilson Dantas e Carlos Lopes.

A todas as pessoas que me ajudaram direta e indiretamente até o presente momento, sendo um amigo, um companheiro, um parceiro e um colo de afeto, me dando forças para persistir e não desistir diante das dificuldades encontradas durante todo o percurso, e a minha companheira Aline do Monte, pelo seu amor, companheirismo e apoio, principalmente emocional.

"É somente através da ajuda mútua e das concessões recíprocas que um organismo agrupando indivíduos em número grande ou pequeno pode encontrar sua harmonia plena e realizar verdadeiros progressos."(Jigoro Kano, 1882).

RESUMO

As interrupções no servidor da Web causadas por ataques de *Denial of Service* (DoS) e *Distributed Denial of Service* (DDoS) aumentaram consideravelmente ao longo dos anos. Os *Intrusion Detection Systems* (IDS) não são suficientes para detectar ameaças no sistema, mesmo quando usados em conjunto com *Intrusion Prevention Systems* (IPS) e conjuntos de dados contendo informações de situação e ataques de serviço do sistema. Realizar análises com uma quantidade muito densa de variáveis observadas pode custar uma quantidade significativa de recursos do hospedeiro. Além disso, os dados de diagnósticos realizados por terceiros, correm o risco de não representar o real comportamento do sistema em uso e nem sempre podem ser compartilhados por conter informações confidenciais, resultando em dados incompletos. Este trabalho apresenta o desenvolvimento de uma metodologia não intrusiva para diagnosticar situações de ataque DDoS em servidores corporativos, dispensando o uso de conjuntos de dados de terceiros. Essa metodologia também auxilia no planejamento da capacidade dos ativos de infraestrutura e na implementação de contramedidas de segurança. A metodologia também permite a geração de perfis de comportamento de ataque DDoS, selecionando os *Hardware Performance Counters* (HPCs) mais influentes na caracterização de ataques, como *L1-dcache-stores*, *LLC-loads* e *dTLB-stores*, que possuem baixo nível de abstração e podem diferenciar as situações de ataque no sistema. A análise combina métodos e técnicas de diferentes segmentos utilizando *Machine Learning* (ML) e análise de dados estatísticos, o que pode melhorar consideravelmente a precisão da detecção de ataques, sendo capaz de diferenciar situações sazonais e de ataque no serviço. Com a metodologia proposta, reduziu-se os HPCs em mais de *26%* em comparação com o grupo inicial de contadores.

Palavras-chaves: Diagnóstico. Metodologia. Ataques de Negação de Serviço Distribuído. Contadores de Desempenho de Hardware. Infraestrutura. Servidor Web.

ABSTRACT

Web server outages caused by Denial of Service (DDoS) attacks and Distributed Denial of Service (DDoS) attacks have increased considerably over the years. The Intrusion Detection Systems (IDS) are not sufficient to detect threats in the system, even when used in conjunction with Intrusion Prevention Systems (IPS) and even considering the use of datasets containing situation information and system service attacks. Performing analyzes with a very dense amount of observed variables can cost the host a significant amount of resources. Furthermore, diagnostic data carried out by third parties run the risk of not representing the actual behavior of the system in use and cannot always be shared, as it may contain confidential information, resulting in incomplete data. This work presents the development of a non-intrusive methodology to diagnose DDoS attack situations on corporate servers, dispensing with the use of third-party datasets. This methodology also assists in the capacity planning of infrastructure assets and in implementing security countermeasures. The methodology also allows the generation of attack behavior profiles DDoS, selecting the most influential Hardware Performance Counters (HPCs) in attack characterization, such as *L1-dcache-stores*, *LLC-loads* and *dTLB-stores*, which have a low level of abstraction and can differentiate attack situations in the system. The analysis combines methods and techniques from different segments using ML and statistical data analysis, which can considerably improve the accuracy of attack detection, being able to differentiate seasonal and service attack situations. With the proposed methodology, reduced the HPCs by more than *26%*, compared to the initial group of counters.

Key-words: Diagnosis. Methodology. Distributed Denial of Service attacks. Security. Hardware Performance Counters. Infrastructure. Web server.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AI	*Artificial intelligence*
API	*Application Programming Interface*
Bot	*Robot*
Botnet	*Robot Network*
CPS	*Cyber-physical System*
CPU	*Central Processing Unit*
DDoS	*Distributed Denial of Service*
DMZ	*Demilitarized Zone*
DNS	*Domain Name System*
DoS	*Denial of Service*
DT	*Decision Tree*
eBPF	*Berkeley Packet Filter*
FCA	*Flash Crowd Attacks*
FN	*Falso Negativo*
FP	*Falso Positivo*
FS	*Feature Selection*
GB	*Gigabyte*
GbE	*Gigabit Ethernet*
HD	*Hard Disk*
HPCs	*Hardware Performance Counters*
HST	*Half-Space Trees*
HTTP	*HyperText Transfer Protocol*
ICMP	*Internet Control Message Protocol*
IDS	*Intrusion Detection Systems*
IIS	*Internet Information Services*
IP	*Internet Protocol*
IPS	*Intrusion Prevention Systems*
KB	*Kilobyte*
LAN	*Local Area Network*
LPE	*Linux perf events*

LTS	*Long-term Support*
MB	*Megabyte*
ML	*Machine Learning*
MS	*Microsoft Windows*
MTD	*Moving Target Defense*
OS	*Operating System*
OSI	*Open System Interconnection*
OTP	*One-time Password*
PCA	*Principal Component Analysis*
PCL	*Performance Counters for Linux*
PMCs	*Performance Monitoring Counters*
PMU	*Performance Monitoring Unit*
QOS	*Quality of Service*
RAM	*Random Access Memory*
RF	*Random Forest*
SI	*Security Information*
SPOF	*Single Point of Failure*
SSH	*Secure Socket Shell*
SSL	*Secure Sockets Layer*
SVM	*Support Vector Machine*
SYN	*Synchronize*
TB	*Terabyte*
TCP	*Transmission Control Protocol*
TOR	*The Onion Router*
UDP	*User Datagram Protocol*
UID	*User Identifier*
URLs	*Uniform Resource Locator*
VM	*Virtual Machines*
VPN	*Virtual Private Network*

SUMÁRIO

1 INTRODUÇÃO

As interrupções no servidor da *web* causadas por DoS e DDoS, estão se tornando cada vez mais comuns. A tendência é crescente e, portanto, os esforços feitos apenas pelos IDS, não são suficientes para detectar ameaças no sistema. Mesmo quando utilizados juntamente com os IPS e dispositivos de *Firewalls* (FADHLILLAH; KARNA; IRAWAN, 2021; TANDON, 2020; DOULIGERIS; MITROKOTSA, 2004). Uma vez que esses dispositivos funcionam diretamente com assinaturas de ataques, sua precisão é diretamente prejudicada quando são expostos a ataques desconhecidos ou de *Zero-Day*[1] (ZHOU; PEZAROS, 2019; KHAN; ABDULLAH; KHAN, 2017; SABAHI; MOVAGHAR, 2008; FUCHSBERGER, 2005). Esses ataques podem simular facilmente o tráfego legítimo sem serem detectados pelos sistemas de proteção, uma vez que não há registros suficientes desses ataques. Por este motivo, o comportamento malicioso pode vir a ser identificado como comportamento legítimo pelos sistemas de proteção, produzindo alertas de *Falso Positivo* (FP) e *Falso Negativo* (FN), que podem impactar negativamente a *Quality of Service* (QOS) para usuários legítimos utilizando o serviço da web ou até mesmo tornando-o indisponível (DAHIYA; GUPTA, 2021; ABDULHAMMED et al., 2019; NDIBWILE et al., 2015).

A detecção de ataques utilizando conjuntos de dados contendo assinaturas de ataques permite a observação das características dos recursos do sistema, para posteriormente serem feitas a classificação das situações e as abordagens que será desenvolvida para detectar esses ataques (REHMAN et al., 2021; SHARAFALDIN et al., 2019; LEE et al., 2008). Abordagens que utilizam conjuntos de dados têm se mostrado bastante desafiadoras, pois muitos desses conjuntos de dados não retratam a realidade do sistema que se deseja aplicar, são incompletos ou não podem ser compartilhados por razões de privacidade, por normalmente serem dados não sensíveis, contendo informações confidenciais sobre os ativos da organização (NASCIMENTO et al., 2021a; SHARAFALDIN; LASHKARI; GHORBANI, 2018; JING et al., 2019). A maioria das abordagens que utilizam conjuntos de dados requerem uma quantidade significativa de recursos do hospedeiro e, dependendo do local que são realizadas essas análises, o ativo específico pode eventualmente se tornar alvo desses ataques. Os ativos sendo alvos dos ataques podem vir a se sobrecarregarem, causando instabilidade ou até indisponibilidade do serviço, prejudicando a QOS provida, como ilustrado na Figura 1 que retrata os possíveis direcionamento dos ataques (HE et al., 2021; SENGUPTA et al., 2020).

A combinação de várias técnicas de análises de dados pode melhorar consideravelmente a precisão na detecção de ataques DoS e DDoS produzir métodos mais eficazes para identificar essas anomalias com maior precisão (LIANG, 2021; XIAO et al., 2015). As

[1] Vulnerabilidade de segurança desconhecida em um *software* de computador, ameaça que ainda não foi corrigida ou tornada pública.

informações coletadas no diagnóstico do sistema, como informações de acesso e utilização da aplicação podem ser utilizadas como suporte da investigação em situações de crimes cibernéticos; como, por exemplo, o endereço de *Internet Protocol* (IP) de acesso e o fluxo de requisições. Assim como nos ataques de DoS e DDoS, os usuários podem ser notificados a título informativo sobre essas eventuais invasões em seus dispositivos. Considerando que seus equipamentos podem estar sendo utilizados para executar esses ataques coordenados e gerenciados dentro de uma rede *Robot Network* (Botnet) sem o seu conhecimento (DHAMOR; BHAT; THENMALAR, 2021; MACIEL et al., 2018).

Os mecanismos de defesa são classificados em três categorias, sendo denominadas *Defesa de Origem e Extremidade*, *Defesa Intermediária da Rede* e *Defesa Baseada no Hospedeiro*, que são classificadas dependendo de onde são implementadas no ambiente. Esses métodos de proteção podem ser uma alternativa interessante, dependendo do tipo de serviço oferecido e dos recursos de equipamento utilizados, uma vez que o ambiente terá o mínimo necessário para o provimento do serviço (ADDEPALLI; KARRI; JYOTHI, 2017; CLARK, 2017).

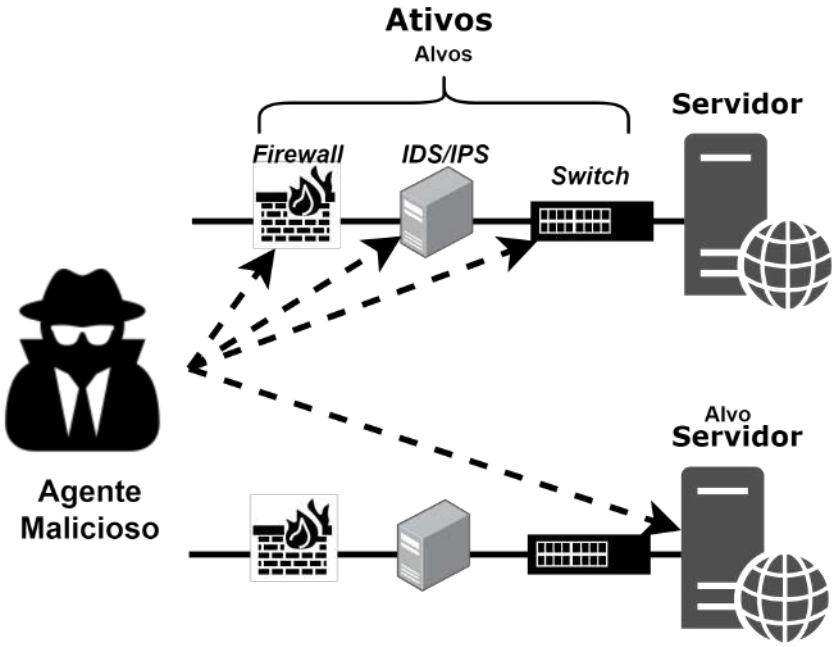


Figura 1 – Alvos do Ataque

1.1 JUSTIFICATIVA

Uma boa representação no diagnóstico das situações de ataque e uso habitual no sistema, onde as observações capturadas sobre os comportamentos do hospedeiro permitem desenvolver mecanismos capazes de executar contramedidas mais eficientes. Utilizando contadores de desempenho presentes na arquitetura dos processadores, os denominados HPCs, que capturam as características do sistema através do monitoramento contínuo.

Por meio desses contadores de baixo nível é possível correlacionar esses eventos com as estatísticas da rede, número de requisições na aplicação, número de usuários conectados e consumo dos recursos do hospedeiro (GRAKOVSKI, 2021; KRISHNAMURTHY; KARRI; KHORRAMI, 2019). Com essas estratégias de análise é possível dar suporte às ferramentas de proteção que apresentam limites de detecção fixos, o que eventualmente as tornam vulneráveis, quando agentes mal-intencionados com conhecimento desses limites podem executar ataques abaixo desse limite. Uma vez que os ataques não são detectados, os recursos do hospedeiro continuam sendo consumidos sem serem detectados pelos sistemas de proteção (COMPTON, 2019). Portanto, é fundamental a realização de diagnósticos mais fiéis e representativos, que possam ser realizados pelos administradores de sistemas. Assim, possibilitando o desenvolvimento de mecanismos capazes de tomar decisões de mitigação ou enviar o tráfego suspeito para verificação da legitimidade de acesso (ADESHINA; SAHA, 2021; SINGH; DE, 2015).

As ameaças na internet têm uma tendência crescente e assustadora. Com o crescimento descontrolado de *malwares* e ataques na internet, é necessário desenvolver contramedidas de proteção (LI; LIU, 2021). Em 2019, o Brasil já era o terceiro país no *ranking* dos que mais sofrem ataques cibernéticos no mundo, ficando atrás apenas da China e dos Estados Unidos (SYMANTEC, 2019). Em 2020, foram registradas mais de 8,4 bilhões de tentativas de ataques cibernéticos no Brasil, de um total de 41 bilhões em toda a América Latina e Caribe (FORTINET, 2021). Com relação às denúncias de crimes cibernéticos, de janeiro a dezembro de 2020 foram registradas mais de 156.692 mil denúncias anônimas, contra mais de 57.428 em 2019 (SAFERNET, 2021). Neste ano, de janeiro a abril, foram detectados mais de 907 mil *spams*, 737 incidentes relacionados a *malwares* e 48 mil *links* suspeitos (INTERPOL, 2020). Ataques por vírus de resgate batem recorde na primeira metade de 2021, havendo 304,7 milhões de investidas detectadas entre janeiro e junho. A quantia é maior do que o total registrado durante todo o ano de 2020. Brasil é o 5º no *rank* (SONICWALL, 2021).

Independentemente das tecnologias e técnicas utilizadas, a correta implantação e configuração dos equipamentos são partes essenciais da proteção do ambiente para garantir o bom funcionamento da prestação do serviço. Assim, deve seguir a Tríade da Segurança da Informação (ou seja, a CIA, visto na Figura 2, que atende aos requisitos de *Confidencialidade*, *Integridade* e *Disponibilidade*) (ROHAN; BASU; KARRI, 2019). Enquanto os agentes maliciosos são motivados a prejudicar o serviço nos pilares da segurança da informação e, consequentemente, ferir o serviço (XING et al., 2021; BAARS et al., 2015).

Figura 2 – Tríade Segurança da Informação

A implementação de mecanismos tradicionais como os *Firewalls* ainda é uma excelente solução entre as formas tradicionais de contramedidas de segurança. Dentre as contramedidas convencionais, os *Firewalls* atuam como um filtro bloqueando, de forma mais abrangente, diversos tipos de ataques. Por exemplo, bloqueando ataques de *IP Spoofing*, uma vez que é realizada a filtragem do tráfego de entrada e saída (KEIJER, 2019). A utilização e configuração de um *Load Balance* também é uma poderosa ferramenta de proteção, sendo um aliado essencial em conjunto com técnicas de proteção mais avançadas (EZENWE; FUREY; CURRAN, 2020)

Diante desses fatos, as contramedidas desenvolvidas de proteção contra ameaças de ataques DDoS são de interesse global, onde diversas comunidades especializadas em *Security Information* (SI), realizam esforços no combate a essas ameaças (ZHANG, 2021; BEHAL; KUMAR, 2016). Além disso, os HPCs têm apresentado excelentes resultados na prevenção e detecção de ataques, tornando-se um recurso promissor na criação de contramedidas para a segurança cibernética (SREENIVASARAO, 2021; FOREMAN, 2018). A escolha entre as diferentes abordagens utilizando esses contadores dependerá muito do serviço fornecido e de quais ameaças se deseja proteger o ambiente. No caso das abordagens baseadas em assinaturas, é necessário conhecimento específico das ameaças ao sistema e seus processos para que elas sejam adicionadas à lista de permissões do sistema (SABOOR; AKHLAQ; ASLAM, 2013). Abordagens baseadas em heurística normalmente são utilizadas quando o conhecimento geral das ameaças é conhecido e as características do sistema são monitoradas; portanto, como um método de investigação para criar diretrizes e regras bem definidas, para que seja possível identificá-las e classificá-las (DEKA; BHATTACHARYYA; KALITA, 2021; BHUYAN et al., 2014).

1.2 OBJETIVOS

Assim, o presente trabalho tem como objetivo desenvolver uma metodologia de diagnóstico não intrusiva para selecionar HPCs em ataques de DDoS de inundação *HyperText Transfer Protocol* (HTTP) em servidores *web* de nível corporativo. Para alcançar este objetivo a aplicação de técnicas estatísticas e de ML permitirá selecionar os contadores mais relevantes para a criação de perfis comportamentais, com o intuito de reduzir a densidade dos contadores monitorados e minimizar os recursos necessários para tal.

A metodologia do presente estudo realizará uma coleta passiva em tempo real, baseada no próprio hospedeiro, sem *hardware* no trajeto do tráfego e coleta de dados. Assim, evitando *Single Point of Failure* (SPOF) e recursos desnecessários no ambiente. Portanto, os procedimentos de aplicabilidade de HPCs de baixo nível, já abordados em estudo anterior de Nascimento et al. (2021b), Nascimento et al. (2021a) apresentam resultados promissores no diagnóstico do sistema.

Os dados coletados serviram para auxiliar no planejamento da capacidade do serviço e nas contramedidas de segurança sem o uso de conjuntos de dados de terceiros, evitando situações em que os dados não representam bem o ambiente e demonstram resultados insatisfatórios de detecção (PEREIRA et al., 2021; PEREIRA et al., 2020).

Em resumo, as principais contribuições deste trabalho são:

- Desenvolver uma metodologia não intrusiva para diagnóstico de sistemas *Linux* para apoiar o planejamento de capacidade de recursos e implementação de contramedidas de segurança utilizando servidores de nível corporativo.
- Desenvolver conjuntos de dados mais representativos do comportamento situacional do hospedeiro, contendo a seleção das características mais influentes entre os HPCs em situações de uso comum e ataques DDoS no sistema. Assim, excluindo a necessidade de utilização de dados de terceiros.

1.3 TRABALHOS RELACIONADOS

Os trabalhos aqui relacionados fazem parte da investigação para o desenvolvimento da metodologia apresentada e os trabalhos relacionados. Abrangendo os temas e soluções de contramedidas de SI, abordando detecção de ataques de DDoS, anomalias e *malwares* no contexto geral, os quais têm como objetivo prejudicar a disponibilidade e a QOS do sistema. A utilização de HPCs serve como suporte no desenvolvimento dessas contramedidas de segurança e o planejamento da capacidade dos recursos do hospedeiro.

Ameaças como os ataques DoS evoluíram consideravelmente em suas abordagens ofensivas, tornando sua detecção ainda mais complexa e de difícil controle. O trabalho realizado por Demoulin et al. (2019) apresenta *FINELAME*, uma estrutura simples e eficaz

independente da linguagem utilizada, para detectar ataques DoS assimétricos. Utilizando os dados coletados no diagnóstico do sistema para treinar um modelo de utilização de recursos usando ML, sem modificar o código-fonte do sistema. A ferramenta usa recursos de diagnósticos recentes do *Linux*, denominado *Berkeley Packet Filter* (eBPF), que permite a interação de trechos de código válidos em pontos designados no *Operating System* (OS) ou na aplicação. Em seguida, correlacionando os dados coletados com os monitores de recursos (como *Central Processing Unit* (CPU), *Random Access Memory* (RAM), armazenamento e rede) durante a execução dos aplicativos no sistema para detectar solicitações ofensivas em andamento no sistema. Com o trabalho proposto, é possível detectar comportamentos anômalos na pilha de software e ataques no sistema, utilizando recursos mínimos do hospedeiro antes mesmo que os ataques consigam tornar o serviço indisponível. O trabalho proposto utiliza diferentes recursos em sua abordagem, distintos da metodologia aqui apresentada, para classificar situações anômalas no sistema.

No trabalho proposto por Oikonomou e Mirkovic (2009) foi possível mostrar em simulação que os modelos propostos podem detectar ataques de DoS e *Flash Crowd Attacks* (FCA). Os ataques DoS sobrecarregam o aplicativo, inundando-o com solicitações legítimas usando vários *Robot* (Bot), gerando um fluxo considerável de requisições. Com os mesmos modelos, Tandon et al. (2019) propôs a implementação em um sistema FRADE e avaliou em três servidores Web com diferentes aplicações e configurações, utilizando *Emulab* [2]. A solução proposta funciona como uma defesa contra ataques DoS e FCA, identificando e colocando os agentes maliciosos em lista de permissões. FRADE observa as principais diferenças de comportamento entre usuários legítimos e Bot, criando assim um perfil de uso e classificando o uso como legítimo ou anômalo, operando em dois estados, aprendendo e classificando. As solicitações do usuário ao serviço Web que não correspondem ao modelo dinâmico são classificadas como Bot pelo sistema e adicionadas em regras específicas de *Firewall*. A solução proposta é baseada na simulação de um único tipo de ataque DoS, diferente da metodologia aqui apresentada, que utiliza diversas ferramentas de ataque.

Agentes maliciosos têm realizado grandes esforços ao longo dos anos em termos de técnicas de amplificação ou uso de Botnet para enviar grandes fluxos de ataques DoS, apresentando uma ameaça significativa à disponibilidade de serviços da web, consumindo seus recursos. O trabalho proposto por Meng et al. (2018) denominado *Rampart* é um mecanismo de defesa que visa proteger aplicações Web de ataques de DoS de exaustão de CPU. A solução proposta usa métodos estatísticos e perfis de utilização para detectar e interromper ataques de DoS em andamento. O *Rampart* bloqueia os ataques subsequentes adicionando filtros de uso e atualizando-os de forma adaptativa. A solução é um sistema de defesa baseado no hospedeiro implementado em uma extensão do mecanismo *PHP*

[2] https://www.emulab.net/

Zend [3], que pode ser aplicado sem modificar o código-fonte da aplicação e utiliza recursos mínimos do hospedeiro. A solução proposta é implementada como uma extensão da aplicação, criando uma dependência que difere da solução proposta aplicada diretamente ao hospedeiro.

O trabalho proposto por Addepalli, Karri e Jyothi (2017) cria uma estrutura de detecção para vários tipos de ataques de DDoS baseada no próprio hospedeiro; isso é chamado de BRAIN (*Behavior-Based Adaptive Intrusion Detection in Networks*). A solução proposta aproveita os HPCs já disponíveis na maioria dos processadores, para modelar o comportamento do sistema e da aplicação. Assim, correlacionando as estatísticas da rede e da aplicação para realizar as análises. As técnicas de ML utilizadas para classificar e selecionar as características mais relevantes, entre os contadores de desempenho disponíveis no processador, podem detectar anomalias no sistema e, assim, identificar ataques DDoS no serviço. A solução proposta é desenvolvida em *desktops* e processadores caseiros, rodando um servidor web.

A metodologia desenvolvida por Krishnamurthy, Karri e Khorrami (2019) é capaz de monitorar em tempo real o *software* executado para detectar anomalias no sistema. A proposta é implementada em um processador embutido em um *Cyber-physical System* (CPS). A abordagem utiliza ML para realizar análises em um dispositivo de *hardware* que realiza monitoramento contínuo e em tempo real de HPCs. As observações são realizadas através dos processos multitarefa do sistema. Com a captura desencadeada pelas interrupções desses processos, criam-se perfis das medições dos HPCs e um perfil temporal do código em execução do sistema é desenvolvido. Anomalias no sistema são detectadas através da análise das séries temporais capturadas, que irão detectar qualquer desvio dos padrões nas características presentes nas séries. A solução desenvolvida utiliza uma abordagem chamada *BlackBox*, onde o código-fonte não está disponível, dificultando a auditoria e a replicação em outros ambientes.

O *Eunomia* foi desenvolvido por Yuan et al. (2011) e é um sistema não intrusivo capaz de detectar ataques baseado em programação orientada a retorno. A detecção inclui ataques de injeção de código e retorno para ataques *libc*, sem qualquer alteração no código-fonte ou adicionar qualquer hardware para esse fim. Com a solução proposta, foi possível mostrar a aplicabilidade prática dos HPCs, amplamente disponíveis em diversos processadores modernos, para detectar brechas de segurança no sistema. Os autores descobriram que várias violações de segurança em sistemas, como visto em ataques de DDoS, normalmente causam um fluxo anormal dentro do sistema e geram um desvio significativo nos dados de HPCs. Esses desvios podem ser identificados nas amostras dos contadores de desempenho capturados e comparados entre a situação comum e a situação de ataque do sistema. A solução visa detectar uma ampla variedade de ataques à segurança, identificando e classificando seus padrões de comportamento. Devido à grande dimensionalidade

[3] https://framework.zend.com/

das variáveis observadas na proposta, a utilização dos recursos do hospedeiro acaba sendo bastante significativa, diferente do método aqui proposto que seleciona as variáveis mais relevantes.

No trabalho proposto por Torres e Liu (2016) é investigada a aplicação de um esquema de detecção de ataques baseado em anomalia. Explorações de dados, também conhecidas como ataques orientados a dados, fazem com que o *software* viole a confidencialidade e integridade dos dados e impacte negativamente os servidores Web, resultando na exposição de dados confidenciais. A abordagem usa informações coletadas dos contadores de desempenho e tempo de execução de aplicativos para detectar os ataques. O autor usa a *Support Vector Machine* (SVM) para classificar os comportamentos habituais e anômalos do sistema no momento exato. Embora o foco da solução aborde explicitamente a vulnerabilidade *Heartbleed*, ele pode investigar a aplicação da solução em um contexto geral de ataques, diferente da metodologia apresentada que trata de ataques DDoS de forma geral.

O trabalho de Alam et al. (2017) propôs uma abordagem que utiliza ML para detectar ataques de canal lateral na microarquitetura. A vulnerabilidade que usa canais secretos para vazar informações é explorada em vários tipos de segmentos de ataque. A solução possui técnicas de série temporal para criar perfis de usuário e coletar eventos de *hardware* de baixo nível, como os HPCs disponíveis na *Application Programming Interface* (API) de eventos de desempenho no *Linux*. A solução possui fases bem definidas onde as anormalidades dentro do sistema são detectadas, analisadas e classificadas. Embora o foco da solução aborde explicitamente a vulnerabilidade dos ataques *Side-channel*, ela não realiza uma análise mais profunda na investigação dos HPCs utilizados, diferente da metodologia apresentada.

No trabalho proposto por Elnaggar, Chakrabarty e Tahoori (2017), Árvores de Meio-espaço foram usadas na análise (TAN; TING; LIU, 2011) onde as árvores constituem o classificador de treinamento online para detectar cavalos de Troia de *hardware* em núcleos de microprocessadores nos dados de fluxos de processamento. A solução desenvolvida e implementada apenas com o *hardware* necessário para o design do ambiente, resolve as limitações das soluções anteriores em relação à escalabilidade das defesas; quando o *hardware* dedicado é responsável pela detecção. A proposta desenvolvida coleta continuamente dados de HPCs, que fornecem informações detalhadas sobre as atividades no nível da microarquitetura do sistema. A proposta desenvolvida coleta continuamente dados de HPCs que fornecem informações detalhadas sobre as atividades no nível da microarquitetura do sistema. O sistema pode detectar cavalos de Troia que causam DoS, o que acaba alterando a funcionalidade do sistema e, consequentemente, afeta negativamente seu desempenho. A solução proposta realiza uma análise na investigação de HPCs utilizando *Half-Space Trees* (HST) Árvores de Meio-espaço, o que difere da metodologia apresentada que utiliza *Random Forest* (RF) Floresta Aleatória, uma análise mais atual no tratamento de ataques

DDoS.

No trabalho de Leng, Zwolinski e Halak (2017) foi desenvolvida uma metodologia baseada em HPCs para detectar erros devido a falhas do sistema. Para garantir a confiabilidade do serviço, os autores traçaram um perfil do comportamento típico do sistema; portanto, qualquer desvio do perfil regular indica uma anomalia do sistema que pode eventualmente levar à falha. A solução mede a utilização de HPCs que ocorrem durante a execução do programa usando *GemFI*, um simulador de arquitetura baseado em *Gem5* com recursos adicionais de injeção de falha (PARASYRIS et al., 2014). A detecção é executada quando qualquer desvio do perfil regular é apresentado, indicando que o sistema se encontra em uma situação anormal e identificando comportamentos anômalos que podem ocasionalmente causar indisponibilidade. Considerando que ataques de DDoS provocam anormalidades no sistema, a metodologia pode ser utilizada para detectar essas ocorrências. Ao contrário da solução proposta, realizamos nossos experimentos em um ambiente rodando um servidor de nível corporativo e não com simulações.

Em Rohan, Basu e Karri (2019) dois algoritmos de detecção de *malware*, *Branch-Store-Load-Arithmetic Instruction Sequences* (BSLA) e *General-Purpose-Registers* (GPR), usam a depuração de *hardware* presente em computadores para extrair recursos de *hardware* confiáveis, como os HPCs. Contadores usados para criar um classificador de *malware*, usando ML, distinguem *malware* de programas benignos. O primeiro algoritmo usa HPCs personalizados e, portanto, é possível avaliar e classificar certos tipos de instruções no binário. O segundo algoritmo monitora os estados dos registros de uso geral, que rastreiam os valores dos registros em diferentes pontos durante os programas em execução. Ao contrário da solução proposta, desenvolvemos nossa metodologia com base no próprio hospedeiro; executando-o em um ambiente com um servidor de nível empresarial, investigando os HPCs que mais influenciam a classificação dos perfis comportamentais.

Com base na Tabela 1, os trabalhos relacionados foram classificados em seis grupos distintos onde, o **Contexto** informa a qual circunstância o trabalho se refere; **Baseado no Hospedeiro** classifica os trabalhos que são desenvolvidos utilizando o próprio hospedeiro como contramedidas, sem a dependência necessária em *hardware adicional*; **Intermediária da Rede** classifica os trabalhos que são desenvolvidos utilizando ativos adicionais no caminho do trafego, como por exemplo a implementação de *Switches*, *Firewalls*, IDS e IPS no ambiente; **Segmento Empresarial** classifica os trabalhos que são desenvolvido utilizando equipamentos de porte empresarial, como servidores de *rack* e processadores exclusivos para servidor. Já a classe **Multivariável** informa se o trabalho contempla várias variáveis em suas análises, como a utilização de variáveis de desempenho dos recursos, como CPU, RAM e rede juntamente com as variáveis de HPCs. O grupo **Simulação** classifica os trabalhos que utilizam simulação no desenvolvimento da sua contramedida, como ferramentas e utilitários que simulam um ambiente ou situações de erros e falhas. Por fim, a classe **HPCs** onde são relacionados os trabalhos que utilizam os contadores de

desempenho de *hardware* em seu trabalho, desenvolvendo análises ou contramedidas em função desse contadores de desempenho.

Tabela 1 – Comparação Trabalhos Relacionados

Trabalhos	Contexto	Baseado no Hospedeiro	Intermediária da Rede	Segmento Empresarial	Multivariável	Simulação	HPCs
(DEMOULIN et al., 2019)	DoS *Asymmetric*	✓		✓	✓		
(TANDON et al., 2019)	DDoS FCA		✓			✓	
(MENG et al., 2018)	DoS *CPU-exhaustion*	✓		✓			
(ADDEPALLI; KARRI; JYOTHI, 2017)	DDoS HTTP	✓			✓		✓
(KRISHNAMURTHY; KARRI; KHORRAMI, 2019)	*Malware*		✓	✓	✓		✓
(YUAN et al., 2011)	*Return-to-libc Attack*	✓			✓		✓
(TORRES; LIU, 2016)	*Heartbleed*	✓					✓
(ALAM et al., 2017)	*Side-channel Attack*	✓		✓	✓		✓
(ELNAGGAR; CHAKRABARTY; TAHOORI, 2017)	*Hardware Trojan*		✓		✓	✓	✓
(LENG; ZWOLINSKI; HALAK, 2017)	*Trojan*/Falhas		✓			✓	✓
(ROHAN; BASU; KARRI, 2019)	*Malware*		✓		✓		✓
Este Trabalho	DDoS HTTP	✓		✓	✓		✓

1.4 CONSIDERAÇÕES FINAIS

O presente trabalho propõe uma metodologia diagnóstica não intrusiva; sem a necessidade de *hardware* adicional para realizar a coleta e análise dos dados. Portanto, tendo vantagens sobre as soluções citadas anteriormente que utilizam *hardware* dedicado para tal afinidade e resolvendo problemas relacionados à escalabilidade.

O trabalho proposto se difere principalmente da solução desenvolvida por Alam et al. (2017), diretamente pelas análises utilizadas para realização do pré-processamento dos dados de HPCs, consequentemente selecionando uma quantidade diferentes de contadores. Ambos os trabalhos utilizam técnicas de ML, mas abordam ataques diferentes em seu

trabalho em comparação a este. A metodologia de diagnóstico faz parte da primeira fase de uma solução proposta para detectar e mitigar ataques DDoS em servidores web de nível corporativo, dando continuidade aos trabalhos onde os diagnósticos são realizados em computadores domésticos. A investigação de HPCs foi abordada em diversos estudos como ferramenta para determinar padrões ou fazer comparações entre os sistemas avaliados (WOO, 2021; DAS et al., 2019; KOSMIDIS et al., 2013).

Conceitos estatísticos como Correlação, teste de Kolmogorov-Smirnov e *Principal Component Analysis* (PCA) são apresentados nos trabalhos de Mirchev e Mirtchev (2020), Kunkel e Dolz (2018) para apoiar a investigação de recursos de desempenho de *hardware*. O uso dessas ferramentas ajuda a reduzir o número de características utilizadas, o que dá suporte ao objetivo da metodologia aqui proposta. O hospedeiro executa as técnicas de diagnóstico estatístico e de ML, os dados coletados dos HPCs são contadores de desempenho de baixo nível, presentes na arquitetura do processador. Como o diagnóstico possui baixa utilização de recursos, é considerado adequado para operar em tempo real em soluções de contramedidas de segurança.

1.5 APRESENTAÇÃO

A seguir a dissertação está organizada da seguinte maneira: No Capítulo 2 abordamos a Fundamentação Teórica, que se faz necessária para o entendimento dos conceitos, definições e técnicas da das áreas da SI, Diagnóstico, Redes de Computadores, Estatística e *Artificial intelligence* (AI) mencionadas no trabalho. O Capítulo 3 apresenta detalhes da Solução Proposta, como os procedimentos utilizados para avaliação e desenvolvimento da metodologia. O Capítulo 4 discute os Estudos de Caso dessa pesquisa e a aplicabilidade da solução desenvolvida apresentada anteriormente. Por fim, O Capítulo 5 apresenta as conclusões, recomendações, lições aprendidas e as considerações finais justamente dos trabalhos futuros que poderão ser desenvolvidos com base no presente trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Esse capítulo abordará definições e conceitos sobre segurança da informação, juntamente de questões estruturais e históricas da internet. Informações que auxiliarão para o entendimento do trabalho aqui apresentado, fornecendo conhecimento prévio necessário para o entendimento como um todo da pesquisa desenvolvida e apresentada nos próximos capítulos.

2.1 SEGURANÇA DA INFORMAÇÃO

A segurança da informação é o processo de controle que envolve várias medidas necessárias para manter e preservar a informação de um determinado conjunto, sendo esse conjunto uma organização ou um ativo. Proteção essa contra agentes maliciosos que ameaçam de forma direta ou indireta os ativos detentores de informações sensíveis e não sensíveis (ALGHAMDI; WIN; VLAHU-GJORGIEVSKA, 2020; VEIGA et al., 2020).

É crescente a má utilização dos computadores e das redes, onde *malwares* estão sendo disseminados deliberadamente pela internet sem muitas formas de controle. Isto representa uma enorme ameaça para a sociedade, uma vez que roubos de identidades, violações de dados, ameaças às vulnerabilidades de smartfones, ataques de *phishing* e engenharia social são exemplos de problemas encontrados pela ausência de responsabilidade das partes envolvidas na conexão. Portanto, existe uma carência enorme por mais conhecimento sobre segurança da informação pela sociedade, além da necessidade de maior capacitação dos profissionais de tecnologia da informação. Numa sociedade com mais usuários treinados com conceitos básicos em segurança, é possível gerenciar com mais segurança seus próprios dados e informações e, consequentemente, os sistemas que utilizam (LI et al., 2020; ASLAN; SAMET, 2020).

Os princípios da segurança da informação são compostos por três componentes fundamentais para sua funcionalidade e aplicação. O triangulo referenciado como CIA em inglês, significando *Confidentiality*, *Integrity* e *Availability*. Correspondendo à sigla CID em português, refere-se à Confidencialidade, Integridade e a Disponibilidade; e juntos são os pilares essenciais para o controle da segurança da informação. Esses pilares fornecem uma maneira direta de entender e resolver problemas relacionados à segurança da informação, apesar da literatura acadêmica não descartar totalmente o modelo da tríade da CIA, mas tenta introduzir vários aprimoramentos a ela servindo de suporte ao modelo (DIESCH; PFAFF; KRCMAR, 2020). Os controles que auxiliam mecanismos de proteção para estarem em conformidade, sendo capaz de tratar riscos, ameaças e vulnerabilidades eminentes e sua ausência sobre os ativos pode comprometer negativamente o ambiente e causar danos irreparáveis (SINGH, 2020; MAČEK; MAGDALENIĆ; REđEP, 2020).

2.1.1 Definições

A seguir podemos observar algumas definições e termos importantes acerca a segurança da informação, que se faz necessária como suporte e conhecimento prévio ao entendimento do assunto abordado posteriormente:

Ativo
É qualquer coisa que tenha valor, seja para organização ou pessoal. Qualquer componente que agregue valor, podendo ser uma informação, pessoas ou equipamentos, o que são necessários para algo ou alguém. Ex.: Funcionários, *Firewall* e computadores.

Ataque
Ação ou tentativa de causar algum dano, podendo ocasionar na destruição, exposição, alteração, inutilizar ou obtenção de acesso não autorizado a ativos. Ex.: Ataques de DoS/DDoS, *Phishing* e *Exposed*.

Ameaça
É qualquer indicação, circunstância ou evento com potencial de causar dano ou perda a ativos e organizações. Ex.: *Malware*, *Ransomware* e *Spyware*

Agente Malicioso
Ativo que possuí, age ou pensa com malícia, que comete atos ilícitos.

Terceiros
Pessoas ou entidades que não estão diretamente relacionadas com a parte envolvida, não estando diretamente ligado ao processo em questão.

Responsabilidade
Atribuição de ações ou decisões que foram ou poderão ser tomadas e suas devidas consequências.

Risco
Probabilidade de insucesso em função de um acontecimento eventual, cuja ocorrência não depende exclusivamente da vontade dos interessados.

Vulnerabilidade
Qualidade ou estado do que é ou se encontra vulnerável, sendo uma particularidade que indica um estado de fraqueza.

2.1.2 Confidencialidade

É a propriedade de que uma informação estará disponível apenas para agentes ou entidades com a devida autorização. É o processo que tem como controle a disponibilização de informações de confiança entre as partes envolvidas, assegurando que elas não serão divulgadas a terceiros sem autorização (YEE; ZOLKIPLI, 2021).

A confidencialidade pode ser provida através de alguns procedimentos como, por exemplo, a criptografia dos dados enviados e armazenados, a implementação de controle de acesso aos dados e a capacitação dos funcionários. Na manipulação de dados, medidas podem ser tomadas para garantir a privacidade das informações disponibilizadas. A autenticação dos usuários por meio de criptografia, controle de acesso e *User Identifier* (UID), como senhas, autorização por dispositivos físicos ou senhas de uso único *One-time Password* (OTP), que concederá acesso seguro às informações (YEE; ZOLKIPLI, 2021; THANDEESWARAN et al., 2020).

2.1.3 Integridade

É a propriedade que se refere que as informações não sejam alteradas de forma não autorizada. A integridade oferece garantias de que as informações não sejam alteradas por pessoas não autorizadas, assegurando que ninguém tenha modificado, adulterado ou corrompido os dados. O devido controle da integridade, garante que apenas usuários autorizados possam acessar ou modificar os dados. No entanto, agentes maliciosos tentam obter essas informações de forma ilegal, por meio de ações humanas ou por meio de *malwares* e fazem com que a integridade do ativo seja comprometida (GŘIVNA; DRÁPAL, 2019; LINKOV; ROSLYCKY; TRUMP, 2019).

Existem algumas formas de garantir a integridade dos ativos, como os dados e informações, principalmente quando essas informações precisam ser deslocadas ou transmitidas de um lugar para o outro. Uma das técnicas bastante conhecidas para reforçar a integridade é denominada *Hashing* (BATENI; SAEIDI, 2019; GIBSON, 2017), que se refere ao processo de geração de uma saída (*output*) de tamanho fixo a partir de uma entrada (*input*) de tamanho variável, realizado através de fórmulas matemáticas (AJISH; KUMAR, 2020; TIWARI; SINGH; PRABHAKAR, 2020; PIEPRZYK; SADEGHIYAN, 1993). Isso consiste em quando os dados nunca forem alterados, o *hash* resultante será sempre o mesmo e alterando em sustações de modificações.

2.1.4 Disponibilidade

É o processo de garantia de que os ativos associados sejam eles, informações, dados, pessoas ou equipamentos estejam disponíveis para os usuários legítimos quando necessário, acessando ou recebendo informações sem interferência ou obstrução de influências externas (SONG; FINK; JESCHKE, 2021). Esse princípio está diretamente relacionado à eficácia

do sistema e do provimento do serviço, onde o ativo precise estar acessível quando for necessário (STANTON et al., 2021).

A disponibilidade que é um dos pilares importantes da SI, é um pilar associado diretamente à parte operacional da organização. Portanto, para garantir a disponibilidade da informação é preciso ter mecanismos e processos como garantia da preservação da disponibilidade (MISHRA; SHARMA; ALOWAIDI, 2021; QADIR; QUADRI, 2016). Ataques de DoS/DDoS são ameaças diretas à disponibilidade, onde agentes maliciosos realizam esforços para interromper ou prejudicar o serviço ofertado. Portanto, é de suma importante a implementação de dispositivos e técnicas de tolerância a falhas e redundância como, duplicação de servidores, *backups*, balanceamento de carga e redundância de discos que dão suporte a garantia da disponibilidade mesmo diante de ações maliciosas ou incidentes (GIBSON, 2017).

2.1.5 Malwares

A palavra *malware* vem da combinação das palavras em inglês *malicious* e "*software*" e se referem a basicamente todos e qualquer software malicioso ou indesejado, como os *vírus*, *worms*, *cavalos de troia* e os *spywares* (HINTZBERGEN et al., 2018). A palavra *malware* também é conhecida por diversos outros nomes como, por exemplo, *software* malicioso ou código malicioso. Referindo a um programa cujo objetivo é causar danos ao computador hospedeiro, o computador no qual é executado ou computadores aos quais este computador está conectado, o que acontece em uma Botnet (HULL; JOHN; ARIEF, 2019).

Uma prática muito comum e essencial na investigação de ataques de *malware* é a utilização de *logging*. O *logging* é capacidade de registrar eventos e produzir evidências capazes de identificar com detalhes tudo o que acontece no sistema. Os logs têm a função de registrar o que aconteceu, quando aconteceu, onde aconteceu e quem o fez. Dessa forma, servindo de suporte na identificação de anomalias no sistema. Um dos maiores problema em relação ao registro desses logs, é a grande quantidade de dados que eles produzem, sendo um grande problema o armazenamento desses registros porque uma quantidade muito densa de dados implica diretamente no tempo necessário para analisá-los. A essência da utilização dos registros de monitoramento de anomalias e eventos é o equilíbrio entre o que é necessário registrar e a quantidade de espaço disponível para o armazenamento (GIBSON, 2017).

Malwares podem tirar proveito de falhas no sistema, podendo ser utilizado para montar ataques DoS/DDoS. O desenvolvimento de soluções para o problema do *malware* é vital para melhorar a segurança da internet mundial, sendo vital para uma infraestrutura crítica para nossa sociedade (LEON et al., 2021; PARK et al., 2021).

2.1.6 Ataques DDoS

Interromper os serviços de um usuário legítimo exaurindo os recursos do servidor é essencialmente o principal motivo desses ataques de inundação no nível do aplicativo, levando a perdas de receita, aumento dos custos de implementação de contramedidas e mitigação e a restauração dos serviços afetados (DAYANANDAM et al., 2019; NASCIMENTO et al., 2021a).

Os ataques DDoS de inundação são classificados em categorias com base no nível, forma e protocolo utilizados. Esses ataques podem ser basicamente divididos em três grandes grupos, sendo eles os Ataques baseados em Volume, Ataques de Exploração de Protocolo e Ataques da camada de Aplicação (MIRCHEV; MIRTCHEV, 2020; ZARGAR; JOSHI; TIPPER, 2013). A Figura 3 descreve de forma visual a arquitetura básica de um ataque de DDoS, onde o agente malicioso tem em seu domínio, maquinas infectadas denominadas Mestre, que através delas é possível executar e enviar procedimentos para várias outras maquinas infectadas em seu controle. Com uma grande quantidade de máquinas à sua disposição, o agente malicioso é capaz de enviar requisições válidas a um determinado alvo e desta forma conseguindo afetar negativamente o desempenho do serviço prestado (ISMAIL et al., 2021)

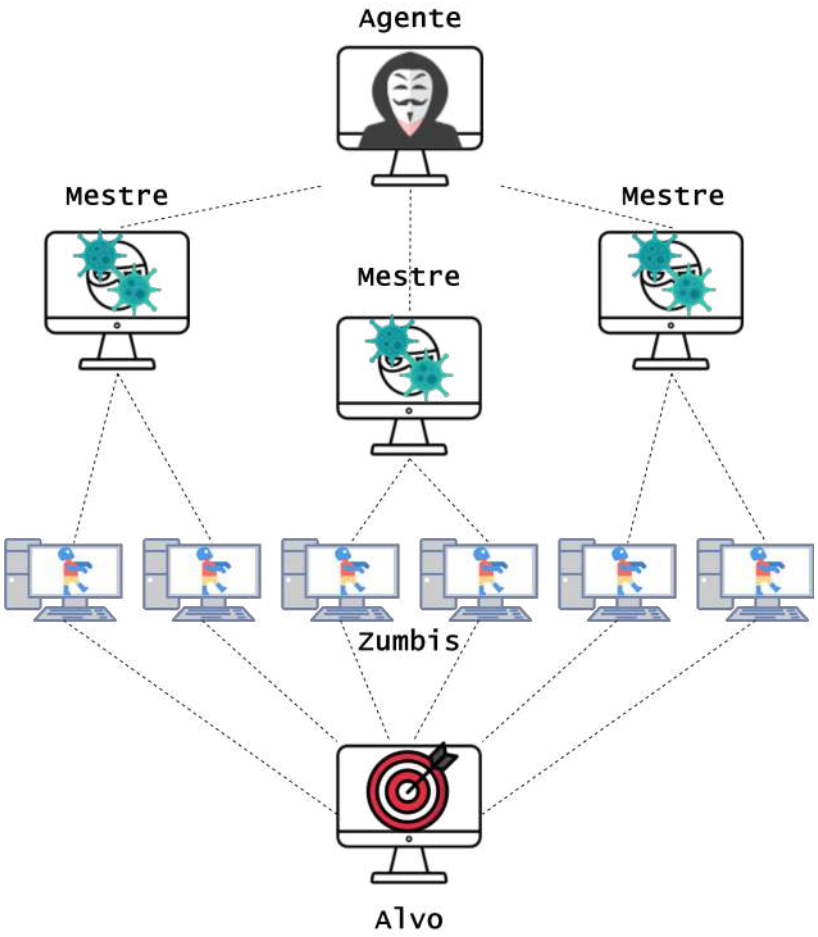


Figura 3 – Arquitetura de um Ataque de DDoS

- **Ataques baseados em Volume**

Tipos mais comuns de ataques de DDoS, funcionam basicamente enviando grandes fluxos de requisições/solicitações ao servidor, com o objetivo de esgotar os recursos do serviço, como a largura de banda do site alvo (ZHANG et al., 2020; AKANJI et al., 2020). Esses ataques são realizados principalmente utilizando pacotes de protocolo *Transmission Control Protocol* (TCP), *User Datagram Protocol* (UDP), *Internet Control Message Protocol* (ICMP) e *Domain Name System* (DNS); e geralmente não exigem grande quantidade de tráfego a ser gerada pelos agentes maliciosos, fazendo com que esse tipo de ataque seja o tipo mais simples de ataque DDoS. Cuja ação normalmente redireciona as solicitações usando um endereço IP de origem falsificado (SANDHYA, 2019; LEE, 2021).

- **Ataques de exploração de protocolo**

Esses ataques se concentram em prejudicar a QOS e a disponibilidade do serviço afetando os usuários legítimos que o utilizam e esgotando os recursos do servidor (NASCIMENTO et al., 2021b). Esses tipos de ataques consomem os recursos do servidor ou dos ativos intermediários da rede, como os *firewalls* e IPS/IDS, com os ataques de inundação *Synchronize* (SYN), ataques de pacotes fragmentados, Ping da Morte e *Smurf* DDoS (MACIEL et al., 2018; RANJAN et al., 2008). Os ataques de DDoS no nível do aplicativo geralmente são mais imperceptíveis em comparação aos ataques volumétricos pelo fato de simularem com precisão o trefego benigno. Isso ocorre principalmente naqueles que utilizam o protocolo HTTP, uma vez que esses ataques são frequentemente mencionados como os principais tipos de ataques de inundação DDoS (JAAFAR; ABDULLAH; ISMAIL, 2019).

- **Ataques da camada de aplicação**

Em ataque na camada de aplicação, os agentes maliciosos enviam um grande número de solicitações válidas para o servidor da Web alvo, visando vulnerabilidades na aplicação, com objetivo de causar instabilidade ou tornar serviço indisponível (NASCIMENTO et al., 2021a). Solicitações como *GET*, *POST*, *HEAD* ou *OPTIONS* são enviadas ao servidor, inspecionando e solicitando as respostas do servidor de cada item do site com o intuito de exaurir completamente seus recursos (NASCIMENTO et al., 2021b; CIRILLO et al., 2021). O ataque parece uma série de consultas legítimas à aplicação, direcionada normalmente ao processo de maior uso da CPU no servidor web, gerando muitas solicitações válidas e utilizando número relativamente pequeno de máquinas de ataque (ODUSAMI et al., 2020). Como uma das suas principais características, tem como consumir lentamente os recursos disponíveis, com baixa utilização da largura de banda do alvo, tornando esses ataques de difícil identificação (LIU et al., 2018; ODUSAMI et al., 2020).

A prevenção, detecção e a mitigação de ataques DDoS são de difícil manipulação devido principalmente à sua natureza distribuída e seu volume. Por esse motivo, as contramedidas de defesa não ficam na mesma intensidade que as ferramentas de ataque. Cada ferramenta utiliza algum mecanismo único para executar o ataque, dificultando a detecção e, consequentemente, a sua mitigação (KUMAR; KUMAR et al., 2016). Existem alguns métodos de proteção de ataques DDoS e um deles bastante conhecido é o CAPTCHA (Teste de Turing Público Totalmente Automatizado para Diferenciar Computadores de Humanos). Esta técnica insere um desafio na seção do usuário, dificultando que os serviços sejam acessados por usuários não humanos. Há a possibilidade de envio de usuários suspeitos para análise posterior, utilizando esta abordagem para confirmar a veracidade dos usuários ou utilizando técnicas como o balanceamento de carga e o redimensionamento do serviço. Assim, a capacidade do servidor é aumentada e consegue-se ter um melhor desempenho nas situações de ataques de DDoS, mesmo diante de grandes fluxos de acessos simultâneos, como as situações de não ataque (AAMIR et al., 2021). As motivações por trás dos ataques de DDoS podem ser diversas, onde os mais comuns são ganhos financeiros, vingança, diferenças ideológicas e política (ROBINSON; THOMAS, 2021; MAHJABIN et al., 2017).

Existem vários tipos de ataques de DDoS de inundação, mas em geral, os ataques de inundação baseados em HTTP têm seu foco principal, entre suas variações de ataques, sobrecarregar um servidor web, forçando-o a se comportar de maneira anormal. Agentes mal-intencionados se apropriam indevidamente dos computadores dos usuários, sem seu conhecimento, para gerar tráfego que normalmente se comporta como um usuário legítimo acessando o sistema. Esses ataques criam um grande volume de conexões e requisições, tornando desafiador a classificação do tráfego legítimo do malicioso, onde as taxas de transmissão e os números de solicitação são geralmente muito maiores do que as solicitações de usuários reais. Assim, causando uma enxurrada de solicitações, consumindo indevidamente os recursos do servidor, como CPU, RAM e largura de banda. Prejudicando as conexões de usuários válidos do sistema, fazendo com que o servidor alvo, deixe de responder adequadamente às solicitações enviadas pelos usuários legítimos, ou até mesmo chegar a tornar o serviço indisponível (DAYANANDAM et al., 2019; SREERAM; VUPPALA, 2019).

2.1.7 Mecanismos de Proteção

Contramedidas de SI refere à prática da mitigar riscos potenciais ou vulnerabilidades eminentes. Podendo ser física como dispositivos de *hardware*, com *hardware* dedicados para proteção, como os *Firewalls* ou logico como os IPS, com a utilização de softwares e suas configurações e procedimentos que elimine ou reduza a probabilidade de vulnerabilidades que um agente malicioso possa vir a explorar (HINTZBERGEN et al., 2018).

Segundo Clark (2017), considerando a estrutura atual da internet, temos ofertas de

baixo nível de segurança entre os serviços oferecidos. Portanto, dentro de um ambiente controlado, é necessário decidir quais problemas têm prioridade sobre os demais para que possam ser resolvidos sistematicamente. Os engenheiros já previram problemas causados por falta de responsabilidades, como os ataques de DoS e DDoS. Soluções foram propostas ao longo do tempo, mas alguns problemas têm níveis alto de complexidade em sua implementação, o que dificulta consideravelmente o desenvolvimento de soluções (GUPTA; DAHIYA, 2021; CHEN et al., 2010).

Em um projeto estrutural de um ambiente mais seguro, é necessário adicionar camadas de segurança onde cada camada de proteção poderá sobrepor à outra em caso de falhas ou erros, dando o devido suporte a integridade e disponibilidade do serviço oferecido (FADHLILLAH; KARNA; IRAWAN, 2021; CHO et al., 2020; BHOSALE; NENOVA; ILIEV, 2017). O número de camadas de segurança ou contramedidas dentro do ambiente cria obstáculos que dificultam o acesso de agentes maliciosos que tentam invadir o ambiente ou prejudicar o serviço de alguma forma. Uma estratégia de defesa amplamente utilizada é a configuração sistemática de ativos de proteção no ambiente, como a *Moving Target Defense* (MTD), *Firewalls*, *Demilitarized Zone* (DMZ) Zona Desmilitarizada, IDS e IPS. Cada dispositivo representa um obstáculo para os agentes, cujas ações são dificultadas por diferentes níveis de controle de segurança. Assim, a falha de um ou mais dispositivos não comprometem a disponibilidade e a qualidade do serviço (ZHOU et al., 2020; SENGUPTA et al., 2020).

Conforme mencionado no trabalho de Addepalli, Karri e Jyothi (2017), antes de se implementar estratégias de segurança no ambiente e definir seus níveis, é necessário inicialmente identificar qual estratégia de defesa se adequará melhor para o ambiente, como ilustrado na Figura 4:

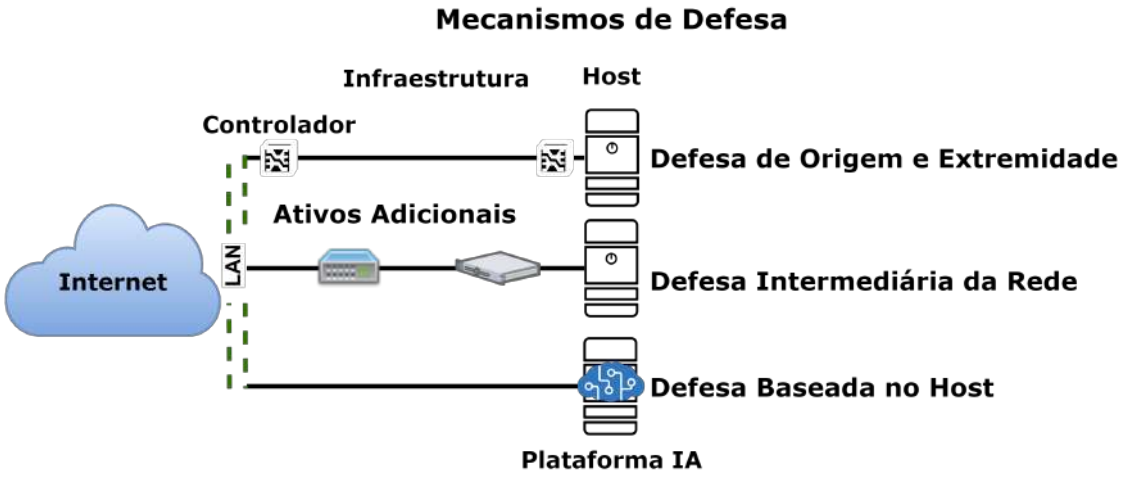


Figura 4 – Mecanismos de Defesa

Defesa de Origem e Extremidade: essa abordagem pode ser implementada no computador do usuário, limitando e otimizando o envio e recebimento dos pacotes de

rede, assegurando um maior controle entre a conexão.

Essa abordagem se torna uma alternativa bastante inviável, pelo simples motivo de que um agente malicioso poder desabilitar ou contornar o dispositivo de segurança antes de iniciar um ataque DoS/DDoS.

Defesa Intermediária da Rede: são os mecanismos intermediários dentro da infraestrutura e do ambiente, como os IDS e os IPS, que oferecem diferentes meios de proteção à infraestrutura e ao serviço.

A adição de ativos à rede, aumenta consideravelmente a complexidade estrutural da infraestrutura e, consequentemente, pode resultar na criação ou aumento de pontos únicos de falha e na possibilidade desses dispositivos serem os alvos dos ataques, acarretando indisponibilidade no serviço.

Defesa Baseada no Hospedeiro: é um método de defesa sem adição de ativos no caminho do tráfego. Nela, o próprio hospedeiro é responsável por sua defesa, que pode ser a prevenção, detecção e mitigação de ameaças. O hospedeiro é capaz de realizar por si só contramedidas de proteção.

Nas abordagens de detecção de ataques em tempo real, a abordagem baseada no hospedeiro se sobressai das alternativas de proteção baseados por excluir a dependência da adição de ativos no ambiente. Isto consequentemente aumenta sua complexidade e os gastos de recursos. Embora esse método de contramedida seja bastante eficaz, vale salientar que ainda é essencial usar parâmetros e características de hardware do hospedeiro como CPU, RAM e consumo de largura de banda da rede para aumentar a precisão dos gatilhos (FURFARO; PACE; PARISE, 2020).

Quando usados em conjunto esses métodos de contramedidas, podem aumentar consideravelmente a precisão e velocidade de detecção de ataques DoS/DDoS e com isso fornece melhores soluções de mitigação (FADHLILLAH; KARNA; IRAWAN, 2021). O comportamento anômalo do ativo, pode vir a causar falhas no sistema e esse comportamento podem ser causados pela existência de malwares no ambiente. Existem várias maneiras de se identificar tais comportamentos anômalos dentro do sistema, sendo uma delas utilizando os HPCs que têm apresentado resultados significativos na captura comportamental do sistema, tornando-se ótimos recursos para o diagnostico dessas ameaças (LENG; ZWOLINSKI; HALAK, 2017).

- **Perfil de Ataque:**

Um ou mais agentes maliciosos estão presentes ou tentando acesso no sistema ou serviço no intuito de prejudicar principalmente sua disponibilidade, seguida da confiabi-

lidade e os outros pilares da segurança da informação. Interferir direta ou indiretamente no serviço prestado, causando instabilidade para os usuários reais conectados.

- **Perfil de uso Comum:**

Nenhum agente malicioso está presente ou utilizando o serviço provido, onde somente usuários reais são encontrados ao utilizar o serviço. Classificando o comportamento como de uso habitual ou sazonal do sistema. Mesmo com o alto consumo de recursos do hospedeiro, vale ressaltar que existe a possibilidade de não se tratar de um ataque DDoS e sim de períodos sazonais, como períodos de registro online em uma instituição de ensino.

Ao centralizar a proteção da segurança de um ambiente em um ou mais dispositivos, todo o sistema tende a ficar vulnerável se esse ou esses equipamentos venham a falhar. Essas falhas de equipamentos é comum ou serem alvos de agentes maliciosos e, por isso, confiar totalmente neles é algo arriscado. Um exemplo são os ataques que consomem totalmente a largura de banda do(s) equipamento(s) com um alto volume de pacotes sobrecarregando o dispositivo e tornando-o temporariamente indisponível, já que a capacidade de processamento desses equipamentos é limitada (KSHIRSAGAR; KUMAR, 2021; CLARK, 2017; LYRA, 2015)

Os mecanismos de defesa podem ser divididos em duas categorias principais: *Mecanismos Proativos* que estão relacionados diretamente com abordagens e técnicas de predição e detecção; e os *Mecanismos Reativos*, que são as abordagens relacionadas a mitigação juntamente com as tomadas de ações (SREERAM; VUPPALA, 2019; ABLIZ, 2011).

- **Mecanismos Proativos:**

São mecanismos que envolvem desde a adição de mais recursos para provimento da disponibilidade do serviço, com o intuito de minimizar o impacto e danos dos ataques de DDoS no serviço. Juntamente com utilização de técnicas tradicionais de administração de redes, como o balanceamento de carga, replicação do serviço e utilização de regras de *Firewall*.

- **Mecanismos Reativos:**

São os mecanismos que após a predição e detecção de ações maliciosas no ambiente, ações proativas são executadas, seja tentando controlar o fluxo das solicitações, bloquear o tráfego malicioso ou tentando localizar os agentes maliciosos responsáveis. Normalmente são utilizadas em suporte para a invocação de ações humanas dos administradores, onde

analisaram a melhor abordagem que poderá ser tomada no momento de um ataque DDoS.

A comunidade acadêmica empenhada no provimento da segurança da informação realiza grandes esforços no desenvolvimento de métodos e técnicas de mecanismos de defesa reativa, realizando esforços significativos no desenvolvimento de contramedidas de segurança contra ameaças cibernéticas (NASCIMENTO et al., 2021b). O conceito de MTD, surgiu como método de defesa proativa, que serve como suporte no provimento da disponibilidade do serviço, mesmo diante de ameaças que têm como objetivo causar a instabilidade e indisponibilidade no serviço (ZHOU et al., 2020; HONG et al., 2018). Levando em consideração inicialmente que os ataques de DDoS não podem ser evitados completamente, o objetivo do MTD é se adaptar no momento da melhor forma possível às situações adversas e se defender dinamicamente delas. Assim, a manipulação de vários ativos e suas configurações no momento do ataque podem contornar um ataque em andamento e fazer com que o sistema se adapte da melhor forma. A aplicação dessa técnica nos momentos certos, aumenta consideravelmente a complexidade da infraestrutura e diminui as oportunidades dos agentes maliciosos de tirar proveito das vulnerabilidades encontradas no sistema, uma vez que o fluxo do ambiente pode ter mudado (CHO et al., 2020)

2.2 HARDWARE PERFORMANCE COUNTERS

Essas características denominadas HPCs são parte de uma unidade especial e dedicada pertencente à estrutura da CPU, chamada *Performance Monitoring Unit* (PMU) Unidade de Monitoramento de Desempenho, também chamados de *Performance Monitoring Counters* (PMCs) Contadores de Monitoramento de Desempenho, que pode acessar informações detalhadas sobre as execuções dos processos no sistema. Nas análises dos dados utilizando esses contadores, é possível identificar comportamentos incorretos do sistema, como a existência de falhas, erros e comportamentos anômalos (GREGG, 2019; WOO, 2021). Desta forma, é possível identificar situações adversas no sistema, na execução das aplicações e nos processos, exatamente quando os padrões criados posteriormente no diagnóstico se desviam dos padrões classificados como típicos (ELNAGGAR; CHAKRABARTY; TAHOORI, 2017). Embora os microprocessadores da atualidade tenham muitos contadores de desempenho, só é possível coletar um número limitado deles. Esses contadores realizam uma contagem cumulativa dos eventos relacionados, só sendo possível monitorar um número limitado desses eventos simultaneamente (KRISHNAMURTHY; KARRI; KHORRAMI, 2019; FOREMAN, 2018).

Conhecidas também como *Performance Counters for Linux* (PCL), *Linux perf events* (LPE) ou *perf events* é uma ferramenta de observação orientadas a eventos, que é capaz de capturar funções avançadas de desempenho no sistema. Dentre os tipos de eventos que a ferramenta é capaz de monitorar estão os *Eventos de Hardware*, que são contadores de monitoramento de desempenho exclusivos da CPU; *Eventos de Software*, que são eventos

baseados nos contadores de *kernel*, como as migrações de CPU e falhas (DIMAKOPOULOU et al., 2016). A PMU é um *hardware* construído dentro de um processador para medir seus parâmetros de desempenho, como *ciclos de instrução*, *acertos de cache*, *erros de cache*, *erros de ramificação* dentre muitos outros como ilustrado na Figura 5 [1] (GREGG; MAURO, 2011).

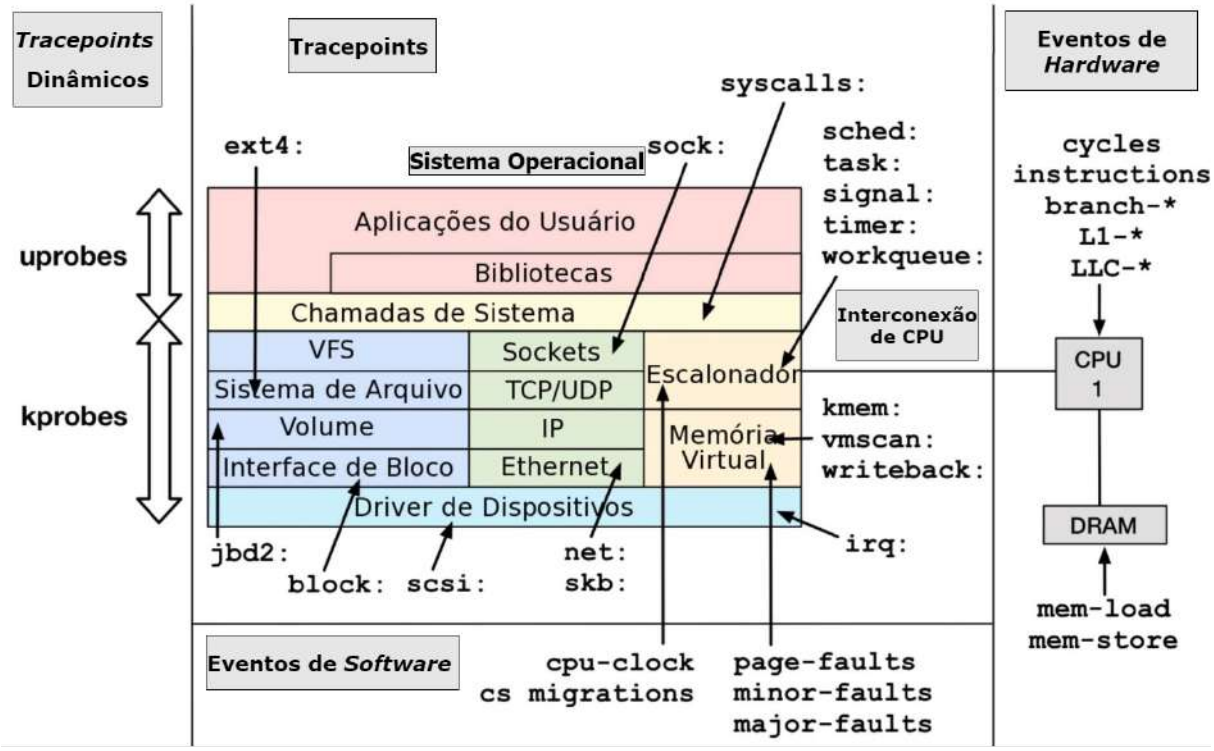


Figura 5 – Mapa do *perf events*

- **Eventos de *Hardware*:** são contadores de monitoramento de desempenho, presentes no *hardware* da CPU, que armazenam informações a respeito dos elementos dos processos, como o número de ciclos, falhas de *cache* e instruções executadas.
- **Eventos de *Software*:** são eventos que dizem respeito às ocorrências no nível do *kernel*, que são contabilizadas a partir das ocorrências de migrações de CPU, falhas menores e falhas graves.
- ***Tracepoints*:** são eventos de *software* predefinidos de instrumentação estáticos no nível do *kernel*, que são codificados em diversos locais lógicos no kernel, como as ocorrências de trocas TCP/UDP, escalonador e chamada de sistemas.
- ***Tracepoints* Dinâmicos:** são eventos de *software* de rastreamento instrumentado dinamicamente para programas e aplicativos no nível do usuário através das interfaces *kprobe* para *software kernel* e *uprobe* para *software* de nível de usuário.

[1] Disponível em: https://www.brendangregg.com/perf.html

Os HPCs foram inicialmente projetados para realizar depuração de *hardware* e de desempenho, auxiliando os desenvolvedores a compreenderem melhor o comportamento das aplicações em execução no sistema e, desta forma, conseguir realizar ajustes no desempenho (NASCIMENTO et al., 2021a). Diagnósticos específicos do sistema podem ser realizados utilizando esses contadores de desempenho que podem ser encontrados em uma ampla variedade de plataformas e arquiteturas de processadores. O tipo e o número de eventos disponíveis e a metodologia para usar esses contadores variam principalmente do seu fabricante, não apenas em sua arquitetura, mas também entre as famílias e versões dos processadores. Mesmo compartilhando a mesma arquitetura, como os processadores *Intel* e AMD de 32 e 64 bits, os contadores podem não ser os mesmos e podem ter quantidades diferentes. Por isso, cada abordagem deve ser modelada individualmente, coletando eventos de *hardware* e outras variáveis durante a execução contando automaticamente os eventos monitorados. Eles são acumulados e medidos ao longo do tempo entre intervalos pré-determinados (KRISHNAMURTHY; KARRI; KHORRAMI, 2019; TORRES; LIU, 2016).

Além de realizar amostragens em um dado intervalo, é possível também obter amostras acionadas por eventos de *hardware* da CPU, sendo uma forma de criação de perfil comportamental da CPU, que pode ser usada para esclarecer mais sobre perdas de cache e ciclos de travamento de memória (TESSELE et al., 2019). Como o número e o tipo de eventos de HPCs dependem principalmente do processador em uso, um subconjunto comum foi utilizado no trabalho, por serem comumente compartilhados pela maioria das CPUs de nível empresarial (ALAM et al., 2017; GRACIOLI; FRÖHLICH, 2011). Ciente da limitação lógica em relação a captura de tofos, os HPCs foram coletados simultaneamente por meio de um *script* de monitoramento desenvolvido especialmente para esta tarefa. A Tabela 2 contém os eventos que foram considerados para o desenvolvimento da metodologia.

Tabela 2 – Contadores de Desempenho de Hardware

branch-instructions	branch-load-misses	branch-loads
branch-misses	bus-cycles	cpu-cycles
ref-cycles	cache-misses	cache-references
dTLB-load-misses	dTLB-loads	dTLB-store-misses
dTLB-stores	instructions	iTLB-load-misses
iTLB-loads	L1-dcache-load-misses	L1-dcache-loads
L1-dcache-stores	L1-icache-load-misses	LLC-load-misses
LLC-loads	LLC-store-misses	LLC-stores

Os contadores de desempenho utilizados na metologia têm diversas características como demonstra a Tabela 3, onde todos esses contadores coletam dados do tipo inteiro. Os HPCs normalmente oferecem suporte a dois modos de utilização, sendo eles contagem e amostragem. Cada núcleo do processador possui um conjunto de HPCs que contam os eventos de desempenho no sistema. No modo de contagem, os contadores são configurados, apagados e iniciados. Então, algum tempo depois, os contadores são parados e o número de eventos é lido e relatado. No modo de amostragem, os contadores são configurados, apagados e iniciados. No entanto, eles são configurados para gerar uma interrupção de amostragem após a contagem de um certo número de eventos. Na interrupção, uma amostra é acumulada e o contador é reiniciado. Após a conclusão da carga de trabalho, as interrupções do contador são desligadas, as amostras são agregadas e as estatísticas resumidas são relatadas (WOO, 2021; BAZM et al., 2018).

Tabela 3 – Descrição dos Eventos de HPCs

Nome	**Descrição**
branch-instructions	Instruções de ramificação retiradas por um núcleo de CPU.
branch-load-misses	Falta de leitura da unidade de previsão de ramificação por um núcleo da CPU.
branch-loads	Acessos de leitura de unidade de previsão de filial por um núcleo de CPU.
branch-misses	Instruções de desvio imprevisíveis por um núcleo de CPU.
bus-cycles	Ciclos de barramento por núcleo de CPU, que podem ser diferentes dos ciclos totais.
cpu-cycles	Ciclos totais de CPU por núcleo de CPU.
ref-cycles	Incrementa a taxa constante, independentemente do *Turbo* da CPU.
cache-misses	Perdas de leitura de cache por um núcleo da CPU. Normalmente, isso indica perdas de *cache* de último nível.
cache-references	Acessos de *cache* por núcleo de CPU. Normalmente, isso indica acessos de cache de último nível, mas isso pode variar dependendo do tipo de CPU.

dTLB-load-misses	*Buffer lookaside** de tradução para erros de leitura de dados (dTLB) por um núcleo de CPU.
dTLB-loads	*Buffer lookaside** de tradução para acessos de leitura de dados (dTLB) por um núcleo de CPU.
dTLB-store-misses	*Buffer lookaside** de tradução para erros de gravação de dados (dTLB) por um núcleo de CPU.
dTLB-stores	*Buffer lookaside** de tradução para gravações de dados (dTLB) por um núcleo de CPU.
instructions	Instruções retiradas por um núcleo de CPU.
iTLB-load-misses	*Buffer lookaside** de tradução para instruções (iTLB) erros de leitura por um núcleo de CPU.
iTLB-loads	*Buffer lookaside** de tradução para acessos de leitura de instruções (iTLB) por um núcleo de CPU.
L1-dcache-load-misses	*Cache* de nível 1 para erros de leitura de dados (L1d) por um núcleo de CPU.
L1-dcache-loads	*Cache* de nível 1 para acessos de leitura de dados (L1d) por um núcleo de CPU.
L1-dcache-stores	*Cache* de nível 1 para gravações de dados (L1d) por um núcleo de CPU.
L1-icache-load-misses	*Cache* de nível 1 para instruções (L1i) erros de leitura por um núcleo de CPU.
LLC-load-misses	Falha de leitura do *cache* de último nível (LLC) por um núcleo da CPU.
LLC-loads	Acessos de leitura de *cache* de último nível (LLC) por um núcleo de CPU.
LLC-store-misses	Perdas de gravação no *cache* de último nível (LLC) por um núcleo da CPU.
LLC-stores	*Cache* de último nível (LLC) grava por um núcleo da CPU.

∗ *Translation Lookaside Buffer* (TLB) destina-se a facilitar a tradução de endereços lineares em endereços físicos, evitando consultas à tabela de páginas localizada na memória do sistema.

Os contadores do monitor de desempenho são usados pelo *software* para contar eventos específicos que ocorrem no processador. Essa classe oferece suporte aos mesmos usos de amostragem de evento baseados em interrupção e contagem com um conjunto menor de eventos disponíveis, não são arquitetônicos e variam de um modelo de processador para outro e são apenas *links* para eventos de *hardware* não refinados. Esses eventos podem ser acessados utilizando o utilitário *perf* como observador de eventos. A maioria das CPU modernas, como as fabricadas pelas empresas *Intel* e AMD, não possuem frequência de operação de processamento, possuindo frequências dinâmicas. Nas CPUs da *Intel*, essa tecnologia é denominada de *Turbo Boost* e nos processadores da AMD é chamada de *Turbo Core* (GREGG, 2019; ILSCHE et al., 2017).

Outros processadores podem ter um número diferente de contadores de desempenho e certos contadores podem ser restritos a um ou mais eventos específicos. Essas informações, e mais detalhes referentes aos HPCs e seus eventos, podem ser vistas no *Intel 64 and IA-32 Architectures Software Developer's Manual Volume 3B: System Programming Guide, Part 2*[2] e no *BIOS and Kernel Developer's Guide (BKDG) For AMD Family 10h Processors*[3]

A detecção de anomalias e ataques de DDoS com base em assinaturas, necessita de uma quantidade significativa de informações das ameaças para que seja possível realizar a detecção. Portanto, abordagens que utilizam assinaturas de ataques de DDoS se tornam ineficazes contra ataques de *Zero-Day* ou falhas de *software* nas aplicações de proteção (GARCIA et al., 2021). Logo, é difícil para um invasor burlar um detector baseado em *hardware*, quando comparado a um equivalente baseado em *software* que a detecção baseada em software tem suas limitações e vulnerabilidades; diferentemente das que se apropriam de recursos de *hardware* e se tornam mais confiáveis como os HPCs (ROHAN; BASU; KARRI, 2019).

As diferentes abordagens de utilização dos HPCs são classificadas em oito grandes grupos, que abordam diferentes formas de se utilizar os contadores de desempenho e seus eventos. Os contadores de desempenho podem ser utilizados de diversas formas voltadas para a detecção e mitigação de ataques e *malwares*. Cada abordagem poderá ser utilizada a depender do serviço prestado, a estrutura do ambiente e as contramedidas que se deseja implementar (FOREMAN, 2018).

Baseada em Assinaturas: é quando os contadores de desempenho coletam dados referente aos processos que estão em execução no sistema. Está abordagem cria perfis com assinaturas das situações de referencia, que são características de comportamento, que na ocasião desses comportamentos ou assinaturas. As informações correspondentes são então classificadas como ataques já conhecidos. Abordagem utilizada normalmente na criação de regras de execução no sistema, lista de bloqueios e antivírus.

[2] Disponível em: https://www.intel.com.br/content/www/br/pt/architecture-and-technology/64-ia-32-architectures-software-developer-vol-3b-part-2-manual.html

[3] Disponível em: https:https://www.amd.com/system/files/TechDocs/31116.pdf

A detecção de ataques por anomalias observa o perfil situacional atual do sistema e compara com as condições registrada das assinaturas de ataques, desta forma identificando comportamentos anormais no sistema causados por ataques. Abordagem bastante utilizada em diversas áreas da Segurança da Informação, como fraudes em cartões de crédito, detecção de falhas em sistemas, vigilância e validação de dados. O principal desafio desta abordagem é a necessidade de alto poder computacional, para o processamento de grandes volumes de dados (CHANDOLA; BANERJEE; KUMAR, 2009).

Baseado em Heurística: ocorre quando os HPCs monitoram os processos em execução no sistema, em busca de dados suspeitos determinados pelo administrador, conseguindo desta forma classificar se esses processos são comuns ou se trata de uma situação anormal. Abordagem normalmente utilizada atribuindo limites predeterminados, podendo ser um valor especifico de um determinado contador ou quantidades de faltas, falhas ou erros no sistema.

Essa abordagem mostra que é capaz identificar diversas violações de segurança, analisando as amostras de desempenho capturadas pelos HPCs presente em diversos processadores, fornecendo detecção direta de ataques, ocorrendo quando a quantidade de certos eventos ultrapassa limites predefinidos, individualmente ou em combinação, tendendo a ter um desempenho melhor na detecção de ataques baseado em heurística (YUAN et al., 2011; TORRES; LIU, 2016).

Baseado em Abordagens Avançadas: utilizada normalmente quando os dados adquiridos dos HPCs já estão tratados e é utilizado em conjunto com diversas análises avançadas. Por exemplo, as análises estatísticas avançadas, mineração de dados ou outras metodologias baseadas em heurética. Existem muitas abordagens que utilizam diversas dessas abordagens avançadas e que comumente necessitam de maiores recursos para processamento, como as abordagens utilizando AI e ML com o aprendizado supervisionado ou não supervisionado (SINGH; RAM, 2021).

Abordagens Híbridas: é a técnica de combinação de uma ou mais abordagens diferentes, juntamente com outros meios de contramedidas de segurança da informação. Normalmente apropriando de técnicas de AI e ML pela vantagem dessas técnicas se adaptarem melhor às situações adversas. Além disso, serve de suporte para outras técnicas de linha de frente são utilizadas como ferramenta de auxilio a contramedidas de proteção já existentes.

Assim como as diferentes abordagens de utilização dos HPCs, se tratando de contramedidas de proteção principalmente para detecção de ataques DDoS, as abordagens de utilização são classificadas em cinco métodos primários (BHUYAN et al., 2014).

Baseado em Estatística: abordagem divididas em três etapas básicas utilizando apenas conceitos estatísticos e matemáticos. A primeira é a coleta dos dados que se deseja analisar; a segunda é o tratamento desses dados, aplicando métodos e técnicas estatísticas tradicionais nos dados coletados; e, por fim, a terceira fase é a tomada de decisão e classificação referente aos dados coletados diante das situações coletadas.

Os métodos estatísticos são classificados em dois tipos, que dependem da forma da sua utilização, podendo ser a detecção baseada num valor de limite fixo de alguns parâmetros e quando esse limite é ultrapassado é considerado que houve anomalia no sistema. Ainda, há a detecção baseada em perfil comportamental, onde o foco são as características de comportamento capturadas anteriormente e sendo comparada com as atuais, onde os desvios mais significativos nos perfis são interpretados como anomalias no sistema (MIRCHEV; MIRTCHEV, 2020).

Abordagem de utilização geral na detecção de anomalias, utilizando da construção de modelos probabilísticos e no uso de métodos matemáticos estatísticos na aplicação das teorias das probabilidades. Normalmente é utilizado junto à criação de perfis de execução enquanto o sistema está em seu funcionamento comum. Assim, é medido constantemente os desvios dos perfis coletados e comparados ao comportamento atual do sistema no intuito de detectar comportamento anômalo no sistema (GOGOI et al., 2011).

Baseado em Conhecimento: também conhecido como método baseado em regras. Os métodos que utilizam dessa abordagem, detectam anomalias no sistema pelo conhecimento prévio do padrão de utilização de recursos do sistema. Padrões já estabelecidos anteriormente servem para classificar o estado atual do sistema como atividade suspeita ou utilização comum.

Os métodos baseados em conhecimento são classificados em dois tipos básicos, sendo, o primeiro, a detecção de anomalias por regras estabelecidas. Este método envolve os registros históricos de auditorias realizadas e assim gerando novas regras automatizadas para identificar novos padrões. E, o segundo, a identificação de invasão baseada em regras, que envolve a identificação e classificação das assinaturas de anomalias conhecidas através do uso de dados de anomalias já identificadas.

Baseado em Computação Suave: são métodos que envolvem um baixo custo e recursos computacionais, existindo duas técnicas básicas de utilização nesse método: as Redes Neurais Artificiais (ANNs), usadas normalmente para desenvolver novos sistemas não-lineares que aceitam uma quantidade muito maior de variáveis de entrada e saída junto com seus relacionamentos; e os métodos de SVM, usados para classificar os dados com base nos relacionamentos independentes e de destino das variáveis.

O método desenvolvido por Chen, Cheng e Hsieh (2010) utilizando de RST (Rough Set Theory) e SVM para detectar invasões no sistema, onde o RST vem como uma solu-

ção de pré-processamento para reduzir as dimensões e densidade espacial nos dados e em seguida são enviados ao modelo de SVM para aprendizagem e teste.

Baseado em Mineração de Dados: são normalmente utilizados em ambientes e sistemas mais complexos, pela necessidade maior de processamento e da utilização de maiores recursos. Normalmente são obtidas taxas inferiores na detecção de anomalias em comparação com métodos baseados em assinaturas, onde detectam padrões em grandes quantidades de dados e usam esses padrões para detectar anomalias futuras em dados semelhantes.

Métodos que usam desse método de contramedida detectam principalmente novas ameaças ou eventos anômalos em conjuntos de dados de maior densidade. Um desafio para técnicas que utilizam dessa abordagem é lidar com grandes quantidades de dados, necessitando extremamente de poder computacional, devido às anomalias serem causadas por um novo evento ou tópico anômalo, que também disponibilizam de um aspecto temporal por serem coletados ao longo do tempo, .

Baseado em Aprendizado de Máquina: utiliza-se de métodos estatísticos específicos para a análise e classificação dos dados. Esse método utiliza uma combinação ampla de métodos e técnicas de AI e modelos matemáticos, como as Redes Neurais Artificiais (ANNs) e RF.

O existem dois métodos de ML, aprendizado supervisionado e o não-supervisionado. Normalmente o aprendizado supervisionado é de modo *offline*, onde o mecanismo de classificação irá detectar e classificar os comportamentos situacionais no sistema e sua classificação é atribuída de modo manual antes da sua implementação. O aprendizado não-supervisionado, comumente chamado de treinamento *online*, justamente por estar já em execução, enquanto o treinamento permanece atualizando a base de informação, acumulada e analisada (BAŞKAYA; SAMET, 2020; BAHADOR; ABADI; TAJODDIN, 2014).

2.3 CONSIDERAÇÕES FINAIS

É possível fortalecer as contramedidas e detecção de ataques de DoS/DDoS utilizando estratégias tradicionais, como as baseadas em estatísticas da rede e detecção de assinaturas, em suporte com novas técnicas e abordagem, para que seja possível se ter contramedidas mais maduras (AWAN et al., 2021; VEDULA et al., 2021). Propostas baseada no próprio hospedeiro, combinando estatísticas da rede e da aplicação, juntamente com técnicas de ML é possível criar mecanismos de proteção com limites de alertas dinâmicos, o que dificulta alguns ataques que exploram esses limites (BASKAR et al., 2021; JYOTHI et al., 2016). Ao se utilizar HPCs para a detecção de anomalias, juntamente com técnicas de estatística e ML para análises em tempo real, é possível prevenir ataques de DoS/DDoS mesmo que eles sejam ataques de *zero-day* (RIOS et al., 2021; GARCIA-SERRANO, 2015).

Abordagens baseadas em ML provam ser uma opção com melhores resultados e desempenho entre as diferentes abordagens de prevenção e detecção por conseguirem se adaptar melhor às situações desconhecidas (AAMIR et al., 2021; TUAN et al., 2020; IDHAMMAD; AFDEL; BELOUCH, 2018). A utilização dessas tecnologias juntamente com o suporte dos HPCs é capaz de proporcionar contramedidas de segurança bastante eficientes. Contadores de desempenho que além de conterem grande parte das soluções desenvolvidas atualmente na área, apresentam resultados promissores na cibersegurança (SINHA et al., 2021; ARFEEN et al., 2021; FILHO et al., 2019). Quando se deseja uma melhor adaptação do ambiente e do sistema, essas são as melhores abordagens geralmente utilizadas como uma segunda camada de proteção ou um pré-filtro e obtendo uma adaptação melhor em situações adversas (LIBRI; BARTOLINI; BENINI, 2021; GE et al., 2018; BAHADOR; ABADI; TAJODDIN, 2014). Três classificadores em particular apresentam excelentes resultados na identificação de anomalias no sistema e proporcionando alta precisão na classificação, sendo eles: K-vizinhos mais próximos (kNN), RF e DT (MASLAN; MOHAMAD; FOOZY, 2020; ROHAN; BASU; KARRI, 2019).

3 METODOLOGIA PROPOSTA: UMA VISÃO GERAL

Os mecanismos de defesa, como mencionado anteriormente na Secção 2, são classificados em três categorias básicas, que são denominadas de acordo onde são implementadas no ambiente. São elas: *Defesa de Origem e Extremidade*, *Defesa Intermediária da Rede* e *Defesa Baseada no Hospedeiro*. A utilização destes métodos podem auxiliar no desenvolvimento de contramedidas de proteção no ambiente, dependendo do tipo de serviço oferecido e dos recursos e equipamento utilizados, e utilizando a melhor abordagem especificamente para o ambiente. Uma vez que o ambiente terá o mínimo necessário para a disponibilização do serviço, com cada recurso e sua correta implementação da segurança tendo como base os pilares da SI.

O presente trabalho propõe uma metodologia de diagnóstico não intrusivo, sem a necessidade de *hardware* adicional para realizar a coleta e as análises dos dados, tendo vantagens sobre as soluções propostas que utilizam *hardware* dedicado para tal finalidade. Além disso, resolvendo problemas principalmente relacionados à escalabilidade dos recursos de *hardware*, como os que ocorrem em casos de maior fluxo de requisições e altas taxas enviadas para o sistema e as contramedidas de proteção não dispõem do aumento flexível de recursos para se adaptar melhor à situação (PAN et al., 2021; WOO, 2021; WANG; KARRI, 2015). A metodologia de diagnóstico faz parte da primeira etapa de uma solução completa para detectar e mitigar ataques DDoS em servidores web de nível corporativo, como ilustrado na Figura 6. Assim, dando continuidade e resolvendo problemas onde as soluções utilizam diagnóstico de terceiros, não são realizados em computadores de nível empresarial ou são executados em *hardware* dedicado. A primeira etapa da solução proposta é a execução do diagnóstico do servidor, capturando seus perfis comportamentais e passando os dados coletados para um classificador, para que seja possível detectar e executar posteriormente as abordagens de mitigação atribuídas. Abordagens de mitigação que tem como intuído principal manter a qualidade do serviço prestado, a disponibilidade e proteger reativamente o sistema de eventuais ameaças.

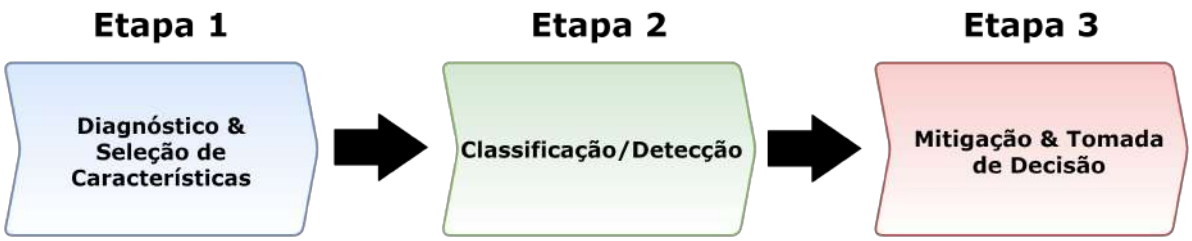


Figura 6 – Etapas da metodologia

O diagnóstico proposto é realizado dentro do próprio hospedeiro, utilizando ferramentas de monitoramento do sistema em conjunto com técnicas estatísticas e de ML para a

realização das análises dos dados. Os dados dos HPCs coletados, têm em sua característica o baixo consumo de recursos para realização do diagnóstico, já que utiliza recursos de monitoramento já existentes no processador. Isso torna esses contadores excelentes recursos para execução de um monitoramento em tempo real (BAZM et al., 2018; CHIAPPETTA; SAVAS; YILMAZ, 2016). Com essa metodologia de diagnóstico, é possível criar conjuntos de dados com maior fidelidade em suas características, visto que os dados são específicos do hospedeiro e do ambiente, dispensando a necessidade de utilização de conjuntos de dados de terceiros. A metodologia cria perfis que poderão ser utilizados na implementação e desenvolvimento de contramedidas de segurança.

Além da execução do diagnóstico do sistema, a metodologia proposta investiga e analisa o comportamento de ataques de DDoS no serviço web, definindo o comportamento de ferramentas de ataque DoS/DDoS de inundação HTTP. Os perfis criados pelo comportamento de cada ferramenta de ataque, poderá ser utilizado para desenvolvimento de contramedidas de segurança. Assim, o entendimento das peculiaridades de cada ferramenta de ataque no serviço mostrará características que normalmente ficam no nível oculto para os administradores de sistema. Os dados coletados das variáveis do sistema servem como suporte para definir ainda mais os perfis capturados nas situações de ataques. Sendo estas variáveis anteriormente utilizadas isoladamente como métricas de proteção e detecção, hoje servem como fonte de auxílio para métodos que utilizam variáveis mais fiéis e de baixo nível de abstração, como no caso dos contadores de desempenho de hardware.

A precisão na diferenciação das situações de ataque e utilização comum do serviço deverá ter bons resultados por conta do esforço computacional necessário que análises recorrentes poderiam causar, ou que causaria um prejuízo considerável em caso de alertas falsos ou da exploração das vulnerabilidades do sistema de classificação. Portanto, é de extrema importância a coleta adequada dos dados situacional do sistema, para que as contramedidas de proteção sejam desenvolvidas para atender a necessidade de proteção do serviço em situações de ataques.

3.1 FASES DA METODOLOGIA

A metodologia proposta é formada por quatro fases, utilizando os contadores de eventos de *hardware* de baixo nível, realizando a modelagem dos perfis comportamentais nas situações de ataque e utilização comum do hospedeiro. Cada fase do processo executa parte fundamental para realização da metodologia de diagnóstico. Extraindo os contadores de desempenho que mais influenciam nas situações comportamentais de ataques DDoS e utilização comum do serviço. A Figura 7 detalha visualmente a metodologia apresentada e suas quatro fases e seus respectivos processos, sendo elas *Coletando Variáveis de HPCs*, *Tratamento de Dados*, *Análise Preliminar* ou *Pré-filtro* e *Extração das Variáveis mais Relevantes*.

Em cada uma das fases acima mencionadas, será apresentada suas especificações seguindo a respectiva ordem ilustrada na Figura 7. A metodologia é dividida em quatro fases para sua realização, onde: na **Fase Um** apresentará como os dados dos HPCs são coletados; na **Fase Dois**, é realizado o tratamento dos dados e mostrará em detalhes como os dados coletados são tratados e os utilitários utilizados para realizar seu processamento; na **Fase Três** apresentará as técnicas de análise utilizadas para selecionar os contadores de desempenho, estabelecendo um pré-filtro para a fase subsequente de análise e extração; na **Fase Quatro** abordará a extração das variáveis mais relevantes dentro do conjunto de amostra inicial dos HPCs, capturando o comportamento do hospedeiro em situações de ataque e a situação típica do sistema.

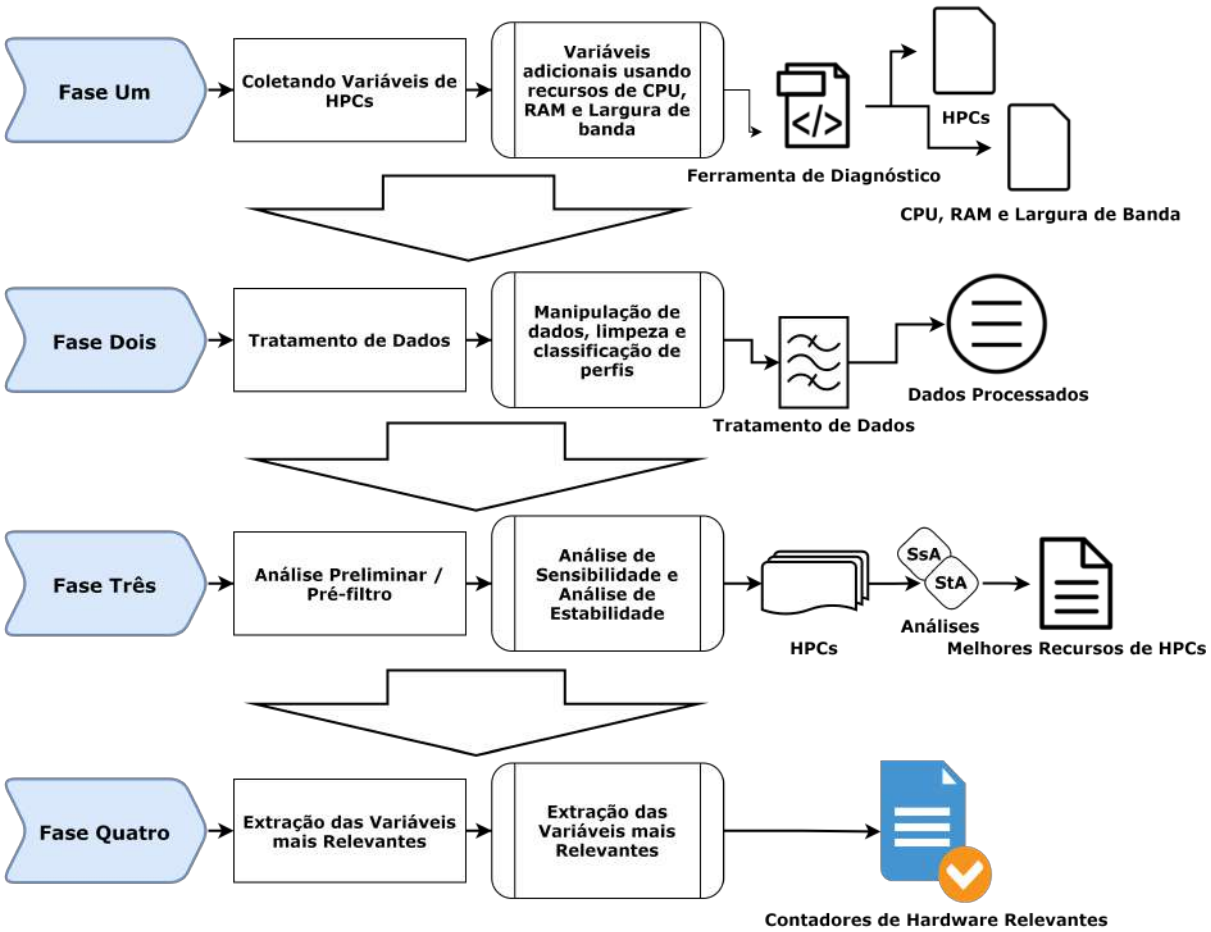


Figura 7 – Definição da Metodologia

A metodologia proposta é baseada em trabalhos anteriores de Rohan, Basu e Karri (2019), Demoulin et al. (2019), Addepalli, Karri e Jyothi (2017), Leng, Zwolinski e Halak (2017), Elnaggar, Chakrabarty e Tahoori (2017), Torres e Liu (2016), Yuan et al. (2011), Oikonomou e Mirkovic (2009) que se mostram eficientes na seleção das características essenciais do sistema, usando principalmente os HPCs. As principais diferenças estão na

aplicação da metodologia em equipamentos de segmento empresarial e sem *hardware* adicional para execução da metodologia; o que gera diagnósticos de perfis comportamentais, dispensando a necessidade de utilização de dados coletados por terceiros. A metodologia apresenta uma proposta de diagnóstico selecionando características e gerando conjuntos de dados com perfis comportamentais para aplicações em serviços *web*. Esses conjuntos de dados são ótimos candidatos para desenvolver soluções para detectar anomalias e ataques DoS/DDoS no sistema. Em suma, o trabalho apresenta uma abordagem diferente dos trabalho presentes na literatura, pois existem contadores de HPCs únicos presentes em equipamentos de nível empresarial. As análises de seleção dos contadores de desempenho foram executadas de forma diferente das executadas nas soluções dos trabalhos relacionados. Dessa forma, acreditamos que conseguimos selecionar melhor os contadores mais influentes. Além de obter uma quantidade diferente de contadores, utilizamos ferramentas de ataques reais, simulando uma estrutura de ambiente real de ataque no servidor.

Toda a estrutura física e lógica criada para o desenvolvimento da metodologia, tem como um dos principais objetivos, a simulação e replicação de um servidor web de segmento empresarial sofrendo ataques de DDoS reais de inundação HTTP, com ferramentas bastante utilizadas para tal finalidade na internet. O ambiente e a metodologia foram desenvolvidos fielmente o mais próximo da realidade de um ambiente de produção, criando assim diagnósticos reais com diferenciação das situações de ataques e situação comum do sistema, juntamente com suas características individuais de cada ferramenta utilizada. A metodologia explora a junção de ferramentas e utilitários comumente utilizada por administradores de rede, criando uma aplicação de diagnostico, que poderá ser utilizada e replicada facilmente em outros ambientes, realizando os demais procedimentos e análises aqui apresentados.

3.1.1 Coletando os Dados

A fase Um da metodologia é a realização de coleta dos dados por meio de um *script* desenvolvido especialmente para essa finalidade e utilizando utilitários do sistema *linux*, como o *perf*, *free*, *ifstat*, *sar* e suas especificações individualmente, ilustrado na Figura 8. Nesta fase, os dados dos HPCs são coletados juntamente com as variáveis de utilização de recursos do sistema, como utilização de CPU e RAM e consumo de largura de banda de rede.

A fase Um é a realização de coleta dos dados por meio de um *script* desenvolvido especialmente para essa finalidade, onde inicialmente só os dados dos HPCs serão analisados na metodologia apresentada e os dados referentes aos recursos serão analisados em outro momento. Os dados coletados estão diretamente relacionados à execução das instruções do processador e são referentes aos eventos de *cache de hardware* e *hardware*. O método de captura dos HPCs foi realizado entre determinados intervalos de dez segundos, por conterem uma quantidade significativa e suficiente de pontos amostrais para realizar a

análise. Cada coleta será realizada separadamente para cada ferramenta (ferramentas de ataque DoS/DDoS e carga de trabalho) selecionada para capturar o comportamento do hospedeiro e criar os perfis comportamentais de cada situação. As condições de ataque e utilização comum do serviço capturadas geram conjuntos de dados para cada ferramenta de ataque utilizada na realização do diagnóstico.

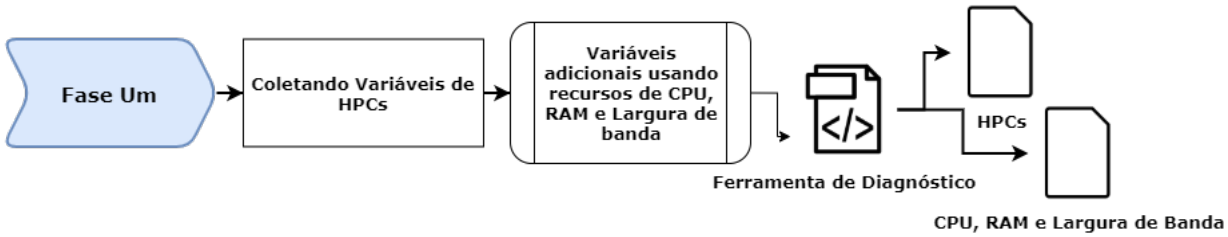


Figura 8 – Fase Um da Metodologia

- **perf stat**

Este utilitário executa um comando que coleta estatísticas do contador de desempenho de referência e a Figura 9 demostra um exemplo de saída da ferramenta utilizada e os dados dos HPCs atribuídos, informando a quantidade de eventos que o contador capturou e o tempo decorrido da coleta. Com o utilitário, é possível obter informações adicionais da execução, como a quantidade de instruções por ciclo de CPU, dividindo o valor de instruções pelo ciclo de CPU. Assim, cada evento do contador de referencia, é somado uma atribuição. Para mais detalhes, ver o Apêndice C.

Sinopse:

```
perf stat [-e <EVENT>|--event=EVENT] [-a] <command>
perf stat [-e <EVENT>|--event=EVENT] [-a]   --<command> [<options>]
perf stat [-e <EVENT>|--event=EVENT] [-a] record [-o file]   --<command> [<options>]
perf stat report [-i file]
```

Exemplo:

```
# perf stat -e cycles,instructions,r80a2,r2b1 gzip file1

 Performance counter stats for 'gzip file1':

     5,586,963,328 cycles                    #    0.000 GHz
     8,608,237,932 instructions              #    1.54  insns per cycle
         9,448,159 raw 0x80a2
    11,855,777,803 raw 0x2b1

       1.588618969 seconds time elapsed
```

Figura 9 – Exemplo de saída do comando *perf*

- **free**

Este utilitário exibe a quantidade total de memória física, de *swap* e compartilhada no sistema, como também exibe o total de memória livre, bem como os *buffers*/*cache* utilizados pelo *kernel*, como mostrado na Figura 10. Para mais detalhes, ver o Apêndice C.

Sinopse:

```
free [-b | -k | -m] [-o] [-s delay ] [-t] [-l] [-V]
```

Exemplo:

```
# free
              total        used        free      shared  buff/cache   available
Mem:       16641840     8643972     7768516       17720      229352     7864136
Swap:      50331648       50588    50281060
```

Figura 10 – Exemplo de saída do comando *free*

- **ifstat**

Pequena ferramenta para relatar atividades da interface de rede, assim como as ferramentas *iostat*/*vmstat* utilizadas para observar outras estatísticas do sistema. A Figura 11 demonstra como a saída da ferramenta é exibida. Para mais detalhes, ver o Apêndice C.

Sinopse:

```
1 ifstat [-a] [-l] [-z] [-n] [-v] [-h] [-t] [-i if0,if1,...] [-d drv[:opt]] [-s [comm@
    ][#]host[/nn]] [-T] [-A] [-w] [-W] [-S] [-b] [-q] [delay[/delay] [count]]
```

Exemplo:

```
/# ifstat -t
  Time           eth0                wifi2
HH:MM:SS    KB/s in  KB/s out   KB/s in  KB/s out
15:20:42       0.00      0.00      0.00       0.00
15:20:43       0.00      0.00      0.00       0.00
15:20:44       0.00      0.00      0.00       0.00
15:20:45       0.00      0.00      0.00       0.00
15:20:46       0.00      0.00      0.00       0.00
15:20:47       0.00      0.00      0.00       0.00
15:20:48       0.00      0.00      0.00       0.00
15:20:49       0.00      0.00      0.00       0.00
```

Figura 11 – Exemplo de saída do comando *ifstat*

- **sar**

O utilitário *sar* grava, na saída padrão, as variáveis de atividade cumulativa no sistema operacional, como a utilização de CPU, RAM e solicitações de leitura/escrita com base nos valores dos parâmetros de contagem e intervalo predeterminados como mostrado no exemplo da Figura 12. Portanto, que tem por padrão todos os dados disponíveis no *kernel* capturados. Para mais detalhes, ver o Apêndice C.

Sinopse:

```
sar [ -A ] [ -b ] [ -B ] [ -C ] [ -d ] [ -h ] [ -i interval ] [ -m ] [ -p ] [ -q ] [ -r ] [ -R ] [ -S ] [ -t ] [ -u [ ALL ] ] [ -v ] [ -V ] [ -w ] [ -W ] [ -y ] [ -n { keyword [,...] | ALL } ] [ -I { int [,...] | SUM | ALL | XALL } ] [ -P { cpu [,...] | ALL } ] [ -o [ filename ] | -f [ filename ] ] [ -s [ hh:mm:ss ] ] [ -e [ hh:mm:ss ] ] [ interval [ count ] ]
```

Exemplo:

```
# sar 3 10
Linux  _x86_64_        (12 CPU)

15:28:14        CPU     %user     %nice   %system   %iowait    %steal     %idle
15:28:17        all      0.50      0.00      0.83      0.00      0.00     98.67
15:28:20        all      1.00      0.00      0.50      0.00      0.00     98.50
15:28:23        all      0.75      0.00      0.67      0.00      0.00     98.59
15:28:26        all      0.64      0.00      1.22      0.00      0.00     98.15
15:28:29        all      0.72      0.00      0.28      0.00      0.00     99.00
15:28:32        all      1.11      0.00      1.14      0.00      0.00     97.76
15:28:35        all      0.61      0.00      0.64      0.00      0.00     98.75
15:28:38        all      1.05      0.00      1.03      0.00      0.00     97.92
15:28:41        all      0.75      0.00      0.39      0.00      0.00     98.87
15:28:44        all      0.64      0.00      0.36      0.00      0.00     99.00
Average:        all      0.78      0.00      0.70      0.00      0.00     98.52
```

Figura 12 – Exemplo de saída do comando *sar*

O tempo de execução do diagnostico comportamental em cada coleta, foi de duas horas para cada ferramenta de ataque, no intervalo de dez segundos para cada captura. O tempo de coleta no intervalo determinado deve ser suficiente para coletar uma quantidade significativa de dados e pontos amostrais para a realização das análises subsequentes. A decisão do tempo de execução foi com base em experiências e testes anteriores de coleta, portanto decidimos utilizar o valor de duas duas horas para realizar o diagnóstico. Outras variáveis do sistema na execução do diagnóstico, como CPU, RAM e consumo de largura de banda, serão utilizadas posteriormente dentro das especificações de duração e intervalo para realização da análise de capacidade do servidor e auxiliar no desenvolvimento de contramedidas de mitigação do hospedeiro.

Inicialmente são coletados 120 minutos, o equivalente a 7.200 segundos, com intervalos de 10 segundos entre cada captura. Destes 120 minutos, os primeiros 20 minutos de cada conjunto de dados (equivalente a 1.200 segundos e 120 amostras) são removidos, excluindo o ruído subsequente de cada coleta e considerando o tempo necessário para executar as

ferramentas, como ilustrado na Figura 13. A metodologia irá trabalhar com base nos 6.000 segundos restantes, com 600 amostras cada coleta, ficando assim 100 minutos de dados coletados do diagnóstico situacional no sistema.

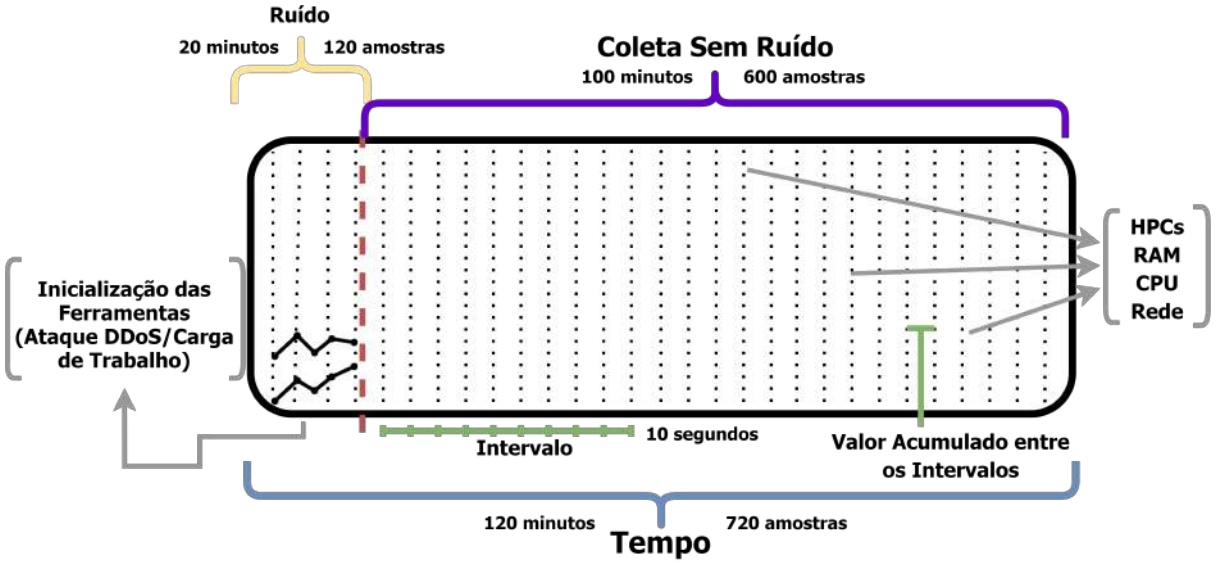


Figura 13 – Janelas de tempo da coleta

A coleta de dados leva 6.000 segundos em cada situação capturada, compreendendo o atraso entre a execução de cada instância de carga de trabalho e as ferramentas de ataque. Isto visando reduzir a variação nos dados entre as situações da não utilização do sistema com a situação que se deseja capturar, que neste caso são as situações de ataque. A remoção dos minutos iniciais visa reduzir a variação dos dados por não uso do sistema em relação à iminência do ataque para capturar a situação de referência de forma adequada. Assim, garantindo que as situações sejam capturadas sem interferência. Com cada situação ocorrendo no sistema, cria-se uma representação comportamental do sistema, desenvolvendo perfis mais definidos. Vale ressaltar que, após cada coleta, a reinicialização do sistema, ou alguma alternativa de limpeza, é realizada preferencialmente limpando os resquícios da carga gerada e dos processos em execução no sistema e, assim, removendo os ruídos subsequentes dentro do sistema. Esse procedimento evita que uma situação não interfira na outra, uma vez que serão utilizadas ferramentas de ataque DoS/DDoS de inundação HTTP, que têm como finalidade sobrecarregar o sistema.

O *script* de monitoramento coletou os dados de diagnóstico dos contadores de desempenho utilizando o utilitário *perf stat*, uma ferramenta de análise de desempenho para ambientes *linux* que já está disponível no *kernel*. Devido à limitação do utilitário, normalmente não é possível coletar todos os HPCs simultaneamente; geralmente só é possível gerenciar simultaneamente um número limitado e fixo de medidores de desempenho. Ciente desta limitação do utilitário, o *script* de coleta não invasivo pode coletar simultaneamente

todos os contadores selecionados sem alterar o código-fonte do sistema hospedeiro. Os dados são coletados no diagnóstico em arquivos separados contendo informações para cada componente presente. Os dados coletados dos HPCs, utilização de CPU, RAM e consumo de largura de banda, com seus respectivos nomes, podem ser facilmente interpretados e manipulados contendo apenas os dados necessários para as análises.

A Figura 14 mostra a sinopse de utilização e as funções de atribuições utilizadas para capturar os dados dos contados de desempenho de *hardware* e redirecioná-los para seus respectivos arquivos utilizando as ferramentas de manipulação de dados do sistema:

```
perf stat -o contador1.txt -e branch-instructions,branch-misses,bus-cycles,cache
-misses,cache-references,cpu-cycles,instructions,ref-cycles -I $x -a -g --
 sleep $y

perf stat -o contador2.txt -e L1-dcache-load-misses,L1-dcache-loads,L1-dcache-st
ores,L1-icache-load-misses,LLC-load-misses,LLC-loads -I $x -a -g -- sleep $y

perf stat -o contador3.txt -e LLC-store-misses,LLC-stores,branch-load-misses,bra
nch-loads,dTLB-load-misses,dTLB-loads -I $x -a -g -- sleep $y

perf stat -o contador4.txt -e
 dTLB-store-misses,dTLB-stores,iTLB-load-misses,iTLB-loads,node-load-misses -I $
 x -a -g -- sleep $y
```

Figura 14 – Coleta dos HPCs

Onde *-o* grava a saída do diagnóstico para um arquivo atribuído, *-e* indica os contadores de desempenho que vão ser capturados, *-I* indica o intervalo entre uma captura e outra atribuindo um valor em segundos a variável de ambiente *$x* em segundos, *-a* coleta todas as CPUs disponíveis no sistema, *-g* habilita recursos adicionais dos contadores ausentes nas pilhas do nível do usuário e por fim a opção *—sleep* que atribui um valor a variável de ambiente *$y*, em segundos, que irá representar o tempo que o *script* esperará em execução antes de ser finalizado.

Os dados relativos ao consumo de CPU, RAM e utilização da rede estão presentes nos dados de diagnóstico, contendo informações detalhadas sobre o desempenho do hospedeiro em cada intervalo que são necessárias para o planejamento da capacidade dos ativos e desenvolvimento de contramedidas. Em casos de alta utilização dos recursos mencionados acima, detalhes adicionais fornecidos pelos HPCs serão necessários para evitar condições de falsos alarmes na classificação comportamental do hospedeiro. Além disso, a aplicação de técnicas como migração de serviço ou obtenção de mais recursos pode diminuir esses falsos alertas. O aumento repentino da utilização da rede pode ser tratado com balanceamento de carga ou escalonamento de serviço, considerando os requisitos de disponibilidade. Os dados dos HPCs caracterizam fielmente o comportamento do hospedeiro, pois são estatísticas de baixo nível de abstração e as variáveis de monitoramento

da utilização de recursos descrevem sua capacidade, sendo variáveis de níveis superiores de abstração. As variáveis de recursos serviram como características de auxílio para diferenciação e detecção de situações anormais.

Toda a coleta ocorre no próprio hospedeiro sem a necessidade de adicionar *hardware* para essa finalidade. Os contadores de desempenho têm um nível satisfatório de integridade devido à sua persistência física na microarquitetura porque já residem no chip do processador implementado em hardware dedicado e são executados no nível de privilégio do *kernel*. Essa característica geralmente proporciona uma utilização mínima dos recursos do processador, fazendo com que a carga gerada pela captura personalizada tenha um baixo consumo de recursos do hospedeiro além de um nível de segurança alto. Os HPCs coletam várias informações sobre o comportamento atual do hospedeiro, onde estão o *branching instructions*, *bus cycles*, *cpu cycles* e *L1-dcache loads*, todas as informações sobre os processos em execução no sistema.

O número e o tipo de eventos de contadores dependem principalmente da CPU utilizada. Como muitos contadores estão disponíveis, vários dispositivos específicos dependerão do fabricante do processador e de sua categoria e, portanto, o OS no qual o processador está sendo executado. Assim, a metodologia aqui apresentada utiliza um conjunto comumente compartilhado pela maioria das CPU e amplamente utilizada em trabalhos relacionados para facilitar o diagnóstico e a replicação da metodologia em outros ambientes.

Importante frisar a importância de todas as ferramentas e utilitários estarem sincronizados, sendo executados no mesmo instante para evitar diagnósticos enviesados e consequentemente análises enganosas. Está primeira fase é de certa forma a mais importante de todo o processo da metodologia, fase em que os dados que vão ser analisados em todo o decorrer do processo. Portanto, está fase necessita de uma atenção especial na execução de todas as ferramentas e utilitários presentes em sua execução.

A execução do diagnostico preferencialmente deverá ser executada de forma remota, utilizando conexões como *Secure Socket Shell* (SSH) ou alguma ferramenta utilizando *Virtual Private Network* (VPN) tanto para facilitar o gerenciamento do ambiente como para evitar possíveis interferência que uma utilização por ambiente virtual pode acarretar, além da memória necessária para a execução de todos os procedimentos.

3.1.2 Processando os Dados

A fase Dois é a realização do tratamento dos dados de diagnóstico que foram coletados na fase anterior na subseção Coletando os Dados 3.1.1, como visto na Figura 15. Juntamente com ferramentas de manipulação de texto para o tratamento e redirecionamento dos dados para que seja possível a realização das análises subsequentes.

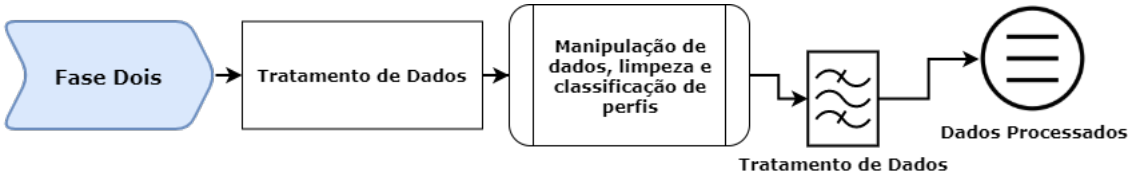


Figura 15 – Fase Dois da Metodologia

Os dados de diagnóstico já foram coletados de forma adequada por meio de um *script* de monitoramento, capturando apenas os dados necessários para as análises e removendo os dados desnecessários. Assim, nenhum esforço significativo é necessário para realizar o processamento dos dados. As informações são salvas em um único arquivo, contendo todos os dados dos contadores de desempenho e as demais variáveis referentes aos recursos do hospedeiro, seguindo os padrões de intervalo e tempo. O processamento de dados é possível com o uso e combinação de várias ferramentas e utilitários de manipulação de dados, que estão disponíveis em várias arquiteturas baseadas em Linux, como o *sed*, *awk*, *cut* e *tr*. As ferramentas são executadas em conjunto com ferramentas de manipulação de dados e utilitários de redirecionamento para uma melhor organização dos dados em um único arquivo, o que facilita os processos de análise.

- **sed**

O *sed* é um editor de fluxo de dados para filtrar e transformar textos, tendo a capacidade de usabilidade que o distingue particularmente de outros tipos de editores, justamente por se fazer possível manipular facilmente um arquivo ou saída e transformar em um arquivo saída já tratado. Para mais detalhes ver o Apêndice C.

Sinopse:

```
1 sed [OPTION]... {script-only-if-no-other-script} [input-file]...
```

- **awk**

O *awk* é uma linguagem de programação que leva em seu nome a abreviatura dos seus criadores, AWK *(Alfred* ***Aho****, Peter J.* ***Weinberger*** *e Brian* ***Kernighan****)*. Ferramentas que utilizam essa linguagem são bastante poderosas e muito utilizada para trabalhos de manipulação com arquivos de texto, geralmente utilizada para extrair algum conteúdo específico do texto e reorganizá-los da forma que se deseja, declarando variáveis e selecionando partes específicas do texto. Por padrão, o *awk* lê a partir de um arquivo e mostra o resultado na saída padrão. Para mais detalhes, ver o Apêndice C.

Sinopse:

```
1 awk [-F sepstring] [-v assignment]... program [argument...]
  awk [-F sepstring] -f progfile [-f progfile]... [-v assignment]...[argument...]
```

- **cut**

O comando *cut* é um utilitário de uso simples e prático, normalmente utilizado para selecionar linhas de um arquivo ou direcioná-lo para a saída padrão utilizando outros comandos. Ótima ferramenta para trabalhar com delimitadores entre os dados do texto, facilitando a sua manipulação. Para mais detalhes, ver o Apêndice C.

Sinopse:

```
cut OPTION... [FILE]...
```

- **tr**

Utilitário básico usado para traduzir ou excluir caracteres específicos da entrada padrão ou redirecionando e gravando para a saída padrão. O comando recebe dados da entrada padrão, localizando os caracteres especificados pelo usuário e os substituindo pelos caracteres atribuídos. Para mais detalhes, ver o Apêndice C.

Sinopse:

```
1 tr [-c|-C] [-s] string1 string2
  tr -s [-c|-C] string1
3 tr -d [-c|-C] string1
  tr -ds [-c|-C] string1 string2
```

Ressaltamos que não existe uma maneira absoluta de se manipular e obter os melhores resultados, o *script* desenvolvido e os utilitários utilizados podem ficar obsoletos e defasados, necessitando de uma atualização e adaptação. Por isso, até mesmo formas melhores podem ser desenvolvidas.

A classificação inicial dos dados de diagnóstico é necessária para comparar os dados de ataque com a situação comum do sistema. As observações feitas analisaram as variações dos HPCs entre os casos de ataque e o uso comum do serviço web. Isso é necessário para classificar os dados de ataque juntando cada arquivo de ataque aos dados da situação sazonal. Assim, a classificação inicial nos dados pode ser realizada no momento da coleta e dentro do próprio *script* de monitoramento ou pode ser posteriormente adicionada de modo manual, acrescentando uma identificação binária aos dados de diagnóstico. A identificação é composta por *0* para dados sazonais e *1* para situações de ataque. Optamos por unir os dados da situação ataque com os dados da situação comum para facilitar a organização dos arquivos. Assim, sendo um para cada ferramenta de ataque, que em fases subsequentes vão ser analisados em suas variações.

3.1.3 Analisando os Dados

Na fase Três, denominada de análise preliminar ou pré-filtro como mostrado na Figura 16, é onde os dados do diagnóstico são submetidos a análises adicionais para a remoção de

contadores menos determinantes para a classificação das situações de ataque e utilização comum no sistema web. O processo de análises ocorre por meio da análise de sensibilidade e estabilidade, realizadas logo após o pré-processamento mencionado nas etapas anteriores.

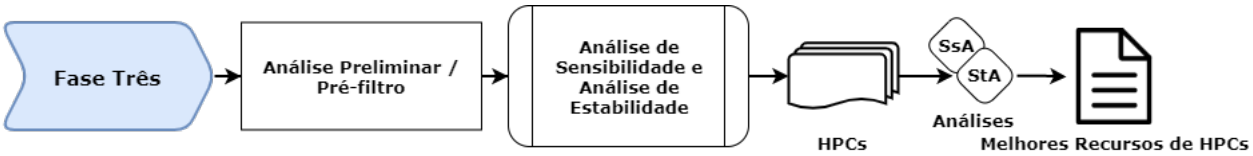


Figura 16 – Fase Três da Metodologia

Análise de Sensibilidade: comparando os casos de Ataque e Situação Comum no sistema, é possível extrair a mudança relativa entre as situações. Isso é necessário para medir a diferença nos valores dos HPCs coletados nas condições observadas no sistema. Contadores com a mudança mais significativa nos valores podem ser ótimos recursos para diferenciar entre ataques e uso sazonal do sistema. A diferença relativa é um indicador de controle utilizado como garantia quantitativa da qualidade dos dados, filtrando contadores com baixíssima sensibilidade, e o restante dos eventos submetidos às demais análises (BENNETT; BRIGGS; BADALAMENTI, 2008). Desta forma, o valor acumulativo dos contadores de desempenho são submetidos a esta análise utilizando o valor de referência da situação que se deseja observar menos a situação comum do sistema.

$$ASe = \frac{SA - SC}{SC}$$

Onde, ASe - Análise de Sensibilidade, SA - Situação de Ataque e SC - Situação Comum.

Análise de Estabilidade: é necessário verificar a consistência nos valores dos contadores de desempenho, usando uma medida de centralidade, os quartis, seus pontos de corte exatos e seus intervalos. A diferença interquartil relativa é utilizada como métrica de observação e é necessária para um melhor entendimento do comportamento dos dados. A execução da redução ou eliminação de alarmes FN e FP entre as situações práticas é possível realizando esta análise e selecionando contadores de *hardware* que demonstram um comportamento mais definido no caso de ataques e situações habituais (ABDULHAMMED et al., 2019; MORETTIN; BUSSAB, 2017). Os contadores selecionados na análise de sensibilidade, são utilizados nesta análise, realizando um filtro para remoção dos contadores com baixo índices.

Ao contrário das métricas estatísticas Desvio Padrão (SD) e Média (Mean), a Mediana (Median) se torna essencial por causa da sua característica de ser resistente a valores extremos nos dados dos HPCs coletados. Sendo assim, uma alternativa bastante interessante é a utilização do Intervalo Interquartil (IQ) sobre a Mediana (Median), com isso conseguindo evitar análises incorretas devido a cargas com valores extremos (HASLWANTER, 2016).

Sendo os Quartis representados por *Q1, Q2* e *Q3*, o *Intervalo Interquartil (II)* é dado por *IIQ = Q3 - Q1*, e a Diferença Relativa Interquartis é dada pela equação abaixo. Onde *Q3* separa os 75% iniciais dos dados dos 25% finais.

$$ASt = \frac{Q3 - Q1}{Q2}$$

Q1, Q2 e Q3 representam os quartis, IIQ = Q3-Q1 dá o intervalo interquartil, e a diferença interquartil relativa é fornecida por [(Q3) - (Q1) / (Q2)].

Utilizaremos preferencialmente na Análise de Sensibilidade 3.1.3, os contadores com os maiores valores em seu coeficiente; e na Análise de Estabilidade 3.1.3, os contadores com os menores valores em seu coeficiente, como mostrado na Figura 17. Após a análise de sensibilidade e estabilidade, é possível filtrar os contadores de desempenho contendo os melhores recursos para a execução das fases subsequentes da metodologia. Portanto, filtrando contadores com sensibilidade muito baixa e coeficientes de estabilidade muito altos após a análise. Os contadores de desempenho restantes são excelentes candidatos para os procedimentos de ML.

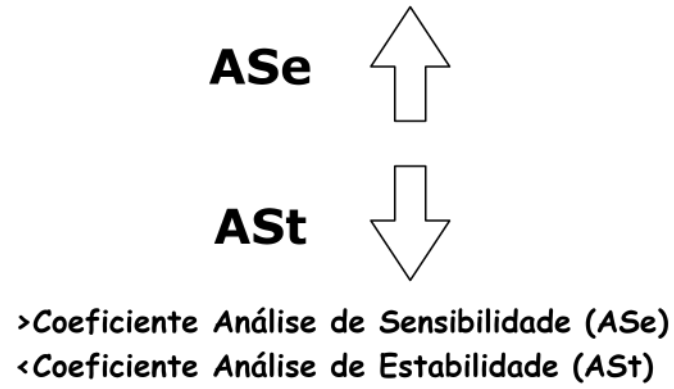


Figura 17 – Regra de utilização dos Coeficientes

Os procedimentos realizados nesta fase têm o objetivo de preparar os conjuntos de dados para as análises de extração e importância, removendo os dados que não influenciam nas demais análises de seleção. Esta fase é denominada de filtro ou pré-filtro, por retirar os contadores que possivelmente impactariam negativamente na análise a seguir. Portanto, a remoção dos contadores de desempenho nesta fase auxilia na criação de um delimitador na consistência dos contadores coletados.

3.1.4 Extração dos Dados

Na execução da fase Quatro, após as variáveis selecionadas nas Análises de Sensibilidade (ASe) 3.1.3 e Análises de Estabilidade (ASt) 3.1.3 como ilustrado na Figura 18, os contadores restantes são submetidos à análise de importância. Aqui todos os procedimentos anteriores se mostram essenciais para a execução da metodologia apresentada.

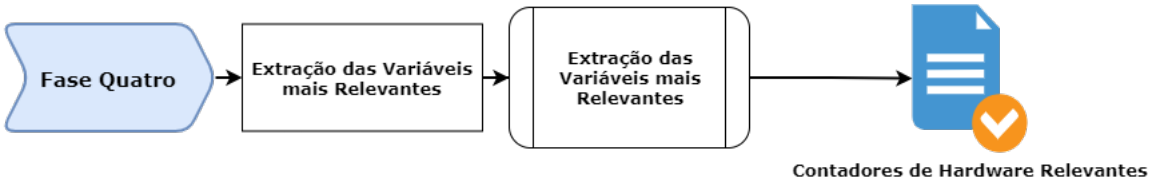


Figura 18 – Fase Quatro da Metodologia

Após a obtenção dos contadores de desempenho, eles são submetidos a análises adicionais para selecionar os contadores mais influentes no sistema usando uma técnica de AI denominada RF. Após a análise de Sensibilidade e Estabilidade, os contadores selecionados são utilizados em estudos subsequentes para extrair os HPCs mais relevante. Como o número de HPCs ainda pode conter inúmeras variáveis e uma abordagem que utiliza vários contadores pode exigir recursos significativos e alto tempo de processamento. O tempo adicional de processamento da análise em tempo real, é um problema quando precisamos de uma classificação/detecção com o menor intervalo possível.

A velocidade na classificação/detecção de contramedidas de proteção, é um fator crucial em sistemas da segurança da informação, principalmente em ataques de DoS/DDoS, uma vez que em poucos minutos o serviço poderá ficar indisponível. Agentes maliciosos se aproveitam das vulnerabilidades do ambiente para praticar atos criminosos. Entre as muitas ameaças, os ataques DDoS são um dos que mais causam danos ao serviço. Com uma resposta rápida na identificação de ameaças no sistema, é possível executar procedimentos de mitigação, dando tempo suficiente para as defesas entrarem em ação, conseguindo efetivamente contornar o tráfego malicioso e garantindo a disponibilidade do serviço (AWAN et al., 2021; PANDE et al., 2021; NASCIMENTO et al., 2021b).

A *Feature Selection* (FS) é uma função da análise de RF que não executa a análise com base em todas as variáveis disponíveis. Ao invés disso, o algoritmo seleciona de maneira aleatória algumas variáveis e então realiza os cálculos com base nas amostras selecionadas.

Assim, o algoritmo define qual dessas variáveis será utilizada no nó raiz. No próximo nó, novamente serão escolhidas duas ou mais variáveis e o processo de escolha se repetirá. Desta forma, a árvore será construída até o último nó como mostrado na Figura 19. Depois que o modelo é gerado, as previsões são feitas a partir de votações, onde cada árvore toma uma decisão a partir dos dados apresentados e a decisão mais votada é a resposta do algoritmo (CHEN et al., 2021; BÉNARD et al., 2021).

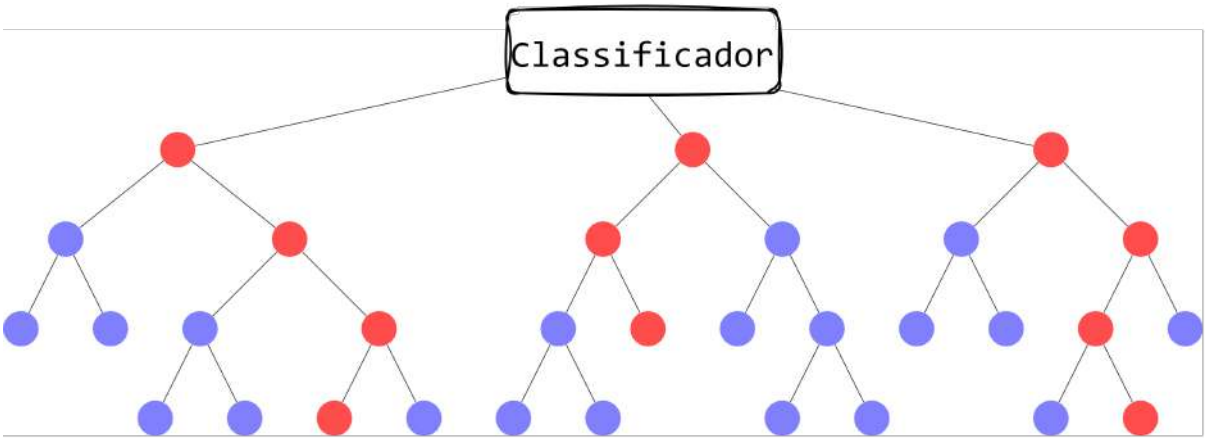


Figura 19 – Arvore de Decisão DT

RF foi utilizado para selecionar as variáveis mais relevantes entre os contadores coletados, determinando se o hospedeiro se comporta de forma anormal durante um ataque ou de forma comum (ROBINSON; THOMAS, 2021). A FS Seleção de Características foi responsável por classificar as variáveis mais relevantes no conjunto de dados após a análise preliminar; que melhora a interpretação e visualização dos dados, permitindo traçar e observar os padrões com mais clareza. Os dados foram divididos em 80% para treinamento e 20% para execução de teste para o RF (BATCHU; SEETHA, 2021). Dividimos os dados em conjuntos de treinamento e teste para evitar estimativas de desempenho tendenciosas de forma otimista. Se usarmos os dados do conjunto de teste para decidir quais recursos são relevantes, teremos vazamento de informações ao usar esses recursos para treinar um classificador. Para obter uma estimativa de desempenho imparcial, os dados de teste não devem ser usados de forma alguma para fazer escolhas sobre o modelo, incluindo FS (AMBROISE; MCLACHLAN, 2002).

Em dados de alta dimensão, é possível extrair padrões e características de cada um dos contadores de desempenho. As técnicas de seleção de recursos reduzem o número de recursos no conjunto de dados inicial sem perder informações essenciais e manter ou mesmo melhorar o desempenho do modelo de classificação. A técnica utilizada na aplicação da seleção de recursos no conjunto de dados e filtrada após análises de sensibilidade e estabilidade, é conhecida como Floresta Aleatória. Sendo este um dos algoritmos mais utilizados para selecionar os recursos mais relevantes entre as variáveis (YOUM; PARK; SHIN,

2021). Assim, escolhe um subconjunto menor de variáveis em um amplo espaço amostral, reduzindo o número de variáveis, diminuindo as dimensões; usando menos recursos, e diminuindo também o tempo e os recursos de computação para treinamento.Além disso, vários algoritmos não funcionam bem quando os dados têm muitas variáveis, usando uma quantidade significativa de recursos. Portanto, é necessário reduzir o número de variáveis para que o algoritmo funcione satisfatoriamente (MASLAN; MOHAMAD; FOOZY, 2020).

Após a obtenção dos contadores mais influentes na ordem de cada conjunto de dados, a união e a interseção são feitas utilizando os contadores de todas as ferramentas. Assim, criando listas exclusivas entre todas as ferramentas de ataque DoS/DDoS executadas. A lista de união corresponde à soma dos contadores de desempenho nos conjuntos de ferramentas de ataque, formados pelos elementos de todos os grupos e componentes. Se os contadores forem repetidos no conjunto de ferramentas de ataque, os contadores aparecerão apenas uma vez na lista de união. A lista final contém a interseção dos contadores de hardware entre as ferramentas de ataque selecionadas. A lista da interseção dos contadores corresponde a elementos repetidos nos conjuntos de HPCs selecionados. Analisando a recorrência com que cada contador aparece após a interseção, cria-se um gráfico contendo os contadores mais comuns entre as ferramentas de ataque e, eventualmente, o grau de influência no sistema. A lista final pode ser utilizada para desenvolver mecanismos de classificação e detecção de ataques DoS/DDoS no hospedeiro, com o auxílio de técnicas de ML, como a chamada DT Árvore de Decisão (SAEED; JAMEEL, 2021; AYTAÇ; AYDIN; ZAIM, 2020).

Diversas análises podem ser realizadas utilizando os contadores de desempenho selecionados, como a extração das variáveis apresenta as variáveis mais influentes de cada ferramenta de ataque, é possível desenvolver contramedidas de proteção especificas para cada ferramenta de ataque. No eventual surgimento de novas ferramentas de ataque, a metodologia apresentada poderá ser executada novamente, desenvolvendo novos conjuntos de dados ou atualizando os já classificados. Todos os dados de diagnóstico coletados, poderá compor um banco de dados crescente, com informações das situações de ataque no hospedeiro, aprimorando ainda mais a defesa contra ameaças que afrentam a disponibilidade do serviço, como os ataques de DDoS. O banco de dados poderá ser analisado utilizado conjunto de técnicas de AI e ML, servindo de apoio nas decisões que o administrador poderá utilizar de maneira proativa (BATCHU; SEETHA, 2021; NASCIMENTO et al., 2021b; NASCIMENTO et al., 2021a).

3.2 CONSIDERAÇÕES FINAIS

A metodologia foi desenvolvida com base em investigações da literatura, onde diversas soluções foram analisadas e avaliadas para se chegar no resultado presente. As análises realizadas na metodologia foram desenvolvidas com base nos trabalhos relacionados, mas ajustando o que acreditamos apresentar uma melhor representação no estudo dos HPCs

e já validados em trabalhos (NASCIMENTO et al., 2021a, 2021a). A metodologia faz parte de uma solução completa de contramedida de proteção em servidores web de segmento empresarial, onde está primeira etapa trata da realização do diagnóstico do sistema, análise das variáveis coletadas e a seleção dos HPCs mais influentes nas situações de ataque e utilização comum do serviço no sistema. As etapas futuras são justamente a utilização do diagnóstico capturado e a classificação de ataques DDoS no sistema, utilizando os recursos coletados e os HPCs selecionados no desenvolvimento de um mecanismo de detecção de ataques. A última etapa da contramedida de proteção é o desenvolvimento de um dispositivo de mitigação e tomada de decisão, onde será atribuído opções de proteção proativas, como balanceamento de carga, replicação e a adição de regras no *Firewall*. Além de mecanismos de proteção reativos, adicionando desafios para os agentes maliciosos que tentam prejudicar a disponibilidade do serviço utilizando dispositivos como *CAPTCHA*[1] ou a utilização de MTD no ambiente.

A elaboração da metodologia desenvolvida, utiliza de ferramentas e utilitários muitas vezes nativos nos OS Linux, com o intuito de facilitar a replicação e aplicação em diversos outros ambientes e possibilitando aos administradores realizarem seus próprios dados de diagnostico do sistema. Utilizando as variáveis de desempenho de recursos no sistema (como CPU, RAM) e rede como auxílio para as variáveis de baixo nível de abstração (como os HPCs) é possível realizar diversas analises de desempenho no sistema, como o planejamento da capacidade do serviço.

Sobre as análises de Sensibilidade e Estabilidade, o critério de corte e seleção dos contadores de desempenho não pôde ser realizado utilizando um valor determinador fixo. Como utilizamos cinco ferramentas de ataques diferentes, cada situação se comportou de forma distinta uma da outra. Cada ferramenta de ataque DDoS obteve dados diferentes e não sendo possível determinar um valor fixo de corte que contemplasse todas as cinco. Diante disso, selecionamos os contadores observando a maior diferença entre os valores no processo de análise dos dados na secção 3.1.3. Contemplando uma quantidade razoável de contadores, tomamos o cuidado de não deixar de fora uma quantidade muito elevada de contadores logo nas primeiras análises; levando em consideração que existem mais análises em todo o decorrer do processo. O conjunto de análises dá suporte à seleção dos contadores de desempenho para garantir que os dados coletados dos HPCs são consistentes.

[1] Acrônimo da expressão *"Completely Automated Public Turing test to tell Computers and Humans Apart"*: um teste de desafio cognitivo, utilizado como ferramenta *antispam*

4 ESTUDO DE CASO

Neste capítulo será abordado e detalhado o ambiente e suas especificações, como também as ferramentas e equipamentos utilizados e suas configurações. Juntamente com a apresentação do estudo de caso em que a metodologia apresentada foi aplicada e os resultados obtidos serão aqui interpretados, juntamente com os devidos comentários e, por fim, as considerações finais.

4.1 DETALHANDO O AMBIENTE

Implementamos três servidores físicos de rack HP ProLiant DL320e Gen8v2 no ambiente. O primeiro servidor contém o OS Debian de 64 bits [1] GNU/Linux 9 (*Stretch*) com Kernel 4.9.0-12, Quad-core Intel Xeon Processador E3-1220 v3, 3100 MHz e 8192 *Kilobyte* (KB) de cache, contendo duas placas de redes Broadcom NetXtreme BCM5720 Gigabit Ethernet PCIe Full-Duplex 1000 Mbps, com 16 *Gigabyte* (GB) de RAM e armazenamento total em *Hard Disk* (HD) de 750 GB. Os outros dois servidores físicos, denominados *node1* e *node2*, contêm configurações idênticas, com OS Ubuntu [2] 18.04.4 *Long-term Support* (LTS) (Bionic Beaver) de 64 bits, com 5.3.0-59 Kernel, Intel Xeon E3-1220 v3 quad-core, processador de 3500 MHz e cache de 8192 KB, contendo duas placas de rede Broadcom NetXtreme BCM5720 *Gigabit Ethernet* (GbE) PCIe Full-Duplex de 1000 Mbps, com 32 GB de RAM e 1 *Terabyte* (TB) de armazenamento HD total. A arquitetura proposta é mostrada na Figura 20.

Os servidores são responsáveis por executar a carga de trabalho, rodando Oracle VirtualBox [3] 6.1, onde os dois servidores rodarão um total de 10 *Virtual Machines* (VM) balanceadas simultaneamente entre os dois dispositivos. Cada coleta situacional contendo as ferramentas de ataque e a ferramenta de carga de trabalho, onde as VM estão rodando OS Linux e Windows de 32 bits, com cinco instâncias de cada sistema, com 2 GB de RAM cada, utilizando 1 CPU e 30 *Megabyte* (MB) de vídeo. O servidor *node1* é responsável por executar o servidor web Apache 2.4.29, sendo o único responsável pelos dados de diagnóstico. Todos os servidores (1 e 2) são conectados em uma *Local Area Network* (LAN) por um Switch GbE TL-SG1016 de 16 portas (ver Apêndice B).

Utilizamos o ambiente previamente definido, mostrado na Figura 20, para diagnosticar o servidor, rodando um servidor web. Os dados dos HPCs coletados no diagnóstico do sistema podem apoiar a realização do planejamento da capacidade do serviço, principalmente em situações de ataque ao sistema e uso regular ou sazonal do serviço. As informações adquiridas no diagnóstico do sistema podem identificar pontos únicos de fa-

[1] https://https://www.debian.org/
[2] https://ubuntu.com/
[3] https://www.virtualbox.org/

lha e possíveis gargalos no ambiente. É possível analisar as vulnerabilidades encontradas em cada tipo de equipamentos. Portanto, podemos abordar as contramedidas de proteção de acordo com os níveis de risco. No entanto, uma proporção muito mais significativa de VM com cargas de trabalho mais altas e um fluxo de ataque DoS/DDoS mais expressivo seria necessária para sobrecarregar o servidor. Por se tratar de uma conexão de rede GbE onde a taxa de transmissão é mais elevada e a latência é baixa, especificamente dentro de uma LAN do que em ambiente e serviços disponibilizados na internet.

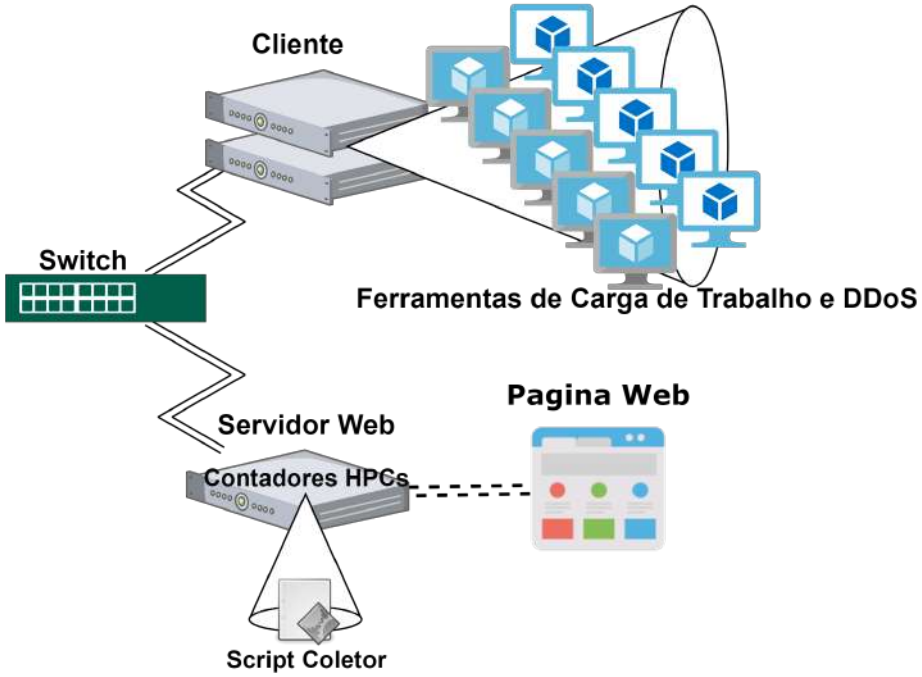


Figura 20 – Estrutura do Ambiente

4.2 FERRAMENTAL

As ferramentas de ataques DoS/DDoS utilizadas na execução do diagnóstico da metodologia, foram selecionadas com base nas evidências da área, sendo algumas das ferramentas de ataque mais populares (FADHLILLAH; KARNA; IRAWAN, 2021; ZHOU; PEZAROS, 2019; SHOREY et al., 2018). A ferramenta da geração da carga de trabalho foi selecionada com base na sua utilização em diversos outros trabalhos quando se trata de análise de desempenho de servidores. As ferramentas de ataques, foram amplamente selecionadas para cobrir diferentes ferramentas de ataques de DoS/DDoS de inundação HTTP. Para uma melhor representação de um cenário mais próximo da realidade, abrangendo diversas ferramentas de ataque. Essas ferramentas de ataque realizam ataques de maneiras diferentes, usando outros protocolos e recursos, embora pertençam ao mesmo tipo de ataque (ver Apêndice C).

4.2.1 Gerando Carga de Trabalho

Para a geração da carga de trabalho, a ferramenta *Apache JMeter v5.3*[4] simula o uso sazonal do sistema. O plano de teste foi realizado com solicitações HTTP ao servidor web. A configuração do ambiente de ataque é composto pelas 10 instâncias de cada VM, executando solicitações *GET*, *POST* e *HEAD* com um temporizador aleatório. Dois planos de testes com temporizadores com valores diferentes foram criados. Os dois planos de testes foram divididos igualmente entre as 10 VMs, visando uma simulação humana mais realista. Cada instância executou *5.000* usuários pelo tempo determinado na metodologia para realizar a coleta de dados, criando o perfil comportamental regular do sistema (GRABOVSKY et al., 2018; MATAM; JAIN, 2017).

4.2.2 Ferramentas de Ataque DDoS

Cinco ferramentas diferentes foram utilizadas na execução do diagnóstico para a coleta comportamental sobre situação de ataque no sistema. Entre as ferramentas de ataques DoS/DDoS mais populares, todas elas executando ataques HTTP de inundação (FADHLILLAH; KARNA; IRAWAN, 2021; ZHOU; PEZAROS, 2019; KUMAR; KUMAR et al., 2016). Executando ataques de DDoS, cada instância das VMs (num total de 10) realizaram simultaneamente um ataque distribuído durante o tempo determinado para cada coleta. Cada coleta foi realizada executando uma ferramenta de ataque por vez em todas as 10 VMs e, assim, gerando cinco conjuntos de dados respectivamente.

- **H.O.I.C (*High Orbit ION Cannon*)**

Ferramenta com interface de uso fácil, utilizada para realização de ataques DoS/DDoS, inundando os serviços do destinatário com solicitações HTTP, *GET* e *POST*, podendo atingir taxas muito alta. Projetada inicialmente para corrigir várias falhas da aplicação Low Orbit Ion Cannon (LOIC), como o problema de não ofuscação dos endereços IP.

- **H.U.L.K. (*HTTP Unbearable Load King*)**

A ferramenta de negação de serviço HTTP, inicialmente desenvolvida em *python*, é usada para atacar servidores web. Executando solicitações de tráfego exclusivo e ofuscado, o que torna difícil detectar, ignorar os mecanismos de cachê e acessar rapidamente os recursos do servidor. Assim, sendo totalmente utilizado para fins de pesquisa e teste de desempenho.

[4] <http://jmeter.apache.org>

- **L.O.I.C. (*Low Orbit Ion Cannon*)**

Esta ferramenta multiplataforma de ataque DDoS escrito em C#, que é bem conhecido e de fácil manuseio, executa ataques com solicitações de vários protocolos, principalmente HTTP. Ela ajuda a realizar testes de desempenho para verificar a estabilidade de um servidor *Web* e a estrutura da rede. Além disso, ela não esconde o IP do invasor.

- ***SwitchBlade***

A ferramenta de ataque de negação de serviço de código aberto da Proactive Risk, OWASP (Open Web Application Security Project) SwitchBlade, anteriormente conhecida como ferramenta HTTP*POST/DoS*, permite que ataques como HTTP *POST* e *GET*, *Slowloris* e renegociação *Secure Sockets Layer* (SSL) sejam executados. Muito útil para testar a estabilidade e o desempenho da rede em aplicativos da *web*.

- ***Tor's Hammer***

A ferramenta de ataque DDoS da camada de aplicação, escrita em *python*, executa ataques HTTP *POST* de baixa taxa. A ferramenta cria muitas conexões de rede, o que dificulta a identificação. A ferramenta envia pequenos pacotes contínuos de HTTP fragmentados ao servidor de destino, que parecem legítimos e acabam consumindo o recurso do destino por longos períodos. Ela pode ser usada junto com o *The Onion Router* (TOR) para identificar dificuldades durante os ataques.

4.3 EXECUÇÃO

Na execução do experimento, 10 VMs são utilizadas por vez em cada situação de ataque DDoS e geração de carga de trabalho. Nessas VMs, ambos sistemas Linux e Windows são executados. Na execução dos ataques, cinco ferramentas de ataque DoSDDoS entre as mais populares serão utilizadas (SINGH; RAM, 2021; BROOKS et al., 2021; PANDE et al., 2021). Cada coleta tem um tempo total de *120 minutos* onde são executados ataques DDoS de inundação HTTP que têm como alvo um servidor web Apache e executando em um servidor com OS *Debian 9 (Stretch)* de 64 bits. Cada ferramenta de ataque é configurada devidamente para executar um fluxo de ataque dentro da capacidade total do servidor, sem fazer com que ele fique indisponível ou negue o provimento do serviço para outros usuários. Sendo o principal propósito do diagnóstico do sistema, a captura comportamental do hospedeiro, observando a utilização de seus recursos entre a situação ataque e o uso sazonal do sistema antes que ele fique totalmente indisponível.

Após a análise de sensibilidade, é possível observar na Tabela 4 os contadores que foram selecionados com base em seus valores. Os contadores com maior coeficiente são selecionados (em azul) e os demais (em amarelo) são removidos neste primeiro filtro com a análise de sensibilidade. Foram executados alguns critérios de cortes nos valores dos contadores, mas nenhum valor de critério fixo pôde ser atribuído para os cinco casos. Portanto, o critério de corte foi executado observando as mudanças mais significativas nos valores em comparação do último e penúltimo valor e os contadores selecionados vão para as próximas fases.

Tabela 4 – Análise de Sensibilidade dos HPCs

Ferramenta H.O.I.C.		Ferramenta H.U.L.K.		Ferramenta L.O.I.C.		Ferramenta SwitchBlade		Ferramenta Tor's Hammer	
Contadores	Valores	Contadores	Valores	Contadores	Valores	Contadores	Valores	Contadores	Valores
LLC-load-misses	0.831639	LLC-store-misses	8.501760	L1-dcache-load-misses	0.433375	iTLB-loads	0.734545	iTLB-loads	0.580811
LLC-stores	0.747267	branch-loads	7.404582	branch-loads	0.410258	LLC-store-misses	0.558837	dTLB-loads	0.524113
cache-references	0.628977	branch-instructions	7.186736	dTLB-loads	0.409512	branch-instructions	0.509917	L1-dcache-loads	0.522034
L1-icache-load-misses	0.541665	instructions	7.166257	instructions	0.408843	LLC-load-misses	0.490810	branch-loads	0.521576
LLC-store-misses	0.496816	dTLB-loads	5.333723	branch-instructions	0.406635	instructions	0.490619	branch-instructions	0.504730
L1-dcache-load-misses	0.461999	L1-dcache-loads	4.902347	L1-dcache-loads	0.400752	branch-loads	0.482196	instructions	0.503584
cache-misses	0.447433	LLC-load-misses	4.617787	LLC-store-misses	0.353569	iTLB-load-misses	0.406850	LLC-store-misses	0.484594
branch-load-misses	0.447349	L1-dcache-load-misses	3.979990	LLC-stores	0.331118	L1-dcache-loads	0.399758	L1-dcache-load-misses	0.431136
branch-misses	0.430665	cache-misses	3.740198	dTLB-store-misses	0.325567	dTLB-loads	0.382757	iTLB-load-misses	0.416264
dTLB-load-misses	0.424703	LLC-stores	2.399282	cache-references	0.288260	cache-references	0.253517	LLC-load-misses	0.373168
LLC-loads	0.401189	cpu-cycles	1.603899	cache-misses	0.279554	dTLB-store-misses	0.243749	dTLB-store-misses	0.316961
dTLB-store-misses	0.334518	LLC-loads	1.480066	LLC-loads	0.273120	LLC-stores	0.227115	LLC-stores	0.250829
iTLB-loads	0.326096	dTLB-stores	1.460504	iTLB-load-misses	0.228475	dTLB-load-misses	0.152770	LLC-loads	0.237057
L1-dcache-loads	0.218981	L1-dcache-stores	1.440068	iTLB-loads	0.208108	LLC-loads	0.143607	dTLB-stores	0.201685
branch-instructions	0.203602	dTLB-store-misses	1.313792	LLC-load-misses	0.197808	dTLB-stores	0.099783	cache-references	0.182771
instructions	0.202270	iTLB-loads	0.912893	dTLB-stores	0.182159	L1-icache-load-misses	0.099262	L1-dcache-stores	0.180684
branch-loads	0.199033	cache-references	0.724595	L1-icache-load-misses	0.181771	L1-dcache-stores	0.097850	cache-misses	0.171936
dTLB-loads	0.198327	iTLB-load-misses	0.711829	L1-dcache-stores	0.164836	branch-load-misses	0.089305	branch-misses	0.100964
L1-dcache-stores	0.130699	L1-icache-load-misses	0.673681	branch-load-misses	0.134738	branch-misses	0.086985	branch-load-misses	0.100882
dTLB-stores	0.129750	dTLB-load-misses	0.390671	branch-misses	0.112903	cache-misses	0.086058	L1-icache-load-misses	0.099099
iTLB-load-misses	0.082735	branch-load-misses	0.327617	dTLB-load-misses	0.099534	L1-dcache-load-misses	0.083231	dTLB-load-misses	0.094320
cpu-cycles	0.055581	branch-misses	0.313515	cpu-cycles	0.073218	cpu-cycles	0.032602	cpu-cycles	0.092658
bus-cycles	0.000085	bus-cycles	0.182584	bus-cycles	0.000221	bus-cycles	0.000333	bus-cycles	0.000079
ref-cycles	0.000000	ref-cycles	0.182322	ref-cycles	0.000000	ref-cycles	0.000000	ref-cycles	0.000000

Após a análise de estabilidade, é possível observar na Tabela 5 os contadores que foram selecionados com base em seus valores. Os contadores com maior coeficiente são removidos (em amarelo) e os demais são selecionados (em azul) neste segundo filtro com a análise de estabilidade. Foram executados alguns critérios de cortes no valores dos contadores, mas nenhum valor de critério fixo pôde ser atribuído para os cinco casos. Portanto, o critério de corte foi executado observando o coeficiente disponibilizado pela equação AS_t, utilizando

os menores valores mais significativos nos valores dos contadores em comparação do último e penúltimo valor, os contadores selecionados vão para as próximas fases.

Após as análises de sensibilidade e estabilidade, os contadores restantes são submetidos a análises para extrair os contadores mais influentes, utilizando a RF. Como mostrado na Tabela 6, os contadores em azul são os contadores mais relevantes dentre todas os HPCs coletados de cada uma das suas respectivas ferramentas de ataques. As análises de sensibilidade e estabilidades são executadas para seleção dos contadores de desempenho mais relevantes dentre o conjunto total de contadores. São utilizados apenas os contadores de referência selecionados e os coeficientes são abstraídos, sendo estas análises feitas a partir dos seus valores em cada observação.

Tabela 5 – Análise de Estabilidade dos HPCs

Ferramenta H.O.I.C.		Ferramenta H.U.L.K.		Ferramenta L.O.I.C.		Ferramenta SwitchBlade		Ferramenta Tor's Hammer	
Contadores	Valores	Contadores	Valores	Contadores	Valores	Contadores	Valores	Contadores	Valores
LLC-load-misses	1.862409	LLC-store-misses	1.925386	branch-instructions	0.447865	iTLB-loads	1.001387	iTLB-loads	0.767863
LLC-store-misses	0.803515	instructions	1.723719	instructions	0.433558	LLC-store-misses	0.893345	LLC-store-misses	0.759307
cache-misses	0.575510	branch-instructions	1.716855	branch-loads	0.417301	LLC-load-misses	0.703191	iTLB-load-misses	0.561002
LLC-stores	0.500088	branch-loads	1.714141	L1-dcache-loads	0.395085	branch-instructions	0.650791	branch-loads	0.560750
dTLB-store-misses	0.433700	LLC-load-misses	1.623506	dTLB-store-misses	0.391533	branch-loads	0.642852	L1-dcache-loads	0.527003
cache-references	0.432292	cache-misses	1.495431	dTLB-loads	0.349153	instructions	0.622818	branch-instructions	0.524486
L1-icache-load-misses	0.393605	dTLB-loads	1.478484	iTLB-load-misses	0.272889	iTLB-load-misses	0.549403	LLC-load-misses	0.524096
branch-load-misses	0.338264	L1-dcache-loads	1.456566	LLC-stores	0.268483	L1-dcache-loads	0.542906	instructions	0.503171
dTLB-load-misses	0.327619	L1-dcache-load-misses	1.158031	LLC-store-misses	0.266983	dTLB-loads	0.518901	dTLB-loads	0.499937
LLC-loads	0.318823	LLC-stores	1.107411	LLC-load-misses	0.247126	dTLB-store-misses	0.281253	dTLB-store-misses	0.380200
L1-dcache-load-misses	0.316147	dTLB-store-misses	0.892838	cache-references	0.227946	cache-references	0.238415	LLC-stores	0.209360
branch-misses	0.303762	cpu-cycles	0.836352	iTLB-loads	0.221480	LLC-stores	0.198089	cache-misses	0.152770
iTLB-loads	0.272449	L1-dcache-stores	0.783574	cache-misses	0.215202	dTLB-load-misses	0.157505	cache-references	0.147227
L1-dcache-loads	0.207527	dTLB-stores	0.777939	LLC-loads	0.202927	LLC-loads	0.154980	L1-dcache-load-misses	0.142020
branch-instructions	0.180562	LLC-loads	0.777892	L1-dcache-load-misses	0.162347			LLC-loads	0.136921
instructions	0.178592	iTLB-loads	0.601526	L1-icache-load-misses	0.148848			dTLB-stores	0.094876
branch-loads	0.177123			dTLB-stores	0.120051			L1-dcache-stores	0.079440
dTLB-loads	0.168997			branch-load-misses	0.119772				
				L1-dcache-stores	0.101047				
				branch-misses	0.084832				

Tabela 6 – Contadores HPCs mais relevantes

Ferramenta H.O.I.C.		Ferramenta H.U.L.K.		Ferramenta L.O.I.C.		Ferramenta SwitchBlade		Ferramenta Tor's Hammer	
Contadores	Valores	Contadores	Valores	Contadores	Valores	Contadores	Valores	Contadores	Valores
L1-dcache-loads	0.306834	dTLB-stores	0.20182	branch-misses	0.332633	dTLB-store-misses	0.486631	cache-references	0.470033
instructions	0.267651	L1-dcache-stores	0.20022	L1-icache-load-misses	0.245063	cache-references	0.282482	LLC-loads	0.190420
branch-instructions	0.176806	LLC-loads	0.19994	L1-dcache-load-misses	0.139998	LLC-stores	0.157114	dTLB-stores	0.110466
dTLB-loads	0.152935	cpu-cycles	0.19949	branch-load-misses	0.111006	dTLB-load-misses	0.063108	L1-dcache-load-misses	0.092970
branch-loads	0.095774	dTLB-store-misses	0.19853	L1-dcache-stores	0.088603	LLC-loads	0.010665	cache-misses	0.070103
		iTLB-loads	0.00000	dTLB-stores	0.082697			L1-dcache-stores	0.066007

Para cada conjunto de dados, os contadores mais relevantes para cada execução de ataque (DoS/DDoS) foram classificados entre as demais variáveis após a realização das análises de FS da RF. É possível observar na Figura 22 os contadores de desempenho mais influentes em cada uma das ferramentas de ataque utilizadas. Duas listas são criadas respectivamente a partir da união e da interseção, com todos os contadores selecionados pertencendo aos conjuntos de cada ferramenta. As variáveis capturadas para a classificação comportamental do hospedeiro são os HPCs, onde cada contador de desempenho é classificado em ordem crescente de acordo com sua pontuação de influência.

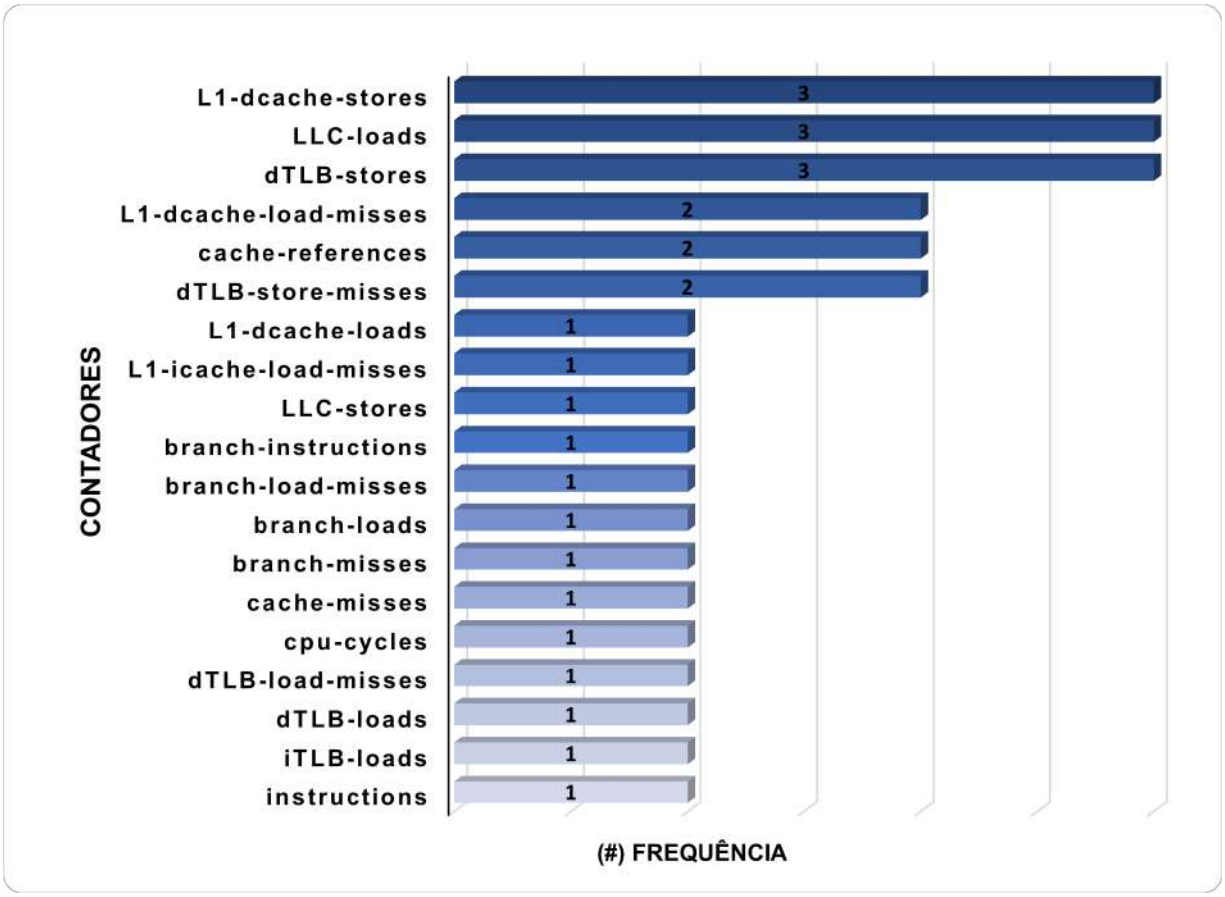


Figura 21 – Frequência dos HPCs

As respectivas listas contêm os contadores mais influentes extraídos dos subconjuntos de dados coletados. Após a análise das variáveis mais relevantes, realizamos a união dos conjuntos de HPCs com todos os contadores das ferramentas de ataque DoS/DDoS. Assim, a lista de união resulta em todas as ferramentas de ataque, sendo seus respectivos HPCs apresentados apenas uma vez.

Depois de desenvolver o conjunto de interseção, pode-se obter outro conjunto contendo a classificação de frequência de HPCs para cada ferramenta de ataque. Após o cruzamento dos conjuntos de HPCs, é possível medir a frequência de cada HPCs dentro do número total de ferramentas utilizadas; os HPCs mais frequentes estão no topo da lista e são organizados por grau de frequência. Conforme observado na Figura 21, três contadores coincidem em três conjuntos distintos de HPCs, onde as ferramentas de ataque DDoS normalmente usam esses contadores com mais frequência. Portanto, exigindo mais atenção, pois eles aparecem em comum em três conjuntos diferentes de ferramentas de ataque. Então é possível observar mais três contadores com frequência de duas repetições cada, que também são de grande importância na classificação do comportamento do hospedeiro em situações de ataque DDoS.

O conjunto inicial contém 24 HPCs, além da variável de referência binária (1 para situações de ataque e 0 para situação de uso comum), responsável por classificar a situação de ataque no sistema e as situações de uso comum. Dependendo dos recursos disponibilizados pelo hospedeiro, as variáveis de HPCs, podem consumir uma quantidade significativa de recursos, ainda mais se forem coletados simultaneamente em tempo real. Com a captura de vários contadores, pode ocorrer de prejudicar significativamente a precisão dos algoritmos de classificação/detecção. Por esse motivo, a redução de dimensionalidade é necessária para selecionar apenas os contadores mais influentes na caracterização, que no presente estudo são ataques DDoS no serviço web. A redução de dimensionalidade é uma técnica que tem como suas principais características a redução do *overfitting*, onde menos dados redundantes significa menos oportunidade de tomar decisões com base no ruído. Além disso, otimiza a precisão, uma vez que menos dados enganosos significa que a precisão da modelagem melhora e consequentemente reduz o tempo de treinamento. Com menos pontos de dados, a complexidade diminui e o algorítimo consegue treinar mais rápido (AAMIR et al., 2021; ABBAS; ALMHANNA, 2021).

O conjunto de união coletou um total de 19 HPCs, conforme mostra a Tabela 7, onde selecionamos os mais relevantes dentro do conjunto inicial de contadores. Após realizar as análises de importância e desenvolver um ranqueamento de influencia dos contadores, foi possível obter uma redução de 26% sobre o grupo inicial de HPCs. O conjunto da interseção selecionou 19 HPCs, conforme mostrado na Tabela 8. Os contadores foram selecionados devido ao cruzamento dos conjuntos de contadores de todas as ferramentas executadas. O conjunto de contadores da interseção também foi reduzido em 26% em relação ao conjunto inicial. Depois de desenvolver o conjunto da intersecção, é possível

obter um novo conjunto contendo a frequência dos contadores de desempenho para cada ferramenta de ataque. O conjunto contendo a frequência mostrará o número de vezes em que cada contador aparece simultaneamente em um ou mais conjuntos de contadores nas ferramentas de ataque.

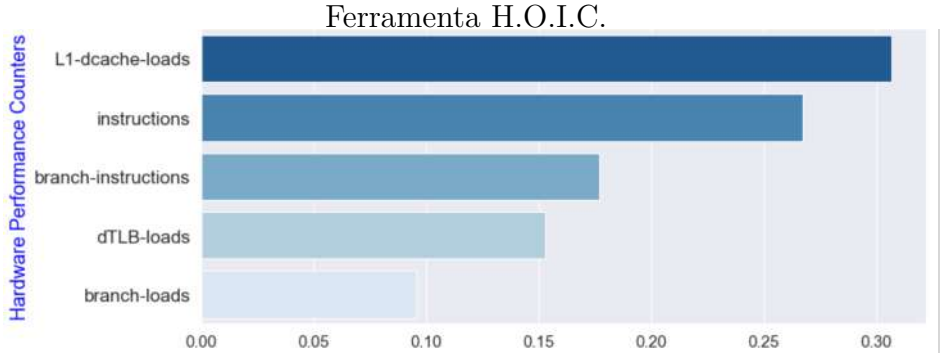
Vale a observação de que as situações de ataque e a situação de uso comum do serviço, como mostrado na Figura 23, não são possíveis de se distinguir apenas visualizando os gráficos das situações. Isso reforça a necessidade das demais análises para filtrar e selecionar os contadores de hardware de cada ferramenta de ataque separadamente. O eixo Y nos gráficos representa a quantidade de amostras e o eixo X representa os contadores de desempenho em cada ferramenta de ataque utilizada.

Essas listas de união, interseção e frequência podem auxiliar a compreender como os contadores se comportam em situações de ataque DDoS no sistema. Com esses conjuntos, podemos fornecer recursos de monitoramento mais significativos, com foco nos contadores que mais influenciaram a classificação comportamental do hospedeiro nas condições que desejamos definir e descartando os contadores que não afetam as situações de ataque de uso regular do sistema.

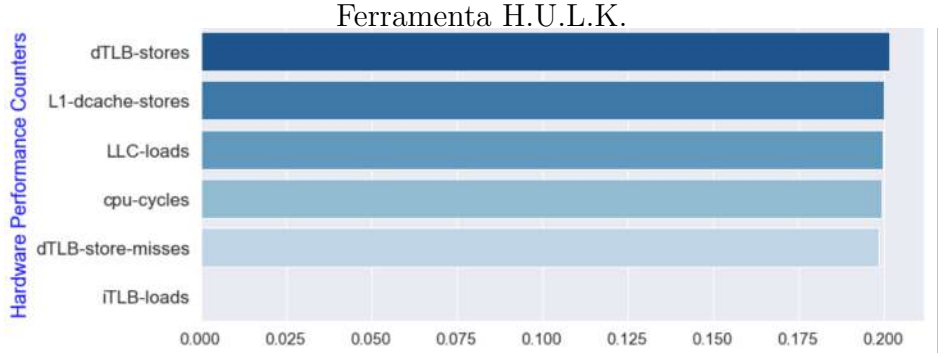
O diagnóstico realizado descarta dispositivos adicionais no caminho de tráfego, onde a conexão entre os clientes e o servidor web é realizada apenas por um *switch* de rede, concentrando-se na coleta e tratamento dos dados de HPCs nas situações de ataque DoS/DDoS e uso comum do serviço web. Dessa forma, é possível criar perfis bem definidos que podem ser desenvolvidos e usados para produzir contramedidas mais eficazes com base no próprio hospedeiro. Por exemplo, a detecção automática ou semiautomática de ataques desenvolvidos para o ambiente, criando conjuntos de dados específicos para o ambiente que podemos avaliar sem a necessidade de conjuntos de dados de terceiros.

O diagnóstico visa captar dados do comportamento de um servidor web fornecidos por um processador de segmento empresarial e realizar análises e observações para obter um melhor desempenho na disponibilidade do serviço, onde agentes maliciosos se esforçam para prejudicar a oferta de serviços disponíveis na internet. Os serviços oferecidos por equipamentos do segmento empresarial oferecem alto nível de performance e disponibilidade, pois costumam possuir recursos mais expressivos, tornando-os alvos preferenciais de agentes maliciosos (GRAKOVSKI, 2021; AWAN et al., 2021). Com os resultados obtidos no diagnóstico, é possível implementar um ambiente com apenas os dispositivos necessários, evitando gargalos do sistema e possíveis alvos para esses ataques.

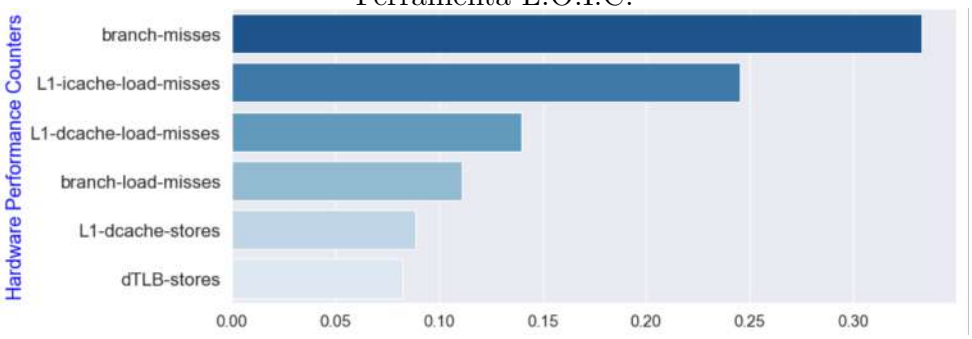
Ferramenta H.O.I.C.



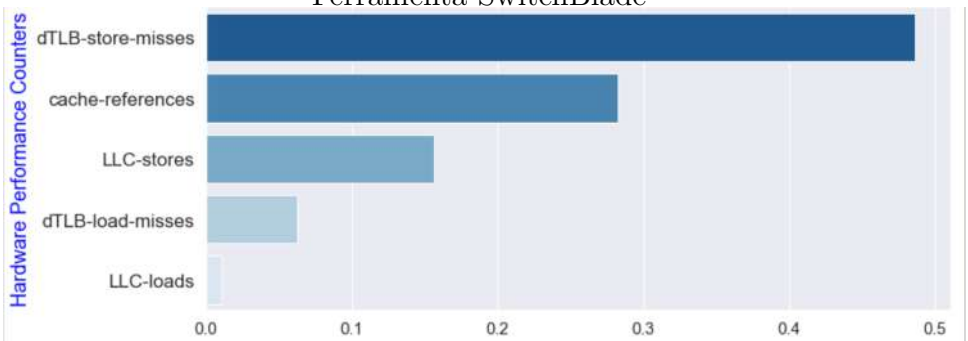
Ferramenta H.U.L.K.



Ferramenta L.O.I.C.



Ferramenta SwitchBlade



Ferramenta Tor's Hammer

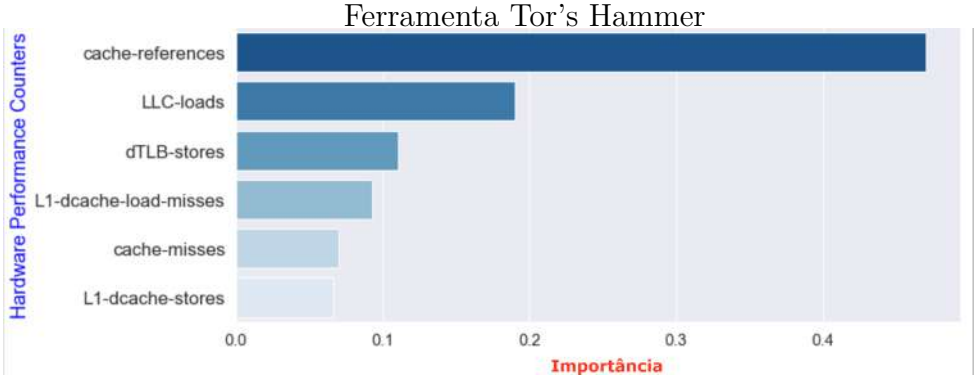


Figura 22 – Gráficos de Importância dos HPCs

Tabela 7 – Lista da União dos HPCs

Ordem	**Contadores**
1	L1-dcache-load-misses
2	L1-dcache-loads
3	L1-dcache-stores
4	L1-icache-load-misses
5	LLC-loads
6	LLC-stores
7	branch-instructions
8	branch-load-misses
9	branch-loads
10	branch-misses
11	cache-misses
12	cache-references
13	cpu-cycles
14	dTLB-load-misses
15	dTLB-loads
16	dTLB-store-misses
17	dTLB-stores
18	iTLB-loads
19	instructions

Tabela 8 – Lista da Interseção dos HPCs

Ordem	**Contadores**
1	L1-dcache-stores
2	LLC-loads
3	dTLB-stores
4	L1-dcache-load-misses
5	cache-references
6	dTLB-store-misses
7	L1-dcache-loads
8	L1-icache-load-misses
9	LLC-stores
10	branch-instructions
11	branch-load-misses
12	branch-loads
13	branch-misses
14	cache-misses
15	cpu-cycles
16	dTLB-load-misses
17	dTLB-loads
18	iTLB-loads
19	instructions

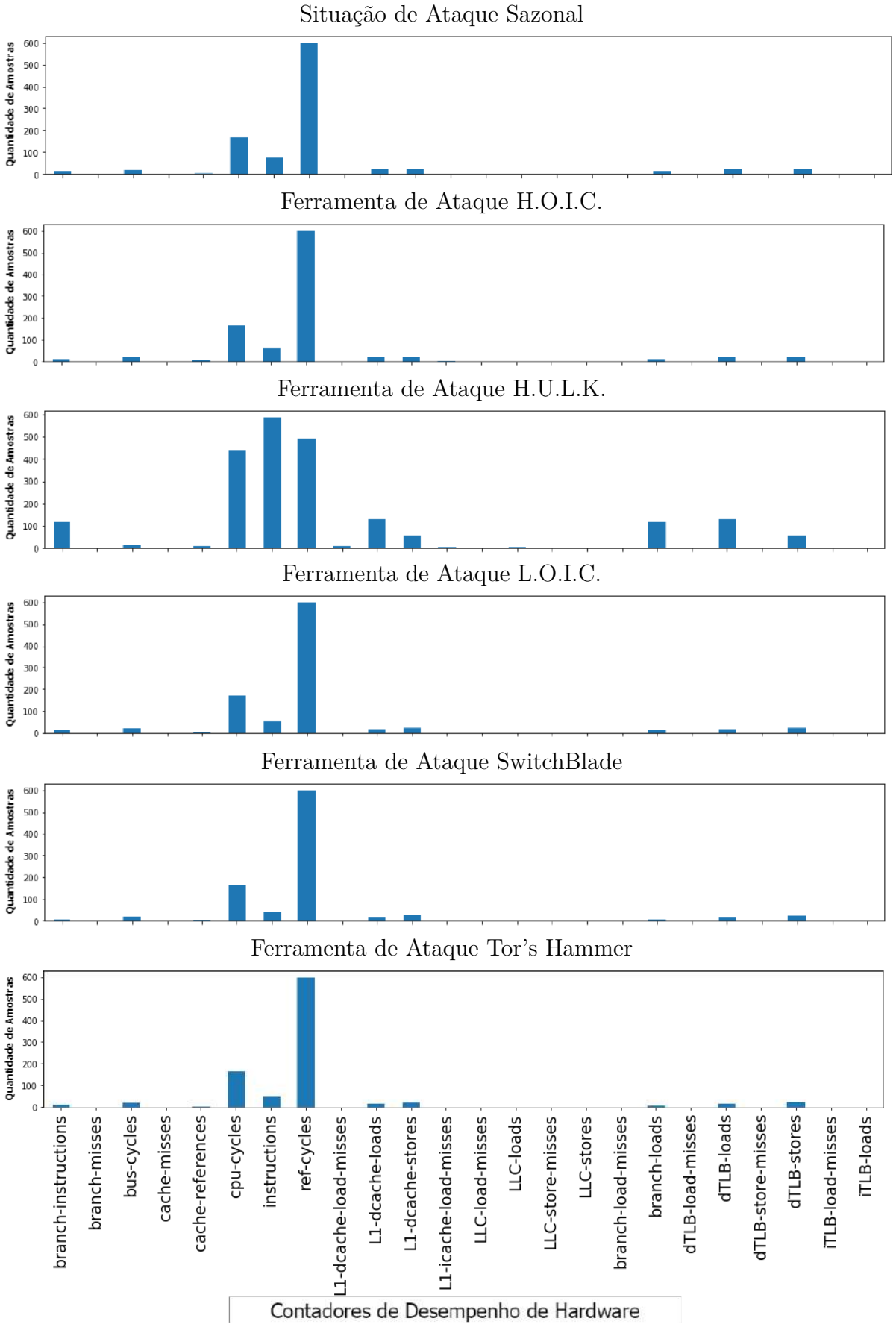


Figura 23 – Gráficos Situacionais

4.4 CONSIDERAÇÕES FINAIS

Diferentemente dos estudos anteriores (ROHAN; BASU; KARRI, 2019; ALAM et al., 2017; ADDEPALLI; KARRI; JYOTHI, 2017; TORRES; LIU, 2016; YUAN et al., 2011), os diagnósticos atuais são realizados em um servidor de nível corporativo, não em desktops domésticos comumente usados por usuários comuns. O diagnóstico proposto realiza um monitoramento não intrusivo, sem fazer alterações na aplicação, seja no código fonte ou binário (TANDON et al., 2019; OIKONOMOU; MIRKOVIC, 2009). Ao contrário do método sugerido em trabalhos anteriores (KRISHNAMURTHY; KARRI; KHORRAMI, 2019; ELNAGGAR; CHAKRABARTY; TAHOORI, 2017), as coletas e análises foram realizadas no hospedeiro sem adição de ativos na via de tráfego. O presente estudo também difere dos trabalhos de (DEMOULIN et al., 2019; ADDEPALLI; KARRI; JYOTHI, 2017) no tratamento das variáveis coletadas no diagnóstico e na escolha dos contadores retirados, mas apresentam métodos diagnósticos semelhantes.

Vale ressaltar que mesmo com consumo elevado de recursos do hospedeiro, não necessariamente a causa pode ser um ataque iminente ao servidor, podendo ser situações sazonais onde o consumo de recursos é normalmente mais utilizado. Enfatizamos que uma proporção muito mais significativa de VMs, com cargas de trabalho maiores e um fluxo de ataque mais significativo, seria necessária para sobrecarregar o servidor e torná-lo indisponível. Como os experimentos foram executados em uma conexão de rede GbE, as taxas de transmissão são muito mais altas e a latência muito mais baixa em uma LAN do que em um serviço disponível na internet. A administração do experimento foi executada por acesso remoto criptografado e neste momento não foi analisado o impacto do acesso no serviço. As ferramentas amplamente selecionadas usadas para realizar ataques DoS/DDoS cobrem diferentes mecanismos de ataque de DoS/DDoS HTTP para representar um cenário melhor e mais próximo da realidade. Essas ferramentas de ataque realizam ataques de diferentes formas, utilizando outros protocolos e recursos, mesmo quando pertencem ao mesmo tipo de ataque.

A metodologia apresentada neste trabalho e este estudo de caso nos permitiram extrair os principais HPCs em um servidor de nível corporativo que hospeda um servidor web. Contadores são ótimos recursos para monitoramento e classificação em tempo real do comportamento do sistema. Este último, onde inicialmente todos os HPCs foram obtidos para cada ferramenta de ataque e selecionados para classificação comportamental do hospedeiro em situações de ataque e uso comum do serviço. Em suma, um total de 24 contadores de desempenho foi utilizado e coletado. A lista de união e interseção selecionou ambas 19 contadores, obtendo uma redução final de 26% na quantidade de HPCs utilizados.

5 CONCLUSÃO

Com o crescimento constante do cibercrime e a complexidade dos ataques de DoS/DDoS, o gerenciamento de rede tradicional e as técnicas de proteção não são mais tão eficazes quanto costumavam ser quando usados separadamente como contramedidas de proteção de serviços da *web*. Portanto, é necessário desenvolver e utilizar tecnologias e técnicas de diferentes áreas emergentes para unir esforços contra agentes maliciosos. Considerando que entre os diversos objetivos desses agentes está dificultar o acesso de usuários e empresas, indisponibilizar seus serviços e impactar negativamente na disponibilidade e QOS. Várias tecnologias, como técnicas de AI, são aliadas poderosas para os administradores de rede e de sistemas. Dessa forma, um ambiente híbrido pode ser o mais favorável, aumentando a probabilidade de proteção de contramedidas e, portanto, uma infraestrutura mais adaptável se comporta melhor em situações adversas. A centralização dos recursos nas características mais relevantes para a classificação das condições é essencial para a detecção de ataques e, consequentemente, para a obtenção de melhor desempenho no combate a essas ameaças de ataques.

É necessário adaptar cada estrutura especificamente ao ambiente, proporcionando níveis de segurança em camadas, onde cada ativo é implementado apenas conforme necessário para a prestação do serviço. Assim, posicionar estrategicamente cada ativo, permitindo que em qualquer vulnerabilidade, um cubra o outro em caso de eventuais falhas. De certa forma, conforme mencionado anteriormente no trabalho, é possível evitar falhas de implantação, onde ativos desproporcionais dentro da rede trazem pontos únicos de falha e correm o risco desses dispositivos se tornarem alvos de ataques DoS/DDoS deixando o ambiente vulnerável.

A utilização da metodologia apresentada, com a execução do *script* desenvolvido, facilitará a replicação e utilização em outros ambientes e infraestruturas. Portanto, servindo como suporte para a realização de estudos de desempenho e na implementação de contramedidas de segurança mais maduras no ambiente. Os dados de diagnóstico capturam os perfis comportamentais das ferramentas de ataques, conseguindo capturar suas características individuais e a tendência geral de um grande conjunto dessas ferramentas. A utilização desta metodologia descarta a necessidade de utilização de conjuntos de dados de diagnósticos realizado por terceiros.

Com este estudo, observamos que os HPCs podem efetivamente identificar ataques ao sistema. A investigação desses eventos de hardware de baixo nível mostrou que eventos de *hardware* de características diferentes tinham peculiaridades distintas para cada ferramenta de ataque estudada. Além disso, após selecionar as variáveis mais relevantes, se a precisão da detecção ainda não for satisfatória, será necessário executar novamente os processos iniciais. Assim, uma seleção mais abrangente ou com variáveis diferentes pode

obter melhores resultados.

Com a necessidade de adquirir características mais fiéis ao comportamento do hospedeiro, uma vez que apenas as variáveis de recursos tradicionais normalmente classificam os comportamentos de forma equivocada; como as variáveis de utilização do sistema, porcentagem de utilização da CPU, RAM e rede. Estas são variáveis de recursos de nível mais elevado de abstração, o que ajuda a reduzir a complexidades das contramedidas de proteção e abranger uma ampla gama de ambientes que possibilita a utilização dessas contramedidas. O ponto crucial do uso de variáveis com alto nível de abstração é o desperdício de características únicas que cada ambiente e sistema pode conter. Isso pode ser de suma importância na classificação de ameaças no sistema, além de que ameaças mais atuais são capazes de driblar a utilização desses monitores de recursos e ofuscar o tráfego malicioso das solicitações dos usuários reais utilizando o sistema.

As variáveis de recursos tradicionais também podem ser utilizadas como recursos de suporte, ao invés do seu uso separadamente como recursos principais de contramedidas de proteção. A extração de características mais sensíveis e fiéis ao comportamento do hospedeiro é o principal desafio da área de segurança da informação. Com o objetivo de fortalecer as contramedidas de proteção, as variáveis tradicionais de monitoramento dos recursos são utilizadas como suporte para variáveis de baixo nível de abstração, como os HPCs. Contramedidas que utilizam variáveis de baixo nível de abstração normalmente necessitam de análises e tratamento adicionais para interpretação. Dessa forma, torna-se aptas a serem utilizadas como variáveis válidas no sistema e assim ser implementadas nas ferramentas de proteção.

Existem muitas variáveis disponíveis no sistema e, por esse motivo, a principal preocupação é selecionar as características que realmente ajudem na classificação comportamental do hospedeiro. Sendo aquelas que contêm um grau de influência significativo suficiente para poder ser utilizadas. Um dos problemas no uso de múltiplas variáveis é o caso particular que cada variável tem em referência ao sistema em uma determinada situação e, assim, nem sempre possibilitando ser correlacionada com as demais variáveis coletadas no sistema. Portanto, é necessário investigar previamente as características e peculiaridades das variáveis que se planeja utilizar. Dessa forma, tomar precauções para não se coletar todas as variáveis disponíveis no sistema e criar conjuntos muito densos de variáveis, que possivelmente prejudicariam as análises e o processamento dos dados. Além disso, sendo fundamental a comparação das variáveis de características e de abstração correlacionadas. Após planejamento das variáveis, é necessário classificar e dividi-las em conjuntos de grupos e subgrupos, definindo o grau de abstração, de referência e de natureza, para que elas possam ser utilizadas na análise e implementação corretas.

Como a metodologia se trata principalmente da coleta de diagnóstico do hospedeiro, com os dados de referência coletados por um *script* contendo utilitários de manipulação de dados e ferramentas de diagnósticos, eventualmente os dados podem tornar-se de difícil

manipulação. Isso pode acontecer quando executados simultaneamente dentro do sistema, uma vez que cada ferramenta se comporta de forma peculiar e acidentalmente podem se comportar de forma a prejudicar a execução de outras ferramentas. Assim, a execução separadamente de cada parte do código que se deseja implementar poderá evitar problemas diversos na execução do diagnóstico e manipulação dos dados. Desta forma, é possível evitar sobrecargas indevidas, geração de carga e dados adicionais e uma análise posterior defasada da situação.

Importante deixar claro que como o principal objetivo da metodologia é a captura e classificação de situações de ataques DoS/DDoS no sistema e na aplicação web, é necessário possibilitar a capturar do comportamento antes mesmo da indisponibilidade do serviço já que, dependendo da carga executada com a ferramenta de ataque, o acesso ao serviço pode ficar completamente indisponível. Com isso, pode-se perder todo o diagnóstico, sem conseguir executar com precisão como o hospedeiro se comporta exatamente em uma situação de ataque. Portanto, é necessário ajustar as ferramentas de ataque utilizadas abaixo da capacidade total do hospedeiro, evitando causar a indisponibilidade do serviço e gerando dados de diagnóstico temporal mais representativos.

Recomendamos atenção na utilização quantidade de ferramentas e utilitários em uso simultâneo na realização do diagnóstico do sistema, dando preferência a recursos mais simples e leves e evitando uma sobrecarga adicional já na utilização das ferramentas para realização. É importante supervisionar toda realização do diagnóstico no ambiente justamente para analisar como o sistema se comporta e evitar ruídos consequentes de uma má utilização de suas ferramentas, garantindo que a coleta foi feita de forma correta. Aconselhamos a realização do gerenciamento do ambiente por acesso remoto para evitar sobrecarga adicional em utilização de um ambiente gráfico e para simular também uma administração real de um ambiente empresarial de servidores. Por fim, frisamos uma atenção maior na quantidade de variáveis coletadas e utilizadas no diagnóstico e, consequentemente, no desenvolvimento das contramedidas de proteção. Além da precaução com o sistema, que pode prejudicar a detecção/classificação da contramedida de proteçã em situações que este disponibiliza uma quantidade muito densa de contadores de HPCs mais as variáveis de recursos, como a utilização da CPU, RAM, largura de banda e requisições na aplicação.

Em suma, conseguimos realizar o desenvolvimento de uma metodologia não intrusiva para diagnóstico de situações de ataques de DDoS em servidores de segmento corporativo. A geração de perfis comportamentais de ataques de DDoS através das características do sistema e da seleção dos HPCs mais influentes. Para isto, foram realizadas diversas técnicas estatísticas e de AI na análises de dados do diagnóstico.

5.1 TRABALHOS FUTUROS E LIMITAÇÕES

Em trabalhos futuros, é possível estender as diferentes formas de ataques utilizados e abordar diferentes servidores da web. Entre os vários usos possíveis desses recursos está a implementação de um detector de ataques de DoS/DDoS em servidores *web* e a aplicação da solução em serviço disponível na internet. Além da interligação do detector de ataque com ferramentas de mitigação. Ainda, a expansão do escopo dos ataques DoS/DDoS e análise de suas características específicas para cada tipo de ataque e suas ferramentas. Desta forma, explorando ainda mais o uso de HPCs para apoiar o desenvolvimento de contramedidas de segurança.

Ainda, pode-se estender o número de ferramentas de ataque de DoS/DDoS conhecidas e realizar os estudos com um número maior de HPCs, uma vez que muitos contadores de desempenho são exclusivos do processador, tornando a utilização específica e distinta das disponíveis na ferramenta. Além de investigar possíveis usabilidade na aplicabilidade da metodologia apresentada em ambiente com OS *Microsoft Windows* (MS). O referido trabalho e a metodologia apresentada utiliza OS baseado em *Linux*, justamente com utilitários e ferramentas do próprio sistema, o que facilita a utilização nos ambientes com as mesmas características, mas dificulta a utilização direta em ambiente com sistema operacionais MS. Portanto, necessitando de algumas modificações e adaptações nesses ambientes.

Por fim, uma investigação profunda da correlação direta e indireta das variáveis coletadas no sistema pode auxiliar na extração características comportamentais e reduzir o uso de variáveis que são correlacionadas. Assim, capturando apenas as características essenciais e mais influentes de cada subgrupo. Além de utilizar mais utilitários e ferramentas de diagnóstico, implementando o *script* desenvolvido para conseguir capturar ainda mais características comportamentais nas diversas situações como ferramentas de análise da rede, ferramentas de monitoramento de rede, como o Zabbix[1] e análises do trafego e ferramentas de monitoramento da aplicação web, como o GoAccess[2], fazendo a interligação da ferramenta desenvolvida com diversas outras.

[1] <https://www.zabbix.com/>
[2] <https://goaccess.io/get-started>

REFERÊNCIAS

AAMIR, M.; RIZVI, S. S. H.; HASHMANI, M. A.; ZUBAIR, M.; AHMAD, J. Machine learning classification of port scanning and ddos attacks: a comparative analysis. *Mehran University Research Journal Of Engineering & Technology*, v. 40, n. 1, p. 215–229, 2021.

ABBAS, S. A.; ALMHANNA, M. S. Distributed denial of service attacks detection system by machine learning based on dimensionality reduction. In: IOP PUBLISHING. *Journal of Physics: Conference Series*. [S.l.], 2021. v. 1804, n. 1, p. 012136.

ABDULHAMMED, R.; MUSAFER, H.; ALESSA, A.; FAEZIPOUR, M.; ABUZNEID, A. Features dimensionality reduction approaches for machine learning based network intrusion detection. *Electronics*, Multidisciplinary Digital Publishing Institute, v. 8, n. 3, p. 322, 2019.

ABLIZ, M. Internet denial of service attacks and defense mechanisms. *University of Pittsburgh, Department of Computer Science, Technical Report*, p. 1–50, 2011.

ADDEPALLI, S. K.; KARRI, R.; JYOTHI, V. *System, method and computer-accessible medium for network intrusion detection*. [S.l.]: Google Patents, 2017. US Patent App. 15/400,568.

ADESHINA, Q. A.; SAHA, B. N. Using machine learning to predict distributed denial-of-service (ddos) attack. *AIJR Proceedings*, AIJR Publisher, p. 159–169, 2021.

AJISH, S.; KUMAR, K. A. Security and performance enhancement of fingerprint biometric template using symmetric hashing. *Computers & Security*, Elsevier, v. 90, p. 101714, 2020.

AKANJI, O. S.; ABISOYE, O. A.; BASHIR, S. A.; OJERINDE, O. A. A survey on slow ddos attack detection techniques. 2020.

ALAM, M.; BHATTACHARYA, S.; MUKHOPADHYAY, D.; BHATTACHARYA, S. Performance counters to rescue: A machine learning based safeguard against micro-architectural side-channel-attacks. *IACR Cryptol. ePrint Arch.*, v. 2017, p. 564, 2017.

ALGHAMDI, S.; WIN, K. T.; VLAHU-GJORGIEVSKA, E. Information security governance challenges and critical success factors: Systematic review. *Computers & Security*, Elsevier, v. 99, p. 102030, 2020.

AMBROISE, C.; MCLACHLAN, G. J. Selection bias in gene extraction on the basis of microarray gene-expression data. *Proceedings of the national academy of sciences*, National Acad Sciences, v. 99, n. 10, p. 6562–6566, 2002.

ARFEEN, A.; KHAN, Z. A.; UDDIN, R.; AHSAN, U. Toward accurate and intelligent detection of malware. *Concurrency and Computation: Practice and Experience*, Wiley Online Library, p. e6652, 2021.

ASLAN, Ö. A.; SAMET, R. A comprehensive review on malware detection approaches. *IEEE Access*, IEEE, v. 8, p. 6249–6271, 2020.

AWAN, M. J.; FAROOQ, U.; BABAR, H. M. A.; YASIN, A.; NOBANEE, H.; HUSSAIN, M.; HAKEEM, O.; ZAIN, A. M. Real-time ddos attack detection system using big data approach. *Sustainability*, Multidisciplinary Digital Publishing Institute, v. 13, n. 19, p. 10743, 2021.

AYTAÇ, T.; AYDIN, M. A.; ZAIM, A. H. Detection ddos attacks using machine learning methods. İstanbul Üniversitesi, 2020.

BAARS, H.; HINTZBERGEN, J.; SMULDERS, A.; HINTZBERGEN, K. *Foundations of information security based on ISO27001 and ISO27002*. [S.l.]: Van Haren, 2015.

BAHADOR, M. B.; ABADI, M.; TAJODDIN, A. Hpcmalhunter: Behavioral malware detection using hardware performance counters and singular value decomposition. In: IEEE. *2014 4th International Conference on Computer and Knowledge Engineering (ICCKE)*. [S.l.], 2014. p. 703–708.

BASKAR, M.; RAMKUMAR, J.; KARTHIKEYAN, C.; ANBARASU, V.; BALAJI, A.; ARULANANTH, T. Low rate ddos mitigation using real-time multi threshold traffic monitoring system. *Journal of Ambient Intelligence and Humanized Computing*, Springer, p. 1–9, 2021.

BAŞKAYA, D.; SAMET, R. Ddos attacks detection by using machine learning methods on online systems. In: IEEE. *2020 5th International Conference on Computer Science and Engineering (UBMK)*. [S.l.], 2020. p. 52–57.

BATCHU, R. K.; SEETHA, H. A generalized machine learning model for ddos attacks detection using hybrid feature selection and hyperparameter tuning. *Computer Networks*, Elsevier, p. 108498, 2021.

BATENI, H.; SAEIDI, P. The effect of information quality integrity on information security risk management. 2019.

BAZM, M.-M.; SAUTEREAU, T.; LACOSTE, M.; SUDHOLT, M.; MENAUD, J.-M. Cache-based side-channel attacks detection through intel cache monitoring technology and hardware performance counters. In: IEEE. *2018 Third International Conference on Fog and Mobile Edge Computing (FMEC)*. [S.l.], 2018. p. 7–12.

BEHAL, S.; KUMAR, K. Trends in validation of ddos research. *Procedia Computer Science*, Elsevier, v. 85, p. 7–15, 2016.

BÉNARD, C.; BIAU, G.; VEIGA, S.; SCORNET, E. Interpretable random forests via rule extraction. In: PMLR. *International Conference on Artificial Intelligence and Statistics*. [S.l.], 2021. p. 937–945.

BENNETT, J. O.; BRIGGS, W. L.; BADALAMENTI, A. *Using and understanding mathematics: A quantitative reasoning approach*. [S.l.]: Pearson Addison Wesley Reading, 2008.

BHOSALE, K. S.; NENOVA, M.; ILIEV, G. The distributed denial of service attacks (ddos) prevention mechanisms on application layer. In: IEEE. *2017 13th International Conference on Advanced Technologies, Systems and Services in Telecommunications (TELSIKS)*. [S.l.], 2017. p. 136–139.

BHUYAN, M. H.; KASHYAP, H. J.; BHATTACHARYYA, D. K.; KALITA, J. K. Detecting distributed denial of service attacks: methods, tools and future directions. *The Computer Journal*, Oxford University Press, v. 57, n. 4, p. 537–556, 2014.

BROOKS, R. R.; YU, L.; OAKLEY, J.; TUSING, N. et al. Distributed denial of service (ddos): A history. *IEEE Annals of the History of Computing*, IEEE, 2021.

CHANDOLA, V.; BANERJEE, A.; KUMAR, V. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, ACM New York, NY, USA, v. 41, n. 3, p. 1–58, 2009.

CHEN, R.-C.; CHENG, K.-F.; HSIEH, C.-F. Using rough set and support vector machine for network intrusion detection. *arXiv preprint arXiv:1004.0567*, 2010.

CHEN, S.; WANG, R.; WANG, X.; ZHANG, K. Side-channel leaks in web applications: A reality today, a challenge tomorrow. In: IEEE. *2010 IEEE Symposium on Security and Privacy*. [S.l.], 2010. p. 191–206.

CHEN, Y.; ZHENG, W.; LI, W.; HUANG, Y. Large group activity security risk assessment and risk early warning based on random forest algorithm. *Pattern Recognition Letters*, Elsevier, v. 144, p. 1–5, 2021.

CHIAPPETTA, M.; SAVAS, E.; YILMAZ, C. Real time detection of cache-based side-channel attacks using hardware performance counters. *Applied Soft Computing*, Elsevier, v. 49, p. 1162–1174, 2016.

CHO, J.-H.; SHARMA, D. P.; ALAVIZADEH, H.; YOON, S.; BEN-ASHER, N.; MOORE, T. J.; KIM, D. S.; LIM, H.; NELSON, F. F. Toward proactive, adaptive defense: A survey on moving target defense. *IEEE Communications Surveys & Tutorials*, IEEE, v. 22, n. 1, p. 709–745, 2020.

CIRILLO, M.; MAURO, M. D.; MATTA, V.; TAMBASCO, M. Application-layer ddos attacks with multiple emulation dictionaries. In: IEEE. *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. [S.l.], 2021. p. 2610–2614.

CLARK, D. D. *Designs for an Internet*. [S.l.]: Draft version, 2017.

COMPTON, R. A. *Distributed denial-of-service attack mitigation with reduced latency*. [S.l.]: Google Patents, 2019. US Patent App. 15/880,522.

DAHIYA, A.; GUPTA, B. B. A reputation score policy and bayesian game theory based incentivized mechanism for ddos attacks mitigation and cyber defense. *Future Generation Computer Systems*, Elsevier, v. 117, p. 193–204, 2021.

DAS, S.; WERNER, J.; ANTONAKAKIS, M.; POLYCHRONAKIS, M.; MONROSE, F. Sok: The challenges, pitfalls, and perils of using hardware performance counters for security. In: IEEE. *2019 IEEE Symposium on Security and Privacy (SP)*. [S.l.], 2019. p. 20–38.

DAYANANDAM, G.; RAO, T.; BABU, D. B.; DURGA, S. N. Ddos attacks—analysis and prevention. In: *Innovations in Computer Science and Engineering*. [S.l.]: Springer, 2019. p. 1–10.

DEKA, R. K.; BHATTACHARYYA, D. K.; KALITA, J. K. Ddos attacks: Tools, mitigation approaches, and probable impact on private cloud environment. *Big Data Analytics for Internet of Things*, Wiley Online Library, p. 285–319, 2021.

DEMOULIN, H. M.; PEDISICH, I.; VASILAKIS, N.; LIU, V.; LOO, B. T.; PHAN, L. T. X. Detecting asymmetric application-layer denial-of-service attacks in-flight with finelame. In: *2019 {USENIX} Annual Technical Conference ({USENIX}{ATC} 19)*. [S.l.: s.n.], 2019. p. 693–708.

DHAMOR, T.; BHAT, S.; THENMALAR, S. Dynamic approaches for detection of ddos threats using machine learning. *Annals of the Romanian Society for Cell Biology*, p. 13663–13673, 2021.

DIESCH, R.; PFAFF, M.; KRCMAR, H. A comprehensive model of information security factors for decision-makers. *Computers & Security*, Elsevier, v. 92, p. 101747, 2020.

DIMAKOPOULOU, M.; ERANIAN, S.; KOZIRIS, N.; BAMBOS, N. Reliable and efficient performance monitoring in linux. In: IEEE. *SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. [S.l.], 2016. p. 396–408.

DOULIGERIS, C.; MITROKOTSA, A. Ddos attacks and defense mechanisms: classification and state-of-the-art. *Computer Networks*, Elsevier, v. 44, n. 5, p. 643–666, 2004.

ELNAGGAR, R.; CHAKRABARTY, K.; TAHOORI, M. B. Run-time hardware trojan detection using performance counters. In: IEEE. *2017 IEEE International Test Conference (ITC)*. [S.l.], 2017. p. 1–10.

EZENWE, A.; FUREY, E.; CURRAN, K. Mitigating denial of service attacks with load balancing. *Journal of Robotics and Control (JRC)*, v. 1, n. 4, p. 129–135, 2020.

FADHLILLAH, A.; KARNA, N.; IRAWAN, A. Ids performance analysis using anomaly-based detection method for dos attack. In: IEEE. *2020 IEEE International Conference on Internet of Things and Intelligence System (IoTaIS)*. [S.l.], 2021. p. 18–22.

FILHO, F. S. d. L.; SILVEIRA, F. A.; JUNIOR, A. de M. B.; VARGAS-SOLAR, G.; SILVEIRA, L. F. Smart detection: an online approach for dos/ddos attack detection using machine learning. *Security and Communication Networks*, Hindawi, v. 2019, 2019.

FOREMAN, J. C. A survey of cyber security countermeasures using hardware performance counters. *arXiv preprint arXiv:1807.10868*, 2018.

FORTINET. *A América Latina sofreu mais de 41 bilhões de tentativas de ataques cibernéticos em 2020. Blog.* 2021. <https://www.fortinet.com/br/corporate/about-us/newsroom/press-releases/2021/latin-america-suffered-more-than-41-billion-cyberattack-attempts-in-2020>.

FUCHSBERGER, A. Intrusion detection systems and intrusion prevention systems. *Information Security Technical Report*, Elsevier Advanced Technology, v. 10, n. 3, p. 134–139, 2005.

FURFARO, A.; PACE, P.; PARISE, A. Facing ddos bandwidth flooding attacks. *Simulation Modelling Practice and Theory*, Elsevier, v. 98, p. 101984, 2020.

GARCIA, N.; ALCANIZ, T.; GONZÁLEZ-VIDAL, A.; BERNABE, J. B.; RIVERA, D.; SKARMETA, A. Distributed real-time slowdos attacks detection over encrypted traffic using artificial intelligence. *Journal of Network and Computer Applications*, Elsevier, v. 173, p. 102871, 2021.

GARCIA-SERRANO, A. Anomaly detection for malware identification using hardware performance counters. *arXiv preprint arXiv:1508.07482*, 2015.

GE, Q.; YAROM, Y.; COCK, D.; HEISER, G. A survey of microarchitectural timing attacks and countermeasures on contemporary hardware. *Journal of Cryptographic Engineering*, Springer, v. 8, n. 1, p. 1–27, 2018.

GIBSON, D. *CompTIA Security+: Get Certified Get Ahead: SY0-501 Study Guide*. [S.l.]: YCDA, LLC, 2017.

GOGOI, P.; BHATTACHARYYA, D.; BORAH, B.; KALITA, J. K. A survey of outlier detection methods in network anomaly identification. *The Computer Journal*, OUP, v. 54, n. 4, p. 570–588, 2011.

GRABOVSKY, S.; CIKA, P.; ZEMAN, V.; CLUPEK, V.; SVEHLAK, M.; KLIMES, J. Denial of service attack generator in apache jmeter. In: IEEE. *2018 10th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*. [S.l.], 2018. p. 1–4.

GRACIOLI, G.; FRÖHLICH, A. A. An embedded operating system api for monitoring hardware events in multicore processors. In: CITESEER. *Workshop on hardware-support for parallel program correctness—IEEE Micro*. [S.l.], 2011.

GRAKOVSKI, A. Using machine learning for dos attacks diagnostics. In: SPRINGER NATURE. *Reliability and Statistics in Transportation and Communication: Selected Papers from the 20th International Conference on Reliability and Statistics in Transportation and Communication, RelStat2020, 14-17 October 2020, Riga, Latvia*. [S.l.], 2021. v. 195, p. 45.

GREGG, B. *BPF Performance Tools*. [S.l.]: Addison-Wesley Professional, 2019.

GREGG, B.; MAURO, J. *DTrace: Dynamic Tracing in Oracle Solaris, Mac OS X, and FreeBSD*. [S.l.]: Prentice Hall Professional, 2011.

GŘIVNA, T.; DRÁPAL, J. Attacks on the confidentiality, integrity and availability of data and computer systems in the criminal case law of the czech republic. *Digital Investigation*, Elsevier, v. 28, p. 1–13, 2019.

GUPTA, B. B.; DAHIYA, A. *Distributed Denial of Service (DDoS) Attacks: Classification, Attacks, Challenges, and Countermeasures*. [S.l.]: CRC Press, 2021.

HASLWANTER, T. An introduction to statistics with python. *With applications in the life sciences. Switzerland: Springer International Publishing*, Springer, 2016.

HE, Q.; WANG, C.; CUI, G.; LI, B.; ZHOU, R.; ZHOU, Q.; XIANG, Y.; JIN, H.; YANG, Y. A game-theoretical approach for mitigatingedge ddos attack. *IEEE Transactions on Dependable and Secure Computing*, IEEE, 2021.

HINTZBERGEN, J.; HINTZBERGEN, K.; SMULDERS, A.; BAARS, H. *Fundamentos de Segurança da Informação: com base na ISO 27001 e na ISO 27002*. [S.l.]: Brasport, 2018.

HONG, J. B.; ENOCH, S. Y.; KIM, D. S.; NHLABATSI, A.; FETAIS, N.; KHAN, K. M. Dynamic security metrics for measuring the effectiveness of moving target defense techniques. *Computers & Security*, Elsevier, v. 79, p. 33–52, 2018.

HULL, G.; JOHN, H.; ARIEF, B. Ransomware deployment methods and analysis: views from a predictive model and human responses. *Crime Science*, Springer, v. 8, n. 1, p. 1–22, 2019.

IDHAMMAD, M.; AFDEL, K.; BELOUCH, M. Semi-supervised machine learning approach for ddos detection. *Applied Intelligence*, Springer, v. 48, n. 10, p. 3193–3208, 2018.

ILSCHE, T.; SCHÖNE, R.; BIELERT, M.; GOCHT, A.; HACKENBERG, D. lo2s—multi-core system and application performance analysis for linux. In: IEEE. *2017 IEEE International Conference on Cluster Computing (CLUSTER)*. [S.l.], 2017. p. 801–804.

INTERPOL. *INTERPOL report shows alarming rate of cyberattacks during COVID-19. Blog*. 2020. <https://www.interpol.int/News-and-Events/News/2020/INTERPOL-report-shows-alarming-rate-of-cyberattacks-during-COVID-19>.

ISMAIL, S.; HASSEN, H. R.; JUST, M.; ZANTOUT, H. A review of amplification-based distributed denial of service attacks and mitigation. *Computers & Security*, Elsevier, p. 102380, 2021.

JAAFAR, G. A.; ABDULLAH, S. M.; ISMAIL, S. Review of recent detection methods for http ddos attack. *Journal of Computer Networks and Communications*, Hindawi, v. 2019, 2019.

JING, X.; YAN, Z.; JIANG, X.; PEDRYCZ, W. Network traffic fusion and analysis against ddos flooding attacks with a novel reversible sketch. *Information Fusion*, Elsevier, v. 51, p. 100–113, 2019.

JYOTHI, V.; WANG, X.; ADDEPALLI, S. K.; KARRI, R. Brain: Behavior based adaptive intrusion detection in networks: Using hardware performance counters to detect ddos attacks. In: IEEE. *2016 29th International Conference on VLSI Design and 2016 15th International Conference on Embedded Systems (VLSID)*. [S.l.], 2016. p. 587–588.

KEIJER, J. *Automated DDoS mitigation based on known attacks using a Web Application Firewall*. Dissertação (B.S. thesis) — University of Twente, 2019.

KHAN, N.; ABDULLAH, J.; KHAN, A. S. Defending malicious script attacks using machine learning classifiers. *Wireless Communications and Mobile Computing*, Hindawi, v. 2017, 2017.

KOSMIDIS, L.; ABELLA, J.; QUINONES, E.; CAZORLA, F. J. A cache design for probabilistically analysable real-time systems. In: IEEE. *2013 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. [S.l.], 2013. p. 513–518.

KRISHNAMURTHY, P.; KARRI, R.; KHORRAMI, F. Anomaly detection in real-time multi-threaded processes using hardware performance counters. *IEEE Transactions on Information Forensics and Security*, IEEE, v. 15, p. 666–680, 2019.

KSHIRSAGAR, D.; KUMAR, S. An ontology approach for proactive detection of http flood dos attack. *International Journal of System Assurance Engineering and Management*, Springer, p. 1–8, 2021.

KUMAR, V.; KUMAR, K. et al. Classification of ddos attack tools and its handling techniques and strategy at application layer. In: IEEE. *2016 2nd International Conference on Advances in Computing, Communication, & Automation (ICACCA)(Fall)*. [S.l.], 2016. p. 1–6.

KUNKEL, J.; DOLZ, M. F. Understanding hardware and software metrics with respect to power consumption. *Sustainable Computing: Informatics and Systems*, Elsevier, v. 17, p. 43–54, 2018.

LEE, K.; KIM, J.; KWON, K. H.; HAN, Y.; KIM, S. Ddos attack detection method using cluster analysis. *Expert systems with applications*, Elsevier, v. 34, n. 3, p. 1659–1665, 2008.

LEE, S. A serverless architecture for frequency-based http request filtering against distributed denial-of-service (ddos) attacks. TechRxiv, 2021.

LENG, E. W. L.; ZWOLINSKI, M.; HALAK, B. Hardware performance counters for system reliability monitoring. In: IEEE. *2017 IEEE 2nd International Verification and Security Workshop (IVSW)*. [S.l.], 2017. p. 76–81.

LEON, R. S.; KIPERBERG, M.; ZABAG, A. A. L.; ZAIDENBERG, N. J. Hypervisor-assisted dynamic malware analysis. *Cybersecurity*, SpringerOpen, v. 4, n. 1, p. 1–14, 2021.

LI, J.; XUE, D.; WU, W.; WANG, J. Incremental learning for malware classification in small datasets. *Security and Communication Networks*, Hindawi, v. 2020, 2020.

LI, Y.; LIU, Q. A comprehensive review study of cyber-attacks and cyber security; emerging trends and recent developments. *Energy Reports*, Elsevier, 2021.

LIANG, X. *An intelligent, distributed and collaborative DDoS defense system*. Tese (Doutorado) — University of Pittsburgh, 2021.

LIBRI, A.; BARTOLINI, A.; BENINI, L. Dig: enabling out-of-band scalable high-resolution monitoring for data-center analytics, automation and control (extended). *Cluster Computing*, Springer, p. 1–12, 2021.

LINKOV, I.; ROSLYCKY, L.; TRUMP, B. D. *Resilience and hybrid threats: Security and integrity for the digital world*. [S.l.]: IOS Press, 2019. v. 55.

LIU, Z.; JIN, H.; HU, Y.-C.; BAILEY, M. Practical proactive ddos-attack mitigation via endpoint-driven in-network traffic control. *IEEE/ACM Transactions on Networking*, IEEE, v. 26, n. 4, p. 1948–1961, 2018.

LYRA, M. R. Governança da segurança da informação. *Brasília: nd*, 2015.

MAČEK, D.; MAGDALENIĆ, I.; REđEP, N. B. A systematic literature review on the application of multicriteria decision making methods for information security risk assessment. *International Journal of Safety and Security Engineering*, v. 10, n. 2, p. 161–174, 2020.

MACIEL, R.; ARAUJO, J.; DANTAS, J.; MELO, C.; GUEDES, E.; MACIEL, P. Impact of a ddos attack on computer systems: An approach based on an attack tree model. In: IEEE. *2018 Annual IEEE International Systems Conference (SysCon)*. [S.l.], 2018. p. 1–8.

MAHJABIN, T.; XIAO, Y.; SUN, G.; JIANG, W. A survey of distributed denial-of-service attack, prevention, and mitigation techniques. *International Journal of Distributed Sensor Networks*, SAGE Publications Sage UK: London, England, v. 13, n. 12, p. 1550147717741463, 2017.

MASLAN, A.; MOHAMAD, K. M. B.; FOOZY, F. B. M. Feature selection for ddos detection using classification machine learning techniques. *IAES International Journal of Artificial Intelligence*, IAES Institute of Advanced Engineering and Science, v. 9, n. 1, p. 137, 2020.

MATAM, S.; JAIN, J. *Pro Apache JMeter: web application performance testing*. [S.l.]: Apress, 2017.

MENG, W.; QIAN, C.; HAO, S.; BORGOLTE, K.; VIGNA, G.; KRUEGEL, C.; LEE, W. Rampart: Protecting web applications from cpu-exhaustion denial-of-service attacks. In: *27th {USENIX} Security Symposium ({USENIX} Security 18)*. [S.l.: s.n.], 2018. p. 393–410.

MIRCHEV, M. J.; MIRTCHEV, S. T. System for ddos attack mitigation by discovering the attack vectors through statistical traffic analysis. *International Journal of Information and Computer Security*, Inderscience Publishers (IEL), v. 13, n. 3-4, p. 309–321, 2020.

MISHRA, S.; SHARMA, S. K.; ALOWAIDI, M. A. Analysis of security issues of cloud-based web applications. *Journal of Ambient Intelligence and Humanized Computing*, Springer, v. 12, n. 7, p. 7051–7062, 2021.

MORETTIN, P. A.; BUSSAB, W. O. *Estatística básica*. [S.l.]: Saraiva Educação SA, 2017.

NASCIMENTO, P. P. D.; COLARES, I. F.; MACIEL, R.; SILVA, H. C. D.; MACIEL, P. Prediction, detection, and mitigation of ddos attacks using hpcs: Design for a safer adaptive infrastructure. In: *Handbook of Research on Cyber Crime and Information Privacy*. [S.l.]: IGI Global, 2021. p. 523–538.

NASCIMENTO, P. P. do; PEREIRA, P.; MIALARET, J. M.; FERREIRA, I.; MACIEL, P. A methodology for selecting hardware performance counters for supporting non-intrusive diagnostic of flood ddos attacks on web servers. *Computers & Security*, Elsevier, p. 102434, 2021.

NDIBWILE, J. D.; GOVARDHAN, A.; OKADA, K.; KADOBAYASHI, Y. Web server protection against application layer ddos attacks using machine learning and traffic

authentication. In: IEEE. *2015 IEEE 39th Annual Computer Software and Applications Conference*. [S.l.], 2015. v. 3, p. 261–267.

ODUSAMI, M.; MISRA, S.; ABAYOMI-ALLI, O.; ABAYOMI-ALLI, A.; FERNANDEZ-SANZ, L. A survey and meta-analysis of application-layer distributed denial-of-service attack. *International Journal of Communication Systems*, Wiley Online Library, v. 33, n. 18, p. e4603, 2020.

OIKONOMOU, G.; MIRKOVIC, J. Modeling human behavior for defense against flash-crowd attacks. In: IEEE. *2009 IEEE International Conference on Communications*. [S.l.], 2009. p. 1–6.

PAN, Z.; SHELDON, J.; SUDUSINGHE, C.; CHARLES, S.; MISHRA, P. Hardware-assisted malware detection using machine learning. In: *Design Automation and Test in Europe (DATE)*. [S.l.: s.n.], 2021.

PANDE, S.; KHAMPARIA, A.; GUPTA, D.; THANH, D. N. Ddos detection using machine learning technique. In: *Recent Studies on Computational Intelligence*. [S.l.]: Springer, 2021. p. 59–68.

PARASYRIS, K.; TZIANTZOULIS, G.; ANTONOPOULOS, C. D.; BELLAS, N. Gemfi: A fault injection tool for studying the behavior of applications on unreliable substrates. In: IEEE. *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. [S.l.], 2014. p. 622–629.

PARK, K.; SAHIN, B.; CHEN, Y.; ZHAO, J.; DOWNING, E.; HU, H.; LEE, W. Identifying behavior dispatchers for malware analysis. In: *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*. [S.l.: s.n.], 2021. p. 759–773.

PEREIRA, P.; ARAUJO, J.; MELO, C.; SANTOS, V.; MACIEL, P. Analytical models for availability evaluation of edge and fog computing nodes. *The Journal of Supercomputing*, Springer, p. 1–29, 2021.

PEREIRA, P.; ARAUJO, J.; TORQUATO, M.; DANTAS, J.; MELO, C.; MACIEL, P. Stochastic performance model for web server capacity planning in fog computing. *The Journal of Supercomputing*, Springer, p. 1–25, 2020.

PIEPRZYK, J.; SADEGHIYAN, B. *Design of hashing algorithms*. [S.l.]: Springer, 1993.

QADIR, S.; QUADRI, S. Information availability: An insight into the most important attribute of information security. *Journal of Information Security*, Scientific Research Publishing, v. 7, n. 3, p. 185–194, 2016.

RANJAN, S.; SWAMINATHAN, R.; UYSAL, M.; NUCCI, A.; KNIGHTLY, E. Ddos-shield: Ddos-resilient scheduling to counter application layer attacks. *IEEE/ACM Transactions on networking*, IEEE, v. 17, n. 1, p. 26–39, 2008.

REHMAN, S. ur; KHALIQ, M.; IMTIAZ, S. I.; RASOOL, A.; SHAFIQ, M.; JAVED, A. R.; JALIL, Z.; BASHIR, A. K. Diddos: An approach for detection and identification of distributed denial of service (ddos) cyberattacks using gated recurrent units (gru). *Future Generation Computer Systems*, Elsevier, v. 118, p. 453–466, 2021.

RIOS, V. de M.; INÁCIO, P. R.; MAGONI, D.; FREIRE, M. M. Detection of reduction-of-quality ddos attacks using fuzzy logic and machine learning algorithms. *Computer Networks*, Elsevier, v. 186, p. 107792, 2021.

ROBINSON, R. R.; THOMAS, C. Low rate multi-vector ddos attack detection using information gain based feature selection. In: *Computer Networks, Big Data and IoT*. [S.l.]: Springer, 2021. p. 685–696.

ROHAN, A.; BASU, K.; KARRI, R. Can monitoring system state+ counting custom instruction sequences aid malware detection? In: IEEE. *2019 IEEE 28th Asian Test Symposium (ATS)*. [S.l.], 2019. p. 61–615.

SABAHI, F.; MOVAGHAR, A. Intrusion detection: A survey. In: IEEE. *2008 Third International Conference on Systems and Networks Communications*. [S.l.], 2008. p. 23–26.

SABOOR, A.; AKHLAQ, M.; ASLAM, B. Experimental evaluation of snort against ddos attacks under different hardware configurations. In: IEEE. *2013 2nd National Conference on Information Assurance (NCIA)*. [S.l.], 2013. p. 31–37.

SAEED, A. A.; JAMEEL, N. G. M. Intelligent feature selection using particle swarm optimization algorithm with a decision tree for ddos attack detection. *International Journal of Advances in Intelligent Informatics*, v. 7, n. 1, p. 37–48, 2021.

SAFERNET. *Indicadores da Central Nacional de Denúncias de Crimes Cibernéticos. Blog*. 2021. <https://helpline.org.br/indicadores/pt/>.

SANDHYA, G. *DEEP LEARNING BASED APPROACH FOR TIME SERIES FORECASTING WITH APPLICATION TO DDoS DETECTION*. Tese (Doutorado) — Andhra University, 2019.

SENGUPTA, S.; CHOWDHARY, A.; SABUR, A.; ALSHAMRANI, A.; HUANG, D.; KAMBHAMPATI, S. A survey of moving target defenses for network security. *IEEE Communications Surveys & Tutorials*, IEEE, 2020.

SHARAFALDIN, I.; LASHKARI, A. H.; GHORBANI, A. A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: *ICISSP*. [S.l.: s.n.], 2018. p. 108–116.

SHARAFALDIN, I.; LASHKARI, A. H.; HAKAK, S.; GHORBANI, A. A. Developing realistic distributed denial of service (ddos) attack dataset and taxonomy. In: IEEE. *2019 International Carnahan Conference on Security Technology (ICCST)*. [S.l.], 2019. p. 1–8.

SHOREY, T.; SUBBAIAH, D.; GOYAL, A.; SAKXENA, A.; MISHRA, A. K. Performance comparison and analysis of slowloris, goldeneye and xerxes ddos attack tools. In: IEEE. *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. [S.l.], 2018. p. 318–322.

SINGH, D. Towards data privacy and security framework in big data governance. *International Journal of Software Engineering and Computer Systems*, v. 6, n. 1, p. 41–51, 2020.

SINGH, K. J.; DE, T. Ddos attack detection and mitigation technique based on http count and verification using captcha. In: IEEE. *2015 International Conference on Computational Intelligence and Networks*. [S.l.], 2015. p. 196–197.

SINGH, R.; RAM, M. *Distributed Denial of Service Attacks: Concepts, Mathematical and Cryptographic Solutions*. [S.l.]: Walter de Gruyter GmbH & Co KG, 2021. v. 6.

SINHA, M.; BHATTACHARYYA, P.; ROUT, S. S.; PRAKRIYA, N. B.; DEB, S. Securing an accelerator-rich system from flooding-based denial-of-service attacks. *IEEE Transactions on Emerging Topics in Computing*, IEEE, 2021.

SONG, H.; FINK, G. A.; JESCHKE, S. *Security and privacy in cyber-physical systems: Foundations, principles, and applications*. [S.l.]: John Wiley & Sons, 2021.

SONICWALL. *Relatório de Ameaças Cibernéticas da SonicWall 2021. Blog*. 2021. <https://www.sonicwall.com/pt-br/2021-cyber-threat-report/>.

SREENIVASARAO, S. Application layer ddos attack detection and defense methods. In: SPRINGER. *International Conference on Emerging Trends and Technologies on Intelligent Systems*. [S.l.], 2021. p. 1–12.

SREERAM, I.; VUPPALA, V. P. K. Http flood attack detection in application layer using machine learning metrics and bio inspired bat algorithm. *Applied computing and informatics*, Elsevier, v. 15, n. 1, p. 59–66, 2019.

STANTON, J.; CALDERA, C.; ISAAC, A.; STAM, K.; MARCINKOWSKI, S. Behavioral information security: Defining the criterion space. SocArXiv, 2021.

SYMANTEC. *2019 Internet Security Threat Report. Blog*. 2019. <https://docs.broadcom.com/doc/istr-24-2019-en>.

TAN, S. C.; TING, K. M.; LIU, T. F. Fast anomaly detection for streaming data. In: *Twenty-Second International Joint Conference on Artificial Intelligence*. [S.l.: s.n.], 2011.

TANDON, R. A survey of distributed denial of service attacks and defenses. *arXiv preprint arXiv:2008.01345*, 2020.

TANDON, R.; PALIA, A.; RAMANI, J.; PAULSEN, B.; BARTLETT, G.; MIRKOVIC, J. Defending web servers against flash crowd attacks. In: IEEE. *2019 IEEE 27th International Conference on Network Protocols (ICNP)*. [S.l.], 2019. p. 1–2.

TESSELE, B. et al. Emprego do perf na medição e análise do tempo de resposta de tarefas no linux. 2019.

THANDEESWARAN, R.; PERUMAL, T.; MA, K.; JEYANTHI, N. *Managing Security Services in Heterogenous Networks: Confidentiality, Integrity, Availability, Authentication, and Access Control*. [S.l.]: CRC Press, 2020.

TIWARI, D.; SINGH, A.; PRABHAKAR, A. Performance analysis of aes, rsa and hashing algorithm using web technology. In: *Computing Algorithms with Applications in Engineering*. [S.l.]: Springer, Singapore, 2020. p. 413–418.

TORRES, G.; LIU, C. Can data-only exploits be detected at runtime using hardware events? a case study of the heartbleed vulnerability. In: *Proceedings of the Hardware and Architectural Support for Security and Privacy 2016*. [S.l.: s.n.], 2016. p. 1–7.

TUAN, T. A.; LONG, H. V.; SON, L. H.; KUMAR, R.; PRIYADARSHINI, I.; SON, N. T. K. Performance evaluation of botnet ddos attack detection using machine learning. *Evolutionary Intelligence*, Springer, v. 13, n. 2, p. 283–294, 2020.

VEDULA, V.; LAMA, P.; BOPPANA, R. V.; TREJO, L. A. On the detection of low-rate denial of service attacks at transport and application layers. *Electronics*, Multidisciplinary Digital Publishing Institute, v. 10, n. 17, p. 2105, 2021.

VEIGA, A. D.; ASTAKHOVA, L. V.; BOTHA, A.; HERSELMAN, M. Defining organisational information security culture—perspectives from academia and industry. *Computers & Security*, Elsevier, v. 92, p. 101713, 2020.

WANG, X.; KARRI, R. Reusing hardware performance counters to detect and identify kernel control-flow modifying rootkits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, IEEE, v. 35, n. 3, p. 485–498, 2015.

WOO, L. L. Hardware performance counters (hpcs) for anomaly detection. In: *Hardware Supply Chain Security*. [S.l.]: Springer, 2021. p. 147–165.

XIAO, P.; QU, W.; QI, H.; LI, Z. Detecting ddos attacks against data center with correlation analysis. *Computer Communications*, Elsevier, v. 67, p. 66–74, 2015.

XING, G.; CHEN, J.; HOU, R.; ZHOU, L.; DONG, M.; ZENG, D.; LUO, J.; MA, M. Isolation forest-based mechanism to defend against interest flooding attacks in named data networking. *IEEE Communications Magazine*, IEEE, v. 59, n. 3, p. 98–103, 2021.

YEE, C. K.; ZOLKIPLI, M. F. Review on confidentiality, integrity and availability in information security. *Journal of ICT in Education*, v. 8, n. 2, p. 34–42, 2021.

YOUM, S.; PARK, S.; SHIN, K.-S. Ddos attack analysis based on decision tree considering importance. In: THE KOREA INSTITUTE OF INFORMATION AND COMMUCATION ENGINEERING. *Proceedings of the Korean Institute of Information and Commucation Sciences Conference*. [S.l.], 2021. p. 652–654.

YUAN, L.; XING, W.; CHEN, H.; ZANG, B. Security breaches as pmu deviation: Detecting and identifying security attacks using performance counters. 2011.

ZARGAR, S. T.; JOSHI, J.; TIPPER, D. A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks. *IEEE communications surveys & tutorials*, IEEE, v. 15, n. 4, p. 2046–2069, 2013.

ZHANG, C. Impact of defending strategy decision on ddos attack. *Complexity*, Hindawi, v. 2021, 2021.

ZHANG, M.; LI, G.; WANG, S.; LIU, C.; CHEN, A.; HU, H.; GU, G.; LI, Q.; XU, M.; WU, J. Poseidon: Mitigating volumetric ddos attacks with programmable switches. In: *the 27th Network and Distributed System Security Symposium (NDSS 2020)*. [S.l.: s.n.], 2020.

ZHOU, Q.; PEZAROS, D. Evaluation of machine learning classifiers for zero-day intrusion detection–an analysis on cic-aws-2018 dataset. *arXiv preprint arXiv:1905.03685*, 2019.

ZHOU, Y.; CHENG, G.; JIANG, S.; ZHAO, Y.; CHEN, Z. Cost-effective moving target defense against ddos attacks using trilateral game and multi-objective markov decision processes. *Computers & Security*, Elsevier, v. 97, p. 101976, 2020.

APÊNDICE A – SCRIPT DE DIAGNÓSTICO

Aqui mostraremos o *script* de diagnostico desenvolvido para realizar a coleta e o monitoramento do sistema. A parte 1 é a atribuições de coleta do diagnóstico. A parte 2 é a realização das coletas simultânea das ferramentas utilizadas, coleta dos contadores de HPCs e variáveis de utilização dos recursos no sistema. A parte 3 adiciona o cabeçalho aos arquivos de diagnostico da rede e da RAM. A parte 4 é a realização do tratamento dos dados coletados, utilizando utilitários de manipulação de texto para realizar a limpeza dos dados. A parte 5 é anião de todos os arquivos que foram coletados e redirecionados separadamente, para um único arquivo contendo todas as informações separadas e organizadas em colunas e linhas.

Código do *Script* de Diagnóstico do Sistema

```
#!/bin/bash

#Parte 1# [Atribuindo as condicoes de coleta]

echo 'Iniciando Coleta dos Dados...'

echo 'Informe o Tempo da Captura /s: (Ex.60 para 1m):'
read y

echo 'Informe o Intervalo de Captura dos Dados de HPCs/RAM/CPU/
   REDE /s: (Ex.10 para 10s):'
read h
x=$(expr $h \* 1000)
z=$(expr $y / 10)

echo 'Coletando os Dados... (HPCs/RAM/CPU/REDE)'
echo '...'

#Parte 2# Contadores de [Hardware Events] & [Hardware Events
   Cache] 4/4 (25 conts)
# Variaveis de utilizacao de recursos [CPU, RAM, REDE]

perf stat -o contador1.txt -e branch-instructions,branch-misses
   ,bus-cycles,cache-misses,cache-references,cpu-cycles,
```

```
    instructions,ref-cycles -I $x -a -g -- sleep $y & perf stat
    -o contador2.txt -e L1-dcache-load-misses,L1-dcache-loads,L1
    -dcache-stores,L1-icache-load-misses,LLC-load-misses,LLC-
    loads -I $x -a -g -- sleep $y & perf stat -o contador3.txt -
    e LLC-store-misses,LLC-stores,branch-load-misses,branch-
    loads,dTLB-load-misses,dTLB-loads -I $x -a -g -- sleep $y &
    perf stat -o contador4.txt -e dTLB-store-misses,dTLB-stores,
    iTLB-load-misses,iTLB-loads,node-load-misses -I $x -a -g --
    sleep $y & free -m -s $h -c $z | sed '1 d' | awk '{$1="";
    print}' | sed "N;s/\n//" | sed -u '/a/d' | awk '{print $1,$2
    ,$3,$4,$5,$6,$7,$8,$9}' > RAM.txt & sar $h $z | sed '1,2 d'|
     sed '$ d' | awk '{$1="";$2=""; print}' | sed 's/\,/\./g' |
    awk '{print $1,$2,$3,$4,$5,$6}' > DadosCPU.txt & ifstat -i
    eno1 $h $z | sed -u '/eno1/d' | sed -u '/KB/d' | awk '{print
     $1,$2}' > REDE.txt

#Parte 3#  [Adicionando Header]

sed -e '1i\' -e 'Total Used Free Shared Buff/Cache Avaliable
    TotalSwp UsedSwp FreeSwp' RAM.txt > DadosRAM.txt & sed -e '1
    i\' -e 'KBpsIn KBpsOut' REDE.txt > DadosREDE.txt


#Parte 4# [Tratamento dos Dados de HPCs]

sed -e '1,3d' < contador1.txt | sed -u '/time/d' | awk '{print
    $1}' | cut -s -d"." -f1 | awk '!x[$0]++' | tr -s '[:space:]'
     > time.txt | sed -e '1,3d' < contador1.txt | sed -u '/time/
    d' |  awk '{print $2}' | sed -e 'N;N;N;N;N;N;s/\n/ /g' |
    tr -s '[:space:]' > 1cont.txt | sed -e '1,3d' < contador2.
    txt | sed -u '/time/d' |  awk '{print $2}' | sed -e 'N;N;N
    ;N;s/\n/ /g' | tr -s '[:space:]' > 2cont.txt |sed -e '1,3d'
    < contador3.txt | sed -u '/time/d' |  awk '{print $2}' | sed
     -e 'N;N;N;N;s/\n/ /g' | tr -s '[:space:]' > 3cont.txt |
    sed -e '1,3d' < contador4.txt | sed -u '/time/d' |  awk '{
    print $2}' | sed -e 'N;N;N;s/\n/ /g' |tr -s '[:space:]' >
    4cont.txt && paste -d " " time.txt 1cont.txt 2cont.txt 3cont
    .txt 4cont.txt > allcont.txt && sed -e '1i\' -e 'Time branch
    -instructions branch-misses bus-cycles cache-misses cache-
```

```
    references cpu-cycles instructions ref-cycles L1-dcache-load
    -misses L1-dcache-loads L1-dcache-stores L1-icache-load-
    misses LLC-load-misses LLC-loads LLC-store-misses LLC-stores
     branch-load-misses branch-loads dTLB-load-misses dTLB-loads
     dTLB-store-misses dTLB-stores iTLB-load-misses iTLB-loads
    node-load-misses' allcont.txt > DadosHPC.txt

#Parte 5# [Redirecionamento dos arquivos]

paste -d " " DadosHPC.txt DadosCPU.txt DadosRAM.txt DadosREDE.
    txt > AllColetores.txt

echo 'Captura dos Dados Finalizada!'
```

APÊNDICE B – INFRAESTRUTURA

Os Servidores HP ProLiant DL320e Gen8 v2[1] é um servidor de rack de pouca profundidade e fácil instalação de nivel empresarial. O servidor suporta os mais recentes processadores Intel® Xeon® série E3-1200 v3[2], slots PCIe 3.0, memória DDR3 de 1600 MHz e até 8 TB de armazenamento interno, além do HP iLO *Management Engine*. Esse equipamento é ideal para pesquisas acadêmicas e implementação de laboratórios executando experimentos e pequenas empresas que requerem um servidor compacto de baixo custo para hospedar sistemas web, versátil o suficiente para lidar com várias cargas de trabalho, incluindo serviços de TI únicos/dedicados justamente por ser de seguimento empresarial, consegue retratar um ambiente que retrata melhor a realidade.

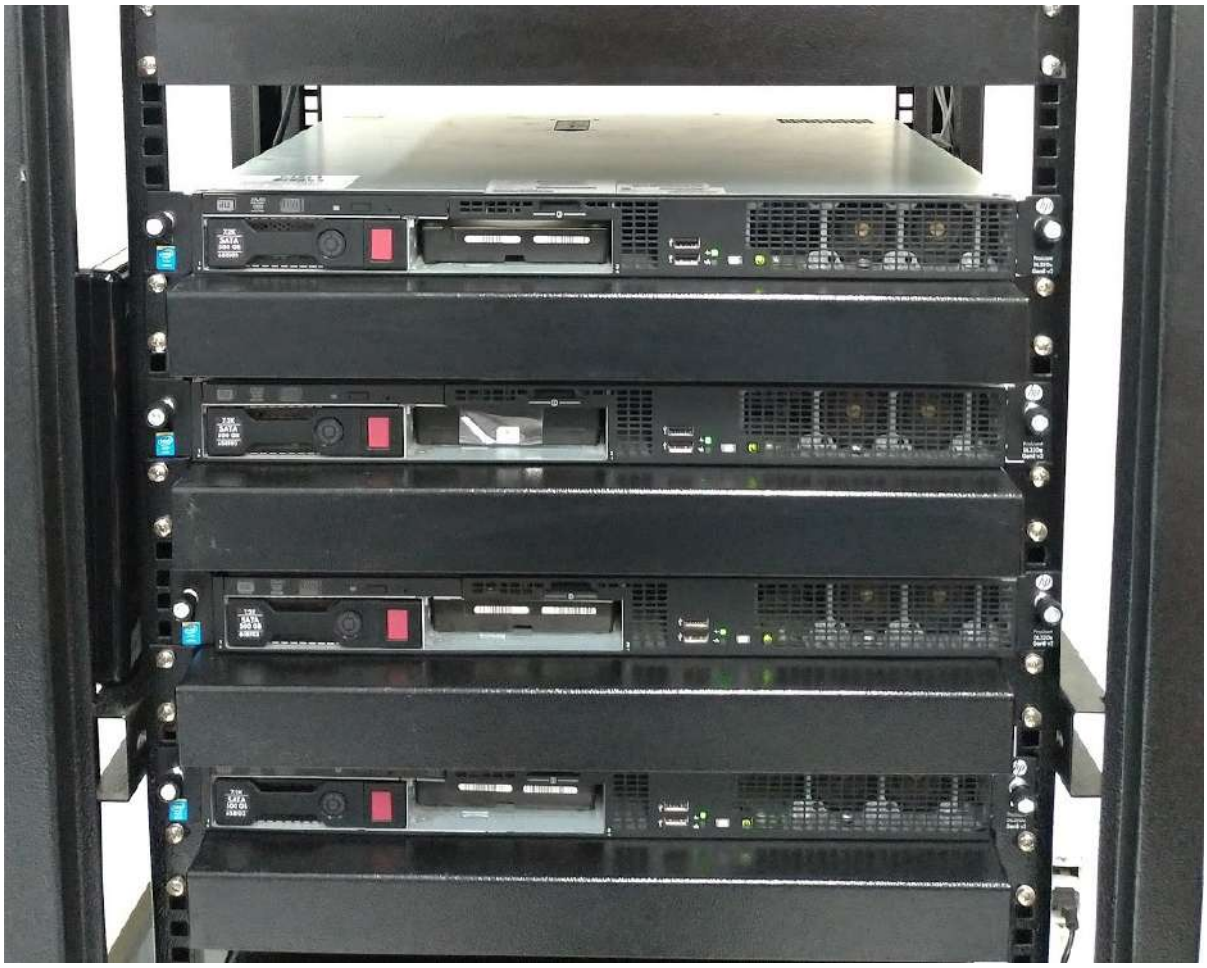


Figura 24 – Servidores

[1] <https://support.hpe.com/hpesc/public/docDisplay?docId=emr_na-c03792388>

[2] <https://ark.intel.com/content/www/br/pt/ark/products/75052/intel-xeon-processor-e3-1220-v3-8m-cache-3-10-html>

Figura 25 – Switch Gigabit TL-SG1016

O Switch TP-Link GbE de 16 portas TL-SG1016[3] de montagem em *rack* fornece uma maneira fácil e pratica de instalação de uma rede GbE. Todas as 16 portas com tecnologia auto MDI/MDIX, auto-negociação de rede (10, 100 ou 1000 Mbps) ajustando automaticamente para um desempenho ideal, descartando a necessidade de configurações adicionais com tipos diferentes de cabeamento. O TL-SG1016 apresenta arquitetura de velocidade a cabo sem bloqueios com uma capacidade máxima de 32/Gbps de transferência de dados. Possui uma tabela de MAC Address de 8mil, fornecendo escalabilidade até para as redes maiores. Ele também suporta controle de fluxo 802.3x em modo *full-duplex*, e controle de fluxo de *back pressure* para o modo *half-duplex* que alivia o congestionamento do tráfego e assegura a transmissão confiável dos dados. Suas características torna-o ideal para as estações de trabalho empresariais ou em implementações em laboratórios acadêmicos.

[3] <https://www.tp-link.com/br/>

APÊNDICE C – FERRAMENTAS

Este apêndice, apresentara com mais alguns detalhes das ferramentas de DoS/DDoS utilizadas e alguns de suas especificações de utilização apresentadas neste trabalho.

Deixamos claro que toda e quaisquer ferramenta aqui utilizada tem como único propósito e objetivo o âmbito acadêmico e cientifico.

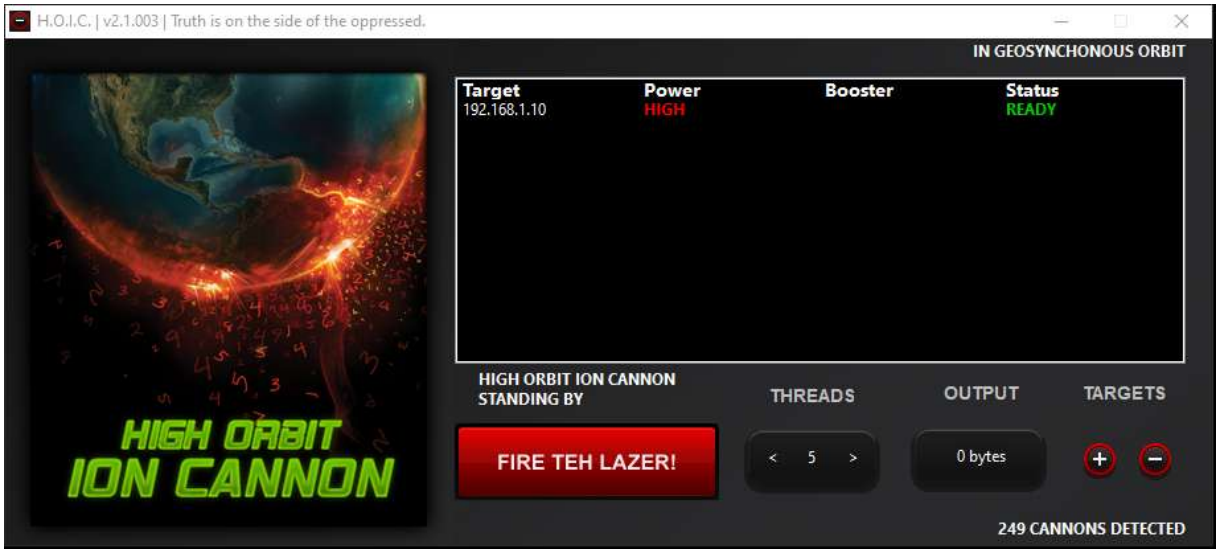


Figura 26 – Ferramenta H.O.I.C.

O *High Orbit Ion Cannon* (HOIC) é um aplicativo de estresse de rede gratuito e de código aberto desenvolvido pelo grupo hacktivista *Anonymous* inicialmente projetado para substituir o seu aplicativo antecessor o *Low Orbit Ion Cannon* (LOIC). Funções complementares adicionais realizada por *scripts* denominados *boosters*, anteriormente não disponíveis em seu antecessor, podem aumentar muito o volume do ataque além de possibilitar a personalização dos ataque para contornar os mecanismos de proteção. Como apresentado na Figura 26, a ferramenta é de fácil utilização, possibilitando usuários sem muito conhecimento utilizarem , passando como parâmetro os alvos e a quantidade de instancias para o ataque selecionando a capacidade individual com três configurações: baixo, médio e alto como visto na Figura 27.

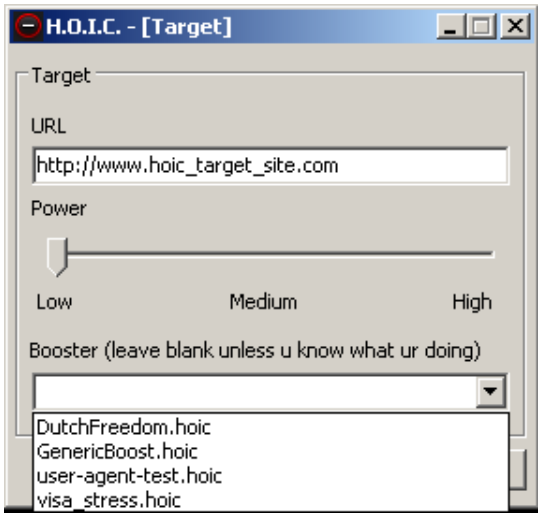


Figura 27 – Seleção de nível Poder de ataque H.O.I.C.

A ferramenta *HTTP Unbearable Load King* (HULK) desenvolvida em *python* para fins acadêmicos e de pesquisa, foi projetado para gerar e enviar enormes volumes de tráfego exclusivo e ofuscado para um servidor web, desviando dos mecanismos de *cache* e atingindo o *pool* de recursos diretos do servidor. A aplicação utiliza várias técnicas para tornar as solicitações dinâmicas, consequentemente mais difíceis de detectar por suas assinaturas, solicitando continuamente um ou vários *Uniform Resource Locator* (URLs) de várias máquinas de ataque de origem, como mostrado na Figura 28. Algumas das características mais marcantes da ferramenta é a ofuscação do cliente de origem usando uma lista de agentes de usuário conhecidos e, para cada solicitação construída, o agente de usuário é um valor aleatório da lista conhecida. Quando os recursos do servidor atingem o limite de conexões simultâneas, o ativo não é capaz de responder as demais solicitações legítimas, negando-as.

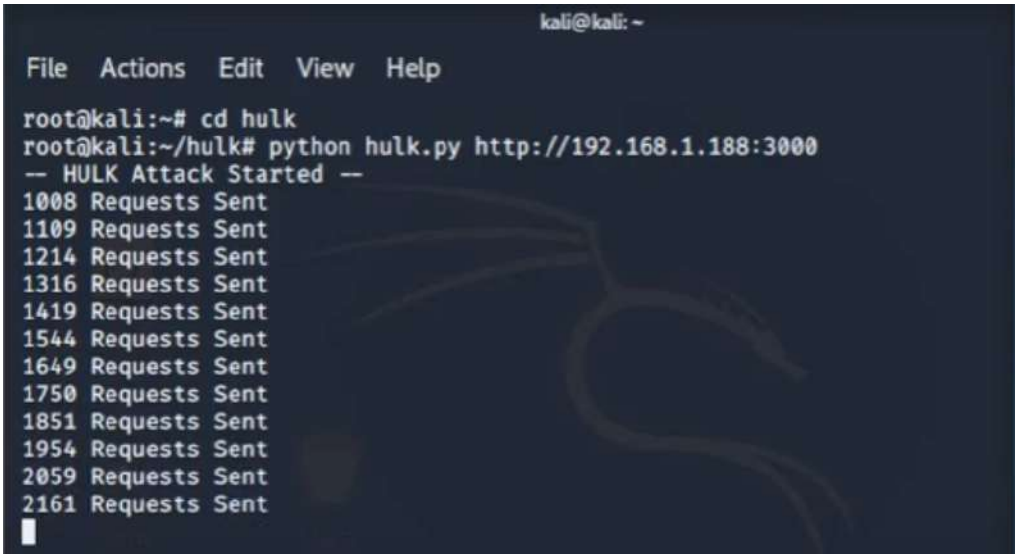


Figura 28 – Ferramenta H.U.L.K.

A ferramenta *Low Orbit Ion Canon* (LOIC) originalmente desenvolvido pela equipe *Praetox Technologies* como uma ferramenta de teste de estresse de rede de licença de código aberto. A aplicação permite realizar tráfego de rede para fins de diagnóstico, ficou também reconhecida pela utilização do grupo hacktivista *Anonymous* como ferramenta de ataque de DDoS. Multiplataforma, a ferramenta de fácil utilização gera uma grande quantidade de tráfego de rede inundando o alvo dos ataques. Esse alto volume e taxa de tráfego de rede, que são facilmente manipuláveis, envia pacotes TCP, UDP ou HTTP ilegítimos, mostrado na Figura 29, resultando na degradação do desempenho e potencialmente afetando negativamente a QOS serviço prestado. Além disso, sua facilidade de uso permite que qualquer pessoa, independentemente de conhecimento, execute ataques potencialmente graves. Embora a ferramenta seja simples e eficaz, ela não aborda falsificação de endereços IP de seus usuários, se tornando completamente visíveis para o alvo, tornando seu rastreamento fácil.

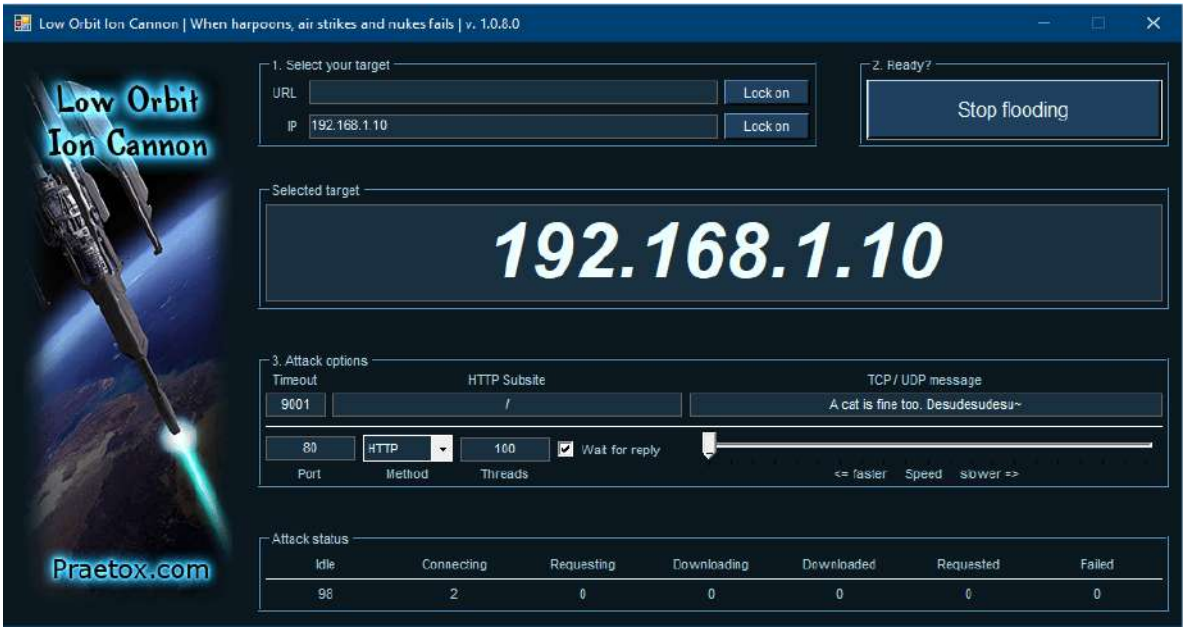


Figura 29 – Ferramenta L.O.I.C.

Tor's Hammer é uma ferramenta de ataque DoS de HTTP *POST* que também funciona na camada de aplicação, sendo a camada 7 no modelo *Open System Interconnection* (OSI). de aplicação gerando solicitações na internet baseado no navegador, que são utilizados para carregar páginas da web. A ferramenta de teste escrita em *python*, realiza ataques de DoS lentos com baixa taxa, que exploram o tempo máximo de conexão atual que os servidores suportam. Podendo também ser utilizada com ferramentas de ofuscação de endereço IP, como o TOR usando um *proxy socks* que permite lançar ataques a partir de endereços IP de origem aleatória, como mostrado na Figura 30. Ferramenta poderosa com varias funcionalidades e opções de ataques, faz com que os *threads* do aplicativo do

servidor da web aguardem o fim das postagens ilimitadas para processá-las, causando o esgotamento dos recursos do servidor da web e fazendo com que ele comece a negar serviço para qualquer tráfego legítimo.

```
root@Note-PPP:/mnt/c/Torshammer 1.0# ./torshammer.py

/*
 * Tor's Hammer
 * Slow POST DoS Testing Tool
 * Version 1.0 Beta
 * Anon-ymized via Tor
 * We are Anonymous.
 * We are Legion.
 * We do not forgive.
 * We do not forget.
 * Expect us!
 */

./torshammer.py -t <target> [-r <threads> -p <port> -T -h]
 -t|--target <Hostname|IP>
 -r|--threads <Number of threads> Defaults to 256
 -p|--port <Web Server Port> Defaults to 80
 -T|--tor Enable anonymising through tor on 127.0.0.1:9050
 -h|--help Shows this help

Eg. ./torshammer.py -t 192.168.1.100 -r 256
```

Figura 30 – Ferramenta Tor's Hammer

A ferramenta *Switchblade* desenvolvido pela empresa *ProactiveRISK* é uma ferramenta de controle de qualidade usada para conduzir testes de capacidade e carga de trabalho em uma aplicação web executando *Internet Information Services* (IIS) ou Apache. O aplicativo com licença de código aberto, inclui técnicas bastante efetivas, incluindo *SSL Half Connect*, *HTTP Post* e *Slowloris*, o que permite a analise especificas dos mecanismos de proteção. As solicitações *GET* são usadas para recuperar conteúdo estático padrão, como as imagens e as solicitações *POST* são utilizadas para acessar recursos gerados dinamicamente. A ferramenta de DDoS ataca servidores Web com grande quantidade de tráfego HTTP podendo utilizar parâmetros aleatórios como mostrado na Figura 31, dificultando ainda mais a sua detecção.

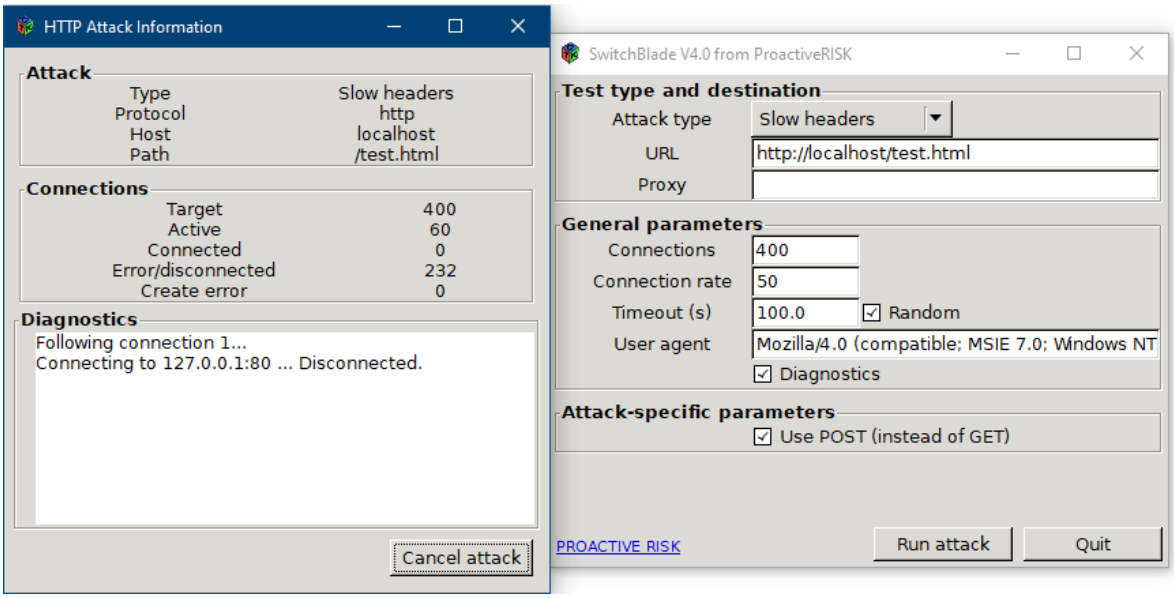


Figura 31 – Ferramenta SwitchBlade

perf-stat

Options:

```
    Any command you can specify in a shell.
-e, --event=
    Select the PMU event. Selection can be a symbolic event
       name (use perf list to list all events) or a raw PMU
       event (eventsel+umask) in the form of rNNN where NNN is
       a hexadecimal event descriptor.
-i, --no-inherit
    child tasks do not inherit counters
-p, --pid=<pid>
    stat events on existing process id (comma separated list)
-t, --tid=<tid>
    stat events on existing thread id (comma separated list)
-a, --all-cpus
    system-wide collection from all CPUs
-c, --scale
    scale/normalize counter values
-r, --repeat=<n>
    repeat command and print average + stddev (max: 100)
-B, --big-num
    print large numbers with thousands separators according to
       locale
-C, --cpu=
    Count only on the list of CPUs provided. Multiple CPUs can
       be provided as a comma-separated list with no space:
       0,1. Ranges of CPUs are specified with -: 0-2. In per-
       thread mode, this option is ignored. The -a option is
       still necessary to activate system-wide monitoring.
       Default is to count on all CPUs.
-A, --no-aggr
    Do not aggregate counts across all monitored CPUs in system
       -wide mode (-a). This option is only valid in system-
       wide mode.
-n, --null
    null run - dont start any counters
-v, --verbose
    be more verbose (show counter open errors, etc)
```

```
-x SEP, --field-separator SEP
    print counts using a CSV-style output to make it easy to
       import directly into spreadsheets. Columns are separated
        by the string specified in SEP.
-G name, --cgroup name
    monitor only in the container (cgroup) called "name". This
       option is available only in per-cpu mode. The cgroup
       filesystem must be mounted. All threads belonging to
       container "name" are monitored when they run on the
       monitored CPUs. Multiple cgroups can be provided. Each
       cgroup is applied to the corresponding event, i.e.,
       first cgroup to first event, second cgroup to second
       event and so on. It is possible to provide an empty
       cgroup (monitor all the time) using, e.g., -G foo,,bar.
       Cgroups must have corresponding events, i.e., they
       always refer to events defined earlier on the command
       line.
-o file, --output file
    Print the output into the designated file.
--append
    Append to the output file designated with the -o option.
       Ignored if -o is not specified.
--log-fd
    Log output to fd, instead of stderr. Complementary to --
       output, and mutually exclusive with it. --append may be
       used here. Examples: 3>results perf stat --log-fd 3 -
       $cmd 3>>results perf stat --log-fd 3 --append - $cmd
```

free

Options:

```
    The -b switch displays the amount of memory in bytes; the -
       k switch (set by default) displays it in kilobytes; the
       -m switch displays it in megabytes.
    The -t switch displays a line containing the totals.
    The -o switch disables the display of a "buffer adjusted"
       line. If the -o option is not specified, free subtracts
       buffer memory from the used memory and adds it to the
       free memory reported.
```

```
    The -s switch activates continuous polling delay seconds
       apart. You may actually specify any floating point
       number for delay, usleep(3) is used for microsecond
       resolution delay times
    The -l switch shows detailed low and high memory statistics
    The -V switch displays version information
```

ifstat

Options:

```
-l
    Enables monitoring of loopback interfaces for which
       statistics are available. By default, ifstat monitors
       all non-loopback interfaces that are up.
-a
    Enables monitoring of all interfaces found for which
       statistics are available.
-z
    Hides interface which counters are null, eg interfaces that
        are up but not used.
-i
    Specifies the list of interfaces to monitor, separated by
       commas (if an interface name has a comma, it can be
       escaped with '\'). Multiple instances of the options are
        added together.
-s
    Equivalent to -d snmp:[comm@][#]host[/nn]] to poll a remote
        host through SNMP . See below for details.
-h
    Displays a short help message.
-n
    Turns off displaying the header periodically.
-t
    Adds a timestamp at the beginning of each line.
-T
    Reports total bandwith for all monitored interfaces.
-A
    Disables use of interface indexes: by default, when polling
        mechanism is index based (snmp, ifmib), ifstat
```

```
       remembers indexes of monitored interfaces to poll only
       them. However, if interfaces indexes change often (new
       interfaces added, etc), you might loose some stats,
       hence this flag. Note that if you ask ifstat to monitor
       a non existent interface, it will poll all interfaces
       until it finds the requested one (regardless of this
       flag) so you can poll for an interface that goes up and
       down.
-w
    Uses fixed width columns, instead of enlarging them if
       needed for interfaces names to fit.
-W
    Wrap lines that are larger than the terminal width (implies
        -w). Wrapped lines are prefixed with a cycling letter
       to ease reading.
-S
    Keep stats updated on the same line if possible (no
       scrolling nor wrapping).
-b
    Reports bandwith in kbits/sec instead of kbytes/sec.
-q
    Quiet mode, warnings are not printed.
-v
    Displays version and the compiled-in drivers.
-d

Specifies a driver to use to gather stats and an eventual
   option for this driver separated of the driver name by a
   colon. If this is not specified, ifstat uses the first
   driver compiled in, with no options.

The following drivers are available (depending on the operating
    system and compile-time options, not all of them might be
   present):
proc
    This driver gets statistics from Linuxs /proc/net/dev file.
        An alternate file name to get stats from can be passed
       as the option.
ifmib
```

```
    This driver gets statistics from FreeBSD's ifmib sysctl. It
        doesn't accept any options.
42  kstat
    This driver gets statistics from Solaris kstat interface.
       It doesn't accept any options.
44  ifdata
    This driver gets statistics using SIOCGIFDATA ioctl under
       IRIX and OpenBSD (different semantics). It doesn't
       accept any options.
46  route
    This driver gets statistics using routing sysctl on BSD
       based systems. It doesn't accept any options.kvm
48     This driver gets statistics by reading the kernel live
       structures. It accepts an option specifying which files/
       devices to use in the following format : [execfile][,[
       corefile][,swapfile]] (see kvm_open(3) for details on
       those fields). If a null string is passed for a
       parameter, the system default will be used for it.

50  Note that for this driver to work, ifstat needs to have read
      access to the system memory device. This is usually done by
      running it as root, or by installing setgid mem or kmem.
      ifstat will NOT install setgid by default; It is up to you
      to decide if you trust it.

52  dlpi
    This driver gets statistics using the DLPI streams
       interface available on HP-UX . An alternate device to
       query statistics from can be passed as the option (
       default is /dev/dlpi).
54  win32
    This driver gets statistics using the GetIfTable interface
       available on Win32 systems. It doesn't accept any
       options.
56  snmp
    This driver gets statistics through SNMP . The option, in
       the form [comm@][#]host[/nn]], specifies the host and
       eventual community to poll. Default community is public,
        but can be changed by prepending "comm@" to the
```

hostname. If host starts by a #, interface names are generated from their index as 'ifNN' (this is a workaround for some equipments that give all interfaces the same description). Default host is localhost, and this will be used by default if snmp is the only available driver.

The driver will try to poll several interfaces at once by grouping requests in SNMP packets. By default interfaces will be polled by group of 8. If this doesn't work well with your equipments, you can lower that number by suffixing the hostname with /nn, where nn is the number of interfaces to poll at once. You can also increase the number if you want to poll a large number of interfaces efficiently and if your server supports it.

delay

delay is the delay between updates in seconds, which defaults to 1. A decimal number can be specified for intervals shorter than a second. (minimum 0.1)

A second delay can also be specified (separated from the first one by a '/'). In that case the first delay will be used for the first poll after start and the second one will be used for all following polls (This can be used to have a "fast" start when running for a long while with a big delay).

count

count is the number of updates before stopping. If not specified, it is unlimited.

sar

Options:

-A

This is equivalent to specifying -bBdqrRSuvwWy -I SUM -I XALL -n ALL -u ALL -P ALL.

-b

Report I/O and transfer rate statistics. The following values are displayed:

tps

Total number of transfers per second that were issued to physical devices. A transfer is an I/O request to a physical device. Multiple logical requests can be combined into a single I/O request to the device. A transfer is of indeterminate size.

rtps

Total number of read requests per second issued to physical devices.

wtps

Total number of write requests per second issued to physical devices.

bread/s

Total amount of data read from the devices in blocks per second. Blocks are equivalent to sectors with 2.4 kernels and newer and therefore have a size of 512 bytes. With older kernels, a block is of indeterminate size.

bwrtn/s

Total amount of data written to devices in blocks per second.

-B

Report paging statistics. Some of the metrics below are available only with post 2.5 kernels. The following values are displayed:

pgpgin/s

Total number of kilobytes the system paged in from disk per second. Note: With old kernels (2.2.x) this value is a number of blocks per second (and not kilobytes).

pgpgout/s

Total number of kilobytes the system paged out to disk per second. Note: With old kernels (2.2.x) this value is a number of blocks per second (and not kilobytes).

fault/s

Number of page faults (major + minor) made by the system per second. This is not a count of page faults that generate I/O, because some page faults can be resolved without I/O.

majflt/s

Number of major faults the system has made per second, those which have required loading a memory page from

```
       disk.
pgfree/s
26     Number of pages placed on the free list by the system per
       second.
pgscank/s
28     Number of pages scanned by the kswapd daemon per second.
pgscand/s
30     Number of pages scanned directly per second.
pgsteal/s
32     Number of pages the system has reclaimed from cache (
       pagecache and swapcache) per second to satisfy its
       memory demands.
%vmeff
34     Calculated as pgsteal / pgscan, this is a metric of the
       efficiency of page reclaim. If it is near 100% then
       almost every page coming off the tail of the inactive
       list is being reaped. If it gets too low (e.g. less than
        30%) then the virtual memory is having some difficulty.
        This field is displayed as zero if no pages have been
       scanned during the interval of time.
-C
36     When reading data from a file, tell sar to display comments
        that have been inserted by sadc.
-d
38     Report activity for each block device (kernels 2.4 and
       newer only). When data is displayed, the device
       specification dev m-n is generally used ( DEV column). m
        is the major number of the device. With recent kernels
       (post 2.5), n is the minor number of the device, but is
       only a sequence number with pre 2.5 kernels. Device
       names may also be pretty-printed if option -p is used (
       see below). Values for fields avgqu-sz, await, svctm and
        %util may be unavailable and displayed as 0.00 with
       some 2.4 kernels. Note that disk activity depends on
       sadc options "-S DISK" and "-S XDISK" to be collected.
       The following values are displayed:
tps
40     Indicate the number of transfers per second that were
       issued to the device. Multiple logical requests can be
```

```
       combined into a single I/O request to the device. A
       transfer is of indeterminate size.
rd_sec/s
    Number of sectors read from the device. The size of a
       sector is 512 bytes.
wr_sec/s
    Number of sectors written to the device. The size of a
       sector is 512 bytes.
avgrq-sz
    The average size (in sectors) of the requests that were
       issued to the device.
avgqu-sz
    The average queue length of the requests that were issued
       to the device.
await
    The average time (in milliseconds) for I/O requests issued
       to the device to be served. This includes the time spent
        by the requests in queue and the time spent servicing
       them.
svctm
    The average service time (in milliseconds) for I/O requests
        that were issued to the device.
%util
    Percentage of CPU time during which I/O requests were
       issued to the device (bandwidth utilization for the
       device). Device saturation occurs when this value is
       close to 100%.
-e [ hh:mm:ss ]
    Set the ending time of the report. The default ending time
       is 18:00:00. Hours must be given in 24-hour format. This
        option can be used when data are read from or written
       to a file (options -f or -o ).
-f [ filename ]
    Extract records from filename (created by the -o filename
       flag). The default value of the filename parameter is
       the current daily data file, the /var/log/sa/sadd file.
       The -f option is exclusive of the -o option.
-h
    Display a short help message then exit.
```

```
-i interval
62      Select data records at seconds as close as possible to the
           number specified by the interval parameter.
-I { int [,...] | SUM | ALL | XALL }
64      Report statistics for a given interrupt. int is the
           interrupt number. Specifying multiple -I int parameters
           on the command line will look at multiple independent
           interrupts. The SUM keyword indicates that the total
           number of interrupts received per second is to be
           displayed. The ALL keyword indicates that statistics
           from the first 16 interrupts are to be reported, whereas
            the XALL keyword indicates that statistics from all
           interrupts, including potential APIC interrupt sources,
           are to be reported. Note that interrupt statistics
           depend on sadc option "-S INT" to be collected.
-m
66      Report power management statistics. Note that these
           statistics depend on sadc option "-S POWER" to be
           collected. The following value is displayed:
MHz
68      CPU clock frequency in MHz.
-n { keyword [,...] | ALL }
70      Report network statistics.
        Possible keywords are DEV, EDEV, NFS, NFSD, SOCK, IP, EIP,
           ICMP, EICMP, TCP, ETCP, UDP, SOCK6, IP6, EIP6, ICMP6,
           EICMP6 and UDP6.
```