DISSERTAÇÃO DE MESTRADO PROFISSIONAL

Prevenindo ameaças persistentes avançadas em redes corporativas utilizando um modelo de segurança baseado em *zero trust* e UEBA

Bruno Carneiro da Rocha

Programa de Pós-Graduação Profissional em Engenharia Elétrica

DEPARTAMENTO DE ENGENHARIA ELÉTRICA
FACULDADE DE TECNOLOGIA
UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia

DISSERTAÇÃO DE MESTRADO PROFISSIONAL

Prevenindo ameaças persistentes avançadas em redes corporativas utilizando um modelo de segurança baseado em *zero trust* e UEBA

Bruno Carneiro da Rocha

ORIENTADOR: Rafael Timóteo de Sousa Jr.

COORIENTADOR: Laerte Peotta de Melo

Dissertação de Mestrado Profissional submetida ao Departamento de Engenharia Elétrica como requisito parcial para obtenção do grau de Mestre em Engenharia Elétrica

Banca Examinadora

Prof. Georges Daniel Amvame Nze, Ph.D FT/UnB _____
Presidente

Prof. Edna Dias Canedo, Ph.D, CIC/UnB _____
Examinador Interno

Prof. Dino Macedo Amaral, Ph.D, Banco do Brasil _____
Examinador Externo

Prof. Flávio Elias Gomes de Deus, Ph.D FT/UnB _____
Suplente

PUBLICAÇÃO: PPEE.MP.026
Brasília/DF, Dezembro - 2022

FICHA CATALOGRÁFICA

CARNEIRO DA ROCHA, BRUNO
Prevenindo ameaças persistentes avançadas em redes corporativas utilizando um modelo de segurança baseado em *zero trust* e UEBA [Distrito Federal] 2022.
xvi, 47 p., 210 x 297 mm (ENE/FT/UnB, Mestre, Engenharia Elétrica, 2022).
Dissertação de Mestrado Profissional - Universidade de Brasília, Faculdade de Tecnologia.
Departamento de Engenharia Elétrica

1. ameaças persistentes avançadas (APT)	2. *zero trust*
3. UEBA	4. *machine learning*
I. ENE/FT/UnB	II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

DA ROCHA, B. C. (2022). *Prevenindo ameaças persistentes avançadas em redes corporativas utilizando um modelo de segurança baseado em zero trust e UEBA*. Dissertação de Mestrado Profissional, Publicação: PPEE.MP.026, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 47 p.

CESSÃO DE DIREITOS

AUTOR: Bruno Carneiro da Rocha
TÍTULO: Prevenindo ameaças persistentes avançadas em redes corporativas utilizando um modelo de segurança baseado em *zero trust* e UEBA.
GRAU: Mestre em Engenharia Elétrica ANO: 2022

Bruno Carneiro da Rocha
Depto. de Engenharia Elétrica (ENE) - FT
Universidade de Brasília (UnB)
Campus Darcy Ribeiro
CEP 70919-970 - Brasília - DF - Brasil

DEDICATÓRIA

Ao meu filho Bernardo. Sem você, jamais teria forças para concluir esta dissertação.

AGRADECIMENTOS

Agradeço primeiramente a Deus, que esteve sempre ao meu lado.

Agradeço a minha esposa Marcela que me apoiou desde o início para eu concluir este mestrado.

Agradeço a minha sogra Lucireis e a minha mãe Ana Cristina, que ficaram com meu filho Bernardo em vários momentos para que eu pudesse pesquisar, estudar e trabalhar.

Sou grato a todo investimento que meus pais Reinaldo e Ana Cristina fizeram em mim para que eu pudesse chegar até aqui.

Um agradecimento especial aos professores Rafael Timóteo de Sousa Jr. e Laerte Peotta de Melo, que me ajudaram bastante com toda experiência e conhecimentos repassados.

RESUMO

Muitas organizações estão sendo alvos de diversos tipos de ataques. Um dos ataques mais perigosos é o chamado Ameaças Persistentes Avançadas (*Advanced Persistent Threats* - APT) pois ele é um ataque silencioso e focado na espionagem e roubo de informações, diferentemente de um ataque de negação de serviço (DoS), por exemplo. A solução proposta aborda a implementação de um modelo de segurança baseado em *zero trust* em conjunto com UEBA para traçar o perfil de comportamento do usuário e encontrar os comportamentos anômalos dos adversários com o objetivo de prevenir ataques APT em redes corporativas. A proposta consiste em utilizar os conceitos de *machine learning* especificamente dentro de cada micro-segmentação e analisar se houve a redução de falsos negativos.

ABSTRACT

Many organizations are being targeted by various types of attacks. One of the most dangerous attacks is called Advanced Persistent Threats (APT) as it is a silent attack and it's main goal is spying and stealing information, different from a denial of service (DoS) attack, por example. The proposed solution addresses the implementation of a security model based on zero trust in conjunction with UEBA to profile user behavior and find anomalous behaviors of adversaries in order to prevent APT attacks on corporate networks. The proposal consists of using machine learning concepts specifically within each micro-segmentation and analyzing whether there was a reduction in false negatives.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

Com o avanço das tecnologias da internet, a segurança da informação tornou-se uma das principais preocupações das corporações. Ataques cibernéticos, como por exemplo o *Advanced Persitent Threats* (APT) estão crescendo cada vez mais e tem atingindo principalmente novas tecnologias como a internet das coisas (IoT), internet dos veículos (IoV) e até mesmo o 5G (1). Apesar das ferramentas tradicionais de segurança (anti-vírus, firewalls, sistemas de detecção de intrusão) serem capaz de detectar e prevenir muitos ataques cibernéticos, os cyber-criminosos estão cada vez mais ágeis, desenvolvendo mais métodos e técnicas avançadas para explorar os seus objetivos, atingindo tanto as comunicações cabeadas e sem fios. (2).

1.2 PROBLEMA DE PESQUISA

O APT explora diversas vulnerabilidades, às vezes desconhecidas pelo fabricantes de *software*, chamados ataques de dia zero (*Zero-Day-Exploits*). (3). Assim que o autor do ataque APT (adversário) se infiltra no sistema e se move lateralmente, ele pode destruir dados importantes e roubar informações sigilosas. (4). Os métodos tradicionais de segurança não conseguem analisar o perfil do adversário (comportamento, capacidades e estratégias) e isso é essencial para evitar os ataques avançados de uma forma automatizada e proativa, a medida que o ataque vai avançando (1).

Então, surge um problema: Como prevenir ameaças persistentes avançadas em redes corporativas, se a maior parte dos métodos de defesa tradicionais respondem passivelmente? Para resolver este problema, foi realizada uma pesquisa utilizando a metodologia *Design Science Research* (DSR), composta por diversas fases, sendo estas:

1. **Fase 1:** Conscientização do problema ;
2. **Fase 2:** Elaboração de uma proposta de solução ;
3. **Fase 3:** Desenvolvimento da solução com a implementação e testes (além da realização dos pequenos ajustes);
4. **Fase 4:** Validação dos testes e análise dos resultados;
5. **Fase 5:** Conclusão.

1.3 OBJETIVOS

Nesta seção, serão detalhados os objetivos desta pesquisa, incluindo o objetivo geral e os objetivos específicos.

1.3.1 Objetivo geral

O objetivo geral deste trabalho é evitar os APTs em redes corporativas utilizando uma abordagem proativa e adaptativa.

1.3.2 Objetivos específicos

Para que o objetivo geral desta pesquisa seja alcançado, alguns objetivos específicos foram enumerados:

1. Pesquisar os APTs que foram feitos, quais os conceitos básicos, e como os ataques foram realizados.
2. Conceituar e discutir as principais técnicas de aplicação de *zero trust* como modelo de segurança e resposta aos incidentes em redes corporativas.
3. Testar diversos algoritmos de *machine learning* para verificar quais aqueles que respondem melhor na tentativa da detecção de APTs.
4. Diminuir a incidência de falsos negativos nas análises de *machine learning* em um modelo de segurança baseado em *zero trust* e UEBA aplicado.

1.4 PUBLICAÇÕES RESULTANTES DESTA PESQUISA

Este trabalho propõe um modelo de segurança de forma que as ameaças avançadas persistentes sejam detectadas e sejam neutralizadas. Durante o desenvolvimento deste modelo, foram geradas duas publicações:

Artigo curto no ICE-B 2021

- **Referência:** ROCHA, B. C. ; MELO, L. P.; SOUSA JR., R. T. *A study on apt in iot networks. 18th International Conference on e-Business* (ICE-B), 2021. (5)
- **Principal contribuição:** Estudo contendo os principais ataques APT, principais ataques em IoT além de uma proposta de um modelo de defesa em um sistema com dispositivos IoT conectados.
- ***Link*:** https://www.scitepress.org/Papers/2021/106152/106152.pdf

Artigo completo no WCNPS 2021

– **Referência:** ROCHA, B. C. ; MELO, L. P. ; SOUSA JR., R. T. *Preventing apt attacks on lan networks with connected iot devices using a zero trust based security model. 2021 Workshop on Communication Networks and Power Systems* (WCNPS), 2021. (6)

– **Principal contribuição:** Proposta de um modelo de segurança baseado em zero trust, como forma de defesa frente a ataques APTs em redes LAN contendo dispositivos IoTs conectados.

– ***Link:*** https://ieeexplore.ieee.org/document/9626270

1.5 ESTRUTURA DA DISSERTAÇÃO

O restante do trabalho está estruturado da seguinte forma: No capítulo 2, serão descritas informações importantes a respeito das tecnologias utilizadas nesta pesquisa além da revisão da literatura. O capítulo 3 irá descrever o modelo proposto baseado em zero trust e UEBA na tentativa de anular os ataques APT e por fim, no capítulo 4 será feita uma análise dos resultados deste modelo. O capítulo 5 irá descrever as conclusões alcançadas nesta pesquisa assim como os possíveis trabalhos futuros.

2 REFERENCIAL TEÓRICO

Este capítulo irá abordar a revisão da literatura além dos trabalhos relacionados que fundamentam o modelo de segurança proposto.

2.1 *ADVANCED PERSITENT THREATS* (APTS)

2.1.1 Definição

O termo APT foi proposto primeiro pela USAF (*United States Air Forces*) em 2006 (7). De acordo com o NIST (*US National Institute of Standards and Tecnology*), a definição de APT é (8) :

> *"An adversary with sophisticated levels of expertise and significant resources, allowing it through the use of multiple different attack vectors (e.g., cyber, physical, and deception), to generate opportunities to achieve its objectives which are typically to establish and extend its presence within the information technology infrastructure of organizations for purposes of continually exfiltrating information and/or to undermine or impede critical aspects of a mission, program, or organization, or place itself in a position to do so in the future; moreover, the advanced persistent threat pursues its objectives repeatedly over an extended period of time, adapting to a defender's efforts to resist it, and with determination to maintain the level of interaction needed to execute its objectives."*

Traduzindo, a definição de APT é: um adversário com níveis sofisticados de especialização e recursos significativos, permitindo-lhe, através do uso de várias ferramentas de ataque diferentes (por exemplo, cibernéticos, físicos e fraudes), gerar oportunidades para atingir seus objetivos que normalmente são estabelecer e estender sua presença dentro do infra-estrutura de tecnologia da informação de organizações para fins de vazamento contínuo de informações e/ou para minar ou impedir aspectos críticos de uma missão, programa ou organização, ou colocar-se em posição de fazê-lo no futuro; além disso, a ameaça persistente avançada persegue seus objetivos repetidamente por um longo período de tempo, adaptando-se aos esforços de um defensor para resisti-la e com determinação para manter o nível de interação necessário para executar seus objetivos.

2.1.2 Ameaça interna

As organizações devem se preocupar basicamente com três tipos de ameaças internas (9):

– **Empregados negligentes:** Empregados que, sem intenção, acabam cometendo erros por não seguir os procedimentos e normas da política de segurança da informação da organização.

– **Empregados maliciosos:** Corresponde ao empregado que tem ou tinha acesso na rede da corporação e intencionalmente realizou algum tipo de ataque à corporação, que ferem os princípios da disponibilidade, integridade ou confidencialidade da informação.

– **Empregados que foram comprometidos:** Neste caso, as credenciais deste empregado ou foram roubadas ou seu dispositivo foi infectado com algum tipo de *malware*, geralmente feito através de ataques *phishing*. Deste ponto, o adversário pode realizar o movimento lateral na rede e escalar privilégios.

2.1.3 Etapas de um APT genérico

Um APT genérico consiste em quatro etapas, sendo: Preparação, Infiltração, Movimento Lateral e Vazamento de Dados (Figura 2.1) (10). A preparação consiste em pesquisar qual organização será o alvo averiguando se tem ativos digitais que possuem uma certa importância. Em seguida, uma análise dos empregados é feita e um *e-mail* genérico é criado para o envio de um *malware*. A infiltração consiste no envio do *e-mail* com o *malware* para os empregados selecionados que seriam potenciais alvos para executarem o *backdoor*. Ao ser executado, o adversário tem o comando e controle com o servidor alvo. O movimento lateral consiste em instalar outros *backdoors* em outros nós da rede para se propagar o acesso pela empresa. E por fim, o vazamento de dados dos nós da rede da organização é feito pelo invasor, sem deixar rastros, de modo que este continue infiltrado na rede e recolhendo dados constantemente.

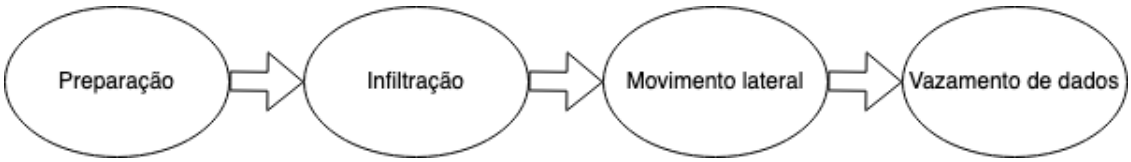

Figura 2.1: Ataque APT genérico. Fonte: Próprio autor

2.1.4 APTs conhecidos

Os ataques APT estão cada vez mais frequentes nas últimas décadas. Antigamente os alvos eram apenas as grandes entidades governamentais dos Estados Unidos, mas agora já foram observados ataques avançados em diferentes países e até mesmo em organizações não governamentais (4). Abaixo seguem alguns exemplos de ataques que ficaram famosos e foram para mídia:

2.1.4.1 *Titan Rain*

Em 2003, uma série de ataques coordenados, apelidados de *Titan Rain*, infiltraram em diversos computadores e redes associadas com a Defesa dos Estados Unidos, com o objetivo principal de roubo de dados sensíveis. Estes ataques ainda estiverem presentes até 2015, roubando informações não classificadas. Não foram reportados roubo de informações classificadas. Este foi o primeiro ataque detectado desse nível, contendo múltiplos vetores. (4)

2.1.4.2 *Hydraq*

Um dos primeiros ataques em organizações comerciais foi o *Hydraq*. Este nome é devido ao trojan utilizado como *backdoor*. Este nome ficou mais popular do que seu nome original chamado "*Operation Aurora*". Este ataque envolveu o uso de diferentes *malwares* que foram encriptados em múltiplas camadas para permanecerem sem serem detectados pelo maior tempo possível. O ataque teve seu início em 2009 e teve como alvo diferentes setores das organizações. O *Google* foi uma das primeiras empresas a anunciar o ataque, seguido pela *Adobe*. O ataque utilizou o *zero-day-exploit* no *Internet Explorer* (CVE-2010-0249) e MS10-002 (11) para estabelecer o controle do sistema. Quando os usuários visitassem o site malicioso, o *Internet Explorer* era infectado e fazia o download de diversos componentes. Um dos componentes do *malware* instalava um *backdoor* no computador, permitindo a entrada de adversários na rede das organizações. (4)

2.1.4.3 *Stuxnet*

Em 2009, uma *worm* sofisticada foi lançada com o objetivo de impedir o projeto nuclear de urânio do Irã. Em um primeiro momento, este *malware* utilizava uma vulnerabilidade *zero-day-exploit* encontrada eu um arquivo LNK do *Windows Explorer*. A *Microsoft* nomeou este *malware* como *Stuxnet* como uma combinação dos nomes dos arquivos encontrados no código malicioso (.stub e MrxNet.sys) e depois reportou esta vulnerabilidade *zero-day-exploit*. Posteriormente foi encontrada uma nova vulnerabilidade no *spooler* de impressora de computadores *Windows* que se espalhava em computadores que estavam compartilhando a mesma impressora. Este *malware* se beneficiava de vulnerabilidades do arquivo do *Windows Keyboard* e do arquivo do *Task Scheduler* para pegar controle total do computador através da escalação de privilégios.(4)

A data final do *Stuxnet* foi por volta de 2012, três anos após ter sido lançado. O Irã descobriu a existência desse *malware* com 500kb na fábrica na cidade de Natanz em 2010. Este *malware* danificou diversas centrífugas e diminuiu a velocidade do processo de geração de armas nucleares.

Diversos estudos como (12) (13) foram feitos nesse período em que os ataques ainda estavam ativos. Todos estes estudos tem em comum que o *Stuxnet* foi um *malware* utilizado pelos adversários para ter acesso remoto aos seus alvos via uma central de comando e controle.

2.1.4.4 *RSA SecureID Attack*

Em 2011, a RSA, uma divisão de segurança do software EMC anunciou um ataque cibernético sofisticado em seus sistemas que envolviam o comprometimento da informação associada ao *SecureID*, um produto de dois fatores de autenticação. Este é mais um ataque de infiltração em rede interna através do envio de *e-mail* com *phishing* aos empregados da organização. Neste ataque, os adversários enviaram *e-mails* aos empregados contendo uma planilha *Excel* contendo uma vulnerabilidade *zero-day-exploit*. Esta planilha instalava um *backdoor* via *Adobe Flash Player* (CVE-2011-0609) no sistema operacional do empregado. Este *backdoor* permitia que os adversários tivesse o controle do computador do empregado remotamente. E a partir daí, conseguiam escalar privilégios, roubar informações. Quando a RSA detectou

este ataque, várias informações já tinham sido vazadas. (4)

2.1.4.5 *Carbanak*

O *Carbanak*, diferente dos ataques discutidos anteriormente, foi um ataque para roubo de dinheiro de instituições financeiras. Os ataques iniciaram em 2013 com os adversários penetrando as redes internas das instituições financeiras e/ou bancárias através de ataques do tipo *phishing*. E estes ataques não foram detectados até 2014. Os *e-mails* enviados aos empregados tinham em anexo arquivos maliciosos que, quando executados, exploravam vulnerabilidades no *Microsoft Office* (CVE-2012-0158, CVE-2013-3906, e CVE-2014-1761) permitindo a instalação do *backdoor* chamado de *Carbanak*. Este *backdoor* permitia o controle remoto dos adversários via uma central de comando e controle e a partir daí, poderiam tentar realizar o movimento lateral através de *key loggers*, falsos formulários. Pesquisadores encontraram posteriormente vídeos gravados pelos adversários dos empregados e enviados para os adversários. Este ataque utilizou diferentes ferramentas além de um protocolo customizado para comunicação com a central de comando e controle. Os adversários criaram diversas transações falsas com o objetivo de esconderem as transações reais com os roubos financeiros. Este ataque durou até 2015 e só foi detectado em 2017. (14) (4)

2.1.4.6 *HAFNIUM*

O *HAFNIUM* é um grupo APT patrocinado pelo Estado, porém operando fora da China. Sua atuação é com base táticas e procedimentos através de comportamentos observados de suas vítimas. A *Microsoft* detectou vários *Zero-Day-Exploits* utilizados para explorar versões locais do *Microsoft Exchange Server* em ataques limitados e direcionados em 2021. Nos ataques observados, o adversário usou essas vulnerabilidades para acessar servidores locais do *Exchange* que permitiram o acesso a contas de *e-mail* e permitiram a instalação de *malware* adicional para facilitar o acesso de longo prazo aos ambientes das vítimas (15).

2.1.4.7 Análise comparativa dos ataques APT

A tabela 2.1 apresenta os vetores de ataque utilizados nos ataques APT relacionados acima. É possível observar que a medida que os anos foram passando, mais técnicas de invasão foram surgindo e sendo usadas nos ataques. Porém, técnicas antigas como o *backdoor* estão em uso até os dias atuais.

Análise de APTs conhecidos			
Ataque APT	**Data**	**Objetivo**	**Vetores Utilizados**
Titan Rain	2003 - 2005	Roubar dados sensíveis das organizações	Engenharia Social e *Backdoors*
Hydraq	2009 - 2011	Roubar dados sensíveis das organizações	Engenharia Social, *Phishing*, *Backdoors*, *Zero-Day-Exploits*
Stuxnet	2009 - 2012	Impedir avanço de armas nucleares do Irã	*Malware* via dispositivos USB, *Zero-Day-Exploits*, *Backdoors*
RSA SecureID Attack	2011	Roubar dados sensíveis das organizações	*Phishing*, *Zero-Day-Exploits*, *Backdoors*
Carbanak	2013 - 2015	Roubo financeiro	Engenharia Social, *Phishing*, *Backdoors*, *Key Loggers*, Falsos Formulários, Vídeos capturados dos empregados, Ferramentas de administração remota
HAFNIUM	2021	Roubar dados sensíveis das organizações	*Malware* ,*Zero-Day-Exploits*, *Backdoors*, vulnerabilidades

Tabela 2.1: Análise comparativa de ataques APT

2.2 SEGURANÇA TRADICIONAL EM REDES CORPORATIVAS

Historicamente, o modelo de segurança de rede baseado em perímetro tem sido o modelo predominante de segurança da informação. Assume-se que os usuários dentro do perímetro da rede corporativa são "confiáveis" e qualquer um do lado de fora é "não confiável". Por várias décadas, essa visão de confiança serviu como base para determinar quais recursos um usuário/dispositivo pode acessar (16).

Porém, nos últimos anos, diversos ataques cibernéticos mostraram que a segurança no perímetro não é suficiente. Foi constatado que o perímetro está cada vez menos relevante devido a diversos fatores como o crescimento da *cloud computing*, mobilidade e mudanças da força de trabalho moderna. Além disso, um adversário externo pode iniciar um ataque de uma central de comando e controle através de um usuário interno que já foi previamente infectado por um *malware* (16) (figura 2.2).

2.3 *ZERO TRUST*

De acordo com o NIST (17), *zero trust* é o termo de cybersegurança para agrupar uma série de paradigmas que transforma a defesa estática, que é focada no perímetro da rede, em defesa direcionada para os usuários, componentes de aplicações e dispositivos (sujeito). A arquitetura *zero trust*, ou *Zero Trust Architecture* (ZTA) utiliza os princípios de *zero trust* para planejar diagramas e infraestrutura para as organizações. O princípio básico de *zero trust* é que não existe confiança em usuários ou contas de acesso baseadas apenas na localização da rede. Não importa se o usuário está na rede local ou na internet, a autenticação e a autorização de usuário é necessária para utilização de recursos ou realizar certas funções.

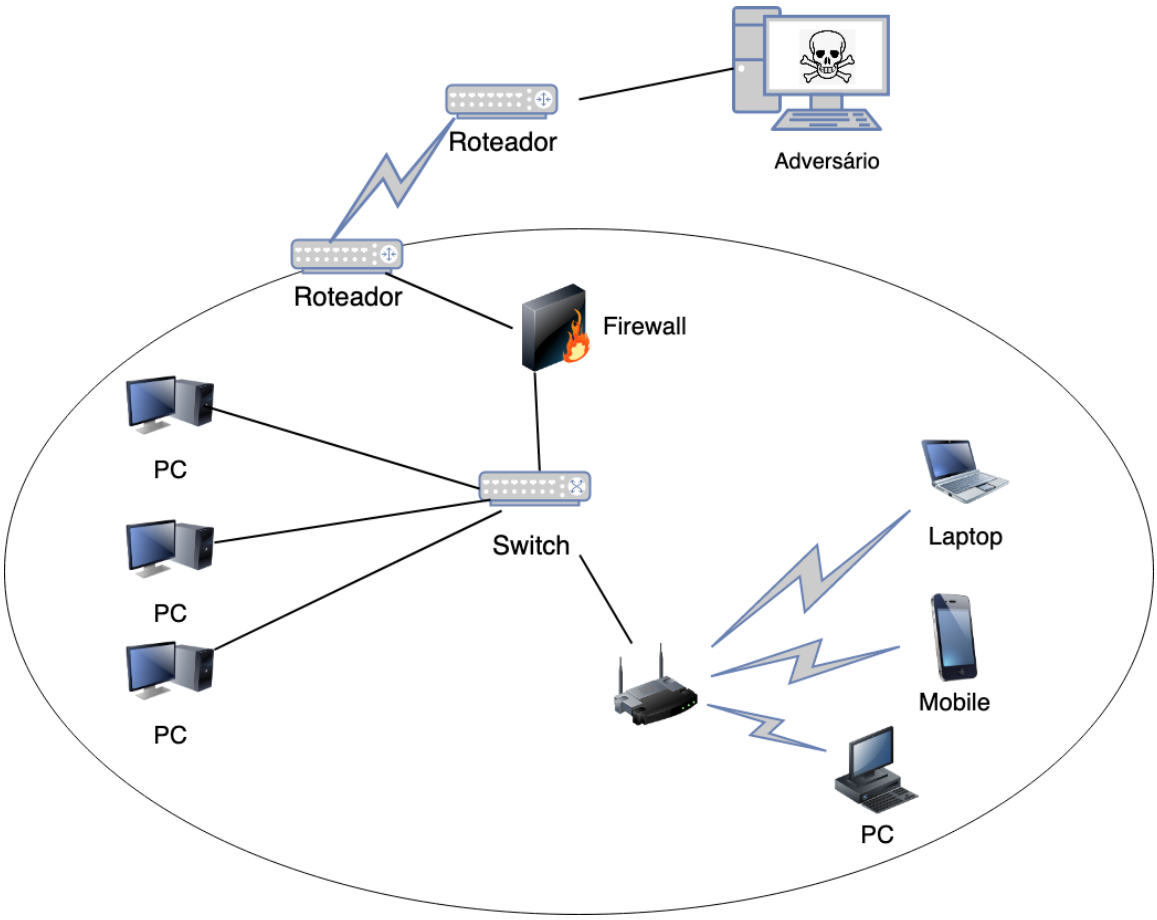

Figura 2.2: Segurança tradicional em rede LAN corporativa. Fonte: próprio autor

O *zero trust* resolve problemas de rede comuns como usuários que estão acessando remotamente a rede local, uso de equipamentos pessoais na rede e o uso de recursos que estão na *cloud*, que não estão disponíveis a menos que o usuário esteja fisicamente dentro do perímetro da rede. A proteção do *zero trust* foca nos recursos (componentes, serviços, fluxogramas, contas de rede, etc.) e o posicionamento dos componentes de rede não é mais o principal insumo de segurança quando esta postura é adotada.

A principal proposta do *zero trust* é prevenir acessos não autorizados aos dados e serviços, de forma que o acesso seja o mais específico possível. Em um modelo de acesso abstrato, quando um sujeito precisa de acesso a um recurso específico (impressoras, computadores, IoTs ou outros dispositivos), este precisa do acesso através do *Policy Decision Point* (PDP) e do seu correspondente *Policy Enforcement Point* (PEP) (figura 2.3).

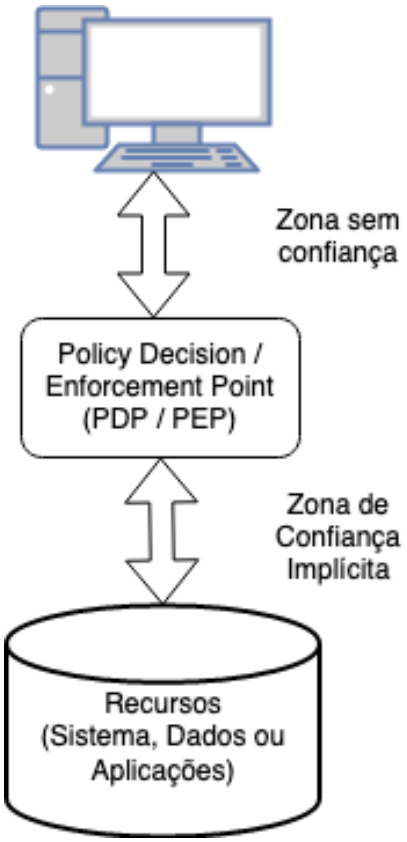

Figura 2.3: Zero Trust: PDP e PEP. Fonte: Próprio autor

A *implicit trust zone* (ou zona da confiança implícita) representa uma área de confiança desde o último *gateway* PDP/PEP. PDP/PEP implementam uma variedade de controles de modo que todo o tráfego de rede depois do PEP tem um nível comum de confiança. O PDP/PEP não pode incluir novas políticas de segurança depois desta localização no tráfego da rede. Para que o PDP/PEP seja o mais específico possível, a *implicit trust zone* precisa ser menor possível. Quanto menor for a implicit trust zone, mais específicos são os controls para cada ativo de rede. E para isso existe o conceito de micro-segmentação de rede.

2.3.1 *Network micro-segmentation*

De acordo com o NIST, (17), existem diversas abordagens que a ZTA pode ser utilizada nos fluxos de rede. Basicamente, existem três abordagens principais na rede: *enhanced identity governance* (governança de identidade aprimorada), *logical micro-segmentation* (micro-segmentação lógica) e *software-based physical micro-segmentation* (micro-segmentação física baseada em software). A melhor abordagem a ser escolhida varia de acordo com cada organização pois cada uma possui suas próprias regras e políticas além de seus próprios componentes de redes. Uma solução completa de *zero trust* inclui elementos de todas as três abordagens. Nesta dissertação, a principal abordagem a ser utilizada será a **micro-segmentação lógica** pois a rede será dividida apenas logicamente sem a utilização de um *software* específico.

Ainda de acordo com o NIST, uma organização pode implementar uma arquitetura de *zero trust* baseada no princípio de separação de recursos individualmente ou em grupos em um segmento de rede protegido por um componente de segurança do *gateway*. Nessa abordagem, a organização desenha seus dispositivos de segurança como *switches* inteligentes, roteadores, *Next-Generation Firewalls* (NGFWs) ou *gateways* para atuarem com PEP, protegendo cada recurso ou um grupo pequeno de recursos.

2.3.2 *Next-Generation Firewalls* (NGFWs)

De acordo com o Gartner (18), *Next-Generation Firewalls* (NGFWs) são firewalls que podem realizar uma inspeção profunda dos pacotes de rede transmitidos. A análise do *firewall* não está limitada na camada de rede 3 e 4 do modelo OSI (rede e transporte), mas também na análise de pacotes na camada de aplicação (camada 7). Além disso, NGFWs possuem sistemas de prevenção de intrusão integrados. Os benefícios de se optar por NGFWs ao invés de *firewalls* tradicionais são vários. NGFWs são capazes de bloquear *malwares* de penetrarem na rede, então é possível concluir que estes respondem muito bem na detecção e prevenção de APTs.

2.4 UBA - *USER BEHAVIOR ANALYTICS* E UEBA - *USER ENTITY BEHAVIOR ANALYTICS*

2.4.1 *Security Analytics*

Quando os ataques são conhecidos, estes podem ser bloqueados através de regras estáticas, verificação de assinaturas de programas maliciosos através do uso de antivírus, sistemas de detecção e prevenção de intrusão (IDS e IPS), *firewalls* e SIEM. Quando os adversários são conhecidos, é possível bloquear seus atributos como IPs ou URLs. Arquivos maliciosos enviados por estes remetentes podem ser bloqueados, apagados ou colocados em quarentena.

Quando nem os ataques e nem os adversários são conhecidos (*zero-day-exploits*), a segurança analítica pode auxiliar no processo de detecção destes ataques com o uso de métodos baseados em estatística como *machine learning*, tornando o processo mais eficiente e mais inteligente (19) (20) .

2.4.2 UBA - *User Behavior Analytics*

A UBA, *User Behavior Analytics*, foi o primeiro produto a ser lançado no mercado pela Gartner. As tecnologias UBA analisam os *logs* históricos de dados dos servidores, incluindo os dados de rede e dados de autenticação dos usuários e estes são armazenados em sistemas de segurança da informação e gerenciamento de eventos (SIEM). A intenção é identificar padrões nos tráfegos causados pelos comportamento dos usuários, tanto normais quanto maliciosos (21).

2.4.3 UEBA - *User Entity Behavior Analytics*

Em agosto de 2015, Aviva Lithan, analista do Gartner, introduziu o termo "*entity*"ao título "*User and Entity Behavioral Analytics*" (UEBA). A UEBA tem as mesma capacidade da UBA, porém a UEBA tem uma técnica analítica mais avançada. Enquanto a UBA tem um enfoque maior em rastreamento de ameaças internas, a UEBA foi desenvolvida para o uso de aprendizagem de máquina com o objetivo de identificar comportamentos anômalos associados às mais diversos ataques, incluindo o APT. Em 2016, a *Gartner Security and Risk Management Summit* categorizou a UEBA como um solução de gerenciamento de riscos. (22). Sistemas que usam UEBA são capazes de rastrear mais do que as atividades de usuário como a UBA faz. A UEBA também rastreia as atividades de dispositivos, aplicações, servidores e dados. Ao invés de analisar somente o comportamento do usuário, a UEBA é bem mais ampla e capaz de combinar o comportamento do usuário com dados de comportamento das mais diversas entidades. (21).

A UEBA trabalha para detectar mudanças no comportamento da comunicação entre o servidor e o *endpoint*, o que pode indicar um ataque. O uso de *unsupervised machine learning* serve para identificar mudanças de comportamento em dispositivos, usuários e rede. Como resposta ao incidente, é possível ter rotinas automatizadas para evitar o ataque ou o incidente pode ser analisado por um administrador de segurança. (22).

2.4.4 Os três pilares da UEBA

O Gartner define a solução UEBA em três dimensões (23) (figura 2.4):

- **Casos de Uso (*Use Cases*):** As soluções UEBA disponibilizam informações a respeito do comportamento dos usuários e outras entidades nas redes corporativas. Estas soluções devem prover monitoração, detecção e alerta de anomalias. Estas soluções podem e devem ser aplicadas para diversos casos de uso como monitoramento de funcionários, redes e fraudes.
- **Repositório de dados (*Data Sources*):** As soluções UEBA são capazes de importar dados de diversos repositórios como *data lake*, *data warehouse*, banco de dados ou até mesmo de um SIEM.
- **Ciência de Dados (*Analytics*):** Para detectar anomalias, podem ser utilizadas diversas formas de ciências de dados modelos estatísticos, aprendizado de máquina, assinaturas de ameaças, regras pré definidas, entre outras.

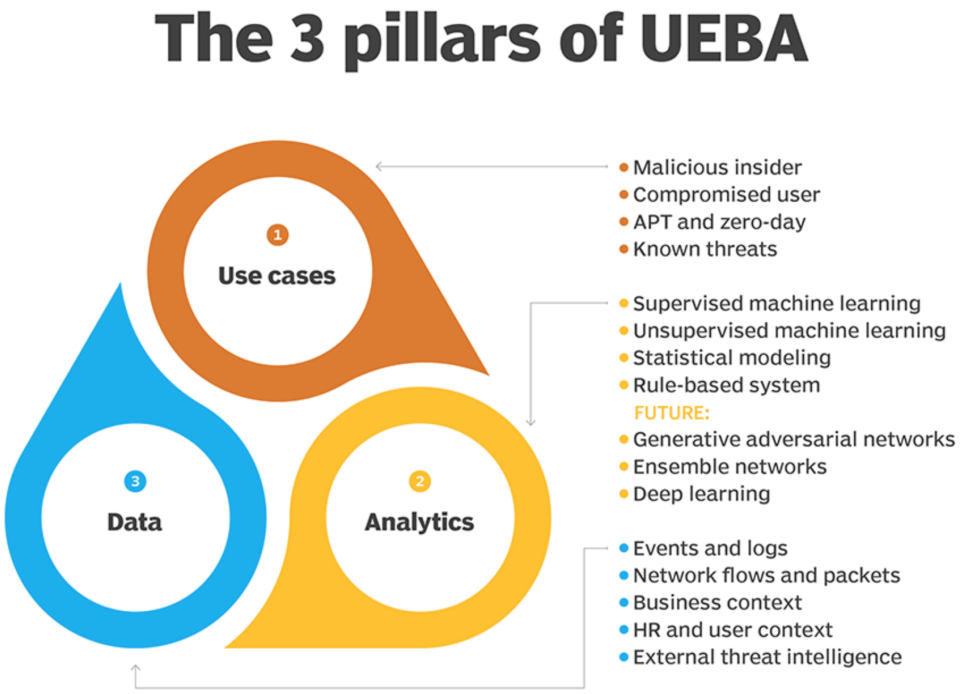

Figura 2.4: Os três pilares da UEBA. Fonte: Gartner

2.4.5 SIEM

SIEM (*Security Information and Event Management*), ou Segurança da Informação e Gerenciamento de Eventos, é combinação entre gerenciamento de informações de segurança (SIM) e gerenciamento de eventos de segurança (SEM). O SIEM oferece monitoramento e análise de eventos em tempo real, bem como rastreamento e registro de dados de segurança para fins de conformidade ou auditoria e por isso é necessário um hardware com grande poder de processamento para análise dos logs citados. (24)

O SIEM trabalha com o gerenciamento de *logs*, correlação de eventos, monitoração de incidentes e alertas de segurança. As mais novas soluções baseadas em SIEM já estão utilizando técnicas de inteligência artificial para aumentar a eficiência da solução. (24)

Anton Chuvakin, analista do Gartner, provoca em seu artigo (25) que uma guerra entre UEBA e SIEM está se aproximando, onde grandes sistemas de informação usam um misto das duas tecnologias. O autor diz que os revendedores de SIEM irão usar algumas soluções UEBA em seus produtos assim como vendedores de produtos UEBA também usarão características de análise e armazenamento de tráfego iguais aos sistemas SIEM (figura 2.5).

2.4.6 *Machine Learning*

O aprendizado de máquina é uma ciência que é utilizada para o desenvolvimento de *softwares* que são capazes de simular o cérebro humano tomando decisões que não estão implícitas no código ou nos dados analisados, de forma que seja possível adaptar o aprendizado para diversos cenários.(26)

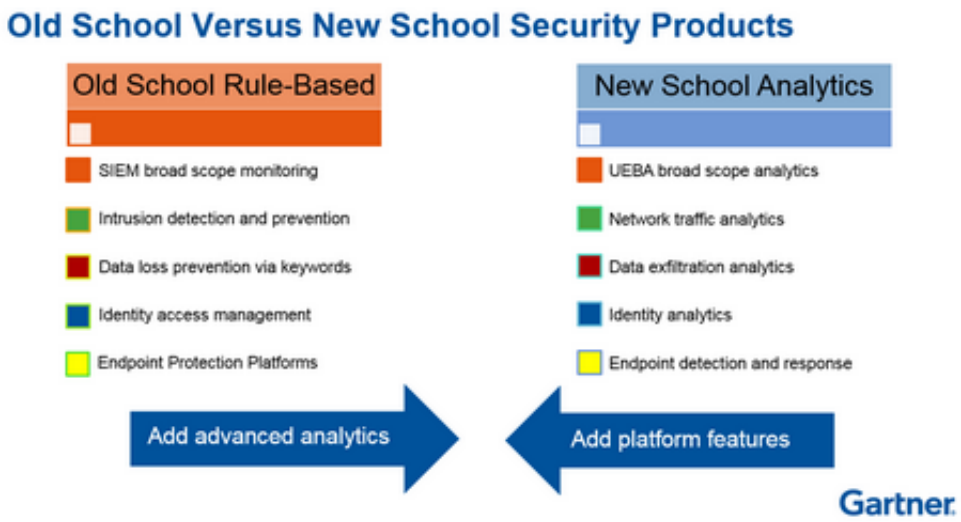

Figura 2.5: SIEM vs UEBA.Fonte: Gartner

2.4.6.1 *Sub-áreas*

Existem três tipos de subáreas principais quando é analisado um problema de *machine learning*: Classificação, Regressão e Agrupamento.

- **Classificação:** O conceito principal da classificação é prever uma categoria que possui variável discreta. (26) Exemplo: Avaliar se foi um "Ataque"ou "Não Ataque".
- **Regressão:** A regressão também prevê uma resposta assim como a classificação, porém possui como resultado uma variável contínua. (26) Exemplo: Estimar a duração em segundos de um ataque com base nos ataques anteriores.
- **Agrupamento (*Clustering*):** Agrupa observações em grupos (*clusters*). Cada *cluster* possui observações simulares. E cada *cluster* possui diferenças em relação aos demais.(26) Exemplo: Agrupar animais em *clusters*, sem saber quais são os animais.

2.4.6.2 Tipos de Aprendizado

Existem dois tipos de aprendizado de máquina: aprendizado supervisionado e aprendizado não supervisionado.

O aprendizado supervisionado utiliza dados já observados que possuem suas respostas rotuladas. Dessa forma, é possível comparar as previsões encontradas na fase de teste com os rótulos reais. Abrange as subáreas de classificação e regressão.

O aprendizado não supervisionado não necessita de rótulos nas respostas. Seu objetivo principal é encontrar grupamentos (ou *clustering*) onde seus dados apresentam um comportamento similar. Abrange a subárea de *Clustering*.

2.4.6.3 Algoritmos

- ***Support Vector Machines* (SVMs):** *Support Vector Machines* (SVMs) são um composto de métodos supervisionados utilizados para classificação, regressão e detecção de *outliers*. A maior vantagem do SVM é o seu poder de classificação em *datasets* de muitas dimensões pois ele busca pela melhor linha que separa todas as diferentes classes presentes nos dados de treino. (27)

- ***K-Nearest Neighbor* (KNN):** Os algoritmos "*Nearest Neighbors*"provém funcionalidades supervisionadas e não supervisionadas. O algoritmo KNN - (k-vizinhos mais próximos) utiliza a classificação supervisionada de cada amostra de um conjunto de dados avaliando a distância em relação aos vizinhos mais próximos. (28)

- **Árvores de Decisão (AD):** Os algoritmos de árvores de decisão representam uma tabela de decisão em forma de árvore. Árvores de Decisão utilizam métodos de classificação e regressão. O objetivo principal é criar um modelo que consegue prever os valores de um objetivo através das regras dos nós que foram criados em tempo de treinamento. (29)

- **Redes Neurais (RN):** Redes Neurais possuem modelos supervisionados e não supervisionados. É um modelo que utiliza o treinamento em camadas semelhantes a um cérebro humano. (30)

- ***Naive Bayes* (NB):** *Naive Bayes* é um composto de algoritmos supervisionados baseados no teorema de *Bayes*. *Naive* vem do inglês que significa ingênuo pois a premissa central do algoritmo é que os atributos considerados não são correlacionados entre si. (31)

- ***Random Forest* (RF):** A idéia do *Random Forest* é criar muitas árvores de decisão de maneira aleatória (formando uma floresta) e cada árvore irá participar da escolha do resultado final em uma espécie de votação. (32)

2.4.7 Métricas Utilizadas

Os sistemas que utilizam *machine learning* possuem métricas para mensuração de sua eficiência. A explicação dos conceitos se dará utilizando o caso de uso desta dissertação, para avaliar se houve ou não ataque e se como o classificador se comportou.

- **Verdadeiro Positivo (*True Positive* - TP):** É uma predição correta de uma atividade que não é ataque. Ou seja, o classificador classificou corretamente uma atividade comum como não ataque.

- **Verdadeiro Negativo (*True Negative* - TN):** Também é uma predição correta. O classificador classifica corretamente uma atividade maliciosa como uma atividade que é realmente um ataque.

- **Falso Positivo (*False Positive* - FP):** É uma predição incorreta de uma atividade comum (que não foi um ataque). Nesse caso, o classificador classificou como ataque uma classe que não era ataque.

- **Falso Negativo (*False Negative* - FN):** Este é o pior cenário quando estamos tratando de ataques. É uma predição incorreta. O classificador classifica como atividade normal uma atividade que realmente era um ataque.

Através desses quatro valores, é possível calcular os indicadores: Acurácia, Precisão, *Recall* e *F1 Score*.

– **Acurácia: (AC)** É a taxa de acerto geral do sistema, calculado pela fórmula (33) (34) :

$$AC = \frac{TP + TN}{TP + TN + FP + FN} \cdot 100 \tag{2.1}$$

Como, no geral, há mais transações comuns (não ataques) do que ataques, então esta métrica tende a ficar desbalanceada no caso de uso desta dissertação.

– **Precisão: (PRC)** É o cálculo da proporção de casos positivos que realmente são positivos. Este indicador prioriza a redução de falsos positivos (35). Estima o quanto de transações autênticas estão sendo consideradas como ataques (Falso Positivo). A precisão é calculada pela fórmula (34):

$$PRC = \frac{TP}{TP + FP} \cdot 100 \tag{2.2}$$

– ***Recall*: (REC)** É a proporção de ataques que foram corretamente classificados como ataques pelo modelo (35). Quanto maior esta métrica, menor a taxa de falsos negativos. O *recall* é calculado pela fórmula (34) :

$$REC = \frac{TP}{TP + FN} \cdot 100 \tag{2.3}$$

É mais aceitável ter algumas transações autênticas serem consideradas como ataques (falso positivo) do que algumas transações de ataques serem consideradas como normais.

– ***F1 Score*: (F1)** É a média harmônica entre o *Recall* e a Precisão (36). O *F1 Score* é calculado pela fórmula (34):

$$F1 = 2 \cdot \frac{(PRC \cdot REC)}{(PRC + REC)} \cdot 100 \tag{2.4}$$

– **Curva ROC** A Curva ROC (*Receiver Operating Characteristic*) é a curva que demonstra a performance de classificação de um modelo de *machine learning*. Pode ser utilizada para comparação entre modelos. O gráfico é composto pela taxa de verdadeiros positivos (TPR) no eixo x e pela taxa de falsos positivos (TFP) no eixo y. A TPR corresponde a sensibilidade e a TFP corresponde a especificidade. (37)

$$TPR = \frac{TP}{(TP + FN)} \cdot 100 \tag{2.5}$$

$$TFP = \frac{FP}{(FP + TN)} \cdot 100 \tag{2.6}$$

– **AUC - *Area Under the Curve*** A área abaixo da Curva ROC é chamada de AUC. A AUC representa o quanto o modelo é capaz de realizar a distinção entre as classes. Quanto maior o valor do AUC, melhor. O AUC varia de 0 a 1. (37)

2.5 TRABALHOS CORRELATOS

2.5.1 Integrando *zero trust* e UEBA

Debanjali Ghosh, uma disseminadora técnica da empresa *Manage Engine* [1], realizou um seminário online (*webinar*) (38) relacionando a utilização de *zero trust* com UEBA. Debanjali informou que o elemento humano é um fator que tende a ser ignorado na elaboração da estratégia de cyber segurança. E esta é uma das principais causas de ataques e ameaças internas. Os adversários cada vez mais exploram técnicas cada vez mais sofisticadas para realizar os ataques. A UEBA, que explora algoritmos de *machine learning* e inteligência artificial com o objetivo de estabelecer um padrão de comportamento, ajuda a detectar uma grande variedade de ataques. Ainda em seu *webinar*, Debanjali relatou que uma arquitetura de *zero trust* (ZTA) tem sido indispensável nas organizações para o estabelecimento uma forte cybersegurança. E foi descrito também com uma solução UEBA se encaixa na ZTA.

2.5.2 Utilizando *machine learning* contra APT

Kim et al. (39) usaram *machine learning* com árvores de decisão para detectar anomalias de ataques APT, baseadas em regras. A proposta envolveu duas etapas. Na primeira etapa, foi usado *machine learning* e árvores de decisão com base em dados estatísticos para gerar regras de comportamento. Na segunda etapa, foi feito um trabalho de detecção do comportamento anômalo a partir das regras pré-estabelecidas na primeira etapa.

Zhao et al. (40) propuseram um sistema para detectar infecções por *malware* de ataques APT. Esta solução contém duas fases, sendo a primeira a detecção da central de comando e controle e em seguida uma análise de IPs associados ao tráfego anômalo. Foi utilizado um algoritmo de árvore de decisão como detector do DNS malicioso. Em seguida, um mecanismo de reputação foi utilizado para identificar as infecções de *malware* por ataques APT.

Friedberg et al. (41) propuseram uma solução que também utiliza *machine learning* para identificação de comportamento anormal nos ataques APT. A solução aprende o comportamento normal de um sistema ao longo do tempo e em seguida descreve todas as ações que diferem do modelo criado. O esquema proposto usa dados de *logs* produzidos por vários sistemas e dados de tráfego de rede.

Yuan (42) apresentou um estudo utilizando técnica de *deep learning*. O autor acredita que o uso de algoritmos convencionais como SVM, árvores de decisão e KNN não ajudam com eficiência no processo de detecção devido ao alto índice de falsos positivos.

Siddiqui et al. (43) assim como Yuan também encontraram um alto índice de falsos positivos e falsos

[1] https://www.manageengine.com/

negativos em algoritmos de *machine learning* tradicionais como o KNN. Foi demonstrado que a combinação de dois data-sets de diferentes origens reduziu a incidência de falsos positivos e falsos negativos baseado no fato de que a correlação entre eles usando a dimensão fractal teve a capacidade de extrair informações escondidas de múltiplas escalas.

Hsieh et al. (44) propuseram uma estrutura para detectar APTs por meio do monitoramento de *logs* do *Active Directory*. E em seguida utilizar *machine learning* como forma de detecção de anomalias com a construção de um modelo probabilístico de Markov. Em geral, o *framework* proposto busca as mudanças no comportamento do usuário ao longo do tempo através da análise de seus *logs* de registro das contas. No entanto, seu modelo de Markov fornece um melhor desempenho de cerca de 66% de taxa de *recall* ou Acurácia.

3 MODELO PROPOSTO

Neste capítulo, o modelo de segurança baseado em *zero trust* e UEBA será proposto com o objetivo final de evitar os APTs.

3.1 DESCRIÇÃO DA METODOLOGIA PROPOSTA

O modelo proposto está exemplificado na figura 3.1. A proposta deste modelo é a aplicação de uma arquitetura baseada em *zero trust* na rede corporativa, agrupando os ativos em segmentos de rede, cada um protegido por um componente de segurança. Para que essa **micro-segmentação lógica** seja possível, o design da arquitetura de rede deve ser com a utilização de *switchs* ou *routers* inteligentes, *Next-Generation Firewalls* ou *gateways* para atuarem como *Policy Enforcement Point* (PEP), realizando a proteção mais específica para cada grupo de ativos de rede.

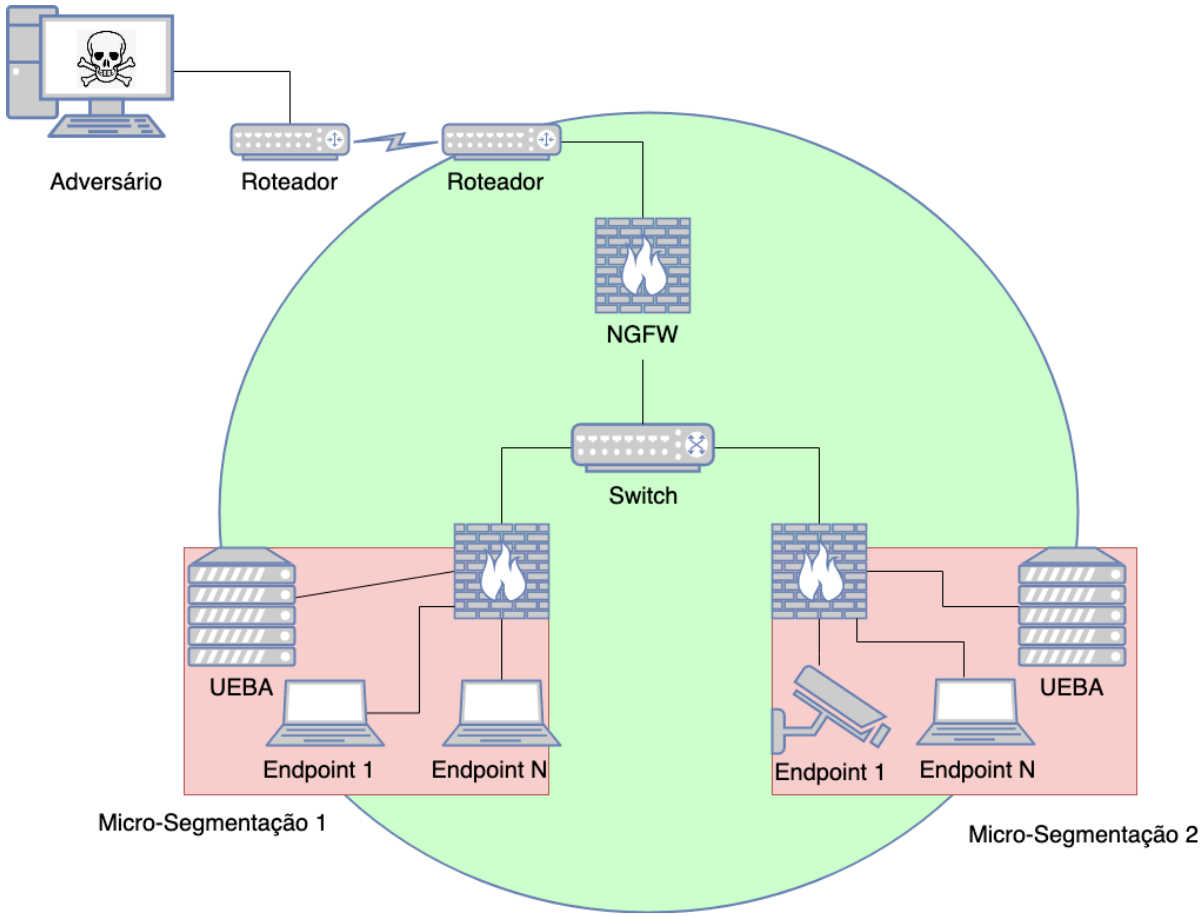

Figura 3.1: Modelo proposto com a aplicação de *zero trust* e UEBA.

Em cada micro-segmentação de rede, deve haver um servidor específico de monitoração com o objetivo principal de detectar comportamentos anômalos através da utilização de UEBA na tentativa de uma redução

de falsos negativos na análise de APTs. As fases para a criação desta arquitetura estão esquematizadas na figura 3.2.

3.2 ETAPA DE TREINO DO ALGORITMO

A etapa de treino consiste em treinar os diversos modelos de *machine learning* com base no comportamento do usuário para se analisar quais os melhores algoritmos serão definidos em cada micro-segmentação de rede. Mas para encontrar o melhor algoritmo, é necessário primeiro entender o caso de uso, ou seja: qual o recurso se está protegendo, qual o valor desse recurso ao negócio e e quais os riscos envolvidos em caso de sucesso do adversário.

3.2.1 Fase 1: Definição do caso de uso

A primeira etapa do processo é definir qual o objeto principal desta análise. Não é possível abranger todas as etapas da infiltração dos adversários. Normalmente, este processo contempla analisar o perfil habitual dos usuários do sistema e tentar encontrar comportamentos que divergem deste comportamento traçado.

O principal método utilizado nas invasões de sistemas corporativos é o *phishing*. A montagem de uma árvore de ataques (45) utilizando o *framework* MITRE ATT@CK [1], é possível visualizar quais as etapas que um adversário pode usar no ataque e a partir desse ponto planejar as etapas seguintes: De onde coletar os dados de tráfego de rede, descobrir quais os serviços mais afetados na tentativa de isolá-los em micro-segmentações de rede mais específicas e mais seguras.

3.2.2 Fase 2: Coletar e preparar os dados de *logs* do tráfego de rede

A fase de coleta de dados é uma das fases mais importantes do modelo. É com base nessa coleta que os algoritmos de *machine learning* são usados no treino dos *datasets*. Quanto maior e mais fidedigna for esta coleta, maior é precisão final.

3.2.3 Fase 3: Testar todos os dados coletados do caso de uso selecionado com diversos algoritmos de *machine learning*

Nesta fase, a experiência do cientista de dados é importante pois ele deve definir quais os melhores algoritmos de *machine learning* que serão utilizados para os testes com os dados coletados.

[1] https://attack.mitre.org/

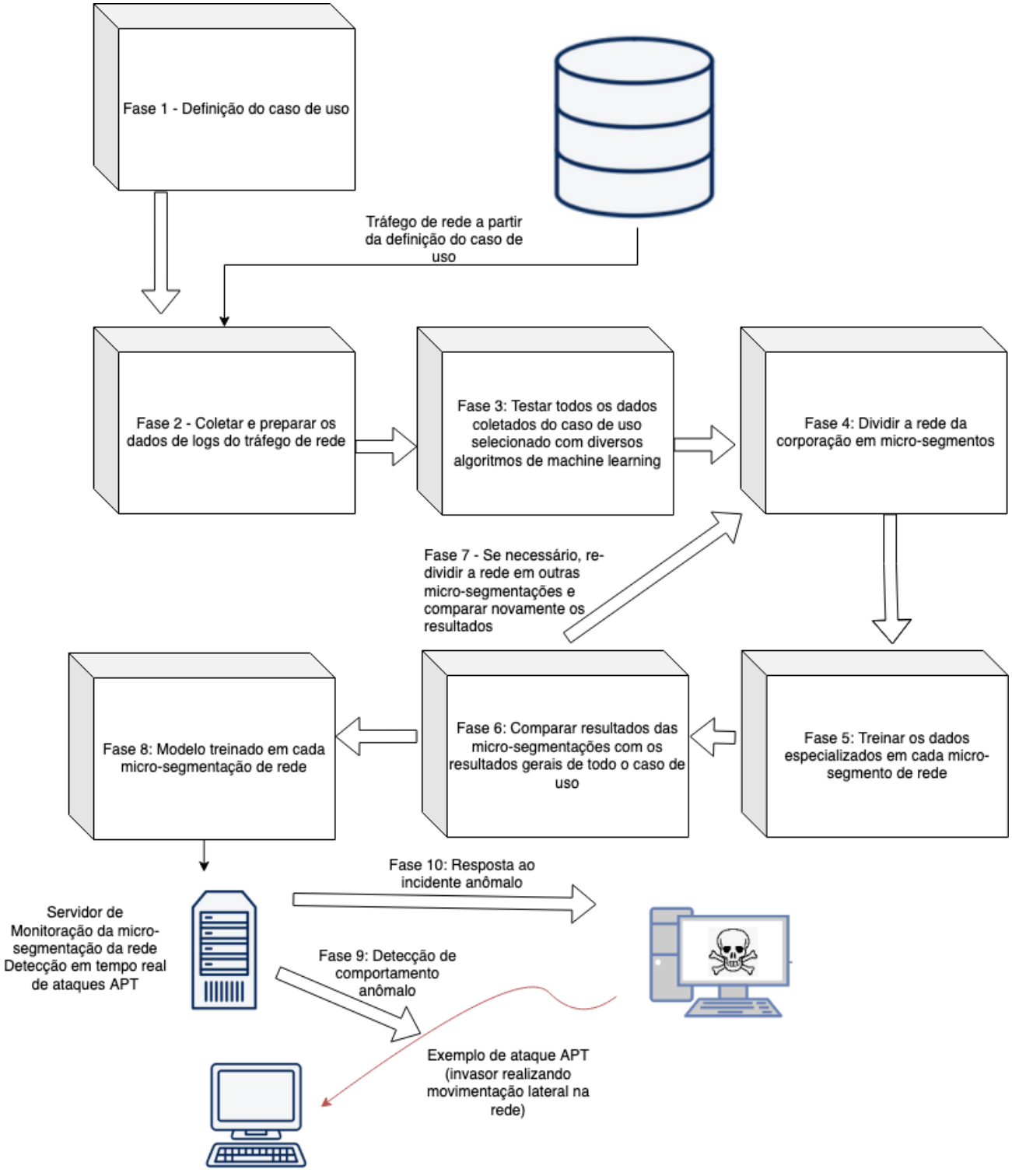

Figura 3.2: Fases do modelo proposto utilizando *zero trust* e UEBA

3.2.4 Fase 4: Dividir a rede da corporação em micro-segmentos

Esta fase depende de uma boa análise da fase 1, do conhecimento do negócio para identificar os ativos mais valiosos, e do conhecimento do analista de redes para saber as melhores pontos a serem micro-segmentados.

3.2.5 Fase 5: Treinar os dados especializados em cada micro-segmento de rede

Em cada um dos micro-segmentos é realizado um novo treinamento do modelo, mas agora com os dados especializados. (Figura 3.3)

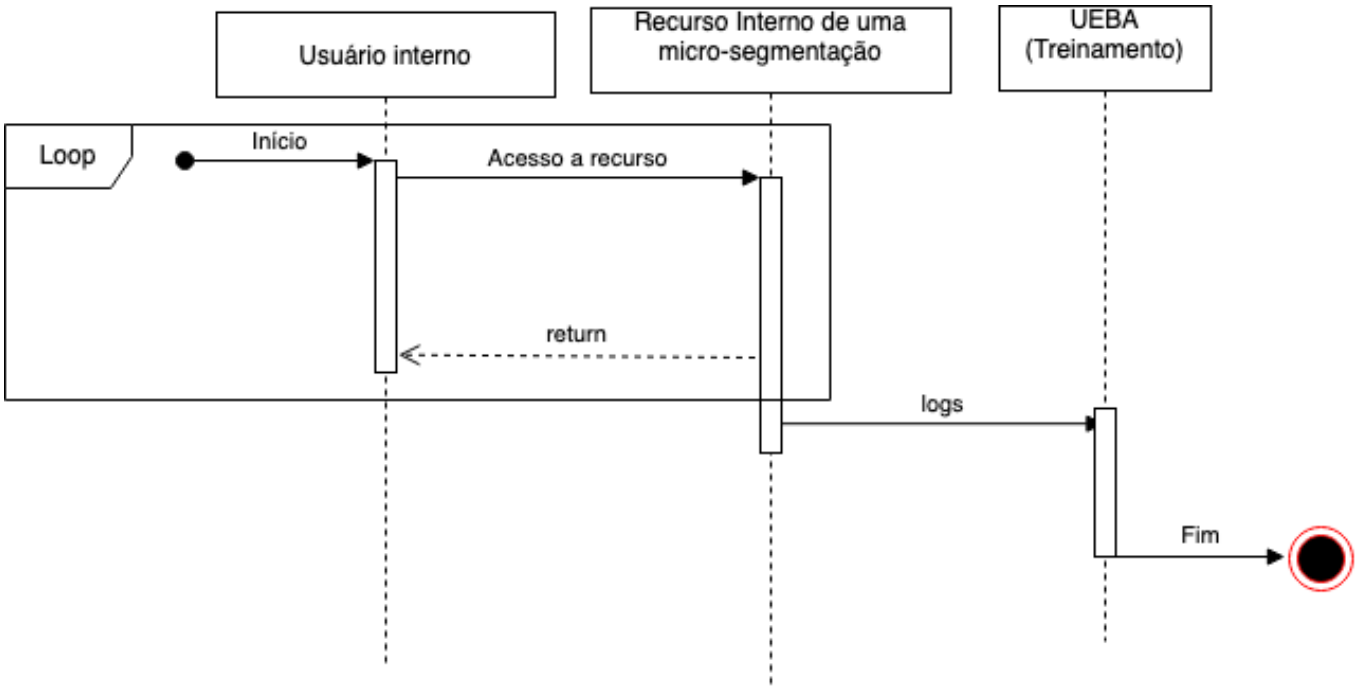

Figura 3.3: Etapa de treino de cada micro-segmento. Fonte: próprio autor

3.2.6 Fase 6: Comparar resultados das micro-segmentações com os resultados gerais de todo o caso de uso

Na fase 6, é comparado o resultado do treinamento de cada micro-segmentação com todo o modelo.

3.2.7 Fase 7: Se necessário, redividir a rede em outras micro-segmentações e comparar novamente os resultados

É importante o modelo estar sempre sendo reavaliado pois a medida que novos dados vão sendo inseridos no treinamento, novas variáveis vão surgindo e os resultados podem ser diferentes. Caso seja necessário, uma nova avaliação das micro-segmentações deve ser refeita.

3.3 ETAPA DE ESCUTA DO ATAQUE

A etapa de escuta é a fase de produção em si. É a fase que o modelo já treinado em cada micro-segmento analisa as conexões de rede e emite alertas em caso de comportamentos anômalos.

3.3.1 Fase 8: Modelo treinado em cada micro-segmentação de rede

A fase 8 é o marco onde delimita que o modelo já foi treinado em cada micro-segmento de rede. Vários testes devem ser feitos e encontrados valores aceitáveis das métricas selecionadas para o caso de uso em questão.

3.3.2 Fase 9: Detecção de comportamento anômalo

A fase 9 é a fase principal de todo o modelo de segurança proposto. O modelo deve realizar um teste no tráfego de rede, em cada conexão e este deve avaliar conforme o treinamento realizado se é um ataque ou não. Caso o modelo não considere como um ataque, este permitirá a execução da transação normalmente. Caso encontre uma atividade anômala, será executada a Fase 10.

3.3.3 Fase 10: Ao encontrar uma atividade anômala, gerar resposta ao incidente

Uma atividade anômala é quando o modelo encontra um desvio de comportamento. Nesse caso, o modelo deve (Figura 3.4):

- **Bloquear o acesso** A primeira resposta ao incidente é bloquear o recurso que está tentando acessar.
- **Solicitar novas credenciais** Solicitar que o usuário troque suas credenciais. Para isso, o usuário deve confirmar sua identidade junto ao administrador de segurança e solicitar novas credenciais.
- **Emitir alerta ao administrador de segurança** Todo incidente deve ser registrado e um alerta deve ser emitido ao administrador de segurança para análise do caso em questão.

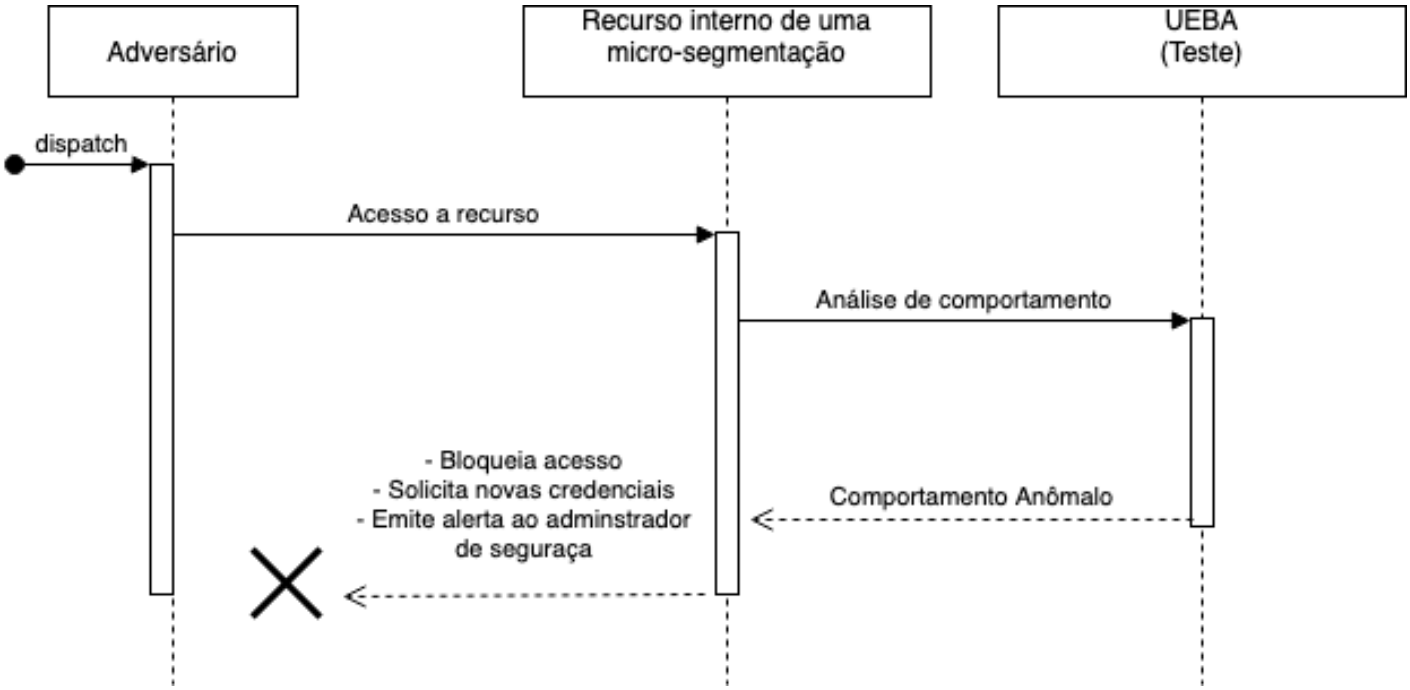

Figura 3.4: Etapa de escuta. Fonte: próprio autor

4 ANÁLISES E RESULTADOS

Nesta capítulo será detalhado um estudo de caso e as ações em cada fase do modelo proposto.

4.1 DATASET ESCOLHIDO - NSL-KDD

O *dataset* escolhido foi o NSL-KDD (46). O NSL-KDD (46) é uma evolução do *dataset* KDD'99 (47). O KDD'99 foi um *dataset* criado para o KDD CUP 1999 (48) onde o objetivo foi criar um modelo preditivo (classificador) capaz de distinguir as conexões de ataque das conexões normais. O arquivo KDD'99 é um arquivo contendo diversos *logs* de conexões TCP que foram simulados em um ambiente de rede militar para uma pesquisa do *1998 DARPA Intrusion Detection Evaluation Program*, realizadas pelo *MIT Lincoln Labs.*

O NSL-KDD foi remapeado pela Universidade de *New Brunswick*, Canadá, e resolveu diversos problemas do KDD'99, como por exemplo:

- Foram retirados registros redundantes no *dataset* de treino (redução de 78%) e de testes(redução de 75%)
- Foram retirados registros duplicados no *dataset* de treino e de testes
- Foram ajustados a quantidade de registros tanto no *dataset* de treino quanto no *dataset* de testes de modo que os testes de *machine learning* possam ser executados no *dataset* como um todo sem a necessidade de escolher aleatoriamente um trecho.

Esta versão NSL-KDD ainda não é perfeita para retratar uma rede de dados real, mas para uma simulação é bastante eficaz no auxílio às pesquisas envolvendo o uso de *machine learning*. O número de registros no *dataset* de treinamento e o número de registros no *dataset* de testes são razoáveis e auxiliam bem no processo de detecção de ataques. A base de treino possui 125.973 registros e a base de testes possui 22.544 registros.

Cada registro do *dataset* corresponde a uma conexão. Esta conexão é uma sequência de pacotes TCP iniciando e terminando em momentos bem definidos, entre os quais os dados fluem de para um endereço IP de origem para um endereço IP de destino sob algum protocolo. Cada conexão é rotulada como normal ou como um ataque. Cada registro de conexão consiste em cerca de 100 bytes (48).

Os ataques se dividem em quatro categorias principais (48):

- **DOS** : Negação de serviço, por exemplo, *sin flood*;
- **R2L** : Acesso não autorizado de uma máquina remota, ex. adivinhar senha;

– **U2R** : Acesso não autorizado a privilégios de superusuário local (root), por exemplo, vários ataques de "estouro de *buffer*";

– ***probing*** : É o chamado *sniffing*, ou sondagem. Exemplo: *port scanning*

O *dataset* de testes não tem a mesma distribuição de probabilidade do *dataset* de treino, além de incluir ataques novos que não estão presentes no *dataset* de treino. Isto torna o *dataset* muito mais realista. O *dataset* de treino possui 24 tipos de ataques diferentes e o *dataset* de teste possui 14 ataques a mais. Acredita-se que ataques considerados novos na maior parte das vezes são variantes de ataques já conhecidos (48).

A lista completa das características presentes neste *dataset* estão descritos na tabela 4.1:

Características individuais das conexões TCP		
feature	**descrição**	tipo
duration	Número em segundos da conexão	contínuo
protocol_type	Tipo de protocolo (TCP, UDP, ICMP...)	discreto
service	Serviço de rede no destino (http, telnet, ftp...)	discreto
src_bytes	Tamanho em bytes da origem para o destino	contínuo
dst_bytes	Tamanho em bytes do destino para origem	contínuo
flag	Conexão classificada em "normal"ou "erro"	discreto
land	1 - Se a conexão tem origem/destino e porta iguais 0 - Se a conexão for diferente	discreto
wrong_fragment	Número de fragmentos inválidos	contínuo
urgent	Número de pacotes urgentes	contínuo

Tabela 4.1: Características individuais das conexões TCP

Além destas informações da conexão TCP, também são descritos no *dataset* as seguintes características:

– número de tentativas de falhas de login ;

– se houve um *login* com sucesso ;

– se o usuário obteve superusuário (*root*) ;

– se o usuário tentou o comando *"su root"* ;

– quantidade de acessos como *root* ;

– quantidade de arquivos criados ;

– quantidade de *shells* abertas ;

– número de operações para acessos em arquivos de controle

– número de comandos inválidos em uma sessão ftp

– se o *login* foi feito como "convidado"ou se está em uma lista "*hot*".

Também foram levantados dados que ocorreram dentro de uma janela de 2 segundos a um mesmo *host* de destino:

- número de conexões na mesma conexão ;
- percentual de conexões com erro "SYN";
- percentual de conexões com erro "REJ";
- percentual de conexões ao mesmo serviço de rede ;
- percentual de conexões em diferentes serviços de rede ;
- número de conexões ao mesmo serviço que a conexão atual nos últimos dois segundos.

E por fim também foram descritos dados que ocorreram em conexões ao mesmo serviço:

- percentual de conexões com erro "SYN";
- percentual de conexões com erro "REJ";
- percentual de conexões para *hosts* diferentes ;

4.2 SIMULAÇÃO

Nesta seção será detalhado todo o processo utilizado para a realizar a simulação do modelo proposto, utilizando o *dataset* NSL-KDD.

4.2.1 Definição e preparação inicial

De acordo com o modelo proposto, a fase 1 consiste na escolha do caso de uso. O caso de uso escolhido foi identificar um usuário que já entrou dentro de um sistema corporativo e está tentando realizar um movimento lateral para uma possível extração de dados. Na fase 2, a coleta e a preparação dos dados de *logs* do tráfego de rede são fundamentais para o treinamento dos diversos algoritmos que serão utilizados no teste.

Neste caso, foi escolhido o dataset NSL KDD para realização da simulação, simulando os dados de logs do tráfego de rede com diversas conexões e seus respectivos atributos. Para realizar o treinamento do modelo e análise dos resultados, foi escolhido o ambiente *Colaboratoy* [1] do Google.

O primeiro passo é importar o *dataset* de treino e o *dataset* de teste, conforme descrito no código *python* abaixo:

[1] https://colab.research.google.com/

```
# UPLOAD Arquivo de TREINO
from scipy.io import arff
from google.colab import files

uploaded = files.upload()

for fn in uploaded.keys():
  print('User uploaded file "{name}" with lengt h {length} bytes'.format(
      name=fn, length=len(uploaded[fn])))

data1 = arff.loadarff('KDDTrain-ajustado.arff')
df = pd.DataFrame(data1[0])


# UPLOAD Arquivo de TESTE
from scipy.io import arff
from google.colab import files

uploaded = files.upload()

for fn in uploaded.keys():
  print('User uploaded file "{name}" with lengt h {length} bytes'.format(
      name=fn, length=len(uploaded[fn])))

data2 = arff.loadarff('KDDTest-ajustado.arff')
dfTest = pd.DataFrame(data2[0])
```

É muito importante preparar os dados para realizar as simulações. Para facilitar o processo de *machine learning* e otimizar a performance de treino e simulações, foi substituído todo o conteúdo que estava em formato texto para formato numérico tanto do *dataset* de treino quanto do *dataset* de testes.

Primeiro, foi realizada a formatação no *dataset* de treino:

```
# REMAP Arquivo TREINO

map = {
    "class" : "resultado",
}
df = df.rename(columns = map)
df.protocol_type = df.protocol_type.map(dict({b'tcp':1,b'udp':2,b'icmp':3}))
df.service = df.service.map(dict({b'aol':1, b'auth':2, b'bgp':3, b'courier':4,
b'csnet_ns':5, b'ctf':6, b'daytime':7, b'discard':8, b'domain':9, b'domain_u':10,
b'echo':11, b'eco_i':12, b'ecr_i':13, b'efs':14, b'exec':15, b'finger':16,
b'ftp':17, b'ftp_data':18, b'gopher':19, b'harvest':20, b'hostnames':21
, b'http':22, b'http_2784':23, b'http_443':24, b'http_8001':25, b'imap4':26,
b'IRC':27, b'iso_tsap':28, b'klogin':29, b'kshell':30, b'ldap':31, b'link':32,
b'login':33, b'mtp':34, b'name':35, b'netbios_dgm':36, b'netbios_ns':37,
b'netbios_ssn':38, b'netstat':39, b'nnsp':40, b'nntp':41, b'ntp_u':42,
b'other':43, b'pm_dump':44, b'pop_2':45, b'pop_3':46, b'printer':47,
b'private':48, b'red_i':49, b'remote_job':50, b'rje':51, b'shell':52,
```

```
b'smtp':53, b'sql_net':54, b'ssh':55, b'sunrpc':56, b'supdup':57, b'systat':58,
b'telnet':59, b'tftp_u':60, b'tim_i':61, b'time':62, b'urh_i':63, b'urp_i':64,
b'uucp':65, b'uucp_path':66, b'vmnet':67, b'whois':68, b'X11':69, b'Z39_50':70}))
df.flag = df.flag.map(dict({b'OTH':1,b'REJ':2,b'RSTO':3,b'RSTOS0':4,
b'RSTR':5,b'S0':6,b'S1':7,b'S2':8,b'S3':9,b'SF':10,b'SH':11}))
df.land = df.land.map(dict({b'0':0, b'1':1}))
df.logged_in = df.logged_in.map(dict({b'0':0, b'1':1}))
df.is_host_login = df.is_host_login.map(dict({b'0':0, b'1':1}))
df.is_guest_login = df.is_guest_login.map(dict({b'0':0, b'1':1}))
df.resultado = df.resultado.map(dict({b'normal':1, b'anomaly':0}))

X_train = df[['duration','protocol_type','service','flag','src_bytes',
'dst_bytes','land','wrong_fragment','urgent','hot','num_failed_logins',
'logged_in','num_compromised','root_shell','su_attempted','num_root',
'num_file_creations','num_shells','num_access_files','num_outbound_cmds',
'is_host_login','is_guest_login','count','srv_count','serror_rate',
'srv_serror_rate','rerror_rate','srv_rerror_rate','same_srv_rate','
diff_srv_rate','srv_diff_host_rate','dst_host_count','dst_host_srv_count',
'dst_host_same_srv_rate','dst_host_diff_srv_rate','dst_host_same_src_port_rate',
'dst_host_srv_diff_host_rate','dst_host_serror_rate','dst_host_srv_serror_rate',
'dst_host_rerror_rate','dst_host_srv_rerror_rate']]
y_train = df['resultado']
```

Em seguida, segue a formatação do *dataset* de testes:

```
# REMAP Arquivo TESTE

map = {
    "class" : "resultado",
}
dfTest = dfTest.rename(columns = map)
dfTest.protocol_type = dfTest.protocol_type.map(dict({b'tcp':1,b'udp':2,
b'icmp':3}))
dfTest.service = dfTest.service.map(dict({b'aol':1, b'auth':2, b'bgp':3,
b'courier':4,b'csnet_ns':5, b'ctf':6, b'daytime':7, b'discard':8, b'domain':9,
b'domain_u':10,b'echo':11, b'eco_i':12, b'ecr_i':13, b'efs':14, b'exec':15,
b'finger':16,b'ftp':17, b'ftp_data':18, b'gopher':19, b'harvest':20,
b'hostnames':21, b'http':22, b'http_2784':23, b'http_443':24, b'http_8001':25,
b'imap4':26,b'IRC':27, b'iso_tsap':28, b'klogin':29, b'kshell':30, b'ldap':31,
b'link':32,b'login':33, b'mtp':34, b'name':35, b'netbios_dgm':36,
b'netbios_ns':37,b'netbios_ssn':38, b'netstat':39, b'nnsp':40, b'nntp':41,
b'ntp_u':42,b'other':43, b'pm_dump':44, b'pop_2':45, b'pop_3':46, b'printer':47,
b'private':48, b'red_i':49, b'remote_job':50, b'rje':51, b'shell':52,
b'smtp':53, b'sql_net':54, b'ssh':55, b'sunrpc':56, b'supdup':57, b'systat':58,
b'telnet':59, b'tftp_u':60, b'tim_i':61, b'time':62, b'urh_i':63, b'urp_i':64,
b'uucp':65, b'uucp_path':66, b'vmnet':67, b'whois':68, b'X11':69, b'Z39_50':70}))
dfTest.flag = dfTest.flag.map(dict({b'OTH':1,b'REJ':2,b'RSTO':3,b'RSTOS0':4,
b'RSTR':5,b'S0':6,b'S1':7,b'S2':8,b'S3':9,b'SF':10,b'SH':11}))
dfTest.land = dfTest.land.map(dict({b'0':0, b'1':1}))
dfTest.logged_in = dfTest.logged_in.map(dict({b'0':0, b'1':1}))
dfTest.is_host_login = dfTest.is_host_login.map(dict({b'0':0, b'1':1}))
```

```
dfTest.is_guest_login = dfTest.is_guest_login.map(dict({b'0':0, b'1':1}))
dfTest.resultado = dfTest.resultado.map(dict({b'normal':1, b'anomaly':0}))



X_test = dfTest[['duration','protocol_type','service','flag','src_bytes',
'dst_bytes','land','wrong_fragment','urgent','hot','num_failed_logins',
'logged_in','num_compromised','root_shell','su_attempted','num_root',
'num_file_creations','num_shells','num_access_files','num_outbound_cmds',
'is_host_login','is_guest_login','count','srv_count','serror_rate',
'srv_serror_rate','rerror_rate','srv_rerror_rate','same_srv_rate','
diff_srv_rate','srv_diff_host_rate','dst_host_count','dst_host_srv_count',
'dst_host_same_srv_rate','dst_host_diff_srv_rate','dst_host_same_src_port_rate',
'dst_host_srv_diff_host_rate','dst_host_serror_rate','dst_host_srv_serror_rate',
'dst_host_rerror_rate','dst_host_srv_rerror_rate']]
y_test = dfTest['resultado']
```

4.2.2 Etapa de Treino e testes dos algoritmos

A fase 3 consiste em treinar o *dataset* do caso de uso selecionado com diversos algoritmos de *machine learning*. E em seguida avaliar cada um dos algoritmos através de métricas conhecidas. Para este caso de uso, serão treinados os algoritmos supervisionados *Support Vector Machine* (Classificador SVC), *K-Nearest Neighbor* (KNN), Árvores de Decisão (AD), Redes Neurais (RN), *Naive Bayes* (NB) e *Random Forest* (RF). A escolha de algoritmos supervisionados deve-se ao fato do *dataset* escolhido estar rotulada com a classe resultado (ataque ou não ataque), então, na fase de testes serão feitas comparações da previsão do algoritmo com o valor verdadeiro.

4.2.2.1 Algoritmo *Support Vector Machine* - SVC

Abaixo, o código fonte em *python* utilizado para o treinamento utilizando o algoritmo SVC (supervisionado, baseado em classificação).

```
from sklearn.preprocessing import StandardScaler
from sklearn import svm
from sklearn.metrics import accuracy_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import precision_score


scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

clf_name = 'SVC'
```

```
clf = svm.SVC(C=1.0)
clf.fit(X_train, y_train)
previsoes = clf.predict(X_test)
acuracia = accuracy_score(y_test, previsoes) * 100
print("Acuracia: %.2f%% " % acuracia)
recall = recall_score(y_test, previsoes) * 100
print("Recall: %.2f%% " % recall )
f1 = f1_score(y_test, previsoes) * 100
print("F1 Score: %.2f%% " % f1 )
precision = precision_score(y_test, previsoes) * 100
print("Precisao: %.2f%% " % precision )
```

Após o treinamento do algoritmo, foram realizados diversos testes com a base que foi destinada para os testes. Os resultados coletados após os testes do algoritmo SVC estão referenciados na tabela 4.2:

Algoritmo *Support Vector Machine* - SVC	
Métrica	**Percentual**
Acurácia	77,32%
Recall	97,80%
F1 Score	78,79%
Precisão	65,97%

Tabela 4.2: Algoritmo *Support Vector Machine* - SVC

4.2.2.2 Algoritmo KNN

Abaixo, o código fonte em *python* utilizado para o treinamento e testes utilizando o algoritmo KNN (supervisionado e baseado em classificação).

```
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import precision_score

scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

clf_name = 'KNN'
clf = KNeighborsClassifier()
clf.fit(X_train, y_train)
previsoes = clf.predict(X_test)
acuracia = accuracy_score(y_test, previsoes) * 100
print("Acuracia: %.2f%% " % acuracia )
recall = recall_score(y_test, previsoes) * 100
print("Recall: %.2f%% " % recall)
```

```
21 f1 = f1_score(y_test, previsoes) * 100
22 print("F1 Score: %.2f%% " % f1 )
23 precision = precision_score(y_test, previsoes) * 100
24 print("Precisao: %.2f%% " % precision )
```

Resultados coletados após os testes do algoritmo KNN estão referenciados na tabela 4.3:

Algoritmo KNN	
Métrica	**Percentual**
Acurácia	76,67%
Recall	97,64%
F1 Score	78,29%
Precisão	65,33%

Tabela 4.3: Algoritmo KNN

4.2.2.3 Algoritmo Árvores de Decisão

Abaixo, o código fonte em python utilizado para o treinamento e testes utilizando o algoritmo Árvores de Decisão (supervisionado e classificador).

```
1  from sklearn.preprocessing import StandardScaler
2  from sklearn.tree import DecisionTreeClassifier
3  from sklearn.metrics import accuracy_score
4  from sklearn.metrics import recall_score
5  from sklearn.metrics import f1_score
6  from sklearn.metrics import precision_score
7
8  scaler = StandardScaler()
9  scaler.fit(X_train)
10 X_train = scaler.transform(X_train)
11 X_test = scaler.transform(X_test)
12
13 clf_name = 'Decision Tree'
14 clf = DecisionTreeClassifier(max_depth=5)
15 clf.fit(X_train, y_train)
16 previsoes = clf.predict(X_test)
17 acuracia = accuracy_score(y_test, previsoes) * 100
18 print("Acuracia: %.2f%% " % acuracia)
19 recall = recall_score(y_test, previsoes) * 100
20 print("Recall: %.2f%% " % recall)
21 f1 = f1_score(y_test, previsoes) * 100
22 print("F1 Score: %.2f%% " % f1)
23 precision = precision_score(y_test, previsoes)  * 100
24 print("Precisao: %.2f%% " % precision)
```

Resultados coletados após os testes do algoritmo de Árvores de Decisão estão referenciados na tabela 4.4:

Algoritmo Árvores de Decisão	
Métrica	**Percentual**
Acurácia	79,12%
Recall	96,53%
F1 Score	79,93%
Precisão	68,20%

Tabela 4.4: Algoritmo Árvores de Decisão

4.2.2.4 Algoritmo Redes Neurais

Abaixo, o código fonte em *python* utilizado para o treinamento e testes utilizando o algoritmo Redes Neurais *Multilayer Perceptron* (supervisionado com a utilização do método de classificação).

```
1  from sklearn.preprocessing import StandardScaler
2  from sklearn.neural_network import MLPClassifier
3  from sklearn.metrics import accuracy_score
4  from sklearn.metrics import recall_score
5  from sklearn.metrics import f1_score
6  from sklearn.metrics import precision_score
7
8  scaler = StandardScaler()
9  scaler.fit(X_train)
10 X_train = scaler.transform(X_train)
11 X_test = scaler.transform(X_test)
12
13 clf_name = 'Redes Neurais'
14 clf = MLPClassifier(alpha=1, max_iter=1000)
15 clf.fit(X_train, y_train)
16 previsoes = clf.predict(X_test)
17 acuracia = accuracy_score(y_test, previsoes) * 100
18 print("Acuracia: %.2f%% " % acuracia)
19 recall = recall_score(y_test, previsoes) * 100
20 print("Recall: %.2f%% " % recall)
21 f1 = f1_score(y_test, previsoes) * 100
22 print("F1 Score: %.2f%% " % f1)
23 precision = precision_score(y_test, previsoes) * 100
24 print("Precisao: %.2f%% " % precision)
```

Resultados coletados após os testes do algoritmo de redes neurais estão referenciados na tabela 4.5:

Algoritmo Redes Neurais - MLP	
Métrica	**Percentual**
Acurácia	76,27%
Recall	97,34%
F1 Score	77,94%
Precisão	64,99%

Tabela 4.5: Algoritmo Redes Neurais - MLP

4.2.2.5 Algoritmo *Naive Bayes*

Abaixo, o código fonte em *python* utilizado para o treinamento e testes utilizando o algoritmo *Naive Bayes*.

```
from sklearn.preprocessing import StandardScaler
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import precision_score

scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

clf_name = 'Naive Bayes'
clf =  GaussianNB()
clf.fit(X_train, y_train)
previsoes = clf.predict(X_test)
acuracia = accuracy_score(y_test, previsoes) * 100
print("Acuracia: %.2f%% " % acuracia)
recall = recall_score(y_test, previsoes) * 100
print("Recall: %.2f%% " % recall)
f1 = f1_score(y_test, previsoes) * 100
print("F1 Score: %.2f%% " % f1)
precision = precision_score(y_test, previsoes) * 100
print("Precisao: %.2f%% " % precision)
```

Resultados coletados após os testes do algoritmo NB estão referenciados na tabela 4.6:

Algoritmo *Naive Bayes*	
Métrica	**Percentual**
Acurácia	77,19%
Recall	91,27%
F1 Score	77,51%
Precisão	67,36%

Tabela 4.6: Algoritmo *Naive Bayes*

4.2.2.6 Algoritmo *Random Forest*

Abaixo, o código fonte em *python* utilizado para o treinamento e testes utilizando o algoritmo *Random Forest*.

```
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
```

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import precision_score

scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

clf_name = 'Random Forest'
clf = RandomForestClassifier(max_depth=5, n_estimators=10, max_features=1)
clf.fit(X_train, y_train)
previsoes = clf.predict(X_test)
acuracia = accuracy_score(y_test, previsoes) * 100
print("Acuracia: %.2f%% " % acuracia)
recall = recall_score(y_test, previsoes) * 100
print("Recall: %.2f%% " % recall)
f1 = f1_score(y_test, previsoes) * 100
print("F1 Score: %.2f%% " % f1)
precision = precision_score(y_test, previsoes) * 100
print("Precisao: %.2f%% " % precision)
```

Resultados coletados após os testes do algoritmo *Random Forest* estão referenciados na tabela 4.7:

Algoritmo *Random Forest*	
Métrica	**Percentual**
Acurácia	73,54%
Recall	97,37%
F1 Score	76,02%
Precisão	62,35%

Tabela 4.7: Algoritmo *Random Forest*

4.2.3 Dividir a rede e treinar novamente

A fase 4 consiste em dividir a rede a corporação em micro-segmentos utilizando os conceitos básicos de *zero trust*. Para realizar esta divisão na simulação, foram escolhidos três serviços com grande massa de dados no *dataset* para que o treinamento pudesse ser feito de forma consistente.

Os serviços (coluna ***service*** do NSL-KDD, referente ao serviço de rede do destino da conexão TCP) escolhidos foram:

- **Servidor WEB - Protocolo HTTP** - Contém 40.338 registros no dataset de treino e 7.853 registros no *dataset* de testes
- **Servidores de envio de *e-mails* - protocolo SMPT** - Contém 7.313 registros no dataset de treino e 934 registros no *dataset* de testes

– **Servidores de DNS (domínio)** - Contém 9.043 registros no *dataset* de treino e 894 registros no *dataset* de testes

Na fase 5, é necessário definir a melhor métrica para ser comparada. Como a necessidade do caso de uso é diminuir o percentual de falsos negativos, a métrica *recall* é a indicada para se observar qual algoritmo de *machine learning* foi melhor treinado. A tabela 4.8 mostra o comparativo de *recall* entre os algoritmos, em ordem decrescente do melhor resultado.

Comparativo *recall* entre os algoritmos de ML	
Algoritmo	**Percentual de *recall***
Support Vector Machine - SVC	**97,80%**
KNN	97,64%
Random Forest	97,37%
Redes Neurais	97,34%
Árvores de Decisão	96,53%
Naive Bayes	91,27%

Tabela 4.8: Comparativo *recall* entre os algoritmos de ML

Tendo escolhido o algoritmo *Support Vector Machine* (SVC) como o melhor algoritmo para este *dataset*, o próximo passo é treinar e testar novamente o modelo especificamente em cada micro-segmentação de rede utilizando este mesmo algoritmo. Para isso, o *dataset* foi dividido de forma que cada micro-segmentação receba apenas os dados do serviço escolhido.

Primeiro foi gerada a curva ROC de todo o *dataset*, comparando a relação das taxas de verdadeiro positivo TPR (Sensibilidade) com as taxas de falsos positivos FPR (Especificidade) (Figura 4.1). O AUC encontrado foi de: 0.9540054332461709.

```
import numpy as np
from sklearn import metrics
fpr, tpr, thresholds = metrics.roc_curve(y_test, y_test_pred, pos_label=1)
print(metrics.auc(fpr, tpr))


import matplotlib.pyplot as plt
from sklearn import datasets, metrics, model_selection, svm
metrics.plot_roc_curve(clf, X_test, y_test)
# print(metrics.auc(fpr, tpr))
print(metrics.roc_auc_score(y_test, y_test_scores))
plt.show()
```

E em seguida foram realizados diversos testes na micro-segmentação referente aos servidores WEB (protocolo HTTP) que corresponde ao serviço 22 no *dataset*:

```
filtro_service = df.service == 22
filtro_service.head()
```

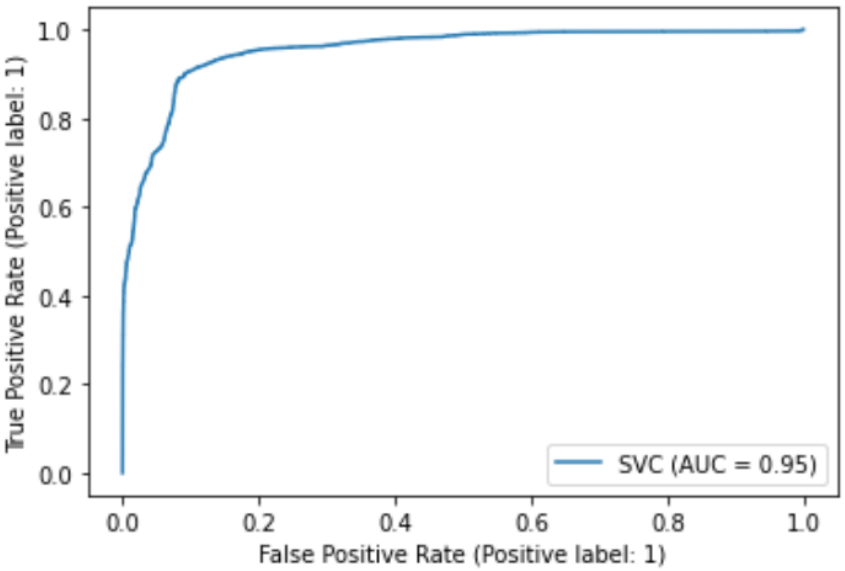

Figura 4.1: Curva ROC e AUC - Modelo geral de todo o dataset

```
df = df[filtro_service]

filtro_serviceTest = dfTest.service == 22
dfTest = dfTest[filtro_serviceTest]

from sklearn.preprocessing import StandardScaler
from sklearn import svm
from sklearn.metrics import accuracy_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import precision_score

scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

clf_name = 'SVC'
clf = svm.SVC(C=1.0)
clf.fit(X_train, y_train)
previsoes = clf.predict(X_test)
acuracia = accuracy_score(y_test, previsoes) * 100
print("Acuracia: %.2f%% " % acuracia)
recall = recall_score(y_test, previsoes) * 100
print("Recall: %.2f%% " % recall )
f1 = f1_score(y_test, previsoes) * 100
print("F1 Score: %.2f%% " % f1 )
precision = precision_score(y_test, previsoes) * 100
print("Precisao: %.2f%% " % precision )
```

Os resultados coletados após o treinamento e testes do algoritmo SVC na micro-segmentação de servi-

dores WEB (protocolo HTTP), serviço 22, estão descritos na tabela 4.9:

Algoritmo SVC na micro-segmentação servidor WEB	
Métrica	**Percentual**
Acurácia	92,02%
Recall	**99,74%**
F1 Score	95,0%
Precisão	91,61%

Tabela 4.9: Algoritmo SVC na micro-segmentação servidor WEB

A curva ROC foi gerada para esta micro-segmentação (Figura 4.2) e foi encontrando um AUC de: 0.9962469552230465

```
import numpy as np
from sklearn import metrics
fpr, tpr, thresholds = metrics.roc_curve(y_test, y_test_pred, pos_label=1)
print(metrics.auc(fpr, tpr))


import matplotlib.pyplot as plt
from sklearn import datasets, metrics, model_selection, svm
metrics.plot_roc_curve(clf, X_test, y_test)
# print(metrics.auc(fpr, tpr))
print(metrics.roc_auc_score(y_test, y_test_scores))
plt.show()
```

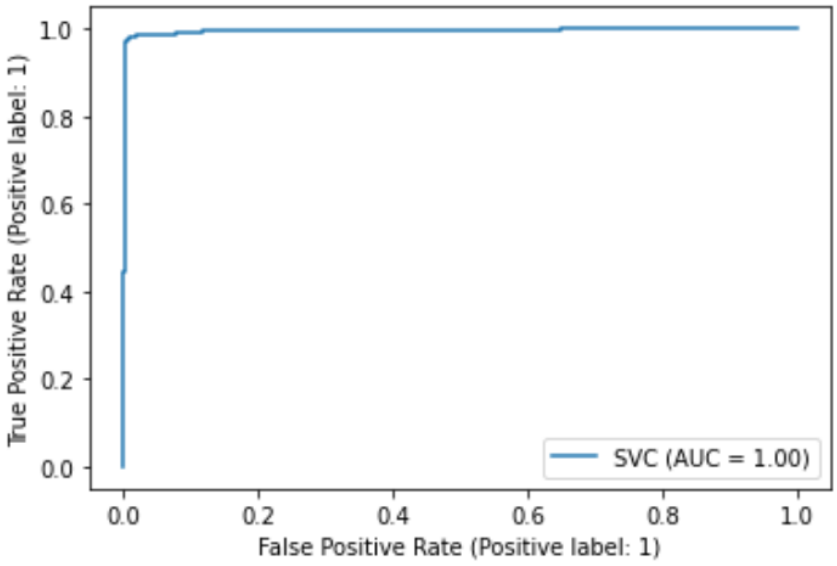

Figura 4.2: Curva ROC e AUC - micro-segmentação servidor HTTP

Depois foram realizados mais testes na micro-segmentação referente aos servidores de envio de e-mails (protocolo SMPT) que corresponde ao serviço 53 no *dataset*:

```
filtro_service = df.service == 53
filtro_service.head()
df = df[filtro_service]

filtro_serviceTest = dfTest.service == 53
dfTest = dfTest[filtro_serviceTest]

from sklearn.preprocessing import StandardScaler
from sklearn import svm
from sklearn.metrics import accuracy_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import precision_score

scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

clf_name = 'SVC'
clf = svm.SVC(C=1.0)
clf.fit(X_train, y_train)
previsoes = clf.predict(X_test)
acuracia = accuracy_score(y_test, previsoes) * 100
print("Acuracia: %.2f%% " % acuracia)
recall = recall_score(y_test, previsoes) * 100
print("Recall: %.2f%% " % recall )
f1 = f1_score(y_test, previsoes) * 100
print("F1 Score: %.2f%% " % f1 )
precision = precision_score(y_test, previsoes) * 100
print("Precisao: %.2f%% " % precision )
```

Os resultados coletados após o treinamento e testes do algoritmo SVC na micro-segmentação de servidores de envio de e-mails (protocolo SMPT), serviço 53, estão descritos na tabela 4.10:

Algoritmo SVC na micro-segmentação servidor de envio de e-mails	
Métrica	**Percentual**
Acurácia	67,77%
Recall	**100,00**%
F1 Score	80,41%
Precisão	67,24%

Tabela 4.10: Algoritmo SVC na micro-segmentação servidor de envio de e-mails

A curva ROC foi gerada para esta micro-segmentação (Figura 4.3) e foi encontrando um AUC de: 0.9462178935725698

```
import numpy as np
from sklearn import metrics
```

```
fpr, tpr, thresholds = metrics.roc_curve(y_test, y_test_pred, pos_label=1)
print(metrics.auc(fpr, tpr))


import matplotlib.pyplot as plt
from sklearn import datasets, metrics, model_selection, svm
metrics.plot_roc_curve(clf, X_test, y_test)
# print(metrics.auc(fpr, tpr))
print(metrics.roc_auc_score(y_test, y_test_scores))
plt.show()
```

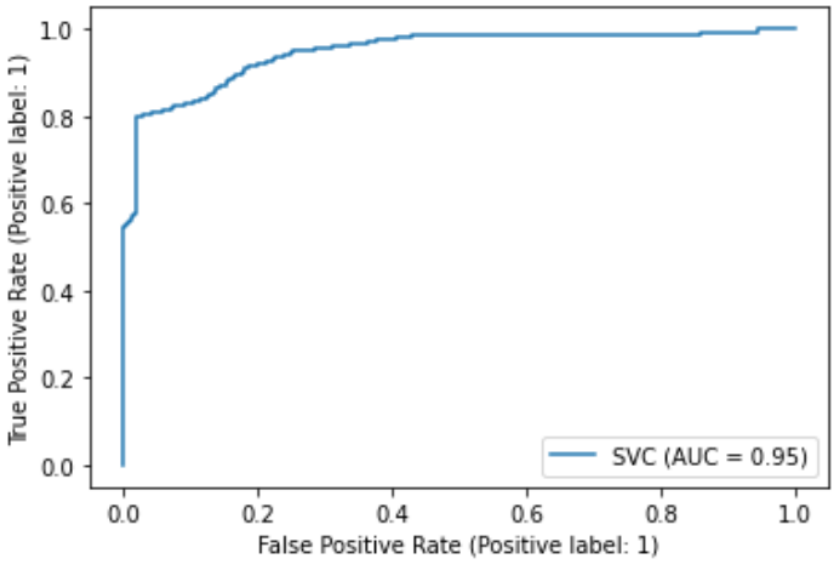

Figura 4.3: Curva ROC e AUC - micro-segmentação servidor SMPT

E por último foram realizados testes na micro-segmentação referente aos servidores de DNS (domínio) que corresponde ao serviço 10 no *dataset*:

```
filtro_service = df.service == 10
filtro_service.head()
df = df[filtro_service]

filtro_serviceTest = dfTest.service == 10
dfTest = dfTest[filtro_serviceTest]

from sklearn.preprocessing import StandardScaler
from sklearn import svm
from sklearn.metrics import accuracy_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import precision_score

scaler = StandardScaler()
scaler.fit(X_train)
```

```
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

clf_name = 'SVC'
clf = svm.SVC(C=1.0)
clf.fit(X_train, y_train)
previsoes = clf.predict(X_test)
acuracia = accuracy_score(y_test, previsoes) * 100
print("Acuracia: %.2f%% " % acuracia)
recall = recall_score(y_test, previsoes) * 100
print("Recall: %.2f%% " % recall )
f1 = f1_score(y_test, previsoes) * 100
print("F1 Score: %.2f%% " % f1 )
precision = precision_score(y_test, previsoes) * 100
print("Precisao: %.2f%% " % precision )
```

Os resultados coletados após o treinamento e testes do algoritmo SVC na micro-segmentação de servidores de DNS (domínio), serviço 10, estão descritos na tabela 4.11:

Algoritmo SVC na micro-segmentação servidor de envio de DNS	
Métrica	**Percentual**
Acurácia	99,77%
Recall	**100,00**%
F1 Score	99,88%
Precisão	99,77%

Tabela 4.11: Algoritmo SVC na micro-segmentação servidor de envio de DNS

A curva ROC foi gerada para esta micro-segmentação (Figura 4.4) e foi encontrando um AUC de: 0.9983183856502242

```
import numpy as np
from sklearn import metrics
fpr, tpr, thresholds = metrics.roc_curve(y_test, y_test_pred, pos_label=1)
print(metrics.auc(fpr, tpr))


import matplotlib.pyplot as plt
from sklearn import datasets, metrics, model_selection, svm
metrics.plot_roc_curve(clf, X_test, y_test)
# print(metrics.auc(fpr, tpr))
print(metrics.roc_auc_score(y_test, y_test_scores))
plt.show()
```

Uma comparação entre os resultados atingidos em cada micro-segmentação de rede juntamente com o resultado total foi realizada cumprindo o requisito da fase 6 4.12.

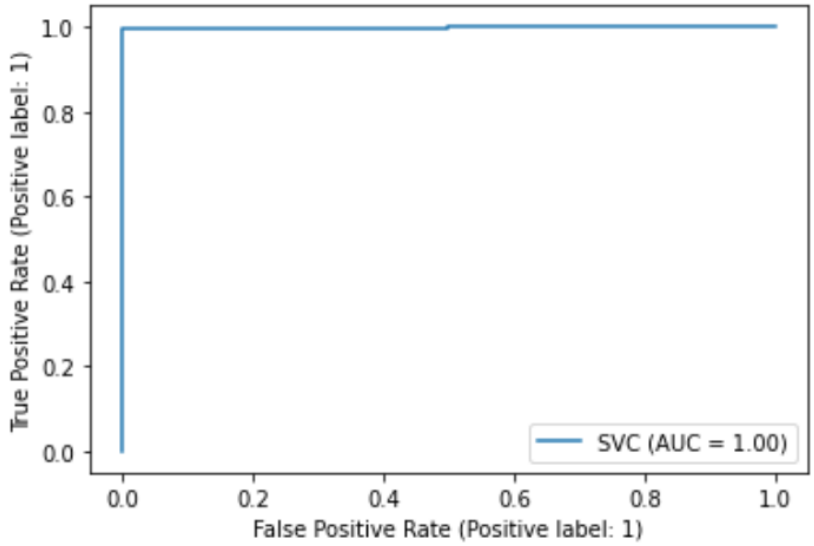

Figura 4.4: Curva ROC e AUC - micro-segmentação servidor DNS

Análise comparativa do algoritmo SVC	
Micro-Segmentação	***recall***
Todo o *dataset*	97,80%
Servidor WEB	99,74%
Servidor de Envio de E-mails	100%
Servidor de Envio de DNS	100%

Tabela 4.12: Análise comparativa do algoritmo SVC

Como foram obtidos valores de *recall* maiores na a micro-segmentação do que em todo o sistema, não foi necessário redividir a rede para cumprir os requisitos da fase 7.

4.2.4 Etapa de Escuta do ataque

A etapa de escuta envolve monitorar todas as atividades de rede, sempre checando no modelo se um tráfego é comum ou ataque, como é descrito na fase 8. A fase 9 e a fase 10 correspondem a detecção e resposta para a atividade anômala, que nesse caso seria bloquear o acesso, solicitar ao usuário gerar novas credenciais de acesso e gerar um alerta de ataque ao administrador de segurança. Após a conclusão da análise, este incidente será adicionado aos treinamento de rede para melhorar a eficácia do modelo.

5 CONCLUSÃO E TRABALHOS FUTUROS

5.1 CONCLUSÃO

O principal objetivo deste trabalho foi evitar o *advanced persitent threats* (APT) em redes corporativas utilizando uma abordagem reativa, diferente do modelo de segurança tradicional. Este objetivo foi atingido com sucesso pois a abordagem foi tanto proativa quanto adaptativa. A abordagem foi proativa pois além da implementação de conceitos de *zero-trust* na rede, como a micro-segmentação lógica e a utlização de NGFWs como PDP/PEP, também foi analisado o comportamento humano e uma análise de possíveis anomalias comportamentais de segurança na rede também foi realizada. A abordagem também foi adaptativa, pois o modelo proposto sugere redividir a rede em outras micro-segmentações na fase 7 e, se for necessário, treinar novamente os dados especializados para que uma nova análise do algoritmo de treinamento seja feita.

É perceptível as vantagens de utilização da filosofia de *zero trust*. Criar políticas de rede separadas para cada micro-segmentação impede a não utilização de recursos de rede por usuários não autorizados. Porém, foi possível observar que a implementação de micro-segmentação exige uma grande complexidade operacional e há pouca automação. A exigência de grande envolvimento humano ainda é necessária nesse tipo de abordagem. Se o trabalho de micro-segmentação não for feito corretamente, a comunicação pode ficar inacessível em algum ponto da rede e depender de uma ação humana para corrigir os possíveis problemas.

Em compensação, a utilização de conceitos de *zero trust* se mostrou bastante eficaz pois além de micro-segmentar a rede e utilizar políticas específicas para cada parte, foi possível especializar o modelo UEBA e de forma que pudesse diminuir a incidência de falsos negativos e aumentasse a detecção de ataques. Em alguns casos, possível observar mais de 5% de aumento na taxa de detecção, o que pode ser considerado uma evidência muito importante quando o assunto é APT.

5.2 TRABALHOS FUTUROS

Para os trabalhos futuros, é proposto:

- Providenciar novas simulações em diferentes bases de dados, se possível, em bases de dados reais para que o modelo proposto seja aplicado em um ambiente mais próximo da realidade;
- Realizar testes em bases de dados não rotuladas para que seja possível analisar o comportamento de algoritmos de *machine learning* não supervisionados;
- Aprofundar o estudo de *zero trust*, aplicando utilizando melhor os benefícios dos NGFWs, que são capazes de analisar a camada de aplicação do modelo OSI;
- Desenvolver mais testes em mais algoritmos de *machine learning* e em um período de tempo maior;
- Pesquisar outros ataques APT às quais o modelo proposto pode estar suscetível, para propor e implementar ações que tornem o modelo mais eficaz;
- A criação deste modelo não se encerra aqui, mas deve ser um processo contínuo de constante desenvolvimento, já que sempre há novas ameaças sempre surgindo. Uma revisão do modelo deve ser feita regularmente a fim de melhorar cada vez mais o treinamento dos algoritmos e a capacidade de detecção de ataques na rede corporativa.

REFERÊNCIAS BIBLIOGRÁFICAS

1 LI, X.; CHEN, T.; CHENG, Q.; MA, S.; MA, J. Smart applications in edge computing: Overview on authentication and data security. *IEEE Internet of Things Journal*, 2021.

2 GHAFIR, I.; HAMMOUDEHC, M.; PRENOSILB, V.; LIANGXIUHANC; HEGARTYC, R.; RABIEC, K.; APARICIO-NAVARROD, F. J. Detection of advanced persistent threat using machine-learning correlation analysis. *Future Generation Computer Systems*, 2018.

3 KASPERSKY. *O que é um ataque de dia zero? – Definição e explicação*. 2022. Acessado em 20 de julho de 2022. Disponível em: <https://www.kaspersky.com.br/resource-center/definitions/zero-day-exploit>.

4 ALSHAMRANI, A.; MYNENI, S.; CHOWDHARY, A.; HUANG, D. A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities. *IEEE Comunications Surveys Tutorials*, 2019.

5 ROCHA, B. C. da; MELO, L. P. de; JR., R. T. de S. A study on apt in iot networks. *18th International Conference on e-Business (ICE-B)*, 2021.

6 ROCHA, B. C. da; MELO, L. P. de; JR., R. T. de S. Preventing apt attacks on lan networks with connected iot devices using a zero trust based security model. *2021 Workshop on Communication Networks and Power Systems (WCNPS)*, 2021.

7 ZHANG, Q.; LI, H.; HU, J. A study on security framework against advanced persistent threat. *7th IEEE International Conference on Electronics Information and Emergency Communication*, 2018.

8 NIST. advanced persistent threat. *URL: <https://csrc.nist.gov/glossary/term/advanced_persistent_threat/> . Acessado em 10 de Novembro de 2022.*

9 ARUBA. The ciso's guide to machine learning user and entity behavioral analytics. *URL: <https://www.arubanetworks.com/assets/CisoGuide.pdf./> . Acessado em 25 de Agosto de 2022.*, 2017.

10 YANG, L.-X.; HUANG, K.; YANG, X.; ZHANG, Y.; XIANG, Y.; TANG, Y. Y. Defense against advanced persistent threat through data backup and recovery. *IEEE Transactions on Network Science and Engineering*, 2020.

11 FERRER, Z.; FERRER, M. C. In-depth analysis of hydraq. *The face of cyberwar enemies unfolds. ca isbu-isi white paper, vol. 37*, 2010.

12 LANGNER, R. Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Security & Privacy, vol. 9, no. 3, pp. 49–51*, 2011.

13 FALLIERE, N.; MURCHU, L. O.; CHIEN, E. W32. stuxnet dossier. *White paper, Symantec Corp., Security Response, vol. 5, no. 6, p. 29*, 2011.

14 JOHNSON, A. L. Cybersecurity for financial institutions: The integral role of information sharing in cyber attack mitigation. *NC Banking Inst., vol. 20, p. 277*, 2016.

15 MICROSOFT. Hafnium targeting exchange servers with 0-day exploits. *URL: <https://www.microsoft.com/en-us/security/blog/2021/03/02/hafnium-targeting-exchange-servers/> . Acessado em 20 de Outubro de 2022.*

16 KERMAN, A.; BORCHERT, O.; ROSE, S.; DIVISION, E.; TAN, A. Implementing a zero trust architecture. *URL: <https://www.nccoe.nist.gov/sites/default/files/legacy-files/zt-arch-project-description-draft.pdf> . Acessado em 2 de setembro de 2021.*

17 NIST. Zero trust architecture. *URL: <https://doi.org/10.6028/NIST.SP.800-207> . Acessado em 20 de agosto de 2021.*

18 GARTNER. Next-generation firewalls (ngfws). *URL: <https://www.gartner.com/en/information-technology/glossary/next-generation-firewalls-ngfws> . Acessado em 25 de Agosto de 2021.*

19 PRITZ, A. Security analytics. *John Wiley Sons*, 2018.

20 CHANDOLA, A. B. V.; KUMAR, V. Anomaly detection: A survey, volume 41. *ACM Computing Surveys*, 2009.

21 LOSHIN, P. Techtarget - what is user (and entity) behavior analytics (uba or ueba)? *URL: <https://www.techtarget.com/searchsecurity/definition/user-behavior-analytics-UBA> . Acessado em 25 de Agosto, 2022.*

22 SALITIN, M. A.; ZOLAIT, A. H. The role of user entity behavior analytics to detect network attacks in real time. *2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, 2018.

23 GONZALEZ, C. Exabeam - user and entity behavior analytics. *URL: <https://www.exabeam.com/ueba/user-and-entity-behavior-analytics/> . Acessado em 25 de Agosto de 2022.*

24 IBM. Security information and event management explained. *URL: <https://www.ibm.com/topics/siem> . Acessado em 3 de Novembro de 2022.*

25 CHUVAKIN, A. The coming uba / ueba – siem war! *URL: <https://blogs.gartner.com/anton-chuvakin/2016/11/07/the-coming-uba-ueba-siem-war/> . Acessado em 3 de Novembro de 2022.*

26 GERON, D. Machine learning: This book includes: Machine learning for beginners, machine learning with python (english edition). *Amazon*, 2019.

27 SCIKITLEARN. Support vector machines. *URL: <https://scikit-learn.org/stable/modules/svm.html> . Acessado em 25 de Janeiro de 2022.*, 2022.

28 SCIKITLEARN. Nearest neighbors. *URL: <https://scikit-learn.org/stable/modules/neighbors.html> . Acessado em 25 de Janeiro de 2022.*, 2022.

29 SCIKITLEARN. Decision trees. *URL: <https://scikit-learn.org/stable/modules/tree.html> . Acessado em 25 de Janeiro de 2022.*, 2022.

30 SCIKITLEARN. Neural network models (supervised). *URL: <https://scikit-learn.org/stable/modules/neural_networks_supervised.html> . Acessado em 25 de Janeiro de 2022.*, 2022.

31 SCIKITLEARN. Naive bayes. *URL: <https://scikit-learn.org/stable/modules/naive_bayes.html> . Acessado em 25 de Janeiro de 2022.*, 2022.

32 SCIKITLEARN. sklearn.ensemble.randomforestclassifier. *URL: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html> . Acessado em 25 de Janeiro de 2022.*, 2022.

33 WU, J.; CHEN, Z. An implicit identity authentication system considering changes of gesture based on keystroke behaviors. *International Journal of Distributed Sensor Networks, v. 11, n. 6, p. 470274*, 2015.

34 J., G. Data science do zero: Primeiras regras com o python. *Alta books, 2016 v.1.*, 2016.

35 POWERS, D. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *Bioinfo Publications*, 2011.

36 HANG, D.; WANG, J.; ZHAO, X. Estimating the uncertainty of average f1 scores. *Proceedings of the 2015 International Conference on The Theory of Information Retrieval. p.317-320*, 2015.

37 TOSHNIWAL, R. Demystifying roc curves. *URL: <https://towardsdatascience.com/demystifying-roc-curves-df809474529a#> . Acessado em 5 de Outubro de 2022.*, 2020.

38 GHOSH, D. Integrating ueba with zero trust: Accounting for the human element. *URL: <https://youtu.be/9jCSggoks3c/> . Acessado em 25 de Agosto de 2022.*, 2022.

39 KIM, H.; KIM, J.; KIM, I.; CHUNG, T. m. Behavior-based anomaly detection on big data. *Australian Information Security Management Conference*, 2015.

40 ZHAO, G.; XU, K.; XU, L.; WU, B. Detecting apt malware infections based on malicious dns and traffic analysis. *IEEE Access, vol. 3, pp. 1132–1142*, 2015.

41 FRIEDBERG, I.; SKOPIK, F.; SETTANNI, G.; FIEDLER, R. Combating advanced persistent threats: From network event correlation to incident detection. *Computers Security, vol. 48, pp. 35–57*, 2015.

42 YUAN, X. Phd forum: Deep learning-based real-time malware detection with multi-stage analysis. *Smart Computing (SMARTCOMP), 2017 IEEE International Conference on. IEEE, pp. 1–2*, 2017.

43 SIDDIQUI, S.; KHAN, M. S.; FERENS, K.; KINSNER, W. Detecting advanced persistent threats using fractal dimension based machine learning classification. *Proceedings of the 2016 ACM on International Workshop on Security And Privacy Analytics. ACM, 2016, pp. 64–69.*, 2016.

44 HSIEH, C.; LAI, C.; MAO, C.; KAO, T.; LEE, K. Ad2: Anomaly detection on active directory log data for insider threat monitoring. *Security Technology (ICCST), 2015 International Carnahan Conference on. IEEE, 2015, pp. 287–292.*, 2015.

45 SCHNEIER, B. Attack trees. *Dr. Dobb's Journal of Software Tools*, 1999.

46 UNB. Nsl kdd dataset. *URL: <https://www.unb.ca/cic/datasets/nsl.html> . Acessado em 2 de Abril de 2022.*

47 TAVALLAEE, M.; BAGHERI, E.; LU, W.; GHORBANI, A. A detailed analysis of the kdd cup 99 data set. *Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, 2009.

48 KDD. Sigkdd : Kdd cup 1999 : Computer network intrusion detection. *URL: <https://kdd.org/kdd-cup/view/kdd-cup-1999> . Acessado em 3 de Novembro de 2022.*