UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Rafael Valentim Silva

Análise Comportamental de Grupos de *Ransomwares* em Ambiente Virtual usando Árvores de Decisão

Uberlândia, Brasil

2021

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Rafael Valentim Silva

Análise Comportamental de Grupos de *Ransomwares* em Ambiente Virtual usando Árvores de Decisão

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Ciência da Computação.

Orientador: Renan Gonçalves Cattelan

Universidade Federal de Uberlândia – UFU

Faculdade de Ciência da Computação

Bacharelado em Ciência da Computação

Uberlândia, Brasil

2021

Rafael Valentim Silva

Análise Comportamental de Grupos de *Ransomwares* em Ambiente Virtual usando Árvores de Decisão

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Ciência da Computação.

Trabalho aprovado. Uberlândia, Brasil, 01 de novembro de 2016:

Renan Gonçalves Cattelan
Orientador

Professor

Professor

Uberlândia, Brasil
2021

Resumo

Na atualidade, com bilhões de dispositivos conectados à Internet, mecanismos capazes de garantir segurança cibernética precisam ser constantemente criados e aperfeiçoados. Dentre as ameaças existentes, os *ransomwares* têm apresentado crescimento vertiginoso e gerado prejuízos para governos, empresas e indivíduos. O objetivo deste trabalho é criar um conjunto de dados rotulado com classes de ransomwares a partir de padrões descobertos na análise dinâmica desse tipo de *malware* em um ambiente controlado, a fim de permitir que técnicas de aprendizado supervisionado possam ser aplicadas. Bem como disponibilizar uma base de dados para incentivar o desenvolvimento de novas pesquisas com foco na identificação de atividade maliciosa através de modelos de predição.

Palavras-chave: segurança da informação, malware, aprendizado de máquina, ransomware

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

RaaS	Ransomware as a Service
C&C	Command-and-Control
SaaS	Software as a Service
HTTP	Hypertext Transer Protocol
ICMP	Internet Control Message Protocol
IP	Internet Protocol
SMTP	Simple Email Transfer Protocol
MFT	Master File Table
NTFS	New Technology File System
MAC	Media Access Control
SVG	Scalable Vector Graphic
HTA	HTML Application
DLL	Dynamic Link Library
API	Application Programming Interface

Sumário

1 Introdução

Com mais de 18 bilhões de dispositivos conectados a Internet(CISCO SYSTEMS, INC, 2020), o acesso a informação vem tomando uma escala jamais vista. Como consequência desse aumento, cada vez mais empresas buscam garantir a segurança e confidencialidade dos dados de seus usuários, visando garantir sua credibilidade.

De acordo com o relatório de segurança da *AV-TEST* de 2019/2020 (AV-TEST, ORG, 2020), o número de ameaças na Internet aumentou mais de 50% se comparado a 2018, atingindo a marca de 677 milhões de ameaças reportadas a plataforma. Essas ameaças são conhecidas como *malware*, que pode ser descrito como "qualquer código adicionado, alterado ou removido de uma aplicação, que busca causar danos ou subverter uma ou mais funções de um sistema" (IDIKA; MARTHUR, 2007). No mesmo relatório, também é evidenciado que a maior parte dos ataques são direcionados a dispositivos com sistema operacional *Windows*, da empresa *Microsoft*.

Ransomware é um tipo especial de *malware* especializado em restringir o uso do dispositivo infectado, através da criptografia parcial ou completa de seu disco. Após a criptografia, o atacante exige o pagamento de uma taxa, geralmente usando criptomoeda, para que o sistema seja descriptografado. Sua instalação no sistema se dá principalmente por anúncios maliciosos, *spams* e arquivos baixados de fontes não confiáveis (ALSHAIKH; RAMADAN; HEFNY, 2020).

Com o aumento da popularidade das criptomoedas, a criação e a propagação de *ransomwares* vêm ganhando popularidade, atingindo, em 2019, 900 mil variações únicas desse tipo de *malware* e 4,5 milhões de exemplares (AV-TEST, ORG, 2020). O fácil acesso a ferramentas de desenvolvimento e interfaces que até mesmo abstraem a criação de *ransomwares* - conhecidas como *Ransomware as a Service* (RaaS) - (ALSHAIKH; RAMADAN; HEFNY, 2020), fazem com que a disseminação e a variação dessa família de *malware* se tornem ainda mais fáceis. O prejuízo com esse tipo de ameaça tende a atingir 20 bilhões de dólares até o fim de 2021 (ALSHAIKH; RAMADAN; HEFNY, 2020).

Técnicas de combate às ameaças cibernéticas também vêm sendo desenvolvidas, entre elas antivírus e monitoração de *endpoints*, assim como empresas têm voltado sua atenção para a segurança de suas soluções, fazendo com que os *softwares* se tornem mais difíceis de serem corrompidos. Soluções como Avast e McAfee são exemplos de antivírus e ofuscação de código é uma técnica usada para dificultar a injeção de código malicioso dentro de aplicações.

Contudo, como anteriormente citado, a variedade de *ransomwares* vem crescendo rapidamente e sua capacidade de imitar comportamentos semelhantes aos de *softwares*

normais torna sua identificação um processo complexo. Por isso, é necessário o desenvolvimento e refinamento de mais ferramentas capazes de se adaptar a "mutação" constante dessas ameaças.

Existem duas principais formas de análise de um *malware*: estática e dinâmica. A primeira diz respeito a uma análise a nível do código executável de uma aplicação afim de encontrar "assinaturas" – termo usado para definir um padrão específico de bytes presente no arquivo executável de um *ransomware* – e pode ser realizada sem o software estar em execução (ALSHAIKH; RAMADAN; HEFNY, 2020). A segunda, também conhecida como análise comportamental, se baseia na execução de uma aplicação em um ambiente controlado e monitorado, afim de determinar padrões de comportamento da aplicação com relação a sua interação com a rede, chamadas de sistema e comportamento específico em relação a arquivos e controles de restauração - como backups - do sistema.

A análise dinâmica pode ser bem imprecisa nos dias de hoje, uma vez que cada vez mais *malwares* têm implementado mecanismos de detecção de ambientes virtuais e agentes conhecidos de monitoramento de atividade suspeita, como chamadas para validar o endereço IP e a presença de um *debugger* ativo no sistema (IJAZ; DURAD; ISMAIL, 2019). Portanto, uma abordagem híbrida, vinculando assinaturas estáticas e dinâmicas se mostra um possível caminho a se seguir.

O objetivo deste trabalho é classificar famílias de *ransomware* utilizando atributos relacionados a manipulação de arquivos coletados a partir de uma ferramenta de análise dinâmica, a fim de evidenciar padrões comportamentais entre *ransomwares* e suas diversas variações. Bem como disponibilizar uma base de dados montado a partir da análise dinâmica de *malwares* para incentivar o desenvolvimento de novas pesquisas com foco na identificação de atividade maliciosa através de modelos de predição.

Para a criação do conjunto de dados, foi usada a ferramenta de código aberto especializada em análise dinâmica de *malwares* chamada *Cuckoo Sandbox*. Uma série de famílias de *ransomwares* já classificados foram submetidas à plataforma em um ambiente virtual com Windows 7 e tiveram seu comportamento observado. Com base na análise emitida pela plataforma, foram aplicados filtros nos dados para determinar suas características e padrões que permitissem a diferenciação entre as famílias de *ransomware*.

Este trabalho é organizado em 5 capítulos. No Capítulo 2 são explicados conceitos necessários para a compreensão do trabalho. No Capítulo 3 é demonstrado como os experimentos foram desenvolvidos. No Capítulo 4 são apresentados os resultados dos experimentos desenvolvidos. Por fim, no Capítulo 5 são apresentadas as conclusões deste trabalhos e sugere trabalhos futuros.

2 Revisão de Literatura

Esse capítulo aborda conceitos fundamentais para a compreensão dos experimentos e resultados obtidos ao longo desse trabalho. Alguns trabalhos que possuem linha de pesquisa na análise dinâmica de *ransomwares* também são citados ao final desse.

2.1 Fundamentação Teórica

2.1.1 *Malware*

Malware vem da definição (**mal**)icious soft(**ware**), cuja tradução direta significa "aplicação maliciosa". Apresentado como qualquer código malicioso que busca causar alguma forma de dano no ambiente no qual se instala. Existem diversas famílias de *malware* como *Trojans*, *Worms*, *Virus*, *Backdoor* e *Ransomwares*.

2.1.2 Ransomware

Ransomare tem origem etimológica nas palavras *"ransom"* - resgate em português- e *"malware"*, que pode ser interpretado com uma aplicação que priva o usuário de utilizar o sistema e pede uma quantia em dinheiro, geralmente em cripto-moeda, para que o sistema seja "resgatado", considerado uma forma de extorsão online através de um *software* mal intencionado. A forma de propagação desse tipo de ameaça é feita principalmente através de arquivos de origem desconhecida que se passam por aplicações reais, aplicações reais com código malicioso injetado - conhecido como *drive-by*, anexos de email e exploração de falhas de rede de sistemas (ALSHAIKH; RAMADAN; HEFNY, 2020).

A forma de atuação desse tipo de *malware* é bem precisa e determinística:

1. O *ransomware* se espalha na rede através de *spams* injetados em aplicações reais.
2. Os dois passos seguintes são executados de acordo com a política de instalação do *ransomware*: se o mesmo usa arquivos como vetor de ataque, eles são executados certamente; enquanto ameaças que já vinculam todo seu código malicioso fazem uso parcial de sua conexão com o servidor remoto, em sua maioria, compartilhando apenas informações de execução e chaves de criptografia.
3. Uma vez instalado no sistema, é estabelecida uma conexão com um servidor remoto, geralmente através da rede Tor, conhecido como servidor de Comando & Controle (C&C) de onde o *malware* baixa seu código de execução. (LISKA; GALLO, 2016).

4. Com o a comunicação com o C&C estabelecida, o *ransomware* então recebe parâmetros de criptografia, como quais extensões de arquivos devem ser criptografadas, tipo de criptografia e trocas da chaves público-privadas, e deve se manter incubado até que receba comando para iniciar o ataque definitivo (LISKA; GALLO, 2016).

5. Antes do processo de criptografia se iniciar de fato, a maioria dos *ransomwares* deletam arquivos de backup automático gerados pelo Windows, também conhecidos como *shadows* do sistema.

6. Uma vez que o comando de ataque é emitido, os dados do usuário começam a ser restringidos. A criptografia dos arquivos do sistema é feita através de métodos assimétricos, então o *malware* tem posse da chave pública e o C&C tem posse da chave privada (LISKA; GALLO, 2016). Portanto, uma vez criptografado, um arquivo só pode ser descriptografado com posse de uma chave que só o atacante que disseminou o *ransomware* tem acesso. Existem também *ransomwares* que usam a geração de chaves de criptografia locais, usando bibliotecas como OpenSSL e API do sistema (CRACIUN; MOGAGE; SIMION, 2019).

7. Por fim, o *malware* busca uma forma de mostrar informações de contato e valor para o resgaste na tela do dispositivo, geralmente por meio da troca de plano de fundo e uso de Widgets com contadores de tempo até que o pedido de resgate expire e o sistema seja corrompido definitivamente, ou o tempo restante até que o preço do resgaste aumente (LISKA; GALLO, 2016). Na Figura 1 é mostrada a tela de recuperação do Cerber, um *ransomware* de 2016 que hoje é disponibilizado como serviço online.

2.1.3 Famílias de *Ransomwares*

Nos dias de hoje, devido à popularização dos *ransomwares* e modernização dos processos pela Internet, existem ferramentas prontas e até mesmo serviços online específicos para o desenvolvimento desse tipo de ameaça, conhecidos como *Ransomware as a Service* (RaaS) (LISKA; GALLO, 2016), inspirado no termo SaaS (Software as a Service) amplantamente utilizado para descrever uso de soluções completas disponibilizadas online. Então, uma grande variedade de *ransomwares* são desenvolvidos a cada dia e pode-se dividi-los em famílias com base no código de referência que usam. Entre essas famílias, as que foram analisadas nesse trabalho foram: WannaCry, GandCrab, Cerber, TeslaCrypt, CryLocker, Sage e Locky.

2.1.4 Técnicas aplicadas por *Ransomwares*

Ransomwares são segregados por famílias, cada uma delas portam técnicas e formas de ataque diferentes e complementares. Tais técnicas dizem respeito a forma de

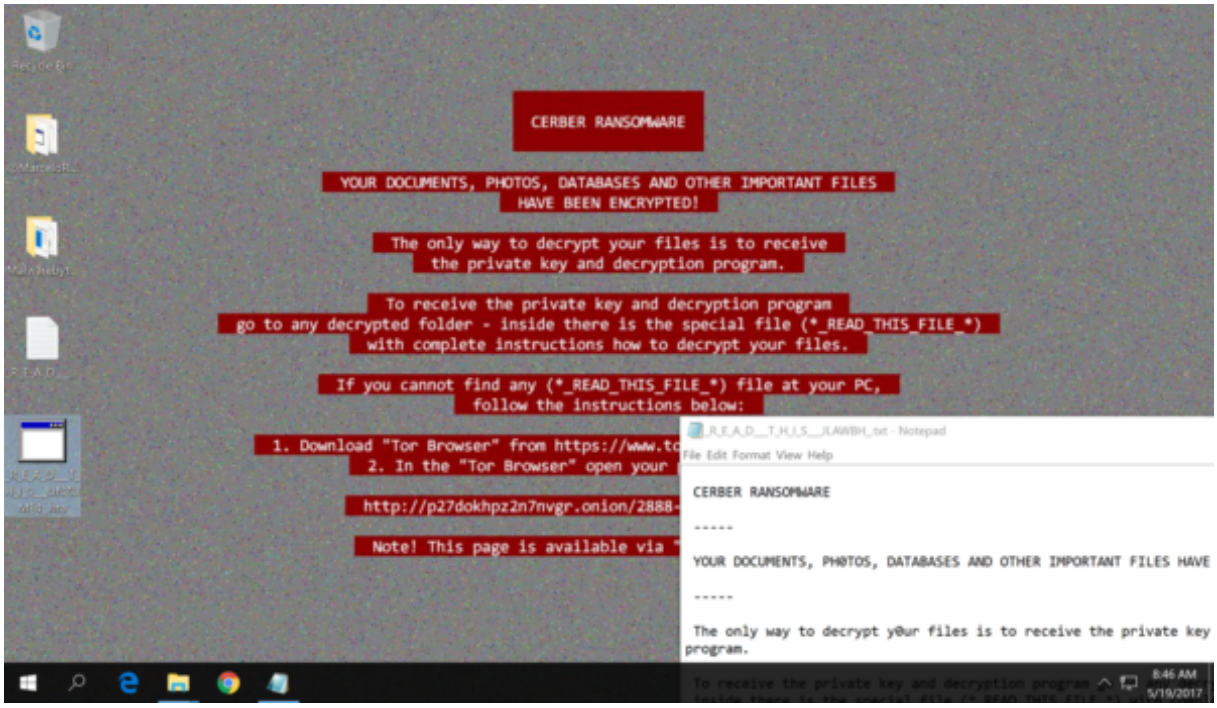

Figura 1 – Tela de resgaste do *ransomware* Cerber. Fonte: do Autor.

instalação, tipo de acesso aos dados do usuário, processo de criptografia e técnicas de evasão (DARGAHI et al., 2019).

2.1.4.1 *Ransomware* baseado em script

Método de instalação de *ransomware* baseado na implantação de um script executável, no qual o código do *malware* já está presente. Esse tipo de ameaça não requere instalação e geralmente é escrita em linguagens de programação interpretadas, como JavaScript, PHP, PowerShell e Python (DARGAHI et al., 2019). Famílias como TeslaCrypt, Locky, Cerber e Sage usam essa abordagem.

2.1.4.2 *Drive-by Ransomwares*

Estratégia referente ao tipo de transporte do código malicioso, o termo *drive-by* é usado para identificar *malwares* que usam outros arquivos, aplicações reais ou documentos para injetar seu código. Essa forma de implantação é mais difícil de ser detectada por antivírus, pois faz uso de aplicações legítimas para se instalar e baixar seu código de execução (DARGAHI et al., 2019). Amostras da família de *ransomware CryptoWall* usam SVGs – um tipo de vetor de imagem – como seu vetor de ataque, assim como algumas amostras do *Cerber* usam SFXs contendo DLLs executávies e amostras do *Locky* usam arquivos do Microsoft World, PDFs e HTAs (DARGAHI et al., 2019).

2.1.4.3 Formas de acesso a arquivos

Ransomwares, em sua generalidade, lidam com criptografia de arquivos para restringir o acesso ao dispositivo infectado, contudo a forma como é feito esse acesso varia, tornando sua identificação por anti-*malwares* um processo complicado. *Ransomwares* da família *Cryptolocker* e *CryptoWall* substituem o arquivo por seu similar criptografado no mesmo *buffer* de leitura - usando acesso de escrita e leitura na mesma chamada de sistema -, outras famílias como *Reverton* e *Filecoder* apagam arquivos através da *MFT* (DARGAHI et al., 2019) do sistema de arquivo NTFS usado pelo Windows. Amostras de *Locky* mudam a extensão do arquivo encriptografado para *.locky* e deleta o original, assim como algumas amostras do *CryptoWall* que mudam a extensão para *.crypt* e amostras do *TeslaCrypt* que usam as extensões *.ecc*, *.ezz* e *.exx* (DARGAHI et al., 2019).

2.1.4.4 Criptografia dos dados

Assim como a forma de acesso e manipulação de arquivos varia, a forma como a criptografia dos mesmos também possui variação. A criptografia dos dados pode ser feita utilizando APIs fornecidas pelo sistema, como o Windows CryptoAPI ou DLLs de criptografia (KHARAZ et al., 2016) e ser feita de maneira distribuída. Amostras do *ransomware Cerber*, por exemplo, usa a Windows CryptoAPI para criptografar os dados e amostras do *Petya* usam a CryptGenRandom API - incluida na CryptoAPI - para gerar a chave de criptografia dos dados (DARGAHI et al., 2019). Contudo, o acesso a APIs de sistema pode ser monitorado por *anti-malwares*, então *ransomwares* usam diferentes abordagens para lidar com tal vulnerabilidade, como o *CryptXXX* que cria múltiplos processos para modificar os arquivos e suspende a criptografia caso detecte que existe outros processos tentando detectar uso de criptografia em massa (LISKA; GALLO, 2016).

2.1.4.5 Técnicas de Evasão

No artigo de Dargahi et al. (2019) são citadas diversas técnicas de evasão ultilizadas por diversas famílias de *ransomwares*. Dentre elas, as mais significantes para o entendimento deste trabalho são:

- Execução Atrasada: Técnica de evasão onde a ameaça suspende sua execução em intervalos de tempo pré-estabelecidos ou devido a alguma reação do sistema, como a inspeção de uso em massa de APIs de criptografia. Boa parte das famílias de *ransomwares* não fazem uso dessa técnica, contudo amostras da família *CryptoXXX* e *Cerber* fazem uso da mesma.

- "Anti-Debbuging": Técnica que consiste na inspeção do sistema afim de identificar componentes de monitoramento ativos. *Debuggers* são ferramentas usadas para analisar a execução de outros programas em um sistema e, em sua maioria, fazem uso

de *flags* de sistema e processos auxiliares, como `procexp.exe`, `regedit` e `msconfig`. Então algumas famílias fazem uso desse comportamento conhecido dos *debuggers* para determinar sua execução. Famílias como *TeslaCrypt*, *Jigsaw*, *Locky* e *CryptoWall* fazem uso dessa técnica suspendo sua atividade caso algum comportamento dessa espécie seja identificado.

- "Anti-sandboxing": Similar a técnica de *anti-debugging*, contudo busca identificar se o *ransomware* está sendo executado em uma *sandbox*. Seu vetor de ataque é a análise dinâmica de ameaças, uma vez que, ao detectar que está sendo executado nesse tipo de ambiente, o *malware* cessa sua execução ou permanece em estado inoperante. A forma de implementação dessa técnica faz uso da checagem do endereço MAC da placa de rede da máquina, validação da quantidade de processos em execução e comparando o ciclos de CPU necessários para executar o programa - pois em ambientes virtuais o tempo de execução é significantemente maior do que em um real. Famílias de *ransomware* como *Cerber*, *WannaCry* e *CryptXXX* fazem uso dessa técnica.

2.1.5 Tipos de Análises de *Malwares*

Um *malware* pode ser analisado estaticamente, sem execução, ou dinamicamente, executado em um ambiente monitorado. Essa sessão define os dois tipos com ênfase na análise dinâmica.

2.1.5.1 Análise Estática

A análise estática de *malware* consiste na identificação de padrões de bytes no código fonte de aplicações. Esses padrões são conhecidos como assinaturas e são retiradas de *malwares* conhecidos e, posteriormente, podem ser usadas para identificação de código malicioso em aplicações. Para a extração dessas assinaturas é necessário ter acesso ao código fonte de *ransomwares* através de engenharia inversa,contudo existem mecânismos de ofuscação de código e variações de código que alteraram a sequência de bytes no código fonte e alterar seu "padrão" (O'Kane; Sezer; McLaughlin, 2011). Assim como os *drive-in ransomware* não carrega seu código malicioso no vetor que usa para infectar a máquina, limitando a cobertura de uma análise a nível de código. Ainda assim, a análise estática apresenta bons resultados ao ser aplicada em *ransomwares* propriamente ditos e em ambientes virtuais, favorecendo uma identificação explícita (IJAZ; DURAD; ISMAIL, 2019). O fluxo desse tipo de análise pode ser verificado na Figura 2 com base em Zimba, Wang e Chen (2018).

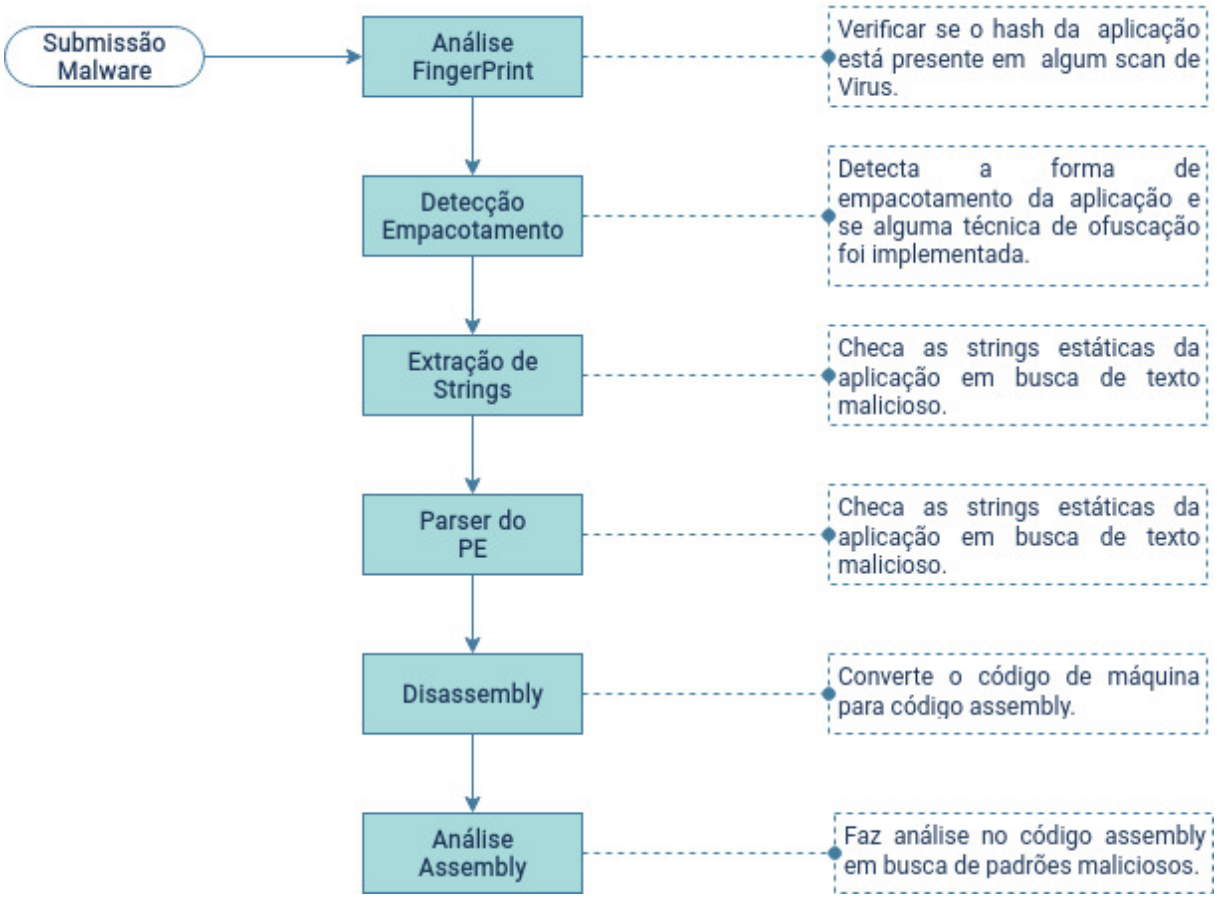

Figura 2 – Fluxo de análise estática. Fonte: adaptada de Zimba, Wang e Chen (2018).

2.1.5.2 Análise Dinâmica

A análise dinâmica consiste na execução de um *malware* e um ambiente virtual e monitorado - uma "sandbox". Agentes de monitoração e "debuggers" - ferramenta usado para inspeção de código e estado da memória de processos - são previamente configurados e geram relatórios com base nos dados observados durante a execução de uma aplicação.

Uma *sandbox* é composta por 2 componentes principais: *guest* e *host*. O *guest* é o ambiente virtual onde a aplicação é executada, virtualizada através um software como KVM (*Kernel Virtual Machine*), Oracle VirtualBox ou VMWare, dotada de agentes de monitoração e instrumentação. Os exportação dos dados é feita por um agente externo instalado na mesma. O *host* é responsável por coordenar e receber os dados emitidos pelo *guest* e aproximar a experiência do ambiente virtual a um cenário real. *Hosts* podem fazem papel de identificador de ameaças também e fornecem relatórios precisos sobre as assinaturas obtidas da execução de um *malware*, como é o caso da *Cuckoo Sandbox* exemplificado na Figura 3.

Em *sandboxes ransomwares* iniciam sua execução e até mesmo realizam todo o processo de criptografia dos arquivos, sendo possível determinar comportamentos em re-

lação a rede, sequência de comandos executados, acesso a arquivos e processos carregados. Muitas dessas ameaças vêm com validações que buscam determinar se as mesmas estão sendo executadas em ambiente monitorado, contudo uma análise estática em ambiente dinâmico pode ser feita, uma vez que o mesmo já começou sua execução (IJAZ; DURAD; ISMAIL, 2019). O próprio comportamento "furtivo" pode ser considerado como um comportamento malicioso e determinar se uma aplicação é confiável de fato.

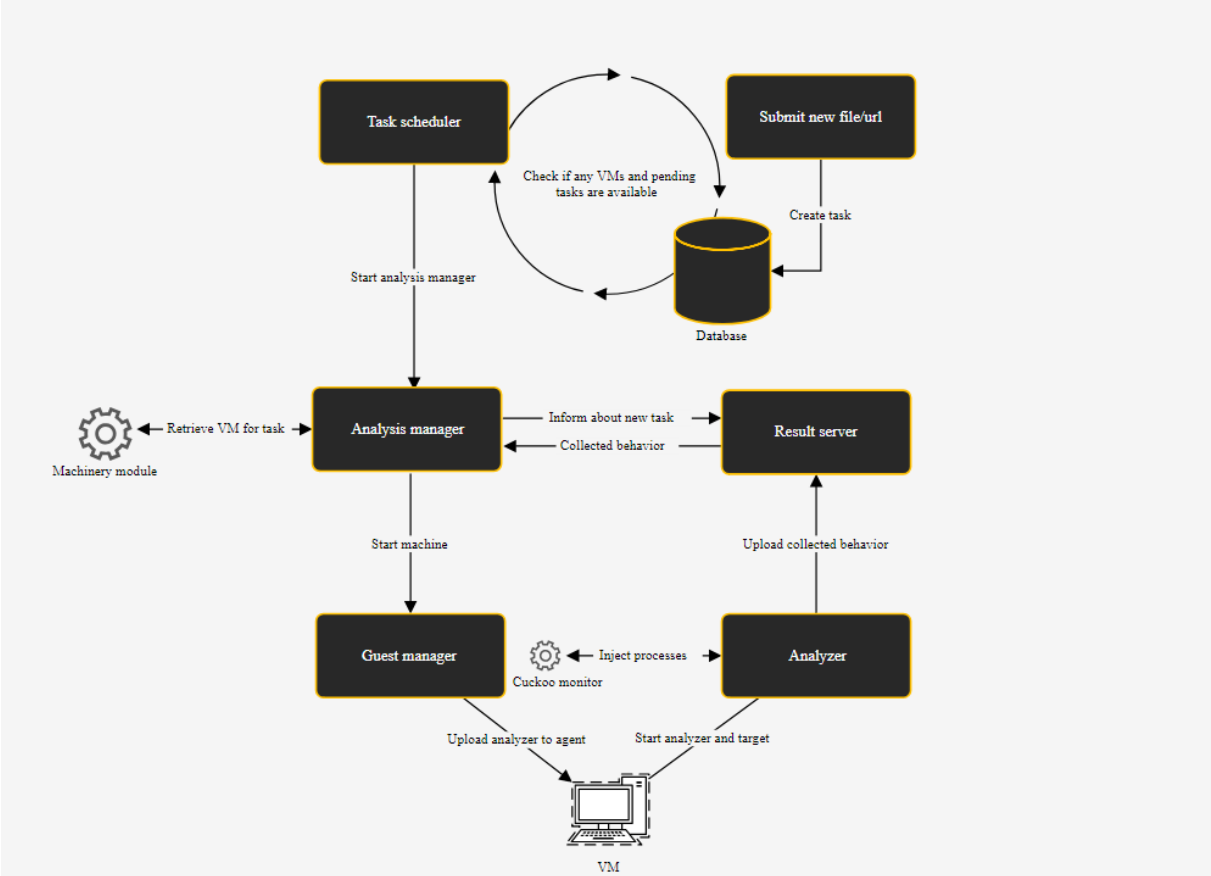

Figura 3 – Exemplo do *workflow* do *Cuckoo Sandbox*. Fonte: extraída de https://eforensicsmag.com/cuckoo-sandbox-architecture.

2.2 Trabalhos Relacionados

Nessa sessão são citados alguns trabalhos na linha de análise dinâmica de *ransomwares* e *malwares* em geral fazendo uso de ambientes virtuais, em especial utilizando *Cuckoo Sandbox*.

No trabalho de Ijaz, Durad e Ismail (2019) é usada uma combinação de atributos obtidos a partir da análise dinâmica de *malwares* executáveis como *datasets* para modelos de aprendizado de máquina comparados com um conjunto de dados composto por atributos extraídos de análise estática. A ferramenta foi a biblioteca em Python chamada

Portable Executable PEFILE e para análise dinâmica foi usada a *Cuckoo Sandbox*. Para a criação dos conjuntos de dados de *fetures* dinâmicas, os autores usaram o relatório emitido pelo *Cuckoo Sandbox* após a análise de uma entrada e extraíram as seguintes informações: relação de chamadas de API de sistema bem-sucedidas, mal-sucedidas e sua soma, contagem de interação com arquivos - quantidade de alterações, aberturas e exclusões -, contagem de IPs e URLs acessados e interação com os *registry* do Windows. Na sequência, combinações desses atributos, sumarizados como APIs, Registry, DLLs e Summary(arquivos e rede), foram submetidos a modelos de predição e a combinação de APIs e Summary obteve maior acurácia, tendo AUC de 95.86%. Para a montagem do conjunto de dados estático, os autores fizeram uso de mais de 92 atributos extraídos a partir da biblioteca PEFILE. Finalmente, o melhor conjuto de dados de atributos dinâmicos e o conjunto de dados de atributos estáticos foram submetidos a modelos de aprendizado de máquina, tendo melhor performance no *Gradient Classifier*, o primeiro tendo AUC 94.84% e o segundo AUC de 99.36%, mostrando que a análise estática foi mais precisa que a dinâmica. Os autores sugerem também que a análise estática é limitada devido a técnicas de ofuscação que os *malwares* usam e que, se somada a um ambiente virtual, os resultados de uma análise estática "dinâmica" pode apresentar melhores resultados no geral.

Em Darshan e Jaidhar (2019) os autores já aplicam um modelo híbrido de extração de atributos estáticos e dinâmicos utilizando o *Cuckoo Sandbox* e uma série de executáveis em Python usados para agregar as duas abordagens. A análise dinâmica desse trabalho é orientada ao estudo de chamadas API do *malware* no formato MIST (*Malware Instruction Set*), um modelo de representação binária semelhante a código de máquina (RIECK et al., 2011). Os atributos extraídos foram selecionadas por um LSVC (Linear Support Vector Classification), uma variação do modelo de aprendizado de máquina SVM (*Support Vector Machine*), que resultou em um dataset mais significativo. O conjunto de dados do experimento atingiu 99.743% de precisão.

Em Zhang et al. (2019), foi explorada a análise estática de *ransomware* compondo n-gramas, uma sequência de opcodes, a partir de toda cadeia de opcodes dos executáveis. No experimento foram usadas 1787 amostras de *ransomwares* divididas em 8 famílias: *cryptolocker*,*petya*,*wannacry*,*cryptowall*,*cryrar*,*locky*,*reveton* e *teslacrypt*. Foram trabalhados n-gramas de tamanho 2, 3 e 4, sendo que, o conjunto formado por 3 opcodes atingiu 99% de revocação, métrica que reflete a quantidade de amostras positivas que foram classificadas corretamente, usando o algoritmo de árvore de decisão.

No trabalho de (MILLER et al., 2017), os autores apresentam uma plataforma de análise dinâmica escalável que usa Cuckoo sandbox para análise dinâmica, bem como os resultados de experimentos realizados para melhorar a configuração dos componentes do sistema e ajudar a melhorar a precisão da análise dinâmica. No entanto, o foco do

trabalho é em malwares de modo geral e não é especializado para ransomware como o trabalho ora proposto.

Há exemplos de trabalhos que focam em apenas uma família de ransomware, como o WannaCry (KAO; HSIAO, 2018) e o Cerber (KARA; AYDOS, 2018). O foco deste trabalho, no entanto, é englobar um número maior de famílias.

3 Metodologia

O processo de coleta de *ransomwares*, seleção da ferramenta de análise dinâmica, a trativa e o refinamento dos dados usados para a construção do conjunto de dados para a predição de famílias de *ransomare* é descrito neste capítulo. Na Figura 4 é representada a visão geral da abordagem do projeto. Primeiramente uma série de amostras foram coletadas em sites da Internet e divididas em famílias, isto é, foram dividas em grupos com o mesmo código base. Em seguida, as amostras foram submetidas à *Cuckoo Sandbox*, uma ferramenta de análise de segurança, através de uma solução desenvolvida em Python. Os relatórios então foram processados por outra solução, também em Python, gerando um conjunto de dados inicial. Por fim, foram utilizados filtros para selecionar apenas amostras que foram executadas e que apresentaram comportamento significativo para o trabalho, construindo-se assim o conjunto de dados final. Uma vez que o conjunto de dados foi montado, ele foi testado usando uma árvore de decisão, um algoritmo de aprendizado de máquina, devido a sua fácil representação, interpretação e bons resultados em problemas multi-classe. Detalhes da implementação serão mostrados nas seções a seguir.

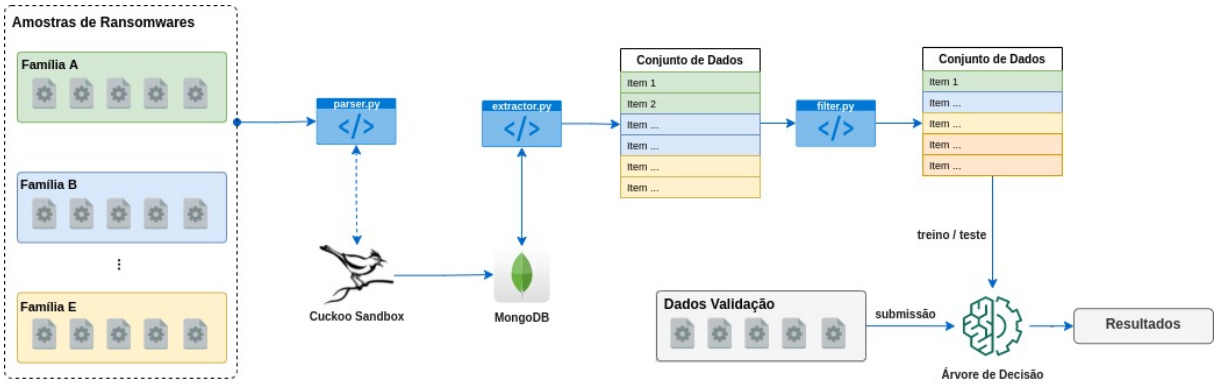

Figura 4 – Visão geral da abordagem. Fonte: do Autor.

3.1 Seleção da ferramenta de análise dinâmica

A ferramenta para análise dinâmica escolhida foi o *Cuckoo Sandbox*. O *Cuckoo Sandbox* é uma ferramenta modular especializada em análise de *malwares* tendo seu funcionamento baseado no conceito de *sandbox* e permite a submissão de arquivos executáveis, PDFs, arquivos WORD e BIN, através de uma interface gráfica ou por API, que posteriormente são executados e analisados em um ambiente isolado. Todo monitoramento dentro do ambiente isolado é feito por um agente que monitora todo tráfego de rede, *proxing*, chamadas de sistema e interações gráficas. O ambiente controlado, por sua vez, é uma

máquina virtual, no caso do projeto foi usada uma imagem de Windows 7 com todos requisitos de segurança desabilitados virtualizado por *Oracle VirtualBox*.

Para simular uma máquina real, configurações de saída para Internet foram realizadas. Todo tráfego de rede que ocorre dentro da máquina virtual foi monitorado através de um *proxy* de rede e um IP estático foi definido para a mesma. Além disso, como o comportamento de inicialização de *ransomwares* pode variar, cada amostra foi observada por 5 minutos.

3.2 Coleta de amostras

Para a coleta de amostras de *ransomwares*, foi usado o site VirusShare e amostras proprietárias. VirusShare é um site de proposito científico que indexa milhões de análises de arquivos testados na ferramenta VirusTotal e permite que sejam pesquisados de acordo com resultados das análises de diversos antivírus sobre tais arquivos. No site, amostras das famílias de interesse foram pesquisadas pelo nome e filtradas usando o critério de, no mínimo, dois antivírus que as classificaram como *ransomware* da família esperada.

Então, as amostras foram agrupadas em grupos determinados por suas respectivas famílias. Ao todo somaram-se 501 amostras e 12 grupos: BadRabbit, CryptoLocker, CryptoWall, Globeimposter, GandCrab, HiddenTear, Locky, Petya, Reveton, Ryuk, Sage, SamSam, Satan, Shade, Sodinokibi, Stop, TeslaCrypt, VegaLocker, WannaCry, WipeLocker e Xorist.

3.3 Submissão e extração de dados da *Cuckoo Sandbox*

Fazendo uso das interfaces de desenvolvedor *Cuckoo Sandbox* (API's para a submissão de arquivos para análise e extração de relatórios), foi realizada a montagem do conjunto de dados. Para a submissão do conjunto amostras de *ransomwares*, foi desenvolvida uma aplicação intermediária em Python que, além de submeter o arquivo, extrai informações conhecidas do arquivo: nome, classe do *ransomware* e se o arquivo está infectado ou não. A extração dos relatórios é feita por outro *script*, que associa as informações já conhecidas sobre o arquivo ao relatório fornecido pela análise do *Cuckoo Sandbox*. Tais procedimentos são executados na seguinte ordem e conforme a Figura 5:

1. Submissão do arquivo pelo *script* em Python para o *Cuckoo Sandbox*.
2. A ferramenta então executa o arquivo em ambiente controlado e escreve os resultados em um arquivo *json*.
3. Posteriormente, outra aplicação em Python recupera o relatório em conjunto com as informações já conhecidas do arquivo.

4. Informações são gravadas em um arquivo *json*.

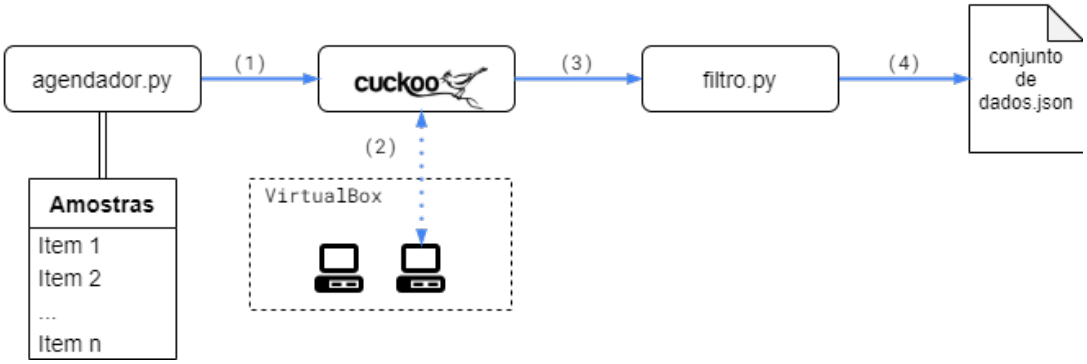

Figura 5 – Fluxo de submissão e extração de dados. Fonte: do Autor.

3.4 Estudo do Relatório do *Cuckoo Sandbox*

Como descrito na seção anterior, o resultado da submissão de um software ao processo desenvolvido é um relatório no formato *json*. Esse relatório contém informações precisas sobre: tempo de execução, análise de rede, análise estática do código, análise comportamental, *logs* de depuração e capturas de tela.

A análise de rede é feita com a ferramenta `pcap`, realizando um levantamento de todos os protocolos de comunicação utilizados (HTTP, SMTP, ICMP, dentre outros). Pacotes transmitidos usando UDP e TCP, quais endereços de IP, *hosts* e servidores a aplicação tentou se conectar também são dados levantados pelo *Cuckoo Sandbox*. Na análise estática do código, são enumeradas as bibliotecas, acesso ao *registry* e recursos encontrados no código fonte do *software* submetido. Como já existem outras ferramentas mais precisas para análise estática e o foco deste trabalho em questão é a análise dinâmica, os dados referentes à análise estática foram pouco explorados.

A análise comportamental, por sua vez, contém informações relacionadas à interação do *software* com o sistema operacional. Dentre elas, as principais são:

- Árvore de processos criada a partir do *software*.
- Arquivos que foram movidos, copiados, modificados, criados ou deletados pelo processo ou qualquer um dos seus filhos.
- Interação com o *registry* do Windows.
- Enumeração e criação de diretórios.
- Bibliotecas carregadas.
- Instruções executadas na linha de comando.

3.5 Definição do Conjunto de Dados

A montagem da conjunto de dados inicial consiste no uso de informações relacionadas, principalmente, ao comportamento das aplicações em relação ao sistema de arquivos da máquina virtual, pois elas representam padrões simples e esperados na execução de um *ransomware*, como a presença de bibliotecas de criptografia, renomeação de arquivos para uma extensão comum, exclusão de mecanismos de recuperação de dados, listagem de todos arquivos do sistema e até mesmo a troca de plano de fundo do dispositivo infectado. Além do mais, os atributos também compreendem mudanças de contexto de um atributo, como:

- A presença de uma biblioteca de criptografia é comum em softwares de médio/grande porte que também apresentam grande quantidade de bibliotecas baixadas, enquanto *ransomwares* fazem baixo uso de bibliotecas;
- Grandes quantidades de arquivos criados, modificados ou deletados dentro de um único subdiretório é uma característica comum de qualquer aplicação. Contudo a modificação em massa em diversos diretórios raiz deixa de ser escopo de uma aplicação comum e é uma atividade suspeita.

A seguir, é apresentada a listagem dos atributos iniciais selecionadas e a Tabela 1 exibe 2 entradas do conjunto de dados, a saber:

1. *dll_downloaded*: Quantidade de bibliotecas baixadas.
2. *crypto_dll_downloaded*: A presença de alguma biblioteca de criptografia carregada durante a execução.
3. *file_opened*: Quantidade de arquivos abertos.
4. *file_created*: Quantidade de arquivos criados.
5. *file_written*: Quantidade de arquivos escritos.
6. *file_read*: Quantidade de arquivos lidos.
7. *file_moved*: Quantidade de arquivos movidos.
8. *file_recreated*: Quantidade de arquivos recriados.
9. *file_renamed*: Quantidade de arquivos renomeados.
10. *only_extension_changed*: Se somente a extensão dos arquivos movidos, recriados e copiados foi alterada.

11. *commonest_orig_extension*: Extensão de arquivo mais comum antes da execução do software.

12. *commonest_orig_extension_count*: Quantidade de arquivos com a extensão mais comum antes da execução do software.

13. *commonest_moved_extension*: Extensão de arquivo mais comum após a execução do software.

14. *commonest_moved_extension_count*: Quantidade de arquivos com a extensão mais comum após a execução do software.

15. *paths_enumerated*: Quantidade total de diretórios que foram enumerados.

16. *major_paths_enumerated*: Quantidade de diretórios raiz que foram enumerados.

17. *enumerate_more_than_3_major_paths*: Se mais de 3 diretórios raiz foram enumerados.

18. *deleted_shadow*: Se os arquivos de *backup* automático do Windows foram apagados.

Para a obtenção dos atributos, uma solução foi desenvolvida em Python. Uma série de extratores específicos foram desenvolvidos para tratar subgrupos de atributos de acordo com suas localizações dentro do relatório *json* emitido pelo *Cuckoo Sandbox*. Para a coleta de informação sobre libs de criptografia, categorizando os atributos 1 e 2, foi analisada a sequência de chamadas de sistema realizadas, presente no *json*, buscando por carregamento de bibliotecas de criptografia. Outra seção do *json*, chamada sumário, retém toda informação acerca da interação com arquivos do sistema, logo um extrator de informação de arquivos foi criado, sendo reponsável pela coleta dos atributos de 3 a 7. Com os dados da mesma seção, envolvendo arquivos, outro extrator com função de levantar dados comparativos do sistema de arquivo antes e depois da execução da amostra foi desenvolvido englobando os atributos de 8 a 12. Ainda na mesma seção, o atributo *diretory_enumerated* permitiu a extração dos atributos 11, 12 e 13 e 14 que representam interação com diretórios. Por fim, os comandos *deleted_shadow*, *mv shadows » NULL* e qualquer interação com o sistema de backup automático do Windows são características comuns em *ransomwares*, portanto um ultimo extrator foi desenvolvido com o intuito de analisar todas as chamadas realizadas em prompts de comando, mesmo que em segundo plano, para identificar comandos suspeitos, permitindo a elaboração dos atributos 15 e 16.

Tabela 1 – Amostra com 2 elementos do conjunto de dados

dll_downloaded	0	0
crypto_dll_downloaded	False	False
file_opened	21	2006
file_created	0	0
file_written	9	1895
file_read	8.0	1906.0
file_moved	0	1884.0
file_recreated	5	5
file_renamed	0	0
only_extension_changed	False	True
commonest_orig_extension	0	py
commonest_orig_extension_count	0	1700
commonest_moved_extension	0	vvv
commonest_moved_extension_count	0	1884
paths_enumerated	4	490
major_paths_enumerated	1	8
enumerate_more_than_3_major_paths	False	True
deleted_shadow	False	True

3.6 Refinamento do Conjunto de Dados

Para determinar a qualidade dos atributos, foram desenvolvidos e aplicados testes empíricos de significância. *Atributos* como a quantidade de bibliotecas baixadas e quantidade de arquivos renomeados apresentaram valor constante 0 e foram eliminadas do conjunto de dados. Uma restrição que filtra somente amostras que apresentam, pelo menos, uma forma de interação que interessa ao modelo, isto é, executa chamadas de sistema para abrir arquivos, escrever arquivos ou listar diretórios, eliminando também amostras que não foram executadas com sucesso no ambiente virtual, também foi aplicada, resultando em 72 amostras no conjunto de dados final e 11 atributos.

Para a manipulação e visualização do conjunto de dados, as bibliotecas *numpy* (NUMPY, 2021) e *pandas* (PANDAS, 2021) , ambas em Python, foram utilizadas. Toda conversão de atributos, localização de inconsitência de dados e aplicação de filtros foram aplicados usando a bibilioteca *pandas*.

3.7 Criação do modelo de Classificação

Dada a natureza multi-classe do projeto, pois é necessário classificar a qual classe uma dada entrada pertence. O algoritmo de Árvore de Decisão, um algoritmo de aprendizado de máquina de classificação supervisionado, foi utilizado. O algoritmo de Árvore de decisão é baseado na estrutura de uma árvore onde cada nó representa uma escolha sobre uma determinado atributo, fazendo com que uma entrada siga um fluxo de múltiplas decisões até que alcance um nó folha. Tal nó folha representa a classe da entrada. Esse método é utilizado principalmente para tarefas de agrupamento e reconhecimento de padrões (JIJO; ABDULAZEEZ, 2021) dadas classes de amostras já conhecidas, por isso é chamado de aprendizado supervisionado. Sua fácil representação visual também

permite uma análise mais refinada acerca dos atributos escolhidos devido à sua estrutura de árvore e decisões comparativas.

Para determinar a qualidade do modelo, 4 métricas foram utilizadas:

- Acurácia: determina o comportamento médio do modelo através da média de amostras por família que foram classificadas de forma correta pelo modelo, verdadeiros positivos e verdadeiros negativos. Tem fórmula:

$$\frac{1}{n}\sum_{i=1}^{n}\frac{VP_i + VN_i}{VP_i + VN_i + FP_i + FN_i} \tag{3.1}$$

- Precisão: determina dentre todas as classificações positivas do modelo, quantas estão realmente corretas, possibilitando a identificação de quais famílias estão sendo classificadas erroneamente. Tem fórmula:

$$\frac{1}{n}\sum_{i=1}^{n}\frac{VP_i}{VP_i + FP_i} \tag{3.2}$$

- *Recall*: determina a sensibilidade do modelo através da média de amostras por família que deveriam ser classificadas como positivas dividido pelas predições do modelo sobre essas amostras. Também auxilia diretamente na determinação de quais amostras estão sendo classificadas erroneamente. Tem fórmula:

$$\frac{1}{n}\sum_{i=1}^{n}\frac{VP_i}{VP_i + FN_i} \tag{3.3}$$

- *F1-Score*: é a média harmônica do modelo. É composta da combinação da Precisão e *Recall* do modelo, estabelecendo uma média entre essas duas métricas. Representa o comportamento médio do modelo em relação tanto a falsos negativos quanto a falsos positivos. Tem fórmula:

$$\frac{1}{n}\sum_{i=1}^{n}\frac{2 * Precisão * Recall}{Precisão + Recall} \tag{3.4}$$

Nas Equações 3.1, 3.2, 3.3 e 3.4, n representa o número de famílias de *ransomware*, VP as instâncias Verdadeiras Positivas, VN as instâncias Verdadeiras Negativas, FP as instâncias Falsas Positivas e FN as Falsas Negativas.

A implementação do modelo foi feita em linguagem de programação Python, juntamente com *scikit-learn* (SCIKIT-LEARN, 2021), uma biblioteca de análise de dados que contém diversos algoritmos estatísticos de aprendizagem de máquina. A modelagem foi dada em 6 passos:

1. Normalização dos dados usando a biblioteca *numpy*, entre -1 e 1.
2. Criação do classificador usando a implementação nativa do *scikit-learn*: *DecisionTreeClassifier*.
3. Divisão do conjunto de dados em amostras de treinamento e amostras de teste seguindo a proporção de 0.7 e 0.3, com as amostras divididas aleatoriamente.
4. Treinamento do modelo com as amostras de treinamento.
5. Teste do modelo com as amostras de teste.
6. Coleta das métricas citadas na seção anterior, por meio das funções nativas do *scikit-learn*.

Com essa modelagem o modelo está completo para análise.

4 Resultados

Para a montagem do conjunto de dados, um conjunto de amostras categorizadas por famílias de *ransomware* foi montado. Em seguida, um conjunto de atributos foi levantado sobre a análise do resultados das amostras quando submetidas a um ambiente simulado. Tais resultados permitiram a definição da afinidade das amostras dentro de cada família e se as mesmas apresentam comportamento tendencioso, resultando no conjunto de dados. Tais resultados, tanto da seleção dos atributos, famílias de amostras e do conjunto de dados são apresentados a seguir.

4.1 Análise de Atributos

Para determinar os resultados sobre a qualidade entre atributos levantados no projeto, uma matriz de correlação foi utilizada. Ela define o grau de afinidade entre cada par de atributos representa quanto tais atributos estão linearmente relacionados, significando que pares com grande afinidade contribuem da mesma forma no modelo e que podem ser agrupados. A matriz de correlação dos atributos na Figura 6 mostra que o grupo referente à manipulação direta de arquivos tem grande afinidade e pode ser agrupado. No entanto, como nuances no comportamento de manipulação de arquivos é um fator determinante para identificar um tipo de *ransomware*, tais atributos não foram agrupados, além do fato da matriz de correlação exprimir somente a relação de proporção linear. Para determinar os resultados acerca da performance do conjunto de amostras dentro de uma mesma família, a média dos atributos selecionados junto com o desvio padrão em cada um foram utilizados.

A Figura 7 é uma amostra do padrão de comportamento exibido por cada amostra das famílias de *ransomwares*. É possível perceber na mesma que, apesar das variações, grande parte das amostras dentro de uma família possuem comportamento similar em relação aos atributos escolhidos.

Na Figura 8, é exibida a média desse comportamento juntamente com o desvio padrão, traçando um aspecto da silhueta do comportamento de cada família de *ransomware*. Na mesma figura, o desvio padrão indica a possível oscilação dos dados entre amostras de cada família, sendo que, quanto menor a barra vertical sobre cada atributo, menor é sua oscilação, indicando um comportamento mais uniforme da mesma entre as diversas amostras.

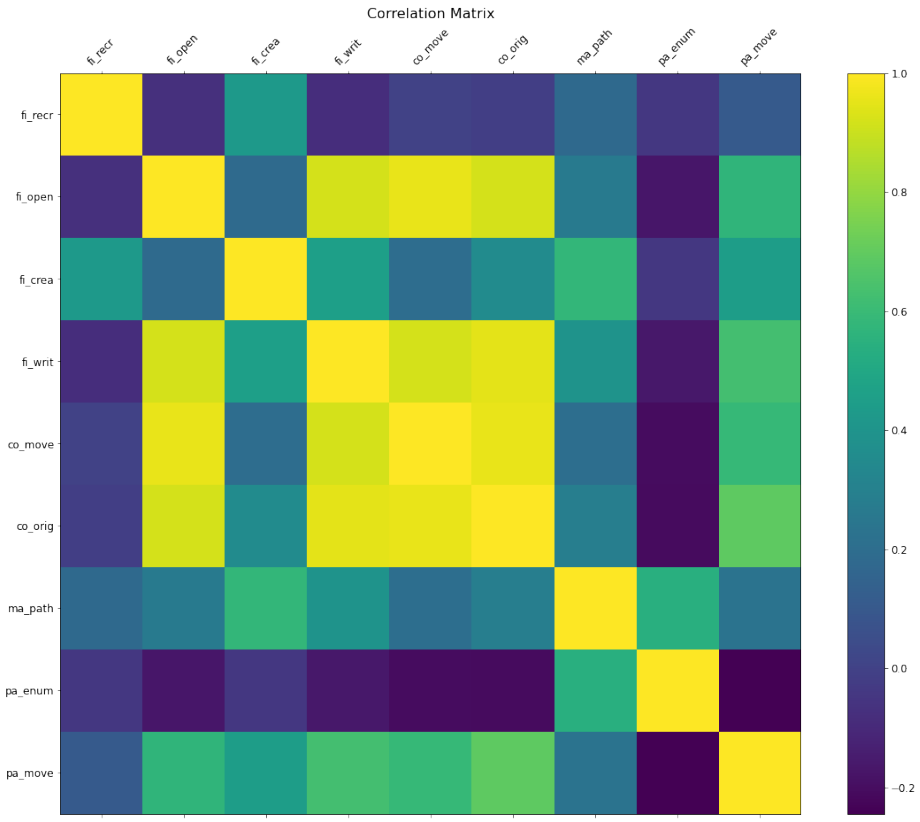

Figura 6 – Matriz de correlação dos atributos. Fonte: do Autor.

4.2 Análise do Conjunto de Dados

Para determinar os resultados da performance da Árvore de Decisão, uma série de iterações de treinamento e teste foram executas no modelo, sendo que, a cada iteração, as métricas precisão, acurácia, *recall* e *f1-score* foram extraídas do modelo treinado. Para o treinamento, foram usados 70% do conjunto de dados, com registros aleatórios a cada execução. Para a avaliação do modelo foram usados os 30% de registros restantes. Por fim, as métricas foram sumarizadas, calculando suas médias e seus respectivos desvios padrão para determinar a estabilidade do modelo.

Após as análises e restrições, o conjunto de dados ficou composto por 73, com 8 famílias de *ransomwares* - CryptoLocker, Globeimposter, HiddenTear, Shade, Sodinokibi, VegaLocker, WannaCry e Xoritst - e com 11 atributos: arquivos recriados, arquivos abertos, arquivos criados, arquivos escritos, deleção do backup *shadow*, se apenas arquivos com a mesma extensão foram alterados, maior número de arquivos de uma mesma extensão antes da execução, maior número de arquivos da mesma extensão após a execução, se a extensão com mais arquivos foi alterada após a execução, se mais de 3 caminhos da raiz

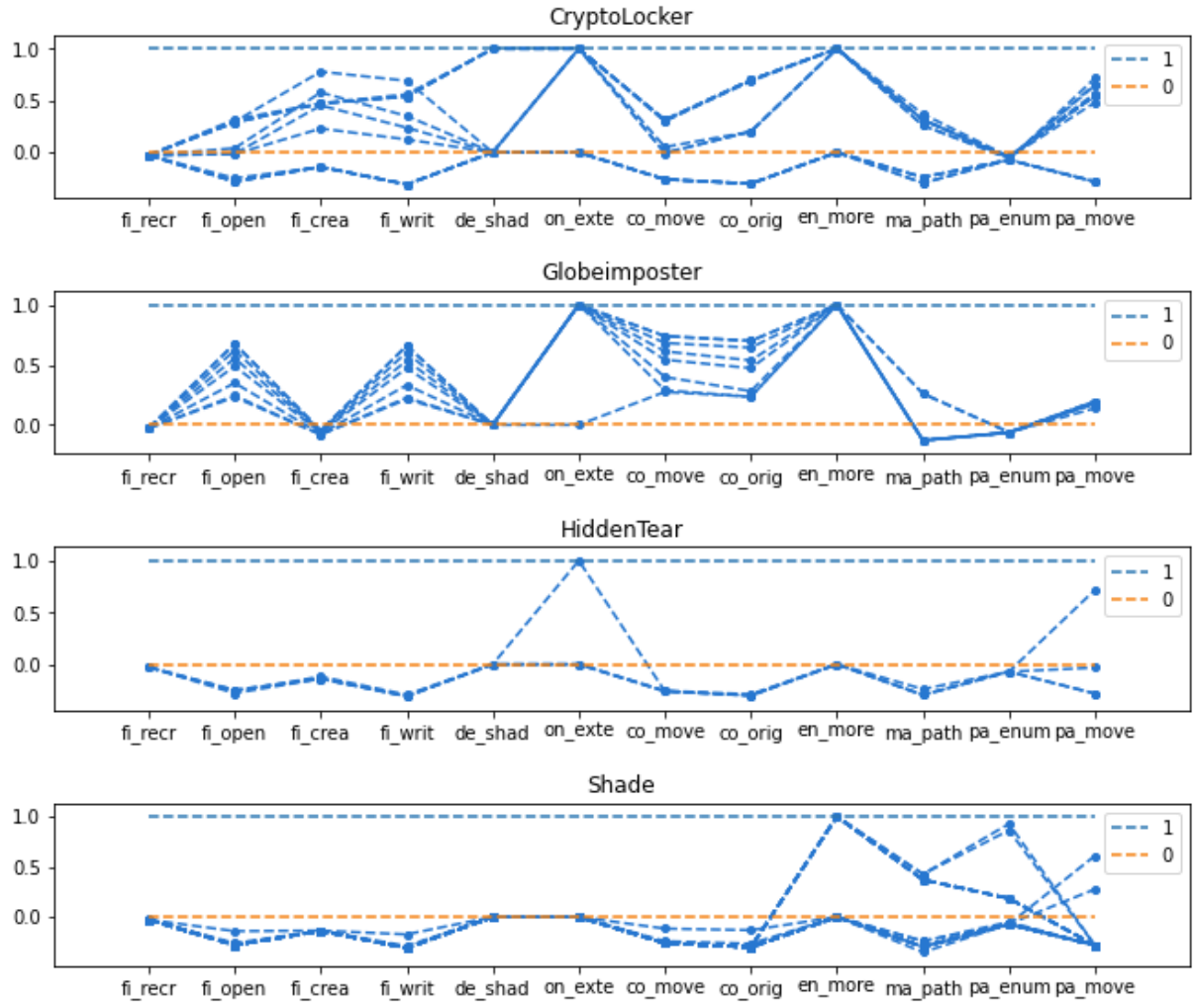

Figura 7 – Padrão de amostras por família. Fonte: do Autor.

do sistema foram listados e número diretórios e sub-diretórios que foram listados.

Na Figura 9 são mostrados os resultados das primeiras 200 iterações da bateria de testes que foi realizada no modelo. O modelo teve desempenho médio de 69% em relação ao *recall*, significando que classificou corretamente 69% de amostras corretamente em suas respectivas famílias. Em relação à precisão, o modelo teve média de 71%, indicando que 29% das amostras foram classificadas erroneamente como pertencentes a uma determinada família. O desempenho médio geral, determinado pela acurácia e *f1-score* foram de 68% e 70%, respectivamente. Pode-se perceber na figura que houve picos em que as métricas ficaram muito acima da média e outras depressões onde as mesmas ficaram muito abaixo. Nos cenários de pico, houve diversos cenários de elitismo e *starvation*, onde famílias com mais amostras foram melhor distribuídas e tiveram grande assertividade, enquanto famílias com menos amostras foram quase ou totalmente eleitas para a fase de treinamento, gerando lacunas nos cenários de testes, promovendo uma maior assertividade por falta de diversidade. Já nos cenários de depressão, ocorreram casos inversos aos de pico: a falta de amostras na fase de treino do modelo gerou um cenário de *under fitting*, definido pela

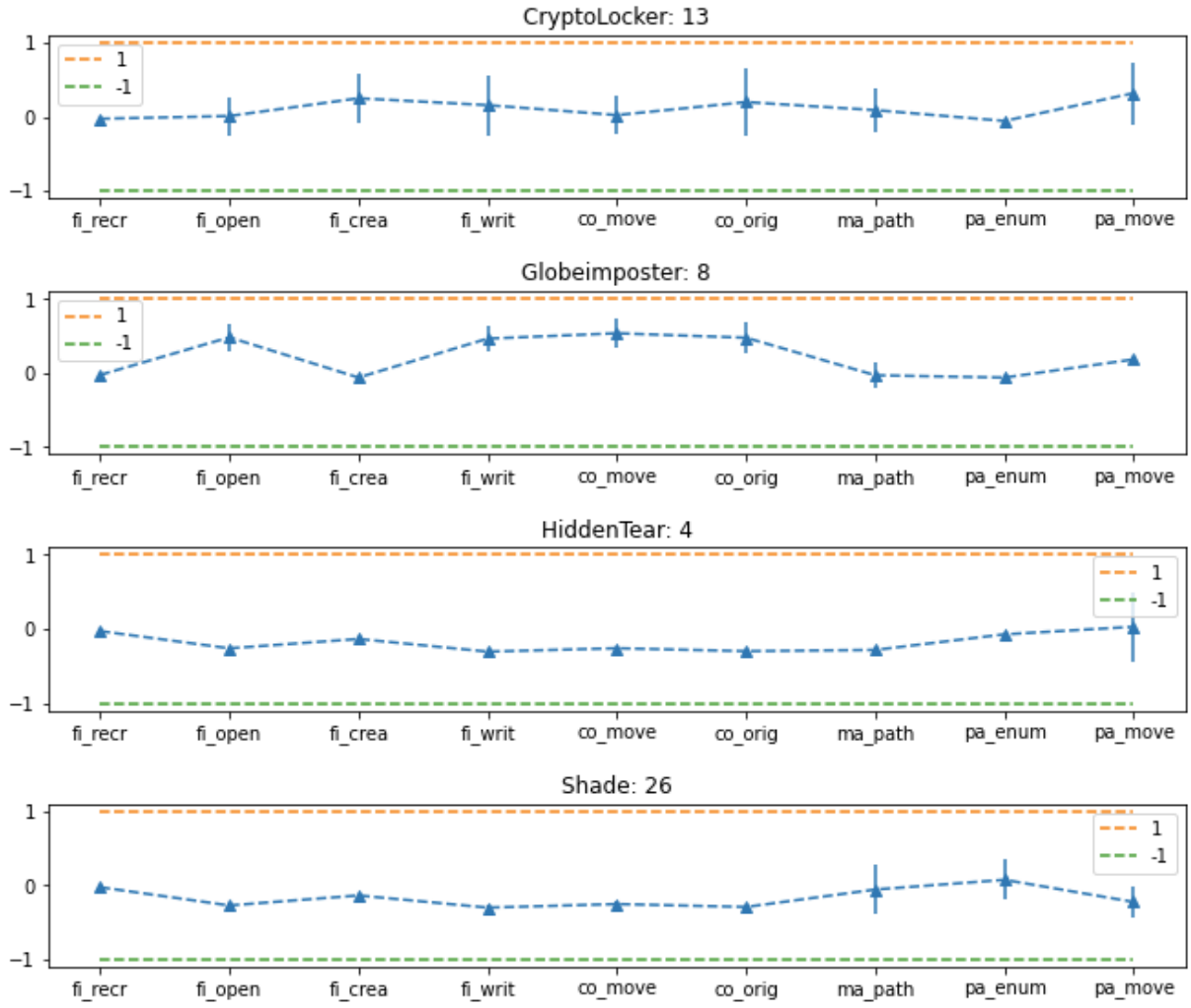

Figura 8 – Média das amostras de cada família com desvio padrão. Fonte: do Autor.

escassez de amostras fazendo que com o modelo ficasse sub-ajustado e não conseguisse extrair os padrões necessários para classificar os dados (BADILLO et al., 2020). Tal evento fez com que, na fase de testes, o modelo tivesse desempenho abaixo da média esperada.

A Figura 10 exibe a matriz de confusão do melhor classificador treinado, com acurácia de 69%, sendo que 13 das 19 amostras de testes foram classificadas corretamente. Devido baixa quantidade de amostras funcionais, a quantidade de amostras de testes não revelou precisamente os motivos dos falsos negativos. Ainda assim, por meio de testes comparativos, mostra-se que amostras entre pares de famílias, como Sodinokibi e Xorist, ambas famílias de *ransomwares*, ainda possuem sobreposições bem evidentes, como mostado na Figura 11

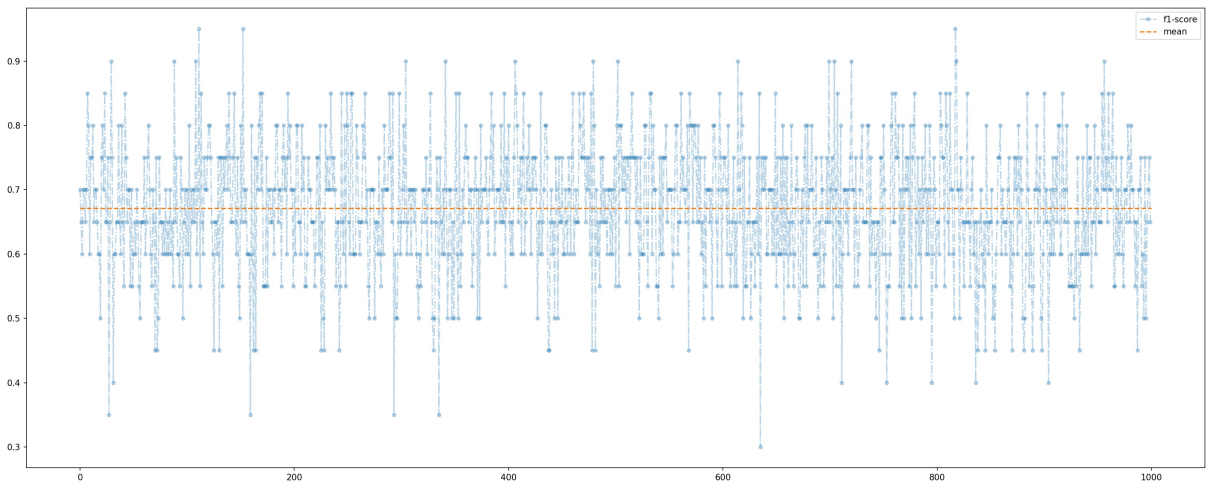

Figura 9 – Resultados Árvore de Decisão. Fonte: do Autor.

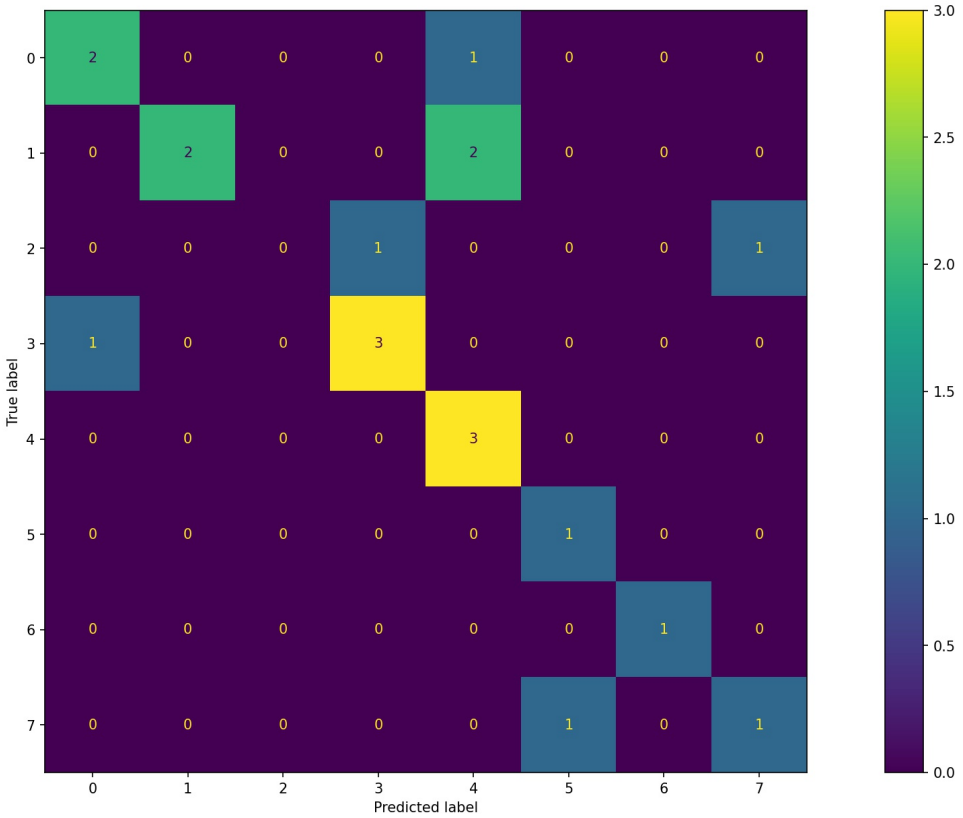

Figura 10 – Matriz de confusão do modelo. Fonte: do Autor.

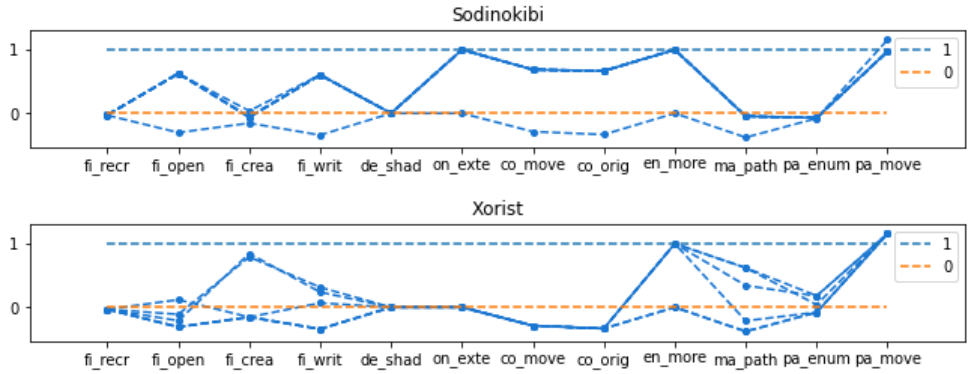

Figura 11 – Comparação amostras da família Sodinokibi e Xorist. Fonte: do Autor.

5 Conclusão

O avanço do aprendizado de máquina e o aumento da disponibilização de dados na Internet tem possibilitado a criação de modelos inteligentes de detecção. A partir dos recursos resultantes dessa evolução, o objetivo deste trabalho foi identificar a qual família um *ransomware* pertence. Toda a coleta de dados foi realizada no VirusShare, um site especializado em indexação de *malwares*, e o ambiente virtualizado utilizado para execução das amostras foi o *Cuckoo Sandbox*, uma ferramenta de código aberto mantida pela comunidade de desenvolvedores na Internet. Além disso, a popularização de técnicas de aprendizagem de máquina possibilitou a exploração de padrões de comportamentos que funções e grafos não conseguem identificar. Neste trabalho, foi explorada uma categoria de atributos de *ransomwares* que é bem inerente ao seu funcionamento, a manipulação de arquivos. Seguindo tal linha, todos os relatórios obtidos do ambiente virtual foram manipulados e compilados em atributos numéricos referentes a manipulação de arquivos: movimentação, cópia, deleção, alteração de nome, extensão, dentre outras. Em sequência, os dados foram analisados e filtrados de maneira a preservar somente os que possuíam relevância e, por fim, os dados foram usados como conjunto de dados para o treinamento de uma árvore de decisão, obtendo resultados de acurácia média de 70%.

Contudo, dificuldades também foram encontradas no desenvolvimento deste trabalho. Das 512 amostras coletadas, mais da metade não executaram corretamente no ambiente virtual, tanto por técnicas de evasão de *sandbox* – artifício em que um executável entra em estado de hibernação e não executa se identificar que está sendo executado em um ambiente virtual –, quanto pelas limitações da aplicação ao executar os arquivos, pois existem diversas formas de execução, como arquivos executáveis, PDF, macros em arquivos WORD e *PowerPoint*, programas em outras linguagens, dentre outros. A aquisição de amostras classificadas de *ransomwares* também é limitada a arquivos relativamente antigos, variando de 2014 a 2016, tornando mais difícil acompanhar o ritmo com que novos *ransomwares* são desenvolvidos. Um trabalho a ser explorado é o desenvolvimento de uma nova ferramenta que permita essas análises e que permita a classificação dos *malwares* submetidos, criando uma base de dados indexada, catalogada e aberta para pesquisa dos arquivos já submetidos. Assim como a exploração do relatório emitido pelo *Cuckoo Sandbox*, a fim de aumentar o número de atributos do modelo e reduzir os falsos negativos, como visto no capítulo anterior.

Os resultados obtidos a partir da árvore de decisão são satisfatórios, mas ainda podem evoluir. Outros modelos mais sofisticados, como redes neurais e e redes profundas podem ser aplicados, abrindo possibilidades para exploração de um conjunto maior de atributos, que vão desde o mapeamento de processos até o estudo da sequência de

procedimentos executados na máquina virtual.

Referências

ALSHAIKH, H.; RAMADAN, N.; HEFNY, H. A. Ransomware prevention and mitigation techniques. *International Journal of Computer Applications*, v. 177, n. 40, p. 31–39, Feb. 2020. Disponível em: <https://doi.org/10.1109/JPROC.2010.2070470>. Acesso em: 27 jan. 2021. Citado 3 vezes nas páginas 8, 9 e 10.

AV-TEST, ORG. *AV Test Security Report 2019/2020*. [S.l.], 2020. Disponível em: <https://www.av-test.org/fileadmin/pdf/security_report/AV-TEST_Security_Report_2019-2020.pdf>. Acesso em: 27 jan. 2021. Citado na página 8.

BADILLO, S. et al. An introduction to machine learning. *Clinical Pharmacology Therapeutics*, v. 107, 03 2020. Disponível em: <https://doi.org/10.1002/cpt.1796>. Citado na página 30.

CISCO SYSTEMS, INC. *Cisco Annual Internet Report (2018–2023) White Paper*. [S.l.], 2020. Disponível em: <https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.pdf>. Acesso em: 27 jan. 2021. Citado na página 8.

CRACIUN, V. C.; MOGAGE, A.; SIMION, E. Trends in design of ransomware viruses. In: LANET, J.-L.; TOMA, C. (Ed.). *Innovative Security Solutions for Information Technology and Communications*. Cham: Springer International Publishing, 2019. p. 259–272. ISBN 978-3-030-12942-2. Disponível em: <https://doi.org/10.1007/978-3-030-12942-2_20>. Citado na página 11.

DARGAHI, T. et al. A cyber-kill-chain based taxonomy of crypto-ransomware features. *Journal of Computer Virology and Hacking Techniques*, v. 15, n. 4, p. 277–305, Dec 2019. ISSN 2263-8733. Disponível em: <https://doi.org/10.1007/s11416-019-00338-7>. Citado 2 vezes nas páginas 12 e 13.

DARSHAN, S. L. S.; JAIDHAR, C. D. Windows malware detection system based on lsvc recommended hybrid features. *Journal of Computer Virology and Hacking Techniques*, v. 15, n. 2, p. 127–146, Jun 2019. ISSN 2263-8733. Disponível em: <https://doi.org/10.1007/s11416-018-0327-9>. Citado na página 17.

IDIKA, N.; MARTHUR, A. P. A survey of malware detection techniques. *Purdue University*, p. 48, 2007. Citado na página 8.

IJAZ, M.; DURAD, M. H.; ISMAIL, M. Static and dynamic malware analysis using machine learning. International Bhurban Conference on Applied Sciences & Technology, Jan. 2019. Doi: <10.1109/IBCAST.2019.8667136>. Citado 3 vezes nas páginas 9, 14 e 16.

JIJO, B.; ABDULAZEEZ, A. M. Classification based on decision tree algorithm for machine learning. *Journal of Applied Science and Technology Trends*, v. 2, p. 20–28, 01 2021. Disponível em: <https://doi.org/10.38094/jastt20165>. Citado na página 24.

KAO, D.-Y.; HSIAO, S.-C. The dynamic analysis of wannacry ransomware. In: *2018 20th International Conference on Advanced Communication Technology (ICACT)*. [s.n.],

2018. p. 159–166. Disponível em: <https://doi.org/10.23919/ICACT.2018.8323682>. Citado na página 18.

KARA, I.; AYDOS, M. Static and dynamic analysis of third generation cerber ransomware. In: *2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)*. [s.n.], 2018. p. 12–17. Disponível em: <https://doi.org/10.1109/IBIGDELFT.2018.8625353>. Citado na página 18.

KHARAZ, A. et al. UNVEIL: A large-scale, automated approach to detecting ransomware. In: *25th USENIX Security Symposium (USENIX Security 16)*. Austin, TX: USENIX Association, 2016. p. 757–772. ISBN 978-1-931971-32-4. Disponível em: <https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/kharaz>. Citado na página 13.

LISKA, A.; GALLO, T. *Ransomware: Defending Against Digital Extortion*. [S.l.]: O'Reilly, 2016. Citado 3 vezes nas páginas 10, 11 e 13.

MILLER, C. et al. Insights gained from constructing a large scale dynamic analysis platform. *Digit. Investig.*, Elsevier Science Publishers B. V., NLD, v. 22, n. S, p. S48–S56, ago. 2017. ISSN 1742-2876. Disponível em: <https://doi.org/10.1016/j.diin.2017.06.007>. Citado na página 17.

NUMPY. 2021. Disponível em: <https://numpy.org>. Citado na página 24.

O'Kane, P.; Sezer, S.; McLaughlin, K. Obfuscation: The hidden malware. *IEEE Security Privacy*, v. 9, n. 5, p. 41–47, 2011. Disponível em: <https://doi.org/10.1109/MSP.2011.98>. Citado na página 14.

PANDAS. 2021. Disponível em: <https://pandas.pydata.org>. Citado na página 24.

RIECK, K. et al. Automatic analysis of malware behavior using machine learning. *Journal of Computer Security*, v. 19, p. 639–668, 06 2011. Disponível em: <https://doi.org/10.3233/JCS-2010-0410>. Citado na página 17.

SCIKIT-LEARN. 2021. Disponível em: <https://scikit-learn.org/stable>. Citado na página 25.

ZHANG, H. et al. Classification of ransomware families with machine learning based onn-gram of opcodes. *Future Generation Computer Systems*, v. 90, p. 211–221, 2019. ISSN 0167-739X. Disponível em: <https://doi.org/10.1016/j.future.2018.07.052>. Citado na página 17.

ZIMBA, A.; WANG, Z.; CHEN, H. Multi-stage crypto ransomware attacks: A new emerging cyber threat to critical infrastructure and industrial control systems. *ICT Express*, v. 4, 01 2018. Disponível em: <https://doi.org/10.1016/j.icte.2017.12.007>. Citado 2 vezes nas páginas 14 e 15.