Detecção de intrusão de Ransomware apoiado por IA

Mateus Mazaro Biscaglia
Curso de Ciência da Computação
Universidade Franciscana
CEP 97010-032 – Santa Maria, RS, Brasil
m.biscaglia@ufn.edu.br

Sylvio André Garcia Vieira
Curso de Ciência da Computação
Universidade Franciscana
CEP 97010-032 – Santa Maria, RS, Brasil
sylvio@ufn.edu.br

Resumo - Este artigo apresenta o desenvolvimento de um software para detecção precoce de ransomware utilizando IA em ambientes Windows, com foco na análise comportamental e estática de processos. O ransomware hoje é uma das maiores ameaças cibernéticas, dando prejuizos bilionários anualmente, sendo um malware que infecta o dispositivo, buscando dados importantes, criptografando-os e em muitos casos solicitando resgate financeiro para entregar a chave da criptografia. A solução proposta foi implementada em Python, utilizando bibliotecas como psutil, pandas e scikit-learn, aliando aprendizado de máquina à observação de dados durante a execução do sistema. O projeto monitora continuamente atividades de CPU, memória, rede e manipulação de arquivos, aplicando modelos de IA treinados com dados reais e investigação de comportamento suspeito. O sistema constatou capacidade de reconhecer padrões associados a ataques antes da criptografia do ransomware, atuando como uma ferramenta preventiva. Os resultados obtidos indicam que a abordagem híbrida, estática e comportamental, pode oferecer uma camada adicional de segurança ao usuário, com baixo impacto no desempenho da máquina e alta precisão na identificação de ameaças.

I. Introdução

Os ataques digitais se tornaram uma das principais ameaças à segurança de dados no mundo todo [1]. Um dos mais perigosos é o malware chamado ransomware, os ataques dele muitas vezes funcionam sequestrando informações importantes dos usuários ou empresas, tornando-as inacessíveis até que alguma exigência seja cumprida, geralmente envolvendo pagamento em moedas virtuais. Estima-se que em 2023, com ao menos 538 variantes novas, houve um prejuízo de mais de 1 bilhão de dólares para empresas de diversos setores [1] . Esse tipo de estratégia tem se espalhado com rapidez, aproveitando falhas em sistemas operacionais populares, como o *Microsoft Windows*, que é o sistema operacional mais usado no mundo, tanto em ambientes domésticos quanto corporativos [2], o que o torna um alvo natural para esse tipo de ataque.

O problema é que muitos usuários ainda não compreendem como esses ataques funcionam, tampouco percebem que estão sendo vítimas de ataques cibernéticos até que o dano já tenha ocorrido. As defesas tradicionais, como antivírus e firewalls, oferecem certa proteção, mas nem sempre são suficientes. Em muitos casos, essas soluções identificam o ataque apenas após a ocorrência do dano. Isso evidencia a necessidade de desenvolver métodos capazes de detectar a ameaça de forma antecipada, ou seja, antes que a infecção e criptografia de fato aconteça.

Entende-se que nem sempre os antivírus tradicionais conseguem detectar esses ataques a tempo, ainda mais sendo um tipo novo de ransomware. Observa-se então, a necessidade de criar um sistema inteligente, capaz de identificar padrões de pré-infecção do malware, algumas vezes até sem conhecer exatamente a ameaça. A detecção precoce, nesse contexto, virou uma peça-chave para minimizar danos e evitar que dados importantes sejam perdidos ou comprometidos. Com o uso de IA, torna-se possível identificar padrões de ataque com base em comportamentos e comandos específicos utilizados pelo vírus.

Foi aplicado o uso de Inteligência Artificial (IA) para criar uma defesa preventiva, identificando os primeiros sinais que o *malware* possa demonstrar ao alterar o comportamento normal do sistema operacional. Para comprovar essa abordagem na prática, foi desenvolvido um protótipo de *software* em python para ser executado no sistema operacional *Windows* que monitora o sistema instantaneamente e dispara alertas automáticos sempre que detecta padrões anômalos. O estudo também buscou entender os padrões utilizados pelos *malwares* do tipo *ransomware*, para então auxiliar na sua detecção precoce, testando métodos que possam emitir o alerta antes que o *malware* entre em ação.

A proposta deste Trabalho de Conclusão de Curso (TCC) foi entender como esses ataques funcionam e propor formas de reconhecer comportamentos suspeitos o mais breve possível, antes que danos mais severos possam acontecer. Então, foram investigados padrões de pré-infecção que alteram o funcionamento normal do sistema operacional, com o intuito de sustentar estratégias voltadas à detecção precoce. Foram definidos como objetivos específicos o estudo de técnicas de Inteligência Artificial (IA) para auxiliar na identificação desses sinais preliminares e a análise de métodos para monitoramento contínuo de processos e atividades no *Windows*. Também, foi pesquisado compreender os padrões característicos de

malwares do tipo *ransomware*, a fim de sustentar a construção de mecanismos capazes de indicar potenciais riscos antes do início da infecção. Com isso, desenvolveu-se um protótipo de *software* destinado a avaliar os comportamentos anômalos do sistema, que serviram como base experimental para testar diferentes abordagens de detecção precoce.

II. Referencial Teórico

Nesta seção, serão descritos conceitos essenciais para a compreensão do trabalho, além das tecnologias que foram utilizadas para o desenvolvimento.

A. Ransomware

Segundo Santos [3], *ransomware*, que vem do inglês "*ransom* (resgate) e *ware* (*malware*)", é um tipo de software malicioso que bloqueia o acesso aos dados da vítima, pedindo um resgate para liberá-los, geralmente em criptomoedas como o Bitcoin. Ele está entre as ameaças digitais mais comuns e perigosas, agem geralmente com o método de criptografar os arquivos da vítima e exigir um valor para liberar o acesso. Com o tempo, os cibercriminosos foram criando variações desse malware que teve inicio em 1989, com o nome Cavalo de Tróia AIDS (ou PC Cyborg Trojan), foi feito a infecção por disquetes, bloqueando arquivos e exigindo o resgate após pagamento de 189 dólares via caixa postal física.

Após décadas de evolução, o ransomware apresenta-se em três tipos mais comuns, de acordo com Santos [3], *Locker ransomware*, que atua bloqueando completamente o acesso ao dispositivo da vítima ao ser executado, *Scareware*: Aparece como um alerta falso, simulando ser uma autoridade ou antivírus, com o objetivo de assustar a vítima e induzi-la a realizar o pagamento. E o tipo em questão que foi base para as pesquisas desse projeto, o *Crypto-ransomware*: É o tipo mais perigoso. Ele criptografa os arquivos do sistema, tornando-os inacessíveis até que o resgate seja pago. A recuperação dos dados depende do pagamento do resgate para obter a chave de decriptação, sem garantia muitas vezes, ou de uma solução técnica alternativa, que normalmente é falha ou raramente eficiente.

B. Comportamento Inicial de Ransomwares

Segundo S.H. Kok [4], os atacantes utilizam diversos métodos para conseguir o acesso inicial a um ambiente. Conforme ilustrado na Figura 1, o acesso inicial pode ser obtido por quatro vetores principais: phishing, password guessing, exploração de vulnerabilidade e e-mail. O phishing é um dos mais comuns, onde e-mails falsos enganam o usuário para fornecer credenciais válidas ou clicar em links maliciosos que levam à obtenção de credenciais válidas. De um modo parecido, o password guessing também visa a obtenção de credenciais válidas. Essas credenciais, por sua vez, podem ser utilizadas para acessar serviços expostos na internet, como servidores mal configurados, que também podem ser explorados diretamente por sua vulnerabilidade.

Vários ataques começam por meio de e-mails que contêm documentos maliciosos como scripts embutidos com o objetivo

Figura 1. Acesso Inicial [5]

de infecção, após essa fase, o ransomware atua facilitando a entrada de acessos não autorizados e ferramentas que garantem a permanência do invasor no sistema. Antes mesmo da criptografia o atacante já estabelece uma base sólida, conhecida como "Preparação e Persistência", que consolida o acesso e prepara o terreno para as próximas ações, como a disseminação do ransomware, reforçando a ilustração da Figura 1.

Com base nas informações da CISA (*Cybersecurity and Infrastructure Security Agency*) [6] , após a fase de preparação do ataque, mas antes da infecção e da criptografia dos arquivos, o *ransomware* geralmente realizará uma série de ações silenciosas, sem ainda causar danos diretos. Contudo, apresentará sinais válidos que exigirão atenção, como processos com nomes incomuns, uso suspeito do *PowerShell*, tentativas de desativação de antivírus, criação e modificação de arquivos em massa e até a remoção de logs do sistema. A proposta foi observar esses comportamentos logo no início e agir preventivamente antes que o ataque seja efetivado.

Inicialmente, o *ransomware* tenderá a se instalar utilizando nomes genéricos e em diretórios ocultos do sistema, como *AppData* ou *Temp*, visando se manter despercebido tanto pelos usuários quanto por algumas soluções antivírus. Outro método comum é o uso de *scripts* em *PowerShell* ou *Prompt* de Comando, com o propósito de desativar o *Windows Defender*, apagar *backups* (como as *Shadow Copies*) e desligar serviços essenciais de proteção do sistema. Alguns *ransomwares* também modificarão configurações no registro do *Windows*, para o preparo do ambiente para a execução completa do ataque.

Um comportamento que chama bastante atenção é que ataques de *ransomware* são projetados para criptografar rapidamente arquivos críticos, tornando-os inacessíveis em questão de minutos. Essa velocidade de execução, antes que humanos possam intervir, destaca a importância de detectar comportamentos atípicos no sistema de arquivos, como acessos e alterações em massa, para prevenir danos irreversíveis [7].

Em determinados casos de ataques mais sofisticados, o *hacker* não inicia a ação maliciosa imediatamente. Antes de criptografar os arquivos, é realizada uma etapa conhecida como reconhecimento, na qual o ambiente é analisado em busca de vulnerabilidades. Essa análise inclui a identificação do tipo de máquina invadida (usuário comum, servidor ou domínio corporativo), os tipos de dados disponíveis, a existência

de *backups*, os processos em execução, os antivírus ativos, entre outros aspectos relevantes [8].

Somente após essa fase o *ransomware* se conecta ao seu servidor remoto de controle, conhecido na área de segurança como C&C ou C2 (*Command and Control*). Essas técnicas permitem que os criminosos monitorem e controlem o ataque de dispositivos comprometidos na rede, além de possibilitar a coleta de dados antes do início da criptografia. [9].

C. Python

Python é uma linguagem de programação de alto nível, criada por Guido van Rossum e lançada em 1991, ela é uma linguagem relativamente simples, fácil de ler e muito prática no seu desenvolvimento, sendo também interpretada e multiparadigma, o que significa que ela permite diferentes estilos de programação, como orientado a objetos e funcional. Além disso, ela conta com uma biblioteca padrão muito rica e um ecossistema cheio de pacotes criados pela comunidade, o que facilita a criação de aplicações para as mais diversas áreas, desde automação e desenvolvimento web até análise de dados, *machine learning*, inteligência artificial e computação em nuvem. Grandes empresas como *Google, Netflix* e até a *NASA* usam *Python* em seus sistemas, o que mostra o quanto ela é robusta e confiável [10].

D. Bibliotecas

Uma biblioteca, em programação *Python*, é um conjunto de funções ou módulos prontos que podem ser reutilizados na criação de *softwares* ou códigos. Elas existem para tornar o processo de desenvolvimento mais prático, permitindo que o programador aproveite funcionalidades que já foram implementadas por outros. Na prática, funcionam como pacotes que podem ser importados para dentro do projeto. Ao fazer isso, o desenvolvedor passa a ter acesso às funções e classes oferecidas pela biblioteca, com a vantagem de economia de tempo no desenvolvimento [11].

Neste projeto, utilizando a linguagem *Python*, as principais bibliotecas empregadas foram as seguintes descritas:

1. *Psutil*, (*Python System and Process Utilities*), é uma ferramenta multiplataforma que permite acessar informações sobre processos em execução e a utilização de recursos do sistema, como CPU, memória, discos, rede e sensores. É utilizada para monitoramento de sistemas, criação de perfis de desempenho e gerenciamento de processos em aplicações *Python* [12].

2. *Pandas* é amplamente utilizada para manipulação e análise de dados. Ela tem uma estruturas de dados eficiente, como os *DataFrames*, que funcionam de forma parecida a tabelas de bancos de dados ou planilhas. Isso permite que os dados sejam organizados de maneira clara e estruturada, facilitando o processo de leitura, limpeza, transformação e análise. Com esses dados organizados, torna-se possível aplicar técnicas de análise exploratória e alimentar os modelos de inteligência artificial com informações bem formatadas. [13].

3. *NumPy*, (*Numerical Python*), é usada na area de computação numérica em *Python*. Ela oferece suporte a *arrays* e matrizes multidimensionais, além de fornecer várias coleções de funções matemáticas otimizadas para operar sobre essas estruturas de dados da melhor forma. Ela é muito usada em operações matemáticas mais complexas, desde cálculos estatísticos até manipulação de grandes volumes de dados numéricos [14].

4. *Scikit-learn* é uma das bibliotecas mais populares e consolidadas para aprendizado de máquina. Ela fornece uma gama de algoritmos prontos para tarefas como classificação, regressão, agrupamento e redução de dimensionalidade, além de ferramentas para avaliação de modelos e pré-processamento de dados. Por ser construída sobre bibliotecas como numpy e pandas, ela se destaca pela eficiência no processamento de dados [15].

5. *Joblib* é voltada principalmente para a serialização de objetos e a execução paralela de tarefas, com foco em desempenho e eficiência no uso de memória. Ela é muito utilizada em aprendizado de máquina, especialmente para salvar modelos treinados de forma rápida e reutilizável. Ela é otimizada para armazenar objetos grandes que contêm *arrays* do NumPy, tornando-o ideal para modelos de aprendizado de máquina e grandes conjuntos de dados. Além disso, a biblioteca permite a execução de tarefas em paralelo com facilidade, o que pode acelerar muito o processos como o treinamento de modelos ou o processamento de dados em etapas independentes[16].

6. *Matplotlib* é uma ferramenta para criação de visualizações estáticas, animadas e interativas em *Python*. Ela foi projetada para tornar tarefas simples de visualização fáceis de executar, ao mesmo tempo em que permite a realização de representações gráficas complexas. Possibilita a geração de gráficos científicos, diagramas e figuras personalizadas. É amplamente utilizada em áreas como ciência de dados, engenharia, estatística e pesquisa, sendo uma das principais ferramentas de visualização da linguagem [17].

E. Inteligência Artificial

Segundo Sichman [18], Inteligência Artificial (IA) é um campo da Ciência da Computação voltado ao desenvolvimento de sistemas capazes de simular aspectos do comportamento humano, como aprender, raciocinar, perceber e tomar decisões. Diferente da computação tradicional, que segue instruções pré-programadas, a IA busca lidar com problemas complexos para os quais nem sempre existem algoritmos bem definidos. Isso permite que os sistemas ajam em situações inéditas, adaptem-se com base na experiência e tomem decisões a partir da análise de dados.

Essa capacidade aproxima o funcionamento da IA do modo

como os seres humanos pensam e agem. Entre as principais técnicas utilizadas estão o aprendizado de máquina, as redes neurais artificiais e o processamento de linguagem natural, que tornam os sistemas mais autônomos e eficientes ao lidar com grandes volumes de informação e variáveis não previsíveis.

F. Aprendizado de Máquina (ML)

Com base no artigo *Machine Learning* na Medicina: Revisão e Aplicabilidade [19], os autores embasam que *Machine Learning* (ML), termo em inglês para aprendizado de máquina, é uma área da inteligência artificial que permite aos computadores aprenderem diretamente a partir de dados, sem a necessidade de serem programados para concluir cada tarefa de forma específica. Em vez de seguir regras especificas definidas por programadores, esses sistemas conseguem analisar grandes volumes de informações, identificar padrões e usar esses padrões para tomar decisões ou fazer previsões. Essa característica tem revolucionado setores como a medicina, finanças, indústria e muitos outros, tornando os processos mais rápidos, eficientes e precisos.

Além disso, a habilidade do ML em lidar com o que chamamos de "*big data*", que se refere a conjuntos de dados muito grandes, variados e que chegam em alta velocidade, possibilita o desenvolvimento de modelos preditivos mais robustos e flexíveis, que conseguem se ajustar conforme surgem novos dados. Para criar esses modelos, são seguidas etapas importantes, como o pré-processamento dos dados, que garante que eles estejam organizados, sejam numéricos e de qualidade; O treinamento, que pode ser supervisionado ou não supervisionado; E, finalmente, a avaliação do modelo, feita com dados que não foram usados no treinamento para verificar sua eficácia. A seguir, são apresentados os tipos e os principais tipos e técnicas de *Machine Learning*: [20]

Tipos de *Machine Learning*:

Aprendizado Supervisionado: Neste método, que foi o usado no TCC, o sistema aprende a partir de um conjunto de dados que já possui exemplos com as respostas corretas (rotuladas). Por exemplo, se for solicitado que um modelo aprenda a identificar um exame médico, onde ele indica uma doença ou não, é fornecido vários exames já classificados e o modelo aprende atraves de um metudo quantitativo de dados a relacionar características dos exames com a classificação correta. Depois de treinado, ele pode classificar novos exames desconhecidos. As tarefas mais comuns são:

Classificação: Categorizar dados em grupos predefinidos (ex: diagnostico de doença).

Regressão: Prever valores contínuos (ex: estimar pressão arterial).

Aprendizado Não Supervisionado: Aqui, o sistema recebe dados sem nenhuma indicação sobre quais seriam as respostas corretas. O objetivo é descobrir padrões, sem orientação externa. Técnicas comuns incluem: *Clustering* (agrupamento) que agrupa dados similares em *clusters* e Redução de dimensionalidade que simplifica dados complexos com muitas variáveis para facilitar a visualização e análise, sem perder informações importantes.

Aprendizado por Reforço: Essa técnica envolve um agente que interage com um ambiente e aprende tomando decisões que maximizam uma recompensa ao longo do tempo. Ele recebe *feedbacks* (recompensas ou punições) e ajusta seu comportamento para obter os melhores resultados possíveis.

As Técnicas mais utilizadas em *Machine Learning*, incluindo tambem as aplicadas no TCC, são:

Redes Neurais Artificiais (RNA): Inspiradas no funcionamento do cérebro humano, compostas por camadas de neurônios artificiais, adequadas para problemas complexos com grandes volumes de dados, como reconhecimento de voz e imagens.

Máquinas de Vetores de Suporte (SVM)**:** Algoritmos que buscam encontrar a melhor fronteira (hiperplano) para separar diferentes categorias nos dados. São muito usados para classificação binária, como diagnosticar a presença ou ausência de uma condição médica, uma das técnicas utilizadas no TCC [21].

Árvores de Decisão ou *Random Forest*: Modelos baseados em perguntas sequenciais para tomada de decisão. Cada árvore é construída a partir de subconjuntos aleatórios de dados e atributos, o que promove diversidade e reduz a correlação entre os modelos individuais. Durante a predição, cada árvore vota em uma classe, e o resultado final é determinado pela maioria dos votos. Esse processo melhora o desempenho e reduz o risco de *overfitting*, tornando o método confiável para diferentes tipos de dados e cenários de classificação, uma das técnicas utilizadas no TCC [22].

Regressão Logística ou *Logistic Regression* (RegLog): Amplamente utilizado para classificação binária, com o objetivo de prever a probabilidade de uma amostra pertencer a uma de duas categorias possíveis, esse modelo aplica a função sigmoide para converter valores contínuos em probabilidades entre 0 e 1, facilitando a interpretação dos resultados, uma das técnicas utilizadas no TCC [23].

Análise de Componentes Principais (PCA): Técnica para reduzir a complexidade dos dados, transformando-os para que as maiores variações sejam destacadas, facilitando a visualização e o processamento sem perder a essência das informações.

G. Métricas de Avaliação

A Curva ROC (Receiver Operating Characteristic) [24], é uma ferramenta fundamental na avaliação de modelos de classificação, pois representa graficamente a relação entre a taxa de verdadeiros positivos (TPR) e a taxa de falsos positivos (FPR) enquanto o limiar de decisão do classificador é variado. Esse gráfico permite visualizar o equilíbrio entre sensibilidade e especificidade do modelo, o que ajuda a escolher o ponto de operação mais adequado quando os erros têm impactos diferentes.

Além disso, sendo a métrica principal utilizada no TCC, a área Sob a curva (AUC) quantifica numericamente a capacidade discriminativa do modelo. Valores próximos de 1,0 indicam excelente separação entre classes, enquanto valores próximos de 0,5 indicam desempenho aleatório, sem capacidade real de discriminação entre as classes (malicioso e be-

nigno), indicando que o modelo não está aprendendo padrões relevantes. No contexto hibrido, a AUC permite comparar de forma consistente o desempenho dos diferentes classificadores em cada etapa do sistema, auxiliando na escolha de quais modelos são mais adequados para compor a solução final.

H. Kaggle

Segundo o CTO da Nextcon, Çağlar Uslu [25], o Kaggle é uma plataforma usada para ciência de dados e aprendizado de máquina que, dentre outros recursos, tem uma gama de datasets e notebooks executáveis na nuvem com suporte a GPU. Criada em 2010 e depois adquirida pelo Google em 2017, hoje é um espaço muito utilizado por pesquisadores, acadÊmicos e profissionais, por facilitar a experimentação, o compartilhamento e a reprodução de resultados. A plataforma hospeda inúmeros conjuntos de dados relacionados à segurança cibernética, incluindo detecção de malware, análise do comportamento de ransomware e classificação de arquivos maliciosos. Como o modelo estático deste trabalho foi treinado usando datasets extraídos da plataforma, ela acabou servindo não só como fonte de dados, mas também como uma referência metodológica para o modelo estático do sistema híbrido, oferecendo exemplos consolidados de extração de características e avaliação de modelos.

I. Metodologia Ágil Scrum

Segundo Carvalho e Mello [26], o *Scrum* é um *framework* ágil utilizado para o desenvolvimento de produtos complexos, especialmente *software*, foi criado por Jeff Sutherland juntamente com Ken Schwaber em 1993, com o propósito de desenvolver *softwares* de uma maneira mais rápida, confiável e eficiente, proporcionando entregas rápidas e incrementais. O método se destaca pela sua capacidade de gerar flexibilidade, colaboração contínua e foco na entrega de valor.

O *Scrum* é estruturado em ciclos curtos, chamados de *Sprints*, que possuem duração fixa (geralmente de uma a quatro semanas). Cada *Sprint* resulta em um incremento funcional do produto, promovendo inspeção, adaptação e transparência ao longo do processo.

O *framework* é composto por três papéis principais, sendo eles o *Product Owner* onde é responsável por definir e priorizar os requisitos do produto, o *Scrum Master* que atua como facilitador do processo e o *Development Team* que é a equipe multidisciplinar responsável pela execução e entrega dos incrementos de *software*. Para finalizar o processo, ele possui uma estrutura onde a ideia seria *Sprint Planning* (Planejamento da *Sprint*) > *Daily Scrum* (Reuniões diárias) > *Sprint Review* (Revisão da *Sprint*) > *Sprint Retrospective* (Retrospectiva da *Sprint*).

III. Trabalhos Relacionados

A. Os ataques ransomware e a camada de proteção em sistemas governamentais

O trabalho de Levi dos Santos [3], teve como meta apresentar diversos casos de ataques *ransomware* aos sistemas eletrônicos de distintas empresas nos Estados Unidos e os meios preventivos existentes para mitigação dos ataques. A abordagem metodológica adotada é a pesquisa bibliográfica, focando na revisão de literatura para discutir o assunto e pela metodologia não foi usado nenhuma linguagem de programação. Como resultados das pesquisas, o artigo conclui que é essencial adotar boas práticas de segurança, investir em profissionais capacitados, manter *backups*, proteger redes e conscientizar contra pagamentos de resgate para mitigar ataques de *ransomware* em sistemas governamentais.

B. Machine Learning Algorithms and Frameworks in Ransomware Detection

O trabalho de Daryle Smith [27] teve como objetivo fornecer uma análise abrangente dos algoritmos e *frameworks* de aprendizado de máquina utilizados na detecção de *ransomware*. O estudo destaca a eficácia dessas técnicas na identificação de padrões maliciosos, contribuindo para o fortalecimento das defesas cibernéticas. Sem metodologia agil especificada o artigo discute *frameworks* como TensorFlow, PyTorch e Scikit-learn e por fim o estudo conclui que algoritmos de ML, como *Random Forest* e *Support Vector Machines*, são ótimos na detecção de *ransomware*. A integração desses algoritmos com *frameworks* robustos permite a identificação de ameaças, reduzindo falsos positivos e melhorando a segurança dos sistemas.

C. Análise Híbrida de Ransomware para Sistema Operacional Windows

O trabalho de Augusto Parisot [28] teve como objetivo avaliar técnicas de aprendizado de máquina para detecção de *ransomware* e classificação em suas respectivas famílias. Foram desenvolvidas ferramentas em *Python* para automatizar a coleta, mineração de dados e extração de informações dos relatórios de execução. O estudo aplicou seis algoritmos de aprendizado de maquina: *Decision Tree, Random Forest, K-Nearest Neighbors, Naive Bayes, Support Vector Machines e Multilayer Perceptron*. Os melhores resultados foram obtidos com os classificadores *Random Forest* e *Decision Tree*, especialmente quando aplicadas técnicas de mineração de texto (TF-IDF).

IV. Metodologia

Para a realização deste trabalho, primeiramente foi conduzida uma revisão bibliográfica, abordando ferramentas, conceitos e estudos relacionados, com o objetivo de aprofundar o entendimento sobre o tema proposto. Além disso, foi adotada a metodologia ágil Scrum, que contribui para um planejamento mais eficiente e uma execução organizada das etapas do projeto dividindo o processo do TCC 2 em *sprints* semanais. Nestes foram definidas tarefas e entregas. Ao final de cada *sprint*, foram realizadas análises dos avanços, identificação de possíveis melhorias e, por fim, planejamento das atividades subsequentes, de forma a

garantir um fluxo de trabalho constante e eficiente. Como apoio à organização e acompanhamento das atividades, foram utilizados recursos digitais, como o *WhatsApp* e o *Meet*, para controle de tarefas e anotações pessoais que auxiliaram na priorização e acompanhamento do desenvolvimento.

O desenvolvimento do sistema seguiu uma abordagem estruturada, que se iniciou com uma etapa de definição dos requisitos funcionais e não funcionais. A partir dos *sprints*, foram definidos os módulos e funcionalidades essenciais do projeto, os quais contemplaram desde o monitoramento do sistema até o desenvolvimento dos modelos de Inteligência Artificial para detecção precoce de *ransomware*. Para isso, foram utilizadas as seguintes tecnologias e ferramentas:

Linguagem *Python*, que foi escolhida pela sua ampla adoção na área de segurança da informação e inteligência artificial, além da vasta disponibilidade de bibliotecas especializadas, usada na IDE Pycharm.

Em complemento à linguagem *Python*, foram utilizadas bibliotecas para fins específicos, como exemplos a seguir, a Psutil, que foi empregada no monitoramento de processos e informações do sistema operacional, permitindo identificar padrões suspeitos associados ao comportamento de *ransomware*. Também Pandas e *Numpy*, que foram responsáveis pela manipulação, análise e tratamento dos dados, fundamentais para a preparação dos *datasets* utilizados no treinamento dos modelos. E *scikit-learn*, que foi utilizada para o treinamento e validação dos modelos de aprendizado de máquina, entre outras, por fim a *joblib*, que foi aplicada na serialização dos modelos de *machine learning* treinados, possibilitando seu reaproveitamento sem necessidade de reprocessamento.

O desenvolvimento foi conduzido em quatro etapas principais. A fase inicial compreendeu o levantamento e estudo dos padrões de comportamento de ransomware no ambiente Windows, visou identificar sinais precoces que antecedem a execução do ataque. Paralelamente, foi desenvolvido o módulo de monitoramento do sistema, com a função de coletar informações baseados em padrões comportamentais de ransomware, como elevação de CPU e Memória RAM, a criação de processos suspeitos, acessos massivos a arquivos e tentativas de desativação de mecanismos de segurança. No contexto de aprendizado de máquina, features são os atributos numéricos extraídos dos dados que servem como entrada para os modelos. Cada feature representa uma observação mensurável do sistema, transformando eventos do ambiente em variáveis que permitem ao algoritmo aprender padrões e realizar predições.

Após, o projeto avançou para a construção e o treinamento do modelo de *machine learning*, com um sistema hibrido de treinamento, usando aprendizado supervisionado mesclando coletas de forma contínua, que tambem foi usado como dataset, foi aplicado tambem treinos estaticos de um conjunto de dados de ransomware adquirido da plataforma web *Kaggle*. Estes modelos foram desenvolvidos com base em dados coletados para classificar comportamentos como malignos ou benignos. A etapa final consistiu na validação, análise de resultados e ajustes, com o objetivo de reduzir falsos positivos e aumentar a taxa de detecção, buscando sempre um equilíbrio entre segurança e usabilidade.

V. Solução desenvolvida e Resultados

Nesta etapa, é apresentado toda solução detalhada, desde o inicio até sua conclusão e resultados, destacando as principais etapas de desenvolvimento, implementação e resultados obtidos ao longo do projeto. O objetivo é oferecer uma visão ampla do funcionamento do sistema desenvolvido até seu funcionamento atual.

O trabalho apresenta uma solução onde é realizado o monitoramento contínuo de processos, arquivos e recursos do sistema operacional Windows, após o treino, é aplicado um modelo híbrido de aprendizado de máquina para prever ações que potencialmente são maliciosas e emitir alertas visuais e sonoros ao usuário ao longo do monitoramento.

Como mostra a Figura 2, o Diagrama de Caso de Uso representa as interações entre o operador do sistema e o software. O operador é responsável por iniciar a aplicação, ajustar limites de CPU, memória e rede, configurar a sensibilidade da Inteligência Artificial e acompanhar o monitoramento em operação. Já o sistema executa o monitoramento contínuo de arquivos e processos, detecta comportamentos suspeitos e emite alertas automáticos quando há indícios de atividade maliciosa. A estrutura demonstra como o sistema foi criado para oferecer uma interface simples ao usuário, enquanto mantém em segundo plano um ciclo automatizado de observação e resposta a ameaças.

O sistema foi implementado em Python, adotando uma arquitetura em modulos para facilitar a manutenção, a expansão e a reutilização de componentes. O diagrama de Componentes, ilustrado na Figura 3, apresenta a estrutura principal do sistema. O módulo `main.py` organiza a execução dos monitores de processos (`process_monitor.py`) e arquivos (`file_monitor.py`), que realizam a coleta contínua de dados. No contexto de aprendizado de máquina, features são os atributos numéricos extraídos dos dados que servem como entrada para o modelo. Cada feature representa uma característica mensurável do sistema, como uso de CPU, quantidade de arquivos modificados ou criação de processos suspeitos, permitindo que o algoritmo aprenda padrões e diferencie comportamentos normais de atividades maliciosas.

As informações obtidas são transformadas em características numéricas pelo módulo `feature_extractor.py`, tornando-as interpretáveis pelos algoritmos de aprendizado de máquina. Os módulos `train_behavior_model.py` e `train_static_model.py` realizam o treinamento supervisionado usando três modelos de aprendizado de máquina (Random Forest (RF), Regressão Logística (LogReg) e Support Vector Machine (SVM)), onde é escolhido o com melhor acurácia, e são combinados em `hybrid_predictor.py`, responsável por gerar previsões mais precisas e reduzir falsos positivos. O módulo `dashboard.py` exibe os resultados na

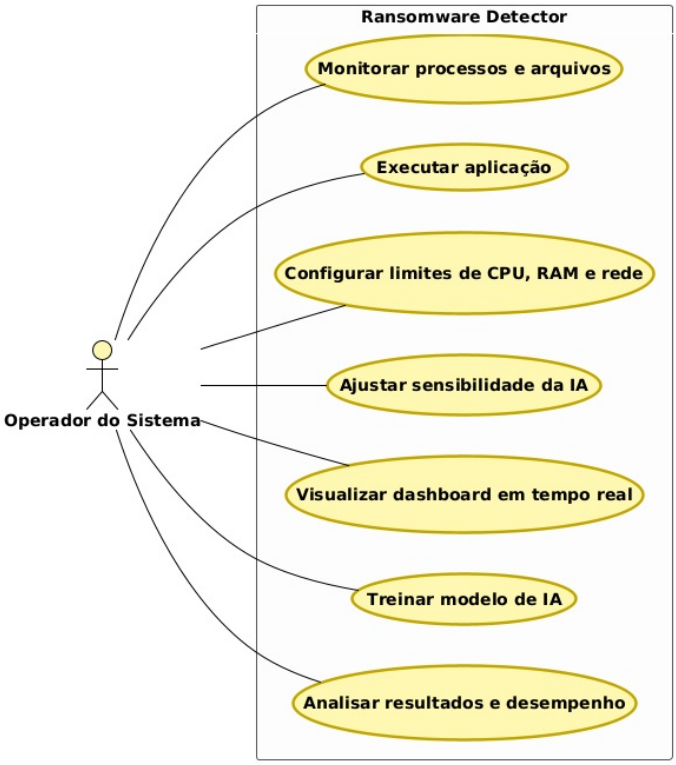


Figura 2. Diagrama de Caso de Uso [5]

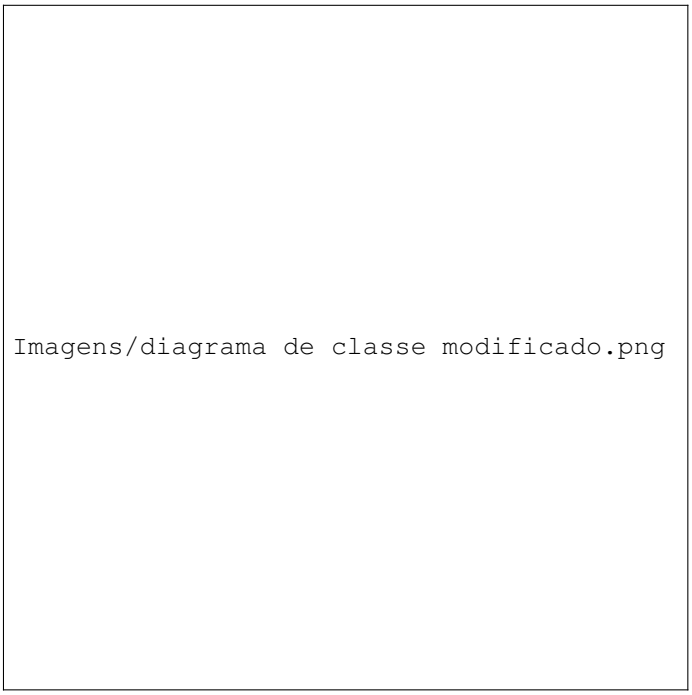


Figura 3. Diagrama de Componentes [5]

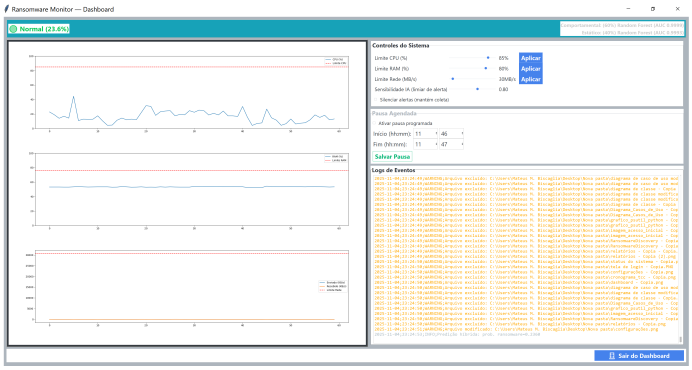


Figura 4. Dashboard [5]

medida em que são gerados, como mostra a Figura 4, enquanto `logger.py` lida com o registro de eventos, o `config.py` a configuração dinâmica de parâmetros e o `state.py` o controle do estado interno do sistema.

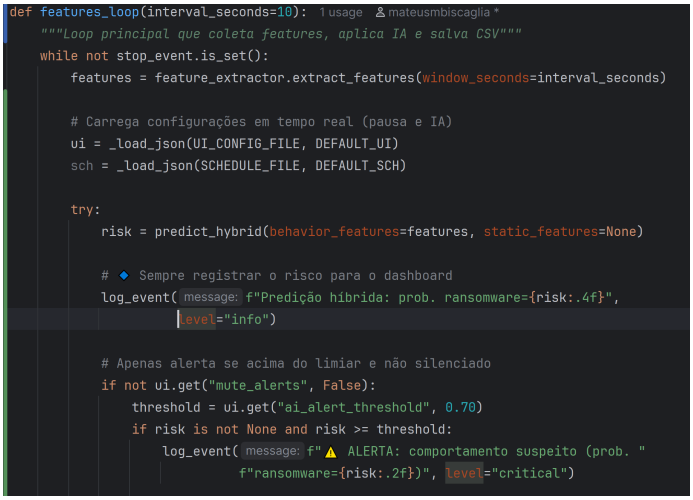
A. Construção e funcionamento do sistema

O funcionamento do sistema começa no arquivo main.py, responsável por iniciar e administrar todos os módulos. Ao ser executado, o sistema carrega os parâmetros definidos em config.json, limites de CPU, memória e rede, e instancia os módulos de monitoramento e previsão. A função principal `(main())` executa um laço contínuo que coleta métricas do ambiente Windows. O monitoramento é dividido em dois componentes principais: `process_monitor.py` que observa continuamente os processos em execução no sistema operacional, analisando nomes, localizações, pais de processo e padrões suspeitos e `file_monitor.py` que acompanha alterações em diretórios e arquivos, registrando eventos como exclusões de backups, modificações simultâneas em massa e criações suspeitas em locais críticos. Os dois monitores enviam logs para `logger.py`, que grava as ocorrências no arquivo events.log para possivel conferencia humana posterior.

A partir das informações coletadas, o Módulo `feature_extractor.py` transforma esses dados brutos em características numéricas, nesse caso o total de métricas usadas no modelo comportamental são dezeseis, como mostra a Figura 6, deixando `Datasets` prontos para o modelo de aprendizado de máquina, quando necessario, treinar a IA. Esse processo garante, um histórico comportamental do sistema, mas tambem uma boa base para treinamentos supervisionados futuros personalizados. Na Figura 5 uma captura de tela de um trecho da função `features_loop()` do main.py, mostrando como as features são salvas em CSV.

Ao mesmo tempo, o módulo `train_static_model.py` treina um modelo com base em um conjunto de dados externo com 18 métricas, como mostra a Figura 7, obtido da plataforma Kaggle. Esse dataset contém mais de 62 mil linhas de amostras de arquivos maliciosos e benignos, permitindo que o sistema aprenda padrões associados a ransomwares conhecidos após treino.

Com base no que se tinha até aqui, foi pensado, para não haver perda de informação, em um modulo híbrido que é responsável por combinar os dois resultados, comportamentais e estáticos, avaliando as probabilidades de risco de



```python
def features_loop(interval_seconds=10):  1 usage  mateusmbiscaglia *
    """Loop principal que coleta features, aplica IA e salva CSV"""
    while not stop_event.is_set():
        features = feature_extractor.extract_features(window_seconds=interval_seconds)

        # Carrega configurações em tempo real (pausa e IA)
        ui = _load_json(UI_CONFIG_FILE, DEFAULT_UI)
        sch = _load_json(SCHEDULE_FILE, DEFAULT_SCH)

        try:
            risk = predict_hybrid(behavior_features=features, static_features=None)

            # 🔹 Sempre registrar o risco para o dashboard
            log_event( message: f"Predição híbrida: prob. ransomware={risk:.4f}",
                      level="info")

            # Apenas alerta se acima do limiar e não silenciado
            if not ui.get("mute_alerts", False):
                threshold = ui.get("ai_alert_threshold", 0.70)
                if risk is not None and risk >= threshold:
                    log_event( message: f"⚠ ALERTA: comportamento suspeito (prob. "
                               f"ransomware={risk:.2f})", level="critical")
```

Figura 5. Captura da função `features_loop()` [5]

Nº	Nome da métrica	Descrição
1	window_seconds	Período de tempo usado para coletar as métricas.
2	cpu_snapshot_percent	Percentual atual de uso da CPU no momento da coleta.
3	mem_snapshot_percent	Percentual atual de uso da memória RAM.
4	num_created_files	Quantidade de arquivos criados durante a janela de coleta.
5	num_modified_files	Quantidade de arquivos modificados.
6	num_moved_files	Quantidade de arquivos movidos ou renomeados.
7	num_deleted_files	Quantidade de arquivos excluídos.
8	num_suspicious_ext_created	Número de arquivos criados com extensões suspeitas
9	num_rapid_file_changes	Velocidade de alteração de arquivos
10	num_deleted_backup_files	Quantidade de arquivos de backup deletados.
11	num_suspect_started	Processos suspeitos iniciados no intervalo.
12	num_parent_not_explorer	Processos iniciados por pais que não são o explorer.exe.
13	bytes_sent_delta	Total de bytes enviados desde a última coleta.
14	bytes_recv_delta	Total de bytes recebidos desde a última coleta.
15	num_external_connections	Conexões ativas com endereços IP externos.
16	num_suspicious_ports_attempts	Conexões feitas em portas conhecidas como críticas
17	num_stopped_services	Quantidade de serviços críticos do Windows parados.

Figura 6. Métricas comportamental para treino [5]

cada abordagem e gerando uma predição final mais robusta. Então o modelo comportamental aprende como cada sistema se comporta, e o modelo estático foca no que está sendo executado, identificando padrões encontrados em amostras maliciosas e benignas obtidas do conjunto de dados do Kaggle. O módulo `hybrid_predictor.py` atua como uma aliança, combinando os resultados das duas abordagens de forma ponderada, utilizando uma proporção de 60% de peso para o modelo comportamental e 40% para o estático. Essa divisão foi definida com base na premissa de que o comportamento enquanto o sistema está ativo, tende a ser mais indicativo na detecção precoce de ransomwares do que a análise estática de arquivos, por ser dados unicos e personalizados de cada rotina sistemica.

Em cada ciclo de monitoramento, o sistema coleta as métricas comportamentais e envia elas ao módulo `hybrid_predictor.py`. A escolha de qual modelo de aprendizado deve ser usado é feita com base nos cálculos da AUC durante o processo de treinamento. Os módulos `train_behavior_model.py` e `train_static_model.py` executam o mesmo processo

Nº	Coluna	Descrição
1	FileName	Nome do arquivo analisado
2	md5Hash	Hash MD5 do arquivo (identificador único)
3	Machine	Identificador de máquina/arquitetura da amostra
4	DebugSize	Tamanho da seção de depuração
5	DebugRVA	Endereço relativo virtual da seção de depuração
6	MajorImageVersion	Versão principal do executável
7	MajorOSVersion	Versão principal do SO alvo
8	ExportRVA	Endereço relativo virtual da tabela de exportação
9	ExportSize	Tamanho da tabela de exportação
10	IatVRA	Endereço relativo virtual da Import Address Table
11	MajorLinkerVersion	Versão principal do linker
12	MinorLinkerVersion	Versão secundária do linker
13	NumberOfSections	Número de seções do executável
14	SizeOfStackReserve	Tamanho reservado para a pilha
15	DllCharacteristics	Flags/características do PE (bitmask)
16	ResourceSize	Tamanho total dos recursos embutidos
17	BitcoinAddresses	Presença/contagem de endereços Bitcoin
18	Benign	Rótulo: 1 = benigno, 0 = ransomware

Figura 7. Métricas estáticas para treino [5]

de comparação entre os três algoritmos supervisionados, na curva, o que tender mais a 1.0 é o escolhido naquele treino para aquele *dataset*.

Cada *dataset* é treinado com 70% dos dados e testado com os 30% restantes. Essa escolha está alinhada às recomendações de várias literaturas, também vista no artigo de Santos (2025) [29], onde proporções como 70/30 são adequadas quando se dispõe de um número grande de amostras, permitindo que o algoritmo receba volume suficiente de dados para aprender padrões relevantes, ao mesmo tempo em que mantém uma parte representativa para avaliar o desempenho preditivo.

Durante o treinamento dos modelos, são avaliadas diversas métricas quantitativas que permitiram mensurar a capacidade preditiva de cada algoritmo. As usadas são:*precision* (precisão), que mostra a proporção de detecções corretas entre os casos classificados como positivos; o *recall* (sensibilidade), que mede a taxa de acertos entre os casos reais que são mesmo positivos; e o *f1-score*, ele representa o equilíbrio entre precisão e sensibilidade. É analisada tambem na matriz de confusão o número de verdadeiros e falsos positivos e negativos, tornando facil a comparação entre modelos.

Ao final do treinamento, o sistema escolhe automaticamente o modelo com o maior valor de AUC de cada (um para o estático e outro para o comportamental) , salvando-os em formato `.pkl`, mantendo os mesmos até novos datasets serem treinados. Essa abordagem automatizada confirma que o modelo final implantado vai ser sempre aquele com melhor desempenho estatístico, com base na ROC, entre os três algoritmos avaliados.

Por fim, foi pensado em um método de combinação ponderada entre os dois modelos, aplicando 60% de peso ao comportamental e 40% ao estático. Essa escolha veio a tona porque os dados comportamentais representam o funcionamento real do sistema e refletem padrões dinâmicos e personalizados,

enquanto o modelo estático atua como apoio complementar baseado em assinaturas conhecidas. Com isso a fusão equilibrada dos dois aumenta a robustez e a confiabilidade da predição final.

B. Resultados das Métricas

Aqui são apresentados os resultados obtidos no treinamento e validação dos dois modelos de detecção avaliados por métricas utilizadas em classificação. Esses resultados permitem comparar o desempenho dos algoritmos e selecionar a abordagem mais eficaz para compor o módulo híbrido final. Na figura 8 é mostrado o arquivo das métricas criadas após o treino dos dois modelos, comportamental e estático.

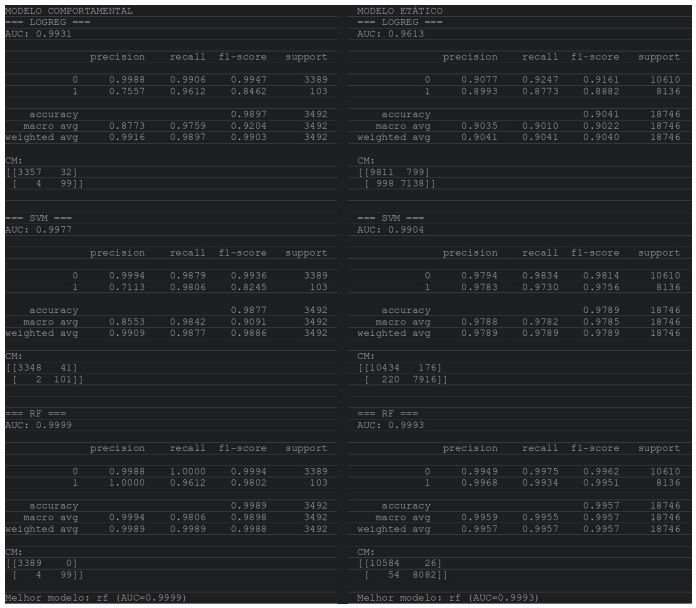


Figura 8. Avaliação dos três algoritmos [5]

Então o sistema identifica ransomware quando vários sinais típicos aparecem ao mesmo tempo, de forma semelhante ao que foi aprendido no treinamento. Em cenários normais são poucas ou nada de modificações/criação de arquivos, CPU e RAM relativamente estável e nenhum processo suspeito, resultando em baixa probabilidade com base nas features conhecidas. Já em um ataque real surgem, simultaneamente, muitas modificações de arquivos, extensões suspeitas, exclusão de backups, processos anormais e serviços críticos sendo interrompidos, elevando a probabilidade calculada pelo modelo. Com isso, a decisão não depende de um único atributo, mas da combinação desses comportamentos ocorrendo juntos.

VI. Considerações Finais

O sistema que foi desenvolvido alcançou o objetivo proposto, sendo construir uma solução de detecção precoce de ransomware capaz de analisar, aprender e reagir a ameaças antes que a infecção se consolide. A utilização conjunta de dois modelos independentes foi de grande importância no projeto, tornando o sistema híbrido em métodos de detecções onde uniu conhecimento prévio sobre padrões maliciosos com a observação contínua do ambiente real em execução. Esse design híbrido se mostrou muito satisfatório pois o diferencial é o modelo comportamental evoluir junto ao sistema monitorado, aprendendo seus hábitos, variações e ritmos naturais, enquanto o modelo estático mantém a capacidade de reconhecer características presentes em amostras já catalogadas.

Esses dois modelos integrados fazem com que o software não apenas detecte anomalias, mas também com o tempo compreenda o contexto em que elas surgem. Com isso, sabendo que o sistema comportamental alem de monitorar a cada dez segundos, salva as *features* monitoradas em um CSV já pronto para treino, tendo a crescente de a cada 24 horas 8.640 linhas, a cada mês 259.200 linhas e a cada trimestre teremos 777.600 linhas prontas no CSV, mesmo sendo manual o treino, podendo ser adaptado em correções futuras, o sistema se fortalece progressivamente conforme novas métricas comportamentais são coletadas, tornando a predição mais refinada, reduzindo falsos positivos, aumentando a confiabilidade da detecção e tornando o sistema totalmente personalizado. O resultado é uma ferramenta que não permanece estagnada, mas sim se adapta, aprende e se torna mais precisa à medida que o ambiente operacional evolui.

Dessa forma, este trabalho demonstra que a união entre monitoramento comportamental, análise estática e modelos de aprendizado supervisionado permitiram construir um mecanismo robusto e capaz de reagir preventivamente a ameaças emergentes. A arquitetura proposta evidencia que é possível ir além da simples identificação de arquivos maliciosos, alcançando um nível de proteção mais inteligente, contextual e alinhado às necessidades reais de cada sistema monitorado. Com isso, o sistema apresentado se coloca como uma solução promissora, eficiente e essencial para ambientes que exigem detecção antecipada de ransomwares, contribuindo, de forma prática e concreta, para o fortalecimento da segurança digital.

Referências

[1] Cointelegraph. “Ransomware: "sequestro virtual"gera lucro recorde de US1$bilhoemcriptoparahackers$”. Em: (2024). Disponível em <https://exame.com/future-of-money/ransomware-sequestro-virtual-gera-lucro-recorde-de-us-1-bilhao-em-cripto-para-hackers>.

[2] Vitor Valeri. “Windows em números: quantas pessoas utilizam o sistema da Microsoft?” Em: (2023). Disponível em <https://www.oficinadanet.com.br/windows/50969-windows-numeros>.

[3] Levi Alves dos Santos. “Os ataques ransomware e a camada de proteção em sistemas governamentais”. Em: *Revista Científica Multidisciplinar, Núcleo do Conhecimento* (2022). Disponível em: https : / / www . nucleodoconhecimento . com . br / tecnologia / ataques - ransomware. Acesso em: 05 jun. 2025. ISSN: 2448-0959.

[4] SH_Kok. "Early detection of crypto-ransomware using pre-encryption detection algorithm". Em: (2022). Disponível em <https://doi.org/10.1016/j.jksuci.2020.06.012>.

[5] Dos Autores. *Desenvolvido pelos autores no decorrer do trabalho*. Em: (2025).

[6] CISA (Cybersecurity e Infrastructure Security Agency). *Guia #StopRansomware*. Acessado em: 01 de junho de 2025. 2023. URL: https://www.cisa.gov/stopransomware/ransomware-guide.

[7] Nikolaos Stavrakis ik Itasoy Valentina Rosenberg. "Ransomware Detection on Windows Using File System Activity Monitoring and a Hybrid Isolation Forest-XGBoost Model". Em: (2024). Disponível em <https://doi.org/10.21203/rs.3.rs-5257558/v1>.

[8] Meriam Senouci. *What Are the Cyber Kill Chain Stages? Stage One: Reconnaissance*. Acessado em: 11 de junho de 2025. 2025. URL: https://www.fortra.com/blog/what-are-cyber-kill-chain-stages-stage-one-reconnaissance.

[9] Palo Alto Networks. *What is a Command and Control Attack?* Acesso em: 05 jun. 2025. n.d. URL: https://www.paloaltonetworks.com/cyberpedia/command-and-control-explained.

[10] "Amazon Web Services - Site oficial da empresa". Em: (2025). Acessado em: 20 de maio de 2025.

[11] Curso de Programação. *O que é biblioteca em programação?* Acesso em: 05 jun. 2025. 2024. URL: https://programae.org.br/cursoprogramacao/glossario/o-que-e-biblioteca-em-programacao.

[12] Giampaolo Rodolà. "psutil - Cross-platform lib for process and system monitoring in Python". Em: (2024). Disponível em <https://github.com/giampaolo/psutil>.

[13] Thiago Coutinho. "Descubra o que é e o porquê de tantos programadores utilizarem a Biblioteca Pandas". Em: (2021). Disponível em <https://voitto.com.br/blog/artigo/biblioteca-pandas>.

[14] Desenvolvedores NumPy. "What is NumPy?" Em: (2024). Disponível em <https://numpy.org/devdocs/user/whatisnumpy>.

[15] Gaio. "Um guia para iniciantes no Scikit-learn". Em: (2025). Disponível em <https://www.hashdork.com/pt/scikit-learn>.

[16] Awari. "Aprenda Python com o Joblib: a Biblioteca Essencial para a sua Carreira Em Tecnologia". Em: (2023). Disponível em <https://awari.com.br/aprenda-python-com-o-joblib-a-biblioteca-essencial-para-a-sua-carreira-em-tecnologia/>.

[17] Matplotlib Development Team. *Matplotlib: Visualization with Python*. Acessado em: 1 de novembro de 2025. 2024. URL: https://matplotlib.org.

[18] Jaime Simão Sichman. "Inteligência artificial e sociedade: avanços e riscos". Em: (2021). Disponível em <https://doi.org/10.1590/s0103-4014.2021.35101.004>.

[19] Gabriela Miana de Mattos Paixão et al. "Machine Learning na Medicina: Revisão e Aplicabilidade". Em: (2022). Disponível em <https://doi.org/10.36660/abc.20200596>.

[20] Pedro Lucas. "Machine Learning: A inteligência das máquinas em detalhes". Em: (2024). Disponível em <https://ligaia.com.br/conceitosfundamentais/machine-learning-inteligencias/>.

[21] Scikit-learn Developers. *1.4. Support Vector Machines*. Acessado em: 1 de novembro de 2025. 2024. URL: https://scikit-learn.org/stable/modules/svm.html.

[22] Scikit-learn Developers. *RandomForestClassifier*. Acessado em: 1 de novembro de 2025. 2024. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

[23] Scikit-learn Developers. *LogisticRegression*. Acessado em: 1 de novembro de 2025. 2024. URL: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html.

[24] Pedro César Tebaldi Gomes. *Curva ROC: Como Avaliar Modelos de Classificação*. Acessado em: 20 de novembro de 2025. 2024. URL: www.datageeks.com.br/curva-roc.

[25] Çağlar Uslu. *O que é o Kaggle?* Acessado em: 18 de novembro de 2025. 2024. URL: www.datacamp.com/pt/blog/what-is-kaggle.

[26] B. V. de Carvalho e C. H. P. Mello. "Aplicação do método ágil Scrum no desenvolvimento de produtos de software em uma pequena empresa de base tecnológica". Em: *Gestão & Produção* 19.3 (2012). Disponível em: https://www.scielo.br/j/gp/a/34xH953TFwLPYDB9BYdJghL. Acesso em: 05 jun. 2025, pp. 557–573.

[27] Daryle Smith, Sajad Khorsandroo e Kaushik Roy. "Machine Learning Algorithms and Frameworks in Ransomware Detection". Em: *IEEE Access* (2022). DOI: 10.1109/ACCESS.2022.3218779.

[28] Augusto Parisot de Gusmão Neto. "Análise Híbrida de Ransomware para Sistema Operacional Windows". Acesso em: 05 jun. 2025. Trabalho acadêmico. Centro de Instrução Almirante Wandenkolk (CIAW) — Marinha do Brasil, 2023. URL: https://www.repositorio.mar.mil.br/handle/ripcmb/846311.

[29] Hellen Geremias dos Santos et al. *Machine learning para análises preditivas em saúde: exemplo de aplicação para predizer óbito em idosos de São Paulo, Brasil*. Acessado em: 20 de novembro de 2025. 2020. URL: https://cadernos.ensp.fiocruz.br/csp/pages/iframe_print.php?aid=792.