BRUNO DE BARROS BULLÉ

Diversidade e Aprendizado de Máquina a Serviço da Detecção de Intrusão em Sistema de Controle Industrial

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná, como requisito parcial à obtenção do título de mestre em Informática.

CURITIBA

2020

BRUNO DE BARROS BULLÉ

Diversidade e Aprendizado de Máquina a Serviço da Detecção de Intrusão em Sistema de Controle Industrial

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná, como requisito parcial à obtenção do título de mestre em Informática.

Área de concentração: *Ciência da Computação*

Orientador: Prof. Dr. Altair Olivo Santin
Coorientador: Prof. Dr. Eduardo Kugler Viegas

CURITIBA

2020

Esboço da Ficha Catalográfica elaborada pela Biblioteca Central da Pontifícia Universidade Católica do Paraná.

N.BC

BULLÉ, Bruno de Barros.

Diversidade e Aprendizado de Máquina a Serviço da Detecção de Intrusão em Sistema de Controle Industrial. Dissertação. – Curitiba, 2020. 60p.

Dissertação (Mestrado) – Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática Aplicada.

1. Segurança em sistemas de controle industrial 2. Tolerância à intrusão 3. Aprendizado de máquina. I. Pontifícia Universidade Católica do Paraná II. Centro de Ciências Exatas e de Tecnologia. III. Programa de Pós-Graduação em Informática Aplicada.

A ficha definitiva será elaborada pela biblioteca central e substituirá esse esboço.

Página reservada à ata de defesa que será fornecida pela secretaria do PPGIA após a defesa da dissertação e depois de efetuadas as correções solicitadas pela banca.

Página reservada ao termo de aprovação que será fornecido pela secretaria do PPGIA após a defesa da dissertação e depois de efetuadas as correções solicitadas pela banca.

Dedico este trabalho à minha família, em especial à minha mãe, aos meus filhos, e à minha amada esposa. Muito obrigado por todo o apoio e compreensão.

Agradecimentos

Agradeço à minha família por todo o apoio que recebi ao longo deste trabalho, em especial à minha mãe, aos meus filhos, e à minha amada esposa. Sou extremamente grato ao meu orientador, Dr. Altair O. Santin, por ter acreditado em mim e por ter sido um grande amigo nos momentos de dificuldade que passei durante essa jornada. Agradeço aos meus amigos, Roger R. dos Santos por ter me auxiliado em meus experimentos, e ao Dr. Eduardo Viegas que, com suas palavras, além de me tirar a paciência, me fez evoluir muito academicamente. Agradeço imensamente a todos os meus demais amigos do SecPLab por todos os estudos, trocas de ideias, apresentações, discussões, e sugestões que contribuíram para que eu me tornasse um aluno melhor. Ao G∴A∴D∴U∴ , pois sem ele nada disso seria possível.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

BFT	*Byzantine Fault Tolerance*
CI	*Critical Infrastructure*
CVE	*Common Vulnerabilities and Exposures*
CVSS	*Common Vulnerability Scoring System*
DCS	*Dynamic Classifier Selection*
DDoS	*Distributed Denial of Service*
DES	*Dynamic Ensemble Selection*
DESlib	*Dynamic ensemble selection library in Python*
DoS	*Denial of Service*
DS	*Dynamic Selection*
HIDS	*Host-based Intrusion Detection System*
ICS	*Industrial Control Systems*
IDS	*Intrusion detection System*
kNN	*K-Nearest Neighbour*
NIDS	*Network-based Intrusion Detection System*
NVD	*National Vulnerability Database*
OT	*Operational Technology*
OWASP	*Open Web Application Security Project*
RTU	*Remote Terminal Unit*
SaaS	*Security-as-a-Service*
SB	*Single Best*
SCADA	*Supervisory Control and Data Acquisition*
SO	Sistema Operacional
SS	*Static Selection*
TI	Tecnologia da Informação
VPN	*Virtual Private Network*
XSS	*Cross-site scripting*

Resumo

Este trabalho apresenta uma nova solução de segurança contra ataques a *Industrial Control Systems* (ICS), valendo-se dos benefícios do uso da diversidade de sistemas, sem a necessidade de realizar a replicação dos servidores de aplicação para prover segurança. Foi possível provar que diferentes sistemas operacionais (SO) reagem de diferentes formas, quando estão sob ataques cibernéticos. Sendo assim, foi desenvolvido um sistema de segurança que visa à proteção do ICS utilizando o aprendizado de máquina para identificar, a partir de métricas geradas sistemas operacionais (SO) Windows e Linux, qual SO é o mais indicado para proteger o ICS de forma mais eficaz. O estudo de caso é proposto para aplicação em *Smart grid.*

Palavras-chave: Segurança em sistemas de controle industrial; Tolerância à intrusão; Aprendizado de máquina.

Sumário

Capítulo 1

Introdução

A seção 1.1 apresenta uma breve contextualização do problema de pesquisa. A seção 1.2. apresenta a motivação e hipótese do estudo. Os objetivos a serem alcançados encontram-se na seção 1.3. Na seção 1.4. é apresentada a contribuição esperada. Finalmente, na seção 1.5., descreve-se como o documento está organizado.

1.1. Contextualização

Infraestrutura crítica (*Critical Infrastructure* – CI) e sistemas de controle industrial (*Industrial Control Systems* - ICS) são utilizados em larga escala em setores da saúde, energia, transporte, sistemas financeiros e militares. Os sistemas utilizados em CI precisam apresentar as qualidades de confiabilidade, resiliência, disponibilidade e segurança, e sua integração às redes públicas inevitável (ZIMBA et al 2018). Devido a isso, tais sistemas acabam se tornando mais expostos aos mais diversos tipos de ataque, aumentando significativamente os desafios na implementação de soluções de segurança da informação.

A última década tem apresentado um elevado número de ciberataques em sistemas - ICS/SCADA (*Supervisory Control and Data Acquisition*). Como as defesas atuais às vezes falham em impedir ameaças mais sofisticadas, é necessário adicionar mecanismos de proteção avançados para garantir que a operação correta seja (sempre) mantida (NOGUEIRA et al., 2018). Diferentemente da segurança da informação em TI, em que a segurança está preocupada com as informações, a segurança em *Operational Technology* (OT) está mais preocupada com a segurança das operações e a segurança das pessoas, já que as falhas de segurança em OT podem causar danos irreparáveis à integridade humana, como por e.g., uma alteração das

informações de pressão de uma caldeira que produziria ferimentos nos funcionários de uma fábrica.

No final do ano de 2015, a Ucrânia enfrentou um apagão de energia causado por um ciberataque no ICS utilizado em suas usinas de energia. O ataque afetou por diversas horas aproximadamente 225.000 clientes. Foi um ataque audacioso e bem-sucedido, onde foi utilizada uma série de técnicas para comprometer as usinas elétricas. Estudos mostraram que o ataque foi possível devido a uma série de fatores, como o uso de uma configuração insegura onde as *Virtual Private Network* (VPN) no ICS da rede comercial não teriam autenticação de dois fatores (LEE et al. 2016) e, com base nos relatórios da mídia, não havia nenhuma tecnologia para monitorar continuamente a rede ICS em busca de atacantes, sem contar que existia uma variedade de informações de código aberto disponíveis na internet, incluindo uma lista detalhada dos tipos de infraestrutura (LEE et al. 2016).

Foi a primeira vez que o mundo viu um ataque em que se visou destruir um sistema de OT na infraestrutura crítica de uma nação (LEE et al. 2016) OT é uma categoria de hardware e software que monitora e controla o desempenho dos dispositivos físicos. Esse ataque, em uma de suas etapas, tinha o objetivo de causar a destruição de equipamentos. Alguns desses equipamentos não existiam em um repositório, e ainda por cima eram fabricados por outros países, não sendo possível a realização uma troca rápida de equipamentos. O tempo para desenvolvimento, entrega e instalação poderiam levar dias, causando danos incalculáveis para a economia do país.

Outro caso em que ciberataques comprometeram ICS foi em 12 de maio de 2017, em que uma onda de ataques afetou mais de 300 mil computadores ao redor do mundo, através de uma vulnerabilidade presente nos SO *Windows*. O *WannaCry* é um *Ransomware*, uma classe de malware (abreviação de "*malicious software*") ou código malicioso, auto propagativo que utiliza criptografia para reter os dados da vítima e pedir resgate em troca. Devido ao seu comportamento de criptografar os dados do computador comprometido, diversos serviços críticos tiveram que ser interrompidos. No Reino Unido, hospitais tiveram de rejeitar pacientes; a montadora Nissan teve que interromper a produção de suas fábricas; a FedEx e empresas de telecomunicação como a *megaforTelefonica* da Espanha também enfrentaram problemas devido ao ataque (MOHURLE, PATIL, 2017).

Um desafio enfrentado por administradores de sistemas e analistas de segurança é o de tornar o seu sistema seguro contra ciberataques. Diversas técnicas são utilizadas para isso, como a utilização de ferramentas de segurança para impedir o acesso de usuários não

autorizados. Também é possível monitorar o ambiente computacional em busca de comportamentos anormais na rede e configurar, de forma segura, as aplicações e servidores, assim como estar em dia com os *patches* de atualização e por dentro das notícias de segurança. Entretanto, os métodos tradicionais de segurança, como o *Intrusion detection System* (IDS) baseado em assinatura (AVASARALA, et al. 2017), simplesmente não detectam um ciberataque originado por uma vulnerabilidade *zero-day*[1], já que tais vulnerabilidades ainda não possuem assinaturas.

1.2. Motivação

Atualmente, diversas são as técnicas utilizadas para aumentar a segurança dos sistemas de ambiente IC. Uma forma já comprovadamente efetiva é o uso da diversificação de tecnologias (GARCIA et al., 2014). Essa técnica funciona usando diferentes tipos de softwares para a execução de uma aplicação. A ideia principal por trás do uso da diversidade está em evitar ao máximo o comprometimento total da aplicação quando surge um ataque proveniente de uma vulnerabilidade *zero-day*.

Uma vulnerabilidade *zero-day* afeta um sistema ou um grupo de sistemas que compartilham da mesma fragilidade (trecho de código vulnerável). Com isso em vista, utilizar um sistema o mais diversificado possível pode evitar que toda uma arquitetura seja comprometida através de uma *zero-day*, pois considerado o elevado número de sistemas diversificados espera-se que pelo menos um deles não seja afetado. Como mencionado, o uso da diversificação é uma técnica de segurança efetiva, mas a sua adoção também apresenta fragilidades. Quando se opta em utilizar um elevado número de softwares para uma determinada aplicação, consequentemente aumentam as chances de um de seus sistemas ser invadido. Se não é mais necessário possuir uma *zero-day* para um sistema específico, agora um atacante pode utilizar uma *zero-day* para os mais diversos sistemas utilizados.

Outra técnica utilizada é a redundância de serviços/servidores. A redundância tem como objetivo evitar a parada total das operações. A técnica se baseia em "clonar" X vezes a

[1] *Zero-day* é uma vulnerabilidade de software não reportada pelo pesquisador responsável pelo seu descobrimento, i.e., que ainda é desconhecida pelos desenvolvedores do software, pelos fabricantes de antivírus e pelos usuários em geral. Ataques *zero-day* são realizados por cibercriminosos que descobriram a vulnerabilidade ou obtiveram conhecimento de sua existência e a exploram de modo sigiloso.

mesma tecnologia utilizada pela aplicação e, caso um dos servidores venha a apresentar problemas, um dos servidores redundantes assume a função, e o trabalho não é interrompido. Trata-se de um método utilizado em larga escala, mas que igualmente apresenta suas fragilidades, como o exemplo de uma vulnerabilidade *zero-day* que causa indisponibilidade para uma das tecnologias utilizadas pelo sistema, bastando que um atacante replique o ataque para cada um dos servidores utilizados para que todos de uma mesma infraestrutura sejam comprometidos. Outro ponto fraco é que esse método não pode ou não é recomendado para todos os casos, a exemplo do estudo realizado por (NOGUEIRA et al., 2018), em que se tinha como objetivo realizar a redundância de um sistema SCADA a fim de aumentar a segurança em ICS. Ao longo desse trabalho foi possível identificar uma série de dificuldades que mostram a inviabilidade na adoção dessa estratégia.

O uso da melhor configuração é, assim como as técnicas acima citadas, uma das abordagens mais utilizadas nos trabalhos analisados. Adotada por administradores de sistemas, essa técnica se baseia na utilização do melhor conjunto de tecnologias, assim como no uso das melhores práticas de segurança. Para utilizar um sistema da maneira mais segura possível, o trabalho de (Gorbenko et al., 2011) emprega informações das vulnerabilidades cadastradas nas bases *National Vulnerability Database* (NVD) e *Common Vulnerabilities and Exposures* (CVE) e calcula a criticidade, em tempo real, de qual sistema deve ser utilizado de acordo com o menor índice obtido. Já no trabalho de Alves et al. (2019), ao invés de utilizar as bases de dados já conhecidas pela comunidade, as notícias referentes às novas vulnerabilidades são obtidas através de mensagens do Twitter. Se com a diversidade é aumentada a área de ataque e diminuído o impacto de uma invasão, adotar a melhor configuração diminui a área de ataque de exploração de um atacante, porém é aumentado o impacto de uma invasão, já que o sistema pode ser comprometido como um todo.

O sistema de detecção de intrusão (IDS) é uma das tecnologias de segurança mais utilizadas para sistemas de controle industrial (HU et al., 2018), já que ele pode detectar atividades maliciosas que violam as políticas de segurança dos sistemas de controle industrial (ICS). Além disso, é possível fornecer evidências para alertar os administradores do sistema para fazer as ações adequadas durante os ciberataques. Sendo assim, um IDS pode efetivamente evitar que o ICS sofra ataques que causem grandes danos. Como resultado, o desenvolvimento de tecnologias eficazes de detecção de intrusão se mostra importante na proteção da segurança do ICS.

Como não foram identificados estudos que mostrassem uma única solução que contemplasse todos os benefícios de cada uma das técnicas de segurança aqui citadas, o presente trabalho torna-se relevante. A motivação por trás deste trabalho está em desenvolver uma solução de segurança para IC que contemplasse os benefícios das técnicas de diversidade de sistemas, do uso da melhor configuração e da redundância, porém, sem que este novo sistema criado também apresentasse os pontos negativos de cada uma destas técnicas.

1.3. Objetivo Geral

O objetivo geral do trabalho é desenvolver um sistema de segurança computacional de CI baseado na diversidade do Sistema Operacional (SO), através do *Host Intrusion Detection System* (HIDS), sem a necessidade da redundância ou da replicação simultânea de todo o sistema. Para tanto, busca-se criar um classificador que utiliza o aprendizado de máquina que seja capaz de identificar, através do comportamento do SO que estará sendo executado para a proteção do servidor SCADA, se este SO é o mais indicado na identificação dos mais diversos ataques contra a IC. Para tanto, objetiva-se gerar diferentes cenários compostos pelos mais diversos tipos de ataques em infraestrutura, e em aplicações web e em protocolos utilizados em TO. Busca-se, além disso, utilizar o aprendizado de máquina na identificação de ataques *zero-day* contra o ambiente de CI desenvolvido.

1.4. Objetivos Específicos

Para alcançar o objetivo geral desse trabalho será necessário atingir os seguintes objetivos específicos:

a) Desenvolver um método para a geração de bases de ataques. A fim de imitar os ataques que explorem vulnerabilidades *zero-day*, estes ataques devem também executar *exploits* já conhecidos, mas que não tiveram seus *patches* de atualização aplicados nos sistemas a serem testados;

b) Analisar nos eventos de comportamento do SO, quais foram as características que mais se alteram durante a execução dos ataques e modelar este comportamento;

c) Definir a tecnologia responsável por direcionar o tráfego para o modelo correto.

1.5. Contribuições

Este trabalho propõe uma solução nova em cibersegurança, cujas principais contribuições são:

- Um ambiente de teste realista e disponível ao público para a construção de diversos IDS para SCADA. O conjunto de dados construído é composto por mais de 100 clientes legítimos e 19 categorias de ataque distintos;
- Avaliação de técnicas tradicionais de ML para detecção de intrusão em SCADA. Os resultados da proposta indicam que a diversidade do sistema operacional pode melhorar significativamente a precisão da detecção para diversas categorias de ataque;
- Um modelo de detecção de intrusão novo e confiável para SCADA com base na diversidade do SO. O modelo proposto pode aproveitar da diversidade do SO para aumentar a precisão de identificação geral do sistema, enquanto também detecta novas categorias de ataque para o modelo de ML subjacente.

1.6. Estrutura do Documento

O presente documento encontra-se organizado da seguinte forma: o capítulo 1 apresenta uma introdução da proposta; o capítulo 2 aborda a fundamentação teórica do trabalho; o capítulo 3 apresenta os trabalhos relacionados com a proposta; o capítulo 4 detalha a proposta realizada e, por fim, o capítulo 5 apresenta a conclusão.

Capítulo 2

Fundamentação teórica

Além do uso da diversidade a nível de SO, serão implementados outros mecanismos para assegurar a segurança do ambiente. O IDS estará focado em identificar um possível ataque através da análise de comportamento do *host* em que se encontra instalado o IDS, caso a análise indique um comportamento anormal no *host*, isto sinalizaria um possível ataque.

Os critérios utilizados durante a escolha das ferramentas se basearam nas ferramentas gratuitas mais utilizadas pela comunidade, de código aberto, que podem ser instaladas nos SOs *Windows* e *Linux*.

2.1. *Critical Infrastructure* – CI

Dá-se o nome de infraestrutura crítica aos serviços e sistemas essenciais para uma sociedade e economia de um país, como os sistemas de energia, distribuição de água, telecomunicação, transporte, militar e finanças. Em CI, utilizamos o *industrial control systems* (ICS) que forma o componente principal da tecnologia operacional (TO). Os ICS são compostos de diferentes tipos de dispositivos, sistemas, controles e redes que são responsáveis por gerenciar uma variada gama de processos industriais.

Diferentemente da tecnologia da informação (TI) em que as informações são os dados importantes, em CI trabalhamos com TO, onde os processos e operações necessitam uma maior atenção. Antigamente, no que tange à segurança em CI, esta não era vista como sendo vital, já que os sistemas de TO não se mostravam conectados através da internet, no entanto, com o passar dos anos e com o aumento de sua conectividade na internet, a segurança em CI passou a ser entendida como um serviço essencial, uma vez que a CI passou a se tornar cada vez mais alvo de inúmeros ciberataques. Tornar-se vítima de um ciberataque é correr o risco de ter a

interrupção ou o colapso do fornecimento de um ou mais insumos essenciais para o pleno desenvolvimento de um país.

2.2. Diversidade

É conhecido que nenhum sistema está totalmente seguro. Novas vulnerabilidades, assim como as ferramentas utilizadas para a sua exploração, surgem todos os dias. Diversas técnicas são adotadas para evitar ao máximo a ocorrência de uma intrusão ou para evitar ao máximo os danos por esta causados. Essa técnica baseia-se no conceito de que softwares distintos apresentam vulnerabilidades distintas (AVIIENIS et al.; 1984).

A diversidade é uma técnica que tem como objetivo fazer o uso de diferentes tecnologias para que uma vulnerabilidade identificada em um dos servidores não seja identificada nos outros servidores. Outro fator que colabora para o aumento de segurança em sistemas de computação ao adotar o uso da diversidade está no fato de que grande parte dos softwares reutiliza trechos de códigos de outros diferentes softwares. Devido a reutilização de códigos, uma vulnerabilidade que afete o sistema A, pode vir a afetar um sistema B, como mostra o trabalho de (GARCIA et al., 2014). Entretanto, apesar de ser considerada uma ótima técnica para evitar o comprometimento de uma arquitetura dos sistemas como um todo, o uso da diversidade pode aumentar as chances de um sistema ser invadido, já que as chances de que do surgimento de uma nova *zero-day* aumentam, conforme novas tecnologias são adotadas para a proteção do sistema.

2.3. *Intrusion Detection System* - IDS

Existem basicamente dois tipos de IDS no mercado: os se baseiam em rede (NIDS – *Network-based Intrusion Detection System*) e os que se baseiam em *host* (HIDS – *Host-based Intrusion Detection System*).

Os NIDS funcionam basicamente analisando o tráfego de rede e emitindo alertas caso sejam identificados ataques, ou atividades maliciosas com características de ataques no segmento de rede, já os HIDS monitoram as atividades a nível de *host*. Um HIDS verifica

informações relativas aos eventos, registros de *logs* e sistema de arquivos (permissão, alteração etc.).

Um IDS pode identificar um ataque através de dois métodos: por *misuse* ou comportamental. No método de identificação por *misuse*, o IDS analisa as informações que coleta e as compara a grandes bancos de dados de assinaturas de ataques. Basicamente, o IDS procura um ataque específico que já foi documentado. *Misuse* é um padrão de fácil detecção, porém, não consegue identificar vulnerabilidades *zero-day*, já que as assinaturas são formadas apenas por ataques conhecidos. O HIDS examinará os arquivos de *log* e de configuração, e o NIDS analisará o tráfego de rede por padrões de ataques e atividades maliciosas. Em geral, o benefício do uso do método de identificação de padrões está em seu elevado desempenho em comparação ao método baseada no comportamento, também pelo fato possuir alto grau de detecção.

A detecção baseada no comportamento procura padrões inesperados ou incomuns de atividades. No caso do HIDS, uma anomalia pode ser identificada através de repetidas tentativas falhas na autenticação, ou através de uma atividade incomum nas portas de um dispositivo, o que implica na varredura de porta. Em contrapartida, em um NIDS, a identificação se dá através do monitoramento da rede, que deverá identificar os comportamentos considerados normais de uso para poder reconhecer um comportamento alterado de uso, que pode ser entendido como um ataque.

Um IDS pode até prever ataques *zero-day* caso haja suficientes dados disponíveis de treinamento e caso os algoritmos sejam bem implementados (DUA et al., 2019). Em geral, a detecção de intrusão é alcançada por quatro módulos sequenciais a saber: (i) aquisição de dados, (ii) extração de recursos, (iii) detecção e (iv) alerta. O módulo de aquisição de dados extrai os dados do ambiente para inspeção adicional, como por exemplo as métricas de uso do SO, como a memória utilizada, o número de processos e os usuários conectados. O módulo de extração de recursos extrai recursos comportamentais dos dados coletados, como por exemplo o número do novo processo e a quantidade de usuários conectados em uma determinada janela de tempo. O módulo de classificação aplica um algoritmo de detecção para a identificação de comportamentos anormais, como por exemplo um modelo de *Machine Learning* (ML) que classifica um determinado evento como normal ou como ataque. Finalmente, o módulo de alerta relata os eventos classificados como ataque.

O modelo de ML é construído a partir de um conjunto de dados de treinamento, por meio de um algoritmo de ML que extrai um modelo comportamental desses dados. A

confiabilidade do modelo de ML depende dos dados de entrada usadospara fins de treinamento, bem como do conjunto de recursos extraídos. Em outras palavras, se um ataque não afetar os valores de recurso extraídos ou se comportar de maneira semelhante ao comportamento normal, o modelo de ML o classificará incorretamente. A classificação incorreta de eventos é normalmente medida por meio de taxas de falso positivo (FP) e falso negativo (FN). A taxa FP denota a proporção de eventos normais classificados incorretamente como um ataque, enquanto a taxa FN denota a proporção de eventos de ataque classificados incorretamente como normais.

Na

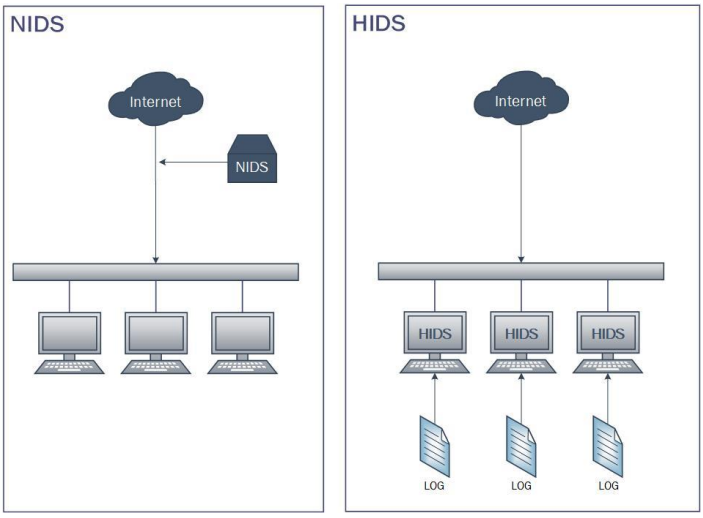

Figura 1, é possível ver a diferença entre os IDS NIDS e HIDS. Como mostra a figura a seguir, o NIDS está fazendo a inspeção dos pacotes do segmento de rede, localizado entre a internet e a rede de computadores. Já o HIDS está fazendo as análises em cada uma das estações de trabalho da rede. Para realizar esta análise, o HIDS está se valendo do comportamento de cada um dos computadores, utilizando arquivos de log para tal.

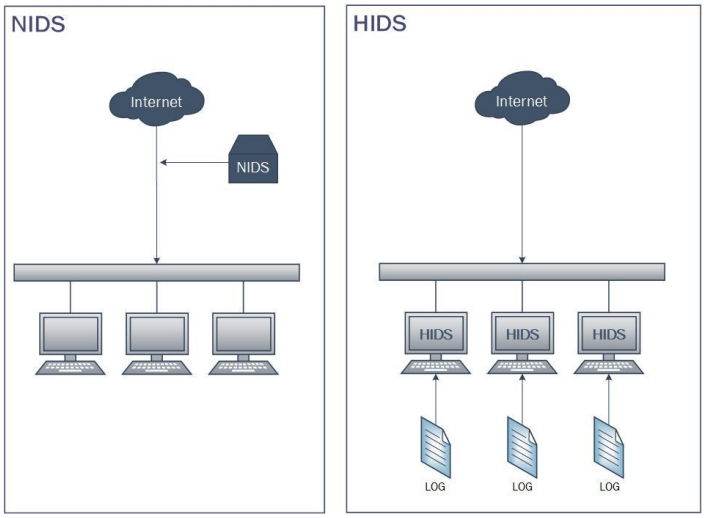

Figura 1 - Diferença entre IDS NIDS e HIDS (AVIIENIS et al.. 1984)

2.4. Aprendizagem de máquina

O aprendizado de máquina é para tarefas de detecção de intrusão é um tópico amplamente estudado na literatura. Em geral, a maioria das abordagens atuais busca maior acurácia de detecção por meio de algoritmos ML customizados ou *pipelines* ML mais complexos, como por exemplo os que utilizam tarefas de pré-processamento. Aprendizado de máquina é o uso de métodos matemáticos para treinar os algoritmos (SUN et al., 2018). Cada um desses ataques possui uma característica própria, como o uso de palavras-chave específicas para explorar determinada vulnerabilidade, ou como a autenticação malsucedida por inúmeras tentativas. O processo de aprendizagem de máquina se dá através desta sequência de passos:

1. Extração das características: processo que consiste em obter um conjunto de atributos que se distinguem de determinada informação;

2. Classificação: processo que consiste em “mostrar” para a máquina qual é o grupo de dados e o fluxo informado ao qual pertence;

3. Pós-processamento (avaliação): etapa que consiste em analisar os mais diversos resultados gerados por diferentes algoritmos com a maior porcentagem de acerto, e a partir dele, escolher os que trazem o melhor resultado;

4. Decisão: nesta etapa é verificado se a taxa de acerto está ou não satisfatória. Caso a acurácia apresente um índice muito baixo de acertos, provavelmente as etapas anteriores não foram bem executadas.

Existem basicamente dois algoritmos em aprendizado de máquina: os de regressão e os de classificação. Algoritmos de regressão são utilizados geralmente para prever valores quantitativos ou numéricos, como por exemplo o valor de um carro. Já os algoritmos de classificação são utilizados para prever valores qualitativos, como por exemplo a característica ou função de um determinado objeto, i.e., dentre os meios de transporte se é um carro, uma moto ou um trem. O aprendizado de máquina é dividido em duas fases:

1 Na primeira fase, o treinamento será realizado em um conjunto de dados, balanceado com as classes ataques e normal. Logo em seguida será feita a extração das características de cada tipo de dado para que o algoritmo consiga identificar se um dado é proveniente de um ataque ou não. Após esse fluxo, espera-se obter o modelo aprendido, que é o conhecimento que o algoritmo gerou;

2 Na segunda fase, será dado ao algoritmo um fluxo de dados e será feita a extração das características pelo próprio algoritmo, baseado no modelo gerado, se os dados são provenientes de um ataque.

2.5. Seleção de características

A seleção de características é um processo de extrema importância que antecede a etapa de criação de modelo. É nesta etapa em que são escolhidos os atributos pertencentes ao dado que será analisado pelo aprendizado de máquina para conseguir distinguir um conteúdo de outro. Uma das técnicas mais utilizadas durante a seleção de características é escolher os atributos que mais apresentam diferenças entre um e outro. Uma boa escolha de características aumenta as chances de acerto de um algoritmo de aprendizado de máquina.

Existem basicamente duas estratégias abordadas para a seleção de características: a *filter* e a *wrapper*. *Filter* é uma seleção que é feita independente do algoritmo de classificação. Esse método avalia cada atributo de forma independente um dos outros, determinando o grau de correlação de cada um com a classe (YANG, PEDERSEN, 1997). *Filter* utiliza as informações da base de treinamento para que assim sejam escolhidos os atributos que serão utilizados posteriormente.

Em *wrapper*, o algoritmo de classificação é executado para cada subconjunto e a avaliação se dá a partir da acurácia obtida. O subconjunto que obtiver o melhor desempenho será utilizado como o melhor subconjunto de atributos. É conhecido por possuir uma boa capacidade preditiva, porém, devido ao elevado número de execuções do algoritmo de classificação, requer um elevado poder computacional, poder este que não se mostra necessário em relação ao método *filter*.

2.6. *Zero-Day*

Um ataque de *zero-day* é executado com uma vulnerabilidade não divulgada na comunidade de segurança. Hackers, por exemplo, podem utilizá-la para explorar e afetar os mais diversos programas de computador. Ataques de *zero-day* tendem a ser devastadores por diversos motivos, dentre os quais pode-se citar: (i) vulnerabilidade não divulgada pelos canais de segurança, a exemplo de quando fábricas de software não contam com *pathes* de segurança disponíveis para que os usuários corrijam a vulnerabilidade em seus sistemas, e (ii) assinaturas de ataques não consideradas por desenvolvedores de soluções de segurança, como por exemplo software de antivírus.

As assinaturas ajudam diversas soluções de segurança na identificação de ataques. Formadas pelos padrões já conhecidos dos ataques, sempre que for identificado um padrão de assinatura sendo executado, as soluções de segurança baseadas em assinaturas acabam por agir no sistema a fim de impedir a execução do ataque.

Em um ataque de *zero-day*, os invasores visam a um ou mais requisitos de segurança ativos. Os atacantes mudam seu vetor de ataque para ocultar seu comportamento e evitar software antivírus sistemático (KARR, 2015), dificultando ainda mais sua identificação.

2.7. Classificadores

São chamados de classificadores os algoritmos responsáveis por criar os modelos utilizados para prever o comportamento de determinado problema. Os algoritmos recebem como entrada uma base de dados, que no caso HIDS é um arquivo contendo o comportamento do SO, que, a partir do qual, são criados os modelos de treinamento e teste.

Quando se trata de aprendizado de máquina, é importante ter conhecimento do teorema "*No free lunch theorem*", que resumidamente quer dizer que não existe um algoritmo universal que solucione todos os problemas da área. Sendo assim, os algoritmos precisam ser testados e analisados, e ter seus resultados comparados para que dessa forma seja feita a melhor escolha dentre os algoritmos a serem utilizados.

Este trabalho focaliza os algoritmos mais conhecidos e mais utilizados nos trabalhos estudados: *Random Forest*, *Extra-Tree*, e *Gradient Boosting*.

2.7.1 *Random Forest*

Floresta Aleatória (*Random Forest* - RF) é um algoritmo de aprendizagem supervisionada criada de forma aleatória. A floresta é a combinação (*ensemble*) de um conjunto de árvores de decisão, treinadas através do método *bagging*.

Este é um algoritmo que pode ser utilizado para classificação e para regressão, como também para a busca pela melhor característica entre um subconjunto aleatório de características. Esse processo tende a criar muita diversidade, resultando assim na geração de melhores modelos.

Outra qualidade que esse algoritmo apresenta é o fato de não sofrer com o *overfitting* (subreajuste). Isso se dá porque se existir um número de árvores suficiente na floresta, o classificador não realizará o *overfitting* do modelo. Entretanto, apesar das qualidades mencionadas, esse algoritmo pode apresentar problemas quando houver grande número de árvores, ocasião em que tende a ficar lento e ineficiente para realização de predições em tempo real.

2.7.2 *Extra-Tree*

O algoritmo *Extra-Tree* (árvores extremamente aleatórias) possui diversas similaridades se comparado aos tradicionais algoritmos de árvores, entretanto se difere destes basicamente em dois pontos: (i) por utilizar o método de escolha aleatória para indicar o melhor ponto de corte, o que tende a um desempenho superior se comparado aos demais algoritmos de árvores; e (ii) por utilizar toda a amostra de aprendizagem para crescer as árvores ao invés de uma réplica *bootstrap*.

Como apresentado em (GHOSH; SAMPALLI, 2019), quando o nível de aleatoriedade é ajustado de forma correta, a variância é reduzida de forma expressiva, enquanto o viés aumenta ligeiramente em relação às árvores padrão.

2.7.3 *Gradient Boosting*

O *Gradient Boosting* (GB) realiza o treinamento de diversos modelos de forma gradual, aditiva e sequencial. Diferentemente do algoritmo *AdaBoost*, em que o modelo de identificação se dá pelas deficiências usando pontos de dados de alto peso, o aumento de gradiente tem o mesmo desempenho usando gradientes na função de perda (y = ax + b + e). Outra característica que o difere dos demais algoritmos de *boosting* é que no algoritmo de GB é utilizada a métrica "erro" para avaliar e mensurar o desempenho do modelo, ao contrário dos demais que permitem ao usuário fazer a especificação de um elevado número de métricas, como erro e perda de *log*.

Por padrão, o modelo de GB assume um valor total 100 árvores, sendo este valor suficiente para fornecer uma boa estimativa de desempenho do modelo. Esse tipo de algoritmo tende a sofrer de *overfitting*, e para evitá-lo, o modelo de GB necessita realizar uma Parada Antecipada que executa uma otimização do modelo, onde seu desempenho é monitorado em um conjunto de dados de teste separado. O processo de treinamento é interrompido quando o desempenho dos dados de teste deixa de obter uma melhora além de um determinado número de iterações.

2.8. *Supervisory Control and Data Acquisition* – SCADA

Os sistemas SCADA são sistemas comumente utilizados para monitorar e controlar diferentes tipos de processos em CI, como indústrias elétricas, nucleares e refinarias. Muitos desses sistemas controlam componentes críticos das mais diversas nações, como transporte público e estações de tratamento de esgoto (GARITANO et al. 2011).

Para que seja possível monitorar e controlar a infraestrutura, é necessário investir em servidores e drivers para a comunicação, sensores que terão seus dados analisados, e atuadores que são utilizados para realizar as devidas ações. Assim, o supervisório capta e armazena vários dados do processo, gerando, por exemplo, telas ou gráficos que entregam informações da automação industrial.

A comunicação utilizada em sistemas SCADA/ICS se dá a partir de mais de 100 protocolos de comunicação, sendo parte deles proprietários e outros abertos. Os protocolos mais populares são o Modbus, o DNP3, o PROFINET/PROFIBUS e o OPC.

Em um dispositivo que utiliza o protocolo Modbus existem mais de um método de armazenamento de dados. *Coils* são os tipos de dados formados por 1 bit, que seriam a variação dos valores 0 e 1. Esses dados podem ser usados tanto para escrita como também para leitura. Os dados em Modbus também podem ser lidos e gravados como registradores (*registers)*, que são dados de 16 bits. Em grande parte, o registro é um número inteiro de 16 bits com ou sem sinal. O registro mais usado é chamado de *Holding Register*, que pode ser lido ou gravado. O outro tipo possível é o *Input Register*, que é somente utilizado para leitura.

Às vezes, mudanças geram insegurança. Profissionais de OT por vezes se deparam com fabricantes que disponibilizam pacote com inúmeras correções de vulnerabilidades, mas não existem garantias de que tais correções são de fato seguras, i.e., uma aplicação de um determinado *patch* pode, por exemplo, resultar em um incidente que leve a óbito algum profissional no chão de fábrica. Uma atualização disponibilizada não garante a segurança para um ambiente em específico. Por esse motivo é tão difícil aplicar novos *patches* de atualização em ICS. Administradores de sistemas não podem exigir que todos os funcionários de uma fábrica se retirem dela para que testes sejam realizados. Ainda que sempre exista a necessidade da obtenção de dados que comprovem que as atualizações disponibilizadas são de fato seguras para a implementação no sistema, esse processo que garante a segurança na atualização de *patches* de segurança não é tão rápido quanto se deseja, deixando assim o sistema inseguro por um período significativo.

Além disso, devido à criticidade dos processos que um SCADA realiza, é crucial que o sistema permaneça disponível. Às vezes, a reinicialização de um SCADA está fora de questão, não sendo possível aplicar os *patches* de atualização que visam à correção de falhas de segurança dos softwares utilizados. Isso se dá pela criticidade dos processos com a qual tais sistemas operam, uma vez que nem sempre sistemas SCADA podem ser reiniciados para a correção de vulnerabilidades, o que implica na necessidade da implementação de maior controle de segurança.

2.9. *Gap* semântico

Na literatura, grande parte das soluções voltadas para sistemas críticos utiliza a replicação de serviços/servidores e a diversidade de tecnologias. Entretanto, apesar de serem

técnicas comprovadamente efetivas, apresentam alguns problemas ou lacunas a serem resolvidos.

Diferentes ferramentas apresentam distintos desempenhos para prevenir um ciberataque. Uma dada vulnerabilidade pode ser bloqueada de forma mais eficaz com uma certa ferramenta se comparada a outras. Em um ambiente tradicionalmente desenvolvido, que utiliza as técnicas de replicação de serviços/servidores e de diversidades de tecnologias, não necessariamente se emprega do melhor conjunto de configurações para identificar e/ou impedir um ciberataque.

Com o objetivo de aumentar a segurança em CI, foi desenvolvido um sistema que fez uso de critérios de diversidade, e composto por diferentes SOs. Para tanto, foram realizados uma série de ataques em cada um dos SO utilizados para proteger o servidor SCADA. Os SO tiveram seu comportamento salvo durante a execução de cada um dos ataques e, posteriormente, este arquivo contendo os comportamentos dos SO foram então analisados pelos algoritmos de ML a fim de identificar qual dos SO obteve uma maior taxa de acerto na identificação do ataque.

Capítulo 3

Trabalhos Relacionados

Nesta seção são apresentados trabalhos relacionados com o escopo desta dissertação.

3.1. Diversidade de Tecnologias

Esta seção apresenta os trabalhos que abordam o uso da diversidade como forma de aumentar a segurança dos sistemas de informação.

3.1.1. *Analysis of operating system diversity for intrusion tolerance*

O trabalho (GARCIA et al., 2014) sugere o uso da diversidade de SOs para a adoção de sistemas tolerantes à intrusão. O estudo analisou 11 SOs ao longo de 18 anos, e buscou pelo sistema que possuía o menor número de vulnerabilidades em comum.

Para obter êxito ao invadir um SO através de uma vulnerabilidade *zero-day*, o esforço necessário para comprometer a infraestrutura que faz uso de uma única versão do SO se resume no tempo de replicação do ataque para os demais ativos de rede. Não obstante, caso sejam utilizados diferentes SOs, o atacante precisará acessar a diferentes vulnerabilidades para invadir cada um dos sistemas utilizados.

O uso da diversidade diminui as chances de um sistema se tornar indisponível, pois ainda que um sistema seja comprometido, um outro sistema que não foi afetado pela vulnerabilidade entra em ação para continuar dando suporte à aplicação. Assim, ao aumentar o número de tecnologias em um ambiente, são aumentadas as chances de uma vulnerabilidade *zero-day* não afetar as demais tecnologias. Um ponto negativo dessa abordagem é não apontar quais tecnologias utilizadas no sistema são mais efetivas para conter um determinado tipo de

ataque, i.e., um ataque pode não ser corretamente identificado, ou ser parcialmente identificado pelas soluções disponíveis.

O uso da diversidade implica em um elevado poder computacional quando comparado a um servidor que utiliza um único sistema para hospedar uma aplicação (uso da melhor configuração). Assim, para adquirir um poder computacional superior, será necessário um investimento elevado em hardware e em energia elétrica, já que um número maior de equipamentos estará em funcionamento.

Na solução proposta por este trabalho, o risco de um usuário se tornar mais uma vítima de uma invasão é reduzido, pois será utilizado o melhor conjunto de configurações possível para bloquear o ataque. Caso um atacante utilize, por exemplo, uma vulnerabilidade que se mostrou com uma taxa de acerto maior no SO Linux, só que durante a execução do ataque estiver em execução o SO *Windows*, o classificador identificará que SO em uso não é o mais indicado, e então sugerirá a mudança para o SO Linux.

3.2. Uso da melhor configuração

3.2.1. *Using diversity in cloud-based deployment environment to avoid intrusions*

O trabalho (Gorbenko et al., 2011) propôs o uso de um gerenciador para escolha de um melhor conjunto de configuração para implantação de serviços Web em nuvem. O gerenciador utiliza informações das vulnerabilidades presentes nas bases NVD e CVE para calcular, em tempo real, a criticidade das vulnerabilidades das tecnologias utilizadas pelo modelo proposto. A infraestrutura é configurada de acordo com a menor pontuação de criticidade calculada pelo gerenciador.

Esse trabalho, em tese, desliga os servidores que apresentam vulnerabilidades, entretanto, a CVE não emite notificações em tempo hábil (ALVES et al., 2019), e não faz uso do melhor conjunto de configurações para mitigar um ciberataque.

3.2.2. *Processing Tweets for Cybersecurity Threat Awareness*

O surgimento de uma nova vulnerabilidade pode colocar em risco toda uma infraestrutura e, por isso, é imprescindível que um administrador de sistemas esteja atualizado sobre as novas ameaças.

Tipicamente, administradores obtém notícias referentes à segurança cibernética através da compra de um *feed* de uma empresa especializada, como por exemplo a *SenseCy*, a *SurfWatch Labs*. Ainda assim essas informações também podem ser coletadas a partir do *Open Source Intelligence* (OSINT) que coleta informações de diversas fontes na internet.

O trabalho de Alves et al. (2019) apresenta uma abordagem diferente das demais analisadas para a escolha da arquitetura mais segura a ser utilizada pela aplicação. O autor propõe uma ferramenta chamada SYNAPSE, que monitora constantemente assuntos relacionados à segurança cibernética por meio da plataforma *Twitter*, que através dos *tweets* coleta, classifica e agrupa mensagens relacionadas a uma infraestrutura específica.

De acordo com o artigo, 43% das *Indicator of compromise* (IoC) descrevem alertas de segurança de alto impacto (*Common Vulnerability Scoring System* (CVSS) ≥ 7,0) e, para metade deles, a publicação de *tweets* precedeu a publicação de vulnerabilidades no NVD em oito dias (em média). Tal período de oito dias pode ser extremamente elevado em determinados casos como em sistemas críticos.

A contribuição desse estudo é significativa, entretanto, trata-se de uma ferramenta de monitoramento de novas vulnerabilidades, e não de mecanismo que identifica um ataque em execução, e que sugere a troca para um sistema que apresenta uma maior taxa de assertividade na identificação.

3.3. SCADA

3.3.1. *On the Challenges of Building a BFT SCADA*

Na última década, os ICS têm sido alvo frequente de ciberataques. Como as defesas atuais às vezes falham em impedir ameaças mais sofisticadas, é necessário adicionar mecanismos de proteção avançados para garantir que a operação correta seja (sempre) mantida.

O artigo de Nogueira et al. (2018) propõe a duplicação do software SCADA, aprimorado com técnicas tolerantes a falhas bizantinas (*Byzantine Fault Tolerance* - BFT), para aumentar a segurança de sistemas críticos. No entanto, ao longo do trabalho, é possível identificar uma série de problemas enfrentados pelos autores que acabam mostrando a inviabilidade de se adotar a estratégia de duplicação de tal sistema. Além de uma tarefa árdua, foi identificada uma queda de desempenho na utilização do modelo proposto, causada principalmente pelo gargalo de serialização de mensagens introduzido para garantir o determinismo, que de acordo com os autores, seria solução suficiente para acomodar cargas de trabalho realistas, apesar da queda no desempenho.

3.3.2. *Analysis of the Cyber Attack on the Ukrainian Power Grid*

O artigo de LEE et al. (2016) apresenta as lições obtidas a partir do ataque aos sistemas utilizados nas indústrias de energia elétrica da Ucrânia, e as medidas pontuais de segurança que foram adotadas para a mitigação de futuras ocorrências.

O ataque afetou aproximadamente 225.000 clientes por horas, e foi quando o potencial destrutivo de um ataque contra sistemas de OT na infraestrutura crítica de uma nação foi presenciado pela primeira vez no mundo. O ataque utilizou uma série de técnicas para comprometer as usinas elétricas da Ucrânia, tais como: envio de e-mails de *spear phishing*, manipulação de documentos do *Microsoft Office* que continha uma variante do malware *BlackEnergy 3* para obter acesso às redes de TI das empresas de eletricidade e, com isso, conseguiu realizar uma coleta das credenciais e das informações para assim acessar à rede ICS.

O *BlackEnergy* é um kit de ferramentas de *malware* usado por cibercriminosos que foi desenvolvido por volta de 2007, originalmente projetado para criar *botnets* para ataques DDoS (*Distributed Denial of Service*), com uso evoluído para oferecer suporte a vários *plug-ins*.

O ataque foi possível devido a uma série de fatores como: uma vasta informação do código-fonte aberto disponível na internet, uma lista detalhada da infraestrutura e de fornecedores de *Remote Terminal Unit* (RTU) e de ICS. Além disso, havia pontos do sistema configurados de forma insegura, já que o *firewall* permitiu que o atacante administrasse remotamente o ambiente, utilizando, para tanto, recursos de acesso remoto nativos do sistema. Adicionalmente, o acesso às VPNs da rede não utilizava autenticação de dois fatores. Como se não bastasse o fato de um vasto número de informações estar disponível na internet, além do uso de uma configuração insegura, nenhuma tecnologia monitorava o sistema para alertar a

presença de intrusos da rede ICS. Por esses motivos, os autores do artigo sugerem que o atacante já havia obtido e mantido o acesso ao ambiente seis meses ou antes do ataque.

Os passos para a execução do ataque seguiram o seguinte fluxo: envio de e-mails que continham documentos *Office* maliciosos em anexo para usuários administradores ou equipe de rede de TI das companhias elétricas. Ao abrir o documento em anexo, uma janela solicitava que o usuário habilitasse a execução de Macros. Ao habilitá-las, o *malware* entrava em ação, instalando o *BlackEnergy* 3 no sistema da vítima, permitindo ao atacante a obtenção de informações do ambiente comprometido. Entre os objetivos do ataque estava o comprometimento do *firmware* de comunicação (*Serial-to-Ethernet*) que impedia os operadores de enviar comandos remotos para colocar as subestações de volta on-line, mesmo que conseguissem obter acesso às suas estações de trabalho.

O artigo focalizou em como ocorreram os ataques na Ucrânia e quais medidas poderiam ser adotadas para mitigar pontualmente as vulnerabilidades no sistema, evitando assim a ocorrência de um novo ataque. Todavia, ataques desse porte dificilmente são executados de maneira idêntica e, portanto, os autores apontaram somente preocupações em relação à necessidade de uma solução para identificação de anomalias na rede, para mapear os privilégios dos operadores na realização de determinadas ações e para derrubar a conexão de um atacante assim que identificada. Assim, uma lacuna desse estudo mencionado é investigar a diversidade de sistemas para o uso das melhores qualidades das tecnologias na defesa contra ciberataques, bem como a gestão de economia de recursos.

3.4. Considerações sobre os trabalhos relacionados

O Quadro a seguir relaciona os principais aspectos de cada abordagem para efeito comparativo:

Tabela 1- Análise comparativa dos trabalhos relacionados

Artigo	**Utiliza diversificação**	**Adota o uso da melhor configuração**	**Escolhe o melhor conjunto de configurações**	**Usa aprendizado de máquina**	**Tolerante à intrusão**
Analysis of operating system diversity for intrusion tolerance	Sim. Busca os SO com maior diversificação	Não	Não	Não	Sim. Um atacante precisa comprometer todos os SO
Processing Tweets for Cybersecurity Threat Awareness	Sim. Diversas tecnologias são utilizadas	Não. O sistema emite alertas quando uma nova vulnerabilidade pode comprometer a infraestrutura	Não	Sim. Para processar as mensagens do Twitter	Não
Using diversity in cloud-based deployment environment to avoid intrusions	Sim. Utiliza a diversificação de tecnologias	Sim. Utiliza a configuração com menor score CVSS.	Sim. Através da base de CVE's	Não	Sim
On the Challenges of Building a BFT SCADA	Busca adotar a duplicação do sistema SCADA	Não	Não	Não	Sim. Tolerante a falhas
Analysis of the Cyber Attack on the Ukrainian Power Grid	Não	Sim. Foram apresentadas sugestões de melhoria a CI	Melhor configuração das tecnologias já utilizadas.	Não	Não

Capítulo 4

Diversidade e Aprendizado de Máquina a Serviço da Detecção de Intrusão em Sistema de Controle Industrial

A proposta deste trabalho consiste em apresentar um novo método que visa identificar de forma mais efetiva os mais diversos ataques em CI, baseando-se em diversidade através de *Host Intrusion Detection System* (HIDS), e Sistema Operacional (SO).

4.1 Novo HIDS para ICS

Nesta seção, é apresentado o novo modelo confiável de detecção de intrusão baseado em host para sistemas SCADA por meio de diversidade de sistemas operacionais. O modelo consiste em duas etapas principais, a saber Classificação de Comportamento do SO e Seletor de Pool de SO, que visam melhorar a segurança do SCADA através da diversidade do OS, aumentando ainda mais a capacidade de detecção de intrusão para detectar novos ataques.

A proposta, apresentada na Figura 1, considera um conjunto de diversos SOs, que atuam como front-end do SCADA. Durante a execução, um único front-end é continuamente usado pelo módulo de Classificação de Comportamento do SO, por exemplo, um SO Windows atuando como o front-end SCADA.

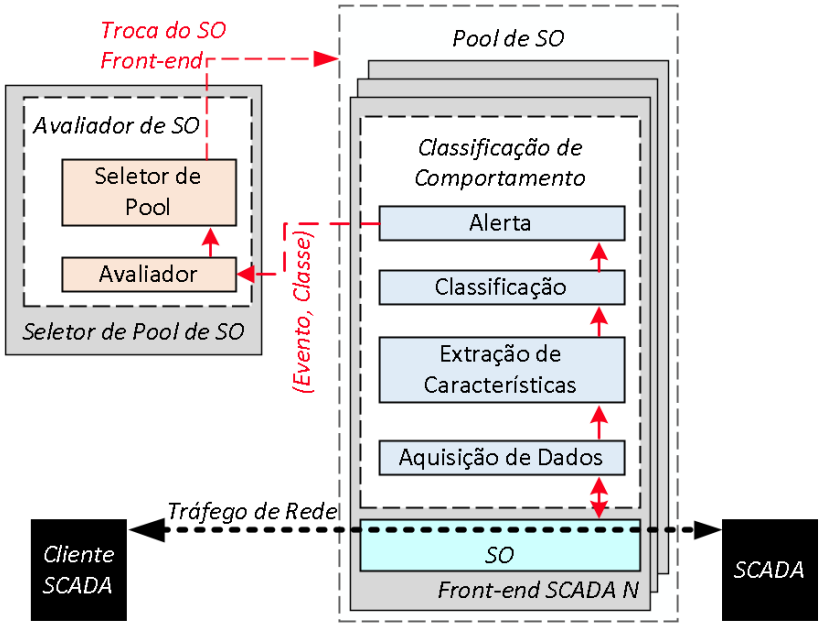

Figura 2 – Modelo proposto

Esta proposta considera a detecção de intrusão baseada em host, um evento que pode ser representado como um conjunto de métricas de uso relacionadas ao SO em uma determinada análise de janela de tempo, por exemplo, uso de CPU no último intervalo de 10 segundos. A janela de tempo é classificada por um modelo de ML para encontrar comportamento anômalo / intrusão. O evento a ser classificado é enviado pelo módulo de Seletor de Pool de SO, que avalia o comportamento do ambiente atual para selecionar o SO mais confiável.

O módulo OS Pool-Selector, por sua vez, visa definir se o SO front-end do SCADA atualmente em uso pode ser utilizado de forma confiável para a detecção de novas tentativas de intrusão. O módulo aplica um avaliador para a identificação de comportamento não confiável (novo para o modelo ML); por exemplo, pode avaliar a confiança da classificação ML para o atual evento classificado.

O comportamento não confiável é considerado como tentativa de intrusão que não pode ser detectada pelo SCADA OS atual front-end; portanto, outro sistema operacional deve ser usado. Portanto, o sistema muda de forma conservadora o front-end do SO SCADA quando eventos não confiáveis são detectados, a fim de melhorar a precisão e confiabilidade geral do sistema. As próximas subseções detalham os módulos Classificação de Comportamento do SO e Seletor de Pool de SO.

4.1.1 Classificação de comportamento do SO

Foi suporto que a diversidade do sistema operacional é capaz de fornecer um escopo de detecção mais preciso, e mais confiável. Em outras palavras, a diversidade do sistema operacional melhora a detecção de outras categorias de ataques. Criar um mecanismo de detecção de intrusão que depende da diversidade do SCADA não é uma tarefa fácil. Esta proposta é baseada em um mecanismo SCADA front-end para facilitar o uso da diversidade. A ideia da proposta é que a detecção de intrusão por meio de diversidade pode ser alcançada no nível de front-end, em vez de exigir a implantação de vários sistemas operacionais com servidores SCADA replicados. A justificativa é que o SO front-end do SCADA também avaliará os pacotes de rede em toda a comunicação cliente / servidor; portanto, as métricas de uso do SO também serão afetadas quando o SCADA estiver sob ataque.

O módulo Classificação de Comportamento do SO é executado continuamente pelo SO atual em execução como o front-end SCADA (SCADA Front-end N, Figura 2). O módulo é executado por meio da etapa de aquisição de dados, que coleta periodicamente as métricas de uso do SO em um determinado período. Portanto, o objetivo do módulo é classificar as métricas de uso do sistema operacional atuais (valor dos recursos), em uma janela de tempo, como normal ou ataque (intrusão).

Os valores coletados das métricas de uso do sistema operacional são usados para compor um vetor de recursos, que é então avaliado pelo módulo de classificação. A saída da classificação final e o vetor de recursos são encaminhados para o módulo Seletor de Pool de SO, para posterior inspeção.

Usando tal abordagem, esta proposta pode fornecer diversas métricas de sistema operacional para a identificação de ataques que ocorrem no servidor SCADA. Cada SO front-end mantém seu modelo de ML, construído de acordo com suas métricas de SO, alcançando diversidade na detecção de intrusão. Além disso, nenhuma alteração adicional é necessária ao servidor SCADA para a implantação desta proposta.

4.1.2 Seletor de pool de SO

O objetivo do módulo Seletor de Pool de SO é definir se o SO front-end SCADA em uso pode detectar o comportamento do ambiente atual de forma confiável. Em outras palavras, o objetivo do módulo é estabelecer qual SO front-end deve ser usado.

O módulo recebe como entrada o evento atual classificado pela Classificação de Comportamento do SO para posterior avaliação. Em seguida, ele avalia se o evento classificado foi detectado de forma confiável pelo SO front-end em uso, definindo se o evento é desconhecido para o modelo de ML subjacente ou não, por exemplo, avaliando a confiança de classificação do modelo de ML.

Se um evento confiável for encontrado, o comportamento será considerado conhecido pelo modelo de ML e o SO do front-end atual permanecerá inalterado. Por outro lado, o módulo alterna o SO do front-end atual em execução se um evento não confiável for encontrado (Troca do SO front-end, Figura 2).

Como resultado, esta proposta pode avaliar continuamente a confiabilidade atual do SO front-end SCADA. Portanto, através da avaliação contínua das classificações atuais do front-end realizadas, o módulo estabelece o SO do front-end mais adequado a ser utilizado para o comportamento do ambiente atual, aumentando a confiabilidade da proposta através da diversidade na detecção de intrusão.

Capítulo 5

Protótipo e avaliação da proposta

Este capítulo tem o propósito de apresentar o desenvolvimento do ambiente utilizado para os testes, assim como apresentar a metodologia para avaliação do ambiente.

5.1 Configuração do ambiente de teste

Para a implementação dessa abordagem de front-end SCADA, foram utilizados dois sistemas operacionais, um Microsoft Windows 10 e um Ubuntu Linux 20.04. Ambos os sistemas operacionais foram implantados como uma máquina virtual com quatro núcleos virtuais e 8 GB de memória.

No ambiente de teste o servidor SCADA foi implantado através do ScadaBR versão 1.0CE, em um computador com o SO Windows 10, e com o servidor de aplicação Apache Tomcat 7, com vários serviços populares normalmente relatados na literatura (GHOSH; SAMPALLI, 2019), como mostra a Figura 3. Tanto as telas de interface do usuário como as de configurações do ScadaBR são acessadas através de um navegador de Internet.

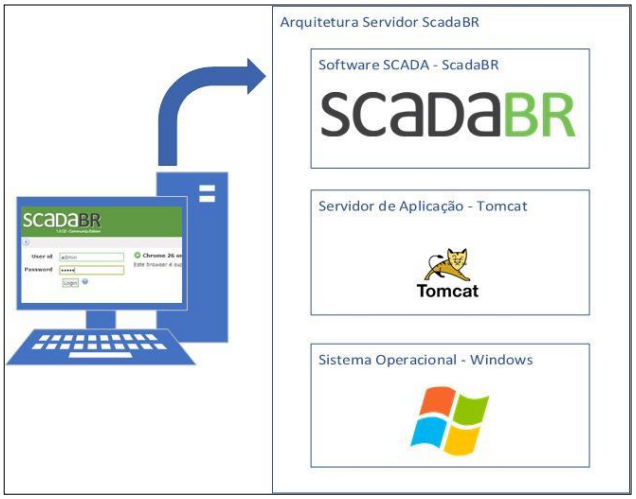

Figura 3 - Arquitetura do servidor utilizado para o sistema ScadaBR

5.2 Escolha do modelo de arquitetura

Um protótipo da proposta foi implementado e implantado em um ambiente distribuído, onde foram utilizadas máquinas virtuais, hospedadas em um servidor composto por 20 CPU de 2,2 GHz, e 255 GB de memória RAM. Foram utilizadas duas versões de SOs, Microsoft Windows 10 e Ubuntu Linux 20.04 para aumentar de forma mais efetiva a cibersegurança do sistema. A **Erro! Fonte de referência não encontrada.** mostra que os dados dos usuários e o tráfego oriundo da rede só chega ao sistema SCADA após a análise do classificador e da definição do modelo a ser utilizado.

Como apresentado na Figura 4, o modelo é composto por um Seletor de Pool de SO, que possui um Avaliador de OS constituído por um Avaliador e um Seletor de Pool. O Pool de SO é composto por dois SOs e um módulo chamado Classificação de Comportamento, formado por módulos de Alerta, Classificação, Extração de Características e Aquisição de Dados.

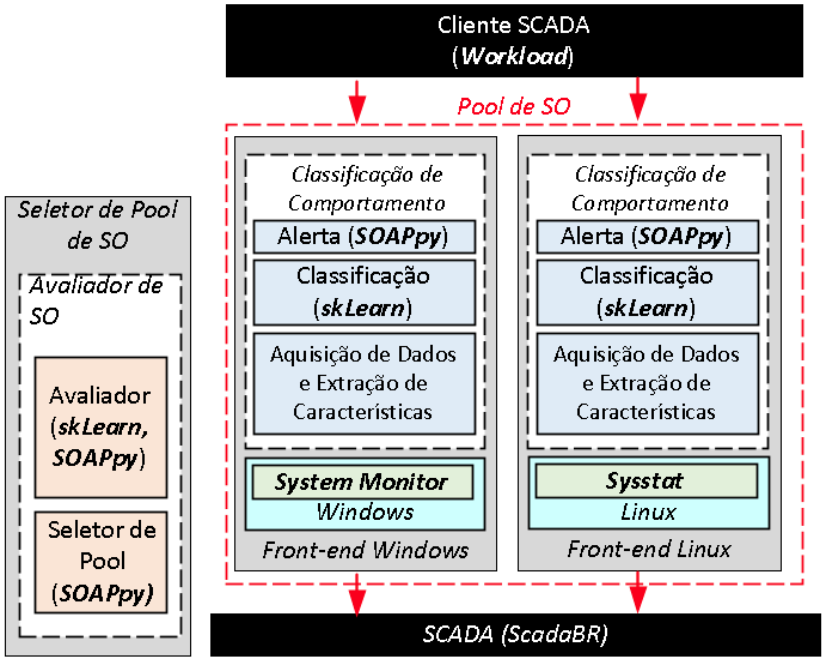

Figura 4 - Arquitetura da proposta

Neste trabalho foi desenvolvida uma camada de proteção capaz de identificar, através de algoritmos de ML, qual é o melhor SO que deve ser adotado pelo sistema a fim de identificar com uma maior acurácia um ataque em IC. A implantação dessa camada de proteção visa desencorajar os administradores na implementação de um sistema SCADA redundante. Optou-se em utilizar o sistema ScadaBR2 em um servidor que contou com o SO Windows 10.

[2] O ScadaBR é um software do tipo SCADA, gratuito e de código-fonte aberto, para desenvolvimento de aplicações de Automação, Aquisição de Dados e Controle Supervisório. Esse software serve para automatizar processos de medição e automação, ou seja: através do ScadaBR é possível acessar e controlar dispositivos físicos como sensores, chaves, motores e outros tipos de máquinas. Dentre outras funcionalidades, também é possível salvar dados dos sensores em base de dados, visualizar os históricos, receber alarmes, e controlar o processo por meio de *scripts* (CERTI s/d).

5.3 Descrição dos Ataques

Uma que o ScadaBR é administrado através de uma aplicação web, o experimento contou com uma série de ataques voltados para aplicações web, além de ataques à infraestrutura que hospeda o ScadaBR e aos dispositivos Modbus. Foram realizados os seguintes ataques:

- Code Injection (Arachni): Trata-se de um ataque onde os usuários inserem códigos que serão interpretados e executados pela aplicação. Este tipo de ataque explora a forma insegura com a qual desenvolvedores lidam com as entradas de dados fornecidas pelos usuários. A injeção de códigos está diretamente relacionada à linguagem de programação utilizada pela aplicação, sendo assim, para que o ataque tenha êxito, um atacante deve modelá-lo para as tecnologias utilizadas no sistema. Assim, um atacante poderia executar funções PHP em uma aplicação desenvolvida por tal linguagem;

- Crawler Autenticado (Acunetix): Web crawler é a técnica utilizada para que um atacante possa ter uma visão geral da estrutura da aplicação, facilitando assim a busca por páginas ou campos vulneráveis. Através desta técnica é possível obter as páginas e diretórios utilizados por determinada aplicação, e caso seja executado um web crawler autenticado (utilizando login de acesso), um atacante pode inclusive obter a estrutura de páginas acessíveis aos privilégios deste usuário. O crawler foi executado para obter o maior número possível de páginas da aplicação, sendo utilizada a credencial de administrador do sistema para tanto;

- Advanced Scanner (Nessus, Nexpose): Advanced Scanner é um tipo de varredura (scan) que pode ser customizada a fim de atender determinado teste. Foram ativados todos os plug-ins contidos nas ferramentas Nessus e Nexpose. Sendo assim, as ferramentas testaram vulnerabilidades tanto a nível de infraestrutura, como também em aplicações web;

- DoS - Write All Coils (Smod): O ataque consiste em realizar um ataque de negação de serviço ao escrever registros binários no dispositivo atacado;

- DoS - Write All Registers (Smod): O ataque consiste em realizar um ataque de negação de serviço ao escrever registros de 16 bits no dispositivo alvo;

- Fuzzing (Smod): Fuzzing é "um método para descobrir falhas de software, fornecendo entrada e monitoramento de exceções inesperadas" (SUTTON et. al, 2007). Trata-se de um processo realizado em grande parte de forma automatizada para ser aplicado nos mais diversos cenários, como na entrada de parâmetros em aplicações web, como pacotes em protocolos de rede, ou em entradas de programas e sistemas computacionais;

- Get Function (Smod): Trata-se de uma função utilizada para a enumeração do dispositivo Modbus, com a qual é possível identificar na rede quais dispositivos se comunicam através do protocolo Modbus, e quais funções são permitidas pelo dispositivo;

- Modbus Client (Metasploit): Através do Modbus Client, presente no framework Metasploit, um atacante pode ler e escrever dados nos dispositivos Modbus. Esta ferramenta foi usada tanto para ler, quanto para escrever dados do dispositivo;

- Modbus Discover (Nmap): Este script auxilia na obtenção de informações do dispositivo Modbus, buscando por Slave IDs (SIDS) dos dispositivos Modbus e por informações adicionais sobre o fabricante e o sobre firmware utilizado;

- Modbus Find Init ID (Metasploit): O módulo envia o comando 0x04 ler registro de entrada para o endpoint do Modbus. Se esse comando for enviado para o unit-id correto, o valor armazenado será retornado. Também é adicionado um valor - BENICE - para fazer com que o scanner hiberne um segundo ou mais entre as execuções, já que a varredura de muitos dispositivos de forma muito rápida pode resultar em um DoS;

- Port Scan (Nmap): Neste teste é feita uma verificação pelas portas que se encontram fechadas, escutando ou abertas. A ferramenta Nmap em seu modo default, i.e., sem que o range de portas seja fornecido, realiza a verificação das portas top 1000 TCP e UDP, ou melhor dizendo, as 1000 portas que são geralmente mais utilizadas de cada protocolo. Sendo assim, para realizar uma

ampla verificação, a ferramenta foi configurada para varrer todas as portas TCP (1 ~ 65535);

- Scanner UID (Smod): Esta função realiza um brute force do UID Modbus, i.e., realiza a leitura dos valores de cada UID identificado no dispositivo;

- SQL Injection (Acunetix, Arachni): O ataque de SQL Injection se vale do mesmo tipo de vulnerabilidade das demais vulnerabilidades de injeção, i.e., explora a forma insegura pela qual desenvolvedores interpretam os dados inseridos pelos usuários. Trata-se de um ataque similar ao Code Injection, diferenciado na parte em que os códigos fornecidos pelos usuários são interpretados e executados pelo Sistema de Gerenciamento de Banco de Dados (SGBD) utilizado pela aplicação. Um atacante consegue interagir fazendo consultadas na base de dados utilizada e, dependendo da tecnologia do SGBD utilizada e dos privilégios da conta de usuário do banco de dados, torna-se possível inclusive a execução de comandos diretamente no SO do servidor vulnerável;

- Cross-Site scripting (XSS) (Acunetix, Arachni): XSS é um ataque do tipo de injeção, no qual hackers enviam scripts maliciosos para usuários da aplicação. Pelo fato de o navegador não conseguir diferenciar o script malicioso de um script da aplicação, este acaba por executá-los.

5.4 Metodologia para geração de ataques

Os dois modelos de arquitetura foram atacados tanto em nível de aplicação, como em nível de rede. A segurança da aplicação web (ScadaBr) foi testada a partir de testes na camada de aplicação, já os ataques em nível de rede tiveram como alvo o servidor que hospeda o ScadaBr, como também os dispositivos Modbus conectados a esse servidor. Para testes na aplicação, foram realizados testes utilizando as vulnerabilidades presente no projeto OWASP Top 10 como guia de referência, i.e., vulnerabilidades como: SQL Injection, Cross-site scripting (XSS), Code Injection etc. Os ataques em nível de rede foram: Denial of Service (DoS), Probing (ataque que visa coletar informações do

sistema para que sejam utilizadas em um futuro ataque), CVEs das tecnologias utilizadas, leitura e escrita de dados em dispositivos Modbus.

Levando em conta a dificuldade de obter uma vulnerabilidade zero-day, mas a fim de executar ataques que simulariam o comportamento de um ataque como este, decidiu-se executar ataques utilizando *exploits*, conhecidos vastamente pela comunidade de segurança em aplicações, que não contam com os seus devidos patches de atualização instalados.

Inicialmente foram gerados 3 tipos de tráfego de dados, sendo:

- Pcap Tráfego de Usuários: Este pcap tinha como objetivo o de simular um tráfego de rede realizado por 100 usuários (IPs distintos). O tráfego foi composto por 100 máquinas clientes que geraram continuamente, ao longo do tempo, uma comunicação utilizando os protocolos HTTP, HTTPS, SSH e SMTP;

- Pcap Modbus: Este pcap salvou a comunicação do servidor ScadaBR, com 3 dispositivos Modbus. Os dispositivos Modbus foram simulados utilizando a ferramenta Modbuspal v.1.6c, e foram configurados da seguinte forma: 2 dispositivos configurados, um com 100 *Registers* e outro com 80 Registers, onde o ScadaBR os lia a cada segundo e a cada cinco segundos, respectivamente; e 1 dispositivo configurado com 50 *Coils*, onde o ScadaBR os lia a cada 3 segundos;

- Pcap Ataques: O arquivo continha 19 tipos de ataques, os quais tinham como objetivo atacar os dispositivos Modbus, a infraestrutura que hospedava o servidor ScadaBr, assim como o painel web utilizado para administrar o ScadaBr.

Tais ataques foram gerados a partir de técnicas comumente utilizadas por profissionais da área de *pentest* e pela execução das mais diversas ferramentas conhecidas e publicamente disponíveis através da internet. O conjunto de ataques gerado foi estabelecido levando em consideração os ataques comumente observados em aplicações Web, infraestruturas e em servidores SCADA, considerando que o SCADA é, em geral, exposto à Internet. No total, 19 máquinas atacantes geram periodicamente ataques de aplicações e de rede direcionados ao servidor SCADA. Da mesma forma, o comportamento do ataque variava na frequência e no rendimento do ataque.

Foram gerados basicamente dois fluxos de dados para a criação do modelo, sendo: (i) Tráfego normal, este composto pelos arquivos pcap Tráfego de Usuários e

Modbus, e (ii) Tráfego de Ataques, este composto pelos arquivos pcap Tráfego de Usuários, Modbus e Ataques. Cada comunicação de serviço normal e de ataque, variou a taxa de transferência do cliente (taxa de transferência de 35%, 70% e 100% de gigabit), a frequência de solicitação (0seg a 4seg) e o conteúdo solicitado (1000 possíveis conteúdos de serviço para cada protocolo). Para realizar o envio dos tráfegos pcap para seus correspondentes destinos, foi utilizada uma máquina com o SO Linux, com a ferramenta TCPReplay.

A base de testes usou as mesmas configurações e ferramentas para o desenvolvimento do protótipo (Figura 4). Portanto, foram realizadas duas execuções de teste para a coleta de dados adequadas de diversos sistemas operacionais. Uma execução usou uma máquina com sistema operacional Windows como front-end para máquinas normais e atacantes. A outra execução realizou o mesmo conjunto de operações normais e de ataque, no entanto usando o sistema operacional Linux como front-end.

A **Erro! Fonte de referência não encontrada.** mostra a quantidade de pacotes de rede gerados e o tempo de execução de cada ataque no ambiente de teste para os cenários Windows e Linux, com duração 96 horas cada. É importante observar que, conforme implementado pelo protótipo, uma instância para classificação é gerada a cada 10 segundos para os front-ends dos sistemas operacionais Windows e Linux.

Tabela 2 - Pacotes de rede gerados e tempo de execução dos ataques

Comportamento (Ferramenta)	Ambiente de teste			
	Windows		Linux	
	Net. Pkt.	Temp. Exec. (m)	Net. Pkt.	Temp. Exec. (m)
Normal (Workload)	5.1B	5760	5.1B	5760
Auth Crawler (Acunetix)	77k	313	77k	311
Nonauth Crawler (Acunetix)	128k	184	129k	182
SQL Injection (Acunetix)	13k	212	13k	210

XSS (Acunetix)	6k	1028	6k	1027
Code Injection (Arachni)	17k	392	17k	391
DoSAllRegisters (Smod)	58k	153	58k	152
Get Functions (Smod)	2k	133	2k	133
Scanner (ModFuzzer)	0.4k	4	0.4k	4
DumpScan (ModFuzzer)	1k	4	1k	4
Portscan (Nmap)	140k	33	140k	33
Modbus Scan (PLCScan)	3k	22	3k	21
Basic scan (Nessus)	45k	52	45k	52
Advanced scan (Nessus)	35k	33	36k	34
Default scan (Nexpose)	150k	56	151k	56
Advanced scan (Nexpose)	38k	86	39k	87
SQL Injection (Arachni)	4k	70	4k	69
Fuzzing (Smod)	320k	20	320k	20
Scanner (Smod)	2k	113	3k	112
DoSAllCoils (Smod)	296k	164	296k	160

5.5 Características dos SOs

Os tráfegos normais e de ataque tinham como destino os sistemas operacionais Windows e Linux, os quais foram configurados para coletar o seu comportamento através das ferramentas System Monitor (Sysmon) e Sysstat, respectivamente. O Sysmon utiliza

um total de 24 ID’s para classificar os eventos por ele monitorados, entretanto, das 24 ID’s monitoradas, foram escolhidas 22 ID’s, já que 2 ID’s não sofreram algum tipo de alteração durante os testes, e através da ferramenta Sysstat, era possível gerar um total de 39 métricas. As métricas geradas pelas ferramentas correspondiam às estatísticas de uso do SO, como tempo de uso da CPU, número de processos, número de pacotes de rede enviados/recebidos, entre outros.

Os dados do Sysmon foram obtidos através do arquivo gerado pela ferramenta Event Viewer, "Log de evento (*.evtx)". Foi desenvolvido um programa em Python responsável por, a cada intervalo de janela de 10 segundos, o programa realizaria o backup do arquivo .evtx, o converteria em dois arquivos XML, sendo um arquivo de todos os eventos identificados pelo Sysmon, e outro arquivo contendo, separadas em grupos, a soma de todas as ID’s. Posteriormente, o programa lia os arquivos XML gerados, e os convertia para o formato CSV, formato este que era então lido pelo algoritmo de aprendizado de máquina. Para obter as 39 métricas foi necessário habilitar, no arquivo de configuração da ferramenta, a opção que gera o relatório completo de eventos. A ferramenta geralmente faz uso da agenda de tarefas **cron** do Linux para executar a leitura de eventos do SO em um período de tempo pré-configurado pelo administrador do sistema, entretanto, a cron utiliza uma janela de tempo mínima de 1 minuto para execução de um evento agendado. Foi utilizado um método que tornava possível a execução do comando a cada 10 segundos, porém, este método não se muito eficaz. Sendo assim, para obter os arquivos de log da ferramenta Sysstat foi necessário desenvolver um arquivo em Python que a cada intervalo de janela de 10 segundos, ele executava o comando da ferramenta para que o Sysstat gerasse um arquivo de todos os eventos do SO Linux. A ferramenta salvava os arquivos em formato de texto, sendo assim, o programa desenvolvido lia estes arquivos texto e os convertia em arquivos CSV, que posteriormente era lidos pelos algoritmos de aprendizado de máquina. O conjunto de recursos coletado foi então classificado pelos algoritmos de ML, conforme implementado por meio da versão 0.23 da API do SkLearn. O evento classificado e seus recursos são enviados ao módulo Seletor de pool de SO por meio da API do serviço web SOAPpy versão 0.9.8. O módulo Seletor de pool de SO avalia o evento recebido por meio do SkLearn. Se um evento não confiável for encontrado, o front-end atualiza as regras de tráfego de rede para encaminhar outras solicitações para outro SO.

5.6 Construção do Modelo

Os conjuntos de dados gerados foram utilizados para a construção de dois modelos de classificação ML relacionados ao SO, para sistemas operacionais Windows e Linux. Os dados do ambiente de teste foram divididos em dados de treinamento, teste e validação. Os conjuntos de dados de treinamento e teste foram compostos por 50% do total de eventos normais e 5 das 17 categorias de ataque (verificação avançada Nexpose), injeção de SQL (Arachni), Fuzzing (Smod), scanner (Smod) e DoSAllCoils (Smod)**Erro! Fonte de referência não encontrada.**, em que 70% desses dados foram usados para treinamento e os 30% restantes para teste.

Os 50% restantes do tráfego normal e 12 categorias de ataque foram usados para a validação final do modelo ML. Ao aplicar essa abordagem, foi possível imitar o comportamento do ambiente de produção, em que o modelo ML encontrou um novo comportamento de ataque (desconhecido). Os modelos de classificação avaliados foram implementados por meio da API SKLearn versão 0.23.

Três modelos de classificação amplamente utilizados foram avaliados, sendo a Random Forest e a Gradient Boosting, ambos com 100 árvores de decisão com base learners e um classificador Extra-Tree. Os classificadores foram treinados através de conjuntos de dados de treinamento e teste, enquanto a precisão final foi relatada através do conjunto de dados de validação. Uma subamostragem aleatória sem substituição foi aplicada nos dados de treinamento para remover o desequilíbrio entre as classes.

5.7 Avaliação

5.7.1 Metodologia para avaliação/testes

O primeiro experimento refere-se à pergunta Q1 e avalia o desempenho do classificador para a classificação de novas categorias de ataque por meio de diferentes conjuntos de recursos do SO. A **Erro! Fonte de referência não encontrada.** mostra a curva ROC (*Receiver Operating Characteristics*) para os três modelos de ML avaliados no conjunto de dados de teste do sistema operacional Windows. É possível observar que

os classificadores avaliados apresentaram taxas de precisão semelhantes em relação aos conjuntos de recursos subjacentes do SO. Portanto, a precisão da classificação de ML dependeu estritamente do conjunto de recursos usados.

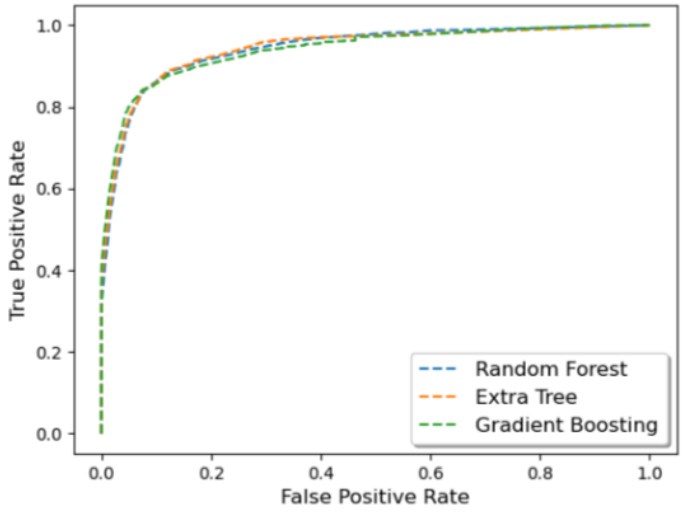

Figura 5 - Curva ROC para três classificadores no SO *Windows*

Para avaliar o impacto da diversidade de SO na performance da classificação, foi comparada a precisão relatada no Windows e no Linux com relação ao conjunto de dados de validação, lembrando que o conjunto de dados de validação é constituído por novos comportamentos de categorias de ataque, ou seja, comportamentos não utilizados na fase de treinamento.

A **Erro! Fonte de referência não encontrada.** (colunas Windows e Linux) mostra a precisão obtida para o classificador Random Forest para cada categoria de ataque e conjunto de recursos do SO. Diferentemente da análise anterior, o desempenho da classificação varia significativamente de acordo com o conjunto de recursos do SO usado. É possível observar que o classificador Linux supera a abordagem do Windows para a detecção de comportamento normal.

Tabela 3 - Performance com o classificador *Random Forest* obtido pelo Windows, Linux e a proposta

Comportamento (Ferramenta)	Acurácia (%)		
	Windows	Linux	Proposta

			Pior caso	Melhor caso
Normal (Workload)	91.40	98.79	91.40	98.79
Auth Crawler (Acunetix)	85.00	100.00	93.76	100.00
Nonauth Crawler (Acunetix)	86.74	83.00	86.72	86.74
SQL Injection (Acunetix)	84.66	100.00	93.66	100.00
XSS (Acunetix)	83.74	77.98	83.73	83.74
Code Injection (Arachni)	90.03	83.50	90.01	90.03
DoSAllRegisters (Smod)	84.05	96.83	92.07	96.83
Get Functions (Smod)	85.71	100.00	94.67	100.00
Scanner (ModFuzzer)	93.75	75.00	93.74	93.75
DumpScan (ModFuzzer)	94.11	54.54	94.07	94.11
Portscan (nmap)	88.41	98.03	98.01	98.03
Modbus Scan (PLCScan)	81.19	55.29	81.13	81.19
Basic scan (Nessus)	72.16	82.48	81.61	82.48
Advanced scan (Nessus)	79.05	95.65	93.85	95.65
Default scan (Nexpose)	70.74	90.15	90.10	90.15
Avg. System Accuracy	84.72	86.08	90.57	92.77

Com relação à classificação de novas categorias de ataque, o modelo ML relacionado ao Windows é capaz de fornecer maior precisão de detecção para seis categorias de ataque, enquanto o modelo relacionado ao Linux atinge taxas de detecção mais altas para oito categorias de ataque. Portanto, a diversidade do SO para a detecção de intrusões ajuda a melhorar a taxa geral de detecção de intrusões para novos ataques.

O segundo experimento refere-se à questão Q2 e visa avaliar a abordagem do Pool Selector do SO para melhorar a precisão através da diversidade do SO. Foi implementada uma abordagem de limite relacionada à classe (CRT) para estabelecer se o sistema operacional atual em uso deve ser mantido ou alterado. Em outras palavras, o avaliador contido na proposta (Evaluator, **Erro! Fonte de referência não encontrada.**) mantém ou altera o SO atual de acordo com seu valor de confiança na classificação do modelo de ML. Por exemplo, o classificador Random Forest gera um valor de confiança de acordo com a proporção de árvores individuais que conferiram o rótulo atribuído ao evento.

A frequência de alteração do SO deve ser realizada de acordo com o critério do administrador. Para os fins de testes, foi selecionado um limite de valor de confiança de 0,9. Portanto, classificações executadas com valor de confiança menor que 0,9 acionaram uma rotina de mudança de SO do front-end.

A **Erro! Fonte de referência não encontrada.** (colunas da Proposta) mostra a precisão obtida usando a proposta, considerando os piores e os melhores cenários e, ao mesmo tempo, informando que o front-end do SO atual deve ser alterado de acordo com sua confiança na classificação. O pior cenário é encontrado quando o sistema operacional atual em uso fornece a menor precisão de detecção para o ataque atual, com uma média de 84,72% de acurácia do sistema. Portanto, a abordagem CRT usada garante o nível de precisão através do limite definido (0,9 para os experimentos). Por outro lado, o melhor cenário é encontrado quando o sistema operacional atual fornece a maior precisão de detecção para o ataque atual. A média de acurácia do sistema é de 92,77%.

É possível observar que a solução proposta é sempre capaz de superar em cerca de 6% o sistema operacional que obteve a menor precisão (pior cenário, média de acurácia do sistema = 90,57 - 84,72 = 5,85%), Linux ou Windows. Ao mesmo tempo, também é possível atingir o mesmo nível das melhores taxas de precisão obtido no melhor sistema operacional de acordo com a categoria de ataque classificado (melhor cenário, média de acurácia do sistema = 92,77 - 84,72 = 8,05%). Em outras palavras, a solução proposta é

capaz de alavancar a melhoria da precisão obtida da diversidade do SO para identificar e alterar automaticamente o SO usado a fim de aumentar a precisão geral da classificação de ataques, conforme mostrado na **Erro! Fonte de referência não encontrada.** (precisão média do sistema).

A terceira e última investigação refere-se à pergunta Q3, que se refere a frequência de mudança do SO do modelo. Essa propriedade é altamente dependente do comportamento atual do ambiente, a exemplo da categoria de ataque Crawler Autenticado (Acunetix) mostrada na **Erro! Fonte de referência não encontrada.**.

Se o sistema operacional Windows estiver sendo usado como front-end, o sistema proposto mudará assim que um evento for classificado com menos de 0,9 de confiança. Posteriormente, o sistema operacional Linux não será alterado, levando em consideração que sempre executará classificações confiáveis. Por outro lado, considerando o ataque do Crawler Não Autenticado (Acunetix), independentemente do sistema operacional sendo usado, várias alterações podem ocorrer. Isso se dá porque ambos os sistemas operacionais podem realizar classificações não confiáveis, como observado pelas taxas de precisão obtidas (~85%). Consequentemente, pode-se usar a alteração de frequência do sistema operacional como uma indicação de que um novo ataque está ocorrendo, enquanto uma baixa precisão está sendo alcançada para sua classificação, exigindo investigação adicional do administrador.

Capítulo 6

Conclusão

A confiabilidade da detecção de intrusões nos sistemas SCADA durante a detecção de novos comportamentos de ataque é frequentemente negligenciada na literatura. Este trabalho apresentou um novo e confiável modelo de detecção de intrusão baseado em *host* para o SCADA por meio da diversidade do SO para a detecção de outras categorias de ataque.

As avaliações realizadas mostraram que a diversidade do conjunto de recursos, através da diversidade do SO, ajuda a melhorar a detecção do novo comportamento do ataque, i.e., os ataques tendem a apresentar diferentes comportamentos em diferentes SO, como por exemplo, o ataque de SQL Injection foi mais facilmente identificado pelo SO Linux, enquanto o ataque de XSS Injection obteve uma maior taxa de assertividade no SO Windows. Isto prova que o uso da diversidade, além de ser utilizada para dar mais confiabilidade ao evitar o comprometido total de um sistema, esta também se mostra útil no reconhecimento das mais diversas ameaças.

Como trabalho futuro, visa-se a implementação do modelo proposto em redes definidas por software para maior transparência e para alavancar maior diversidade de SO e assim executar a detecção por meio de um conjunto de recursos de diversidade de SO. Adicionalmente, busca-se desenvolver um script que seja capaz de alternar automaticamente de SO, quando o algoritmo de ML identificar que o SO em uso não é o mais indicado para lidar com o ataque em execução. Busca-se também implementar toda a arquitetura desenvolvida em um servidor baseado em nuvem, e com isso tornar mais dinâmica a execução do sistema desenvolvido, assim como evitar o uso simultâneo de diversos servidores.

Referências Bibliográficas

ALVES, Fernando; BETTINI, Aurélien; FERREIRA, Pedro M.; BESSANI, Alysson. Processing tweets for cybersecurity threat awareness. arXiv preprint arXiv:1904.02072, 2019.

AVASARALA, Bhargav R.; BOSE, Brock D.; DAY, John C.; STEINER, Donald. System and method for automated machine-learning, zero-day malware detection. U.S. Patent Application No. 14/038,682.

AVIZIENIS, Algirdas; KELLY, John P. J.. Fault Tolerance by Design Diversity: Concepts and Experiments, Computer, vol. 17, no. 8, pp. 67-80, Aug. 1984.

CERTI - ScadaBR. Documento de Referência do ScadaBR. Disponível em: https://sites.google.com/a/certi.org.br/certi_scadabr/requisitos

CERTI – ScadaBR. Documento de Referência do ScadaBR.. Minicurso de ScadaBR. Disponível: https://sites.google.com/a/certi.org.br/certi_scadabr/home/minicursos/iniciando-scadabr

DUA, Mohit; Kunal. Machine Learning Approach to IDS: A Comprehensive Review. In: 2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA). IEEE, 2019. p. 117-121.

GARCIA, Miguel; BESSANI, Alysson; GASHI, Ilir; NEVES, Nuno; OBELHEIRO, Rafael. Analysis of operating system diversity for intrusion tolerance. Software: Practice and Experience, v. 44, n. 6, p. 735-770, 2014.

GARITANO, Iñaki; URIBEETXEBERRIA, Roberto; ZURUTUZA, Urko. Review of SCADA anomaly detection systems." In Soft computing models in industrial and environmental applications, 6th international conference SOCO 2011, pp. 357-366. Springer, Berlin, Heidelberg, 2011.

GHOSH, Sagarika; SAMPALLI, Srinivas. A survey of security in SCADA networks: Current issues and future challenges. IEEE Access, v. 7, p. 135812-135831, 2019.

GORBENKO, Anatoliy; KHARCHENKO, Vyacheslav; TARASYUK, Olga; ROMANOVSKY, Alexander. Using diversity in cloud-based deployment environment to avoid intrusions. In: International Workshop on Software Engineering for Resilient Systems. Springer, Berlin, Heidelberg, 2011. p. 145-155.

HU, Yan; YANG, An; LI, Hong; SUN, Yuyan; SUN, Limin. A survey of intrusion detection on industrial control systems 2018. International Journal of Distributed Sensor Networks, v. 14, n. 8, 2018.

KARR, The IT security vicious cycle of "Assuming Compromise, Apr 2015, Disponível em: http://www.itproportal.com/2015/02/security-vicious-cycle-assuming-compromise/.

LEE, Robert M.; ASSANTE, Michael J.; CONWAY, Tim. TLP: White Analysis of the Cyber Attack on the Ukrainian Power Grid. E-ISAC, Tech. Rep, 2016.

MOHURLE, Savita; PATIL, Manisha. A brief study of wannacry threat: Ransomware attack 2017. International Journal of Advanced Research in Computer Science, v. 8, n. 5, 2017.

NOGUEIRA, André; GARCIA, Miguel; BESSANI, Alysson; NEVES, Nuno. On the Challenges of Building a BFT SCADA. In: 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). IEEE, 2018. p. 163-170.

SUN, Chih-Che; HAHN, Adam; LIU, Chen-Ching. Cyber security of a power grid: State-of-the-art. International Journal of Electrical Power & Energy Systems, v. 99, p. 45-56, 2018.

SUTTON, Michael, Adam Greene, and Pedram Amini. Fuzzing: brute force vulnerability discovery. Pearson Education, 2007.

YANG, Yiming; PEDERSEN, Jan O. A comparative study on feature selection in text categorization. In: Icml. 1997. p. 35.

ZIMBA, Aaron; WANG, Zhaoshun; CHEN, Hongsong. Multi-stage crypto ransomware attacks: A new emerging cyber threat to critical infrastructure and industrial control systems. Ict Express, v. 4, n. 1, p. 14-18, 2018.

Apêndice

Definição das ferramentas de ataque

A base de testes que foi utilizada para testar vulnerabilidades na camada de aplicação foi formada por tipos de ataques presentes no projeto OWASP Top 10. A OWASP é uma comunidade aberta que se dedica a ajudar as organizações a desenvolver, adquirir e manter suas aplicações seguras. O OWASP Top 10 é um dos projetos mantidos pela OWASP, onde são compiladas as ameaças mais comuns encontradas nas aplicações web.

Foram utilizadas as ferramentas automatizadas para verificar a presença de vulnerabilidades em aplicações web, nos servidores, infraestrutura e no sistema SCADA, nas seguintes soluções: *Acunetix, Arachni, Nessus, Nexpose, NMap, Metasploit Framework, e Smod.*

Servidor ScadaBR

No projeto foi utilizado o *software* ScadaBR, versão 1.0CE, disponível em https://sourceforge.net/projects/scadabr/files/.

A **Erro! Fonte de referência não encontrada.** apresentou o modelo de arquitetura utilizado no servidor que hospeda o *software* ScadaBR que foi instalado em um computador com o SO *Windows*, e com o servidor de aplicação *Apache Tomcat*, denominado de "Servidor ScadaBR". Tanto as telas de interface do usuário como também as de configurações do ScadaBR foram acessadas através de um navegador de Internet. (CERTI s/d).