CURSO DE CIÊNCIA DA COMPUTAÇÃO – TCC	
(X) PRÉ-PROJETO ( ) PROJETO	ANO/SEMESTRE: 2023/2

APLICAÇÃO DE ENGENHARIA REVERSA PARA DETECÇÃO DE INDÍCIOS MALICIOSOS EM BINÁRIOS

Paulo Rubens de Moraes Leme Júnior

Prof. Danton Cavalcanti Franco Junior – Orientador

1 INTRODUÇÃO

A cibersegurança é uma área que tem ganhado destaque pelo crescente número de ameaças e incidentes. Devido ao cibercrime ser um modelo de negócio extremamente rentável para malfeitores ou organizações criminosas, nota-se uma frequente tendência de necessidade de investimentos na área de proteção, especialmente quando as empresas do mercado tendem a sofrer enormes prejuízos, financeiros e legais, quando estão envolvidas em algum incidente de cibersegurança. Em 2015, por exemplo, o cibercrime já tinha atingido a marca de 3 trilhões de dólares de danos causados, sendo que, devido a fatores como a rápida tendência de digitalização de negócios, proveniente dos efeitos da pandemia de Covid-19, houve um aumento de 600% no número de ataques, totalizando um dano estimado de 6 trilhões de dólares em 2021 (Liu *et al.*, 2022).

Também impulsionada pela pandemia, a prática do trabalho remoto contribuiu para os funcionários estarem mais vulneráveis e expostos a ciberataques, uma vez que, muitas vezes, as camadas de proteção presentes convencionalmente em um ambiente de trabalho físico são inexistentes ou ineficazes para funcionários remotos. Além disso, a ampla gama de configurações de ferramentas de segurança, como antivírus, *intrusion detection systems* (IDS) ou *intrusion prevention systems* (IPS), acaba gerando margem para *misconfiguration*, que consiste na ineficiência da solução por causa da falta de configurações ou políticas adequadas. Em diversos casos, a performance acaba recebendo maior prioridade do que a segurança, o que leva equipes de cibersegurança a configurarem as soluções para que executem de uma forma mais branda, evitando impacto nos recursos do *host*, mas comprometendo o nível prático de segurança.

Buscando seguir as boas práticas de cibersegurança, é fundamental que seja adotada a estratégia de *defense in depth* (defesa em profundidade). Essa estratégia destaca a importância de múltiplas camadas de proteção para mitigar riscos e ameaças e baseia-se na premissa de que uma única medida de proteção não é suficiente ou eficiente como proteção contra ameaças cada vez mais avançadas. Ao se eliminar o ponto único de falha, é possível ter ganho considerável de segurança, além de servir como empecilho para ataques em andamento (Mughal, 2018).

Baseando-se no fortalecimento da cobertura de proteção, é de extrema importância que as soluções utilizadas estejam com a maior visibilidade possível e estejam com suas assinaturas de detecção de *Malware* devidamente atualizadas. Apesar disso, grande parte das soluções de mercado possuem um banco de assinaturas próprio, que, muitas vezes, pode não estar abrangendo o que é necessário para um bom nível de eficácia na detecção e bloqueio de ameaças. Devido à grande variedade de instituições pesquisadoras e pesquisadores independentes, podem existir casos em que o interesse da equipe de segurança ou do usuário seja comparar uma amostra suspeita com as suas próprias fontes de interesse ou correlacionar esses dados com uma combinação de técnicas de análise do arquivo.

Diante deste cenário, esse trabalho propõe o desenvolvimento de uma aplicação para ser utilizada como validação e investigação de indícios de *Malware* em arquivos e análise de comportamentos de risco durante a execução de arquivos em ambiente controlado, contribuindo, portanto, para a detecção de riscos de cibersegurança no ambiente empresarial ou doméstico.

1.1 OBJETIVOS

O objetivo deste trabalho é desenvolver uma aplicação que seja capaz de analisar um binário em busca de indícios maliciosos de *Malware*, tanto de forma estática, através do arquivo em si, quanto de forma dinâmica, através de análise de execução em ambiente controlado.

Os objetivos específicos são:

a) disponibilizar uma plataforma para envio de amostras a serem testadas e controle das fontes de inteligência de ameaças;
b) disponibilizar um agente destinado para análise dinâmica em execução;
c) analisar comportamentos suspeitos de *Malware*, tráfego de rede e estrutura de arquivo;
d) apresentar um veredito do arquivo, baseando-se na correlação de todas as análises realizadas.

2 TRABALHOS CORRELATOS

Neste capítulo são apresentados trabalhos com características semelhantes aos principais objetivos do estudo proposto. Na seção 2.1, são apresentadas técnicas para otimização de detecção de assinaturas de *Malware* através de regras YARA (Naik *et al.*, 2020). A seção 2.2 apresenta um sistema desenvolvido para detecção de tráfego de rede malicioso (Ghafir *et al.*, 2018). Por fim, a seção 2.3 demonstra as técnicas utilizadas pelos autores Patil *et al.* (2019) no desenvolvimento de um *framework* para detecção de *Malware* em máquinas virtuais.

2.1 YARA RULES: STRENGTHENING YARA RULES UTILISING FUZZY HASHING AND FUZZY RULES FOR MALWARE ANALYSIS

Os autores Naik *et al.* (2020) apresentam técnicas utilizadas para análise de *Malware* e como obter um resultado otimizado. De início, é apresentada a utilização de regras YARA convencionais, que têm como propósito a identificação de *Malware* através da comparação do código atual do arquivo, pasta ou processo com assinaturas ou *strings* de amostras de *Malware* já conhecidas. As regras são definidas em três blocos: *meta*, *strings* e *condition*, conforme mostra a Figura 1.

Figura 1 – Exemplo de regra YARA

```
rule RuleName
{
meta:
  description = "descriptions of rule"
  author = "name"
  date = "dd/mm/yyyy"
  reference = "url"

strings:
  $text_string1 = "text1 you wish to find in malware"
  $text_string2 = "text2 you wish to find in malware"

  $hex_string1 = {hex1 you wish to find in malware}
  $hex_string2 = {hex2 you wish to find in malware}

  $reg_exp_string1 = /regular expressions1 you wish to find in malware/
  $reg_exp_string2 = /regular expressions2 you wish to find in malware/

condition:
  $text_string1 or $text_string2 or
  $hex_string1 or $hex_string2 or
  $reg_exp_string1 or $reg_exp_string2
}
```

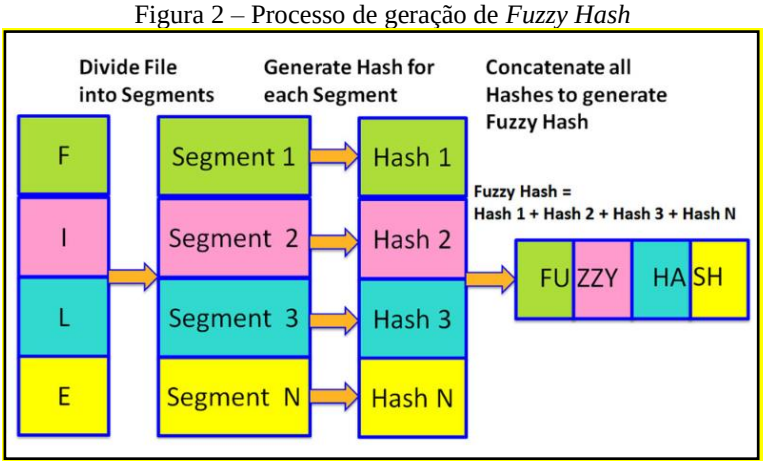
```
rule WannaCry
{
meta:
  description = "Generic Signature of WannaCry"
  author = "Nitin Naik"
  date = "01/06/2018"
  reference = "www.mydomain.com"

strings:
  $text_string1 = "encrypt"
  $text_string2 = "bitcoin"

  $hex_string1 = {B6 D3 56 A5 78 43}
  $hex_string2 = {E8 27 F9 83 C4 82}

  $reg_exp_string1 = /md5: [0-9a-fA-F]{32}/
  $reg_exp_string2 = /state: (on|off)/

condition:
  $text_string1 or $text_string2 or
  $hex_string1 or $hex_string2 or
  $reg_exp_string1 or $reg_exp_string2
}
```

Fonte: Naik *et al.* (2020)

Segundo Naik *et al.* (2020), as *strings* podem ser definidas em formato texto, hexadecimal e expressões regulares. As *strings* em hexadecimal servem para representar *bytes* enquanto *strings* em formato texto costumam representar um texto legível. Uma das funcionalidades adicionadas na versão 2.0 foi a possibilidade de utilizar expressões regulares, compostas por uma *string* convencional com a adição de modificadores, que permitem um *match* dinâmico. Com as *strings* já definidas, é possível criar as condições de *match* que são compostas por possíveis combinações das definições anteriores, semelhante a condições booleanas.

Outra técnica de análise de *Malware* apresentada pelos autores Naik *et al.* (2020) é chamada de *Fuzzy Hashing*. O objetivo de sua utilização é a investigação através da comparação de integridade e similaridade entre arquivos, já que é comum que desenvolvedores de *Malware* reutilizem recursos já implementados em outros *Malwares* previamente desenvolvidos e disponibilizados no mercado. Como qualquer simples alteração de um arquivo gera um *hash* de integridade completamente diferente do original, utilizar *hashing* dessa forma não ajuda a identificar *malwares* semelhantes, apenas idênticos. Para conseguir analisar a similaridade, separa-se o arquivo de interesse em diversos blocos e é calculado um *hash* individual de cada bloco, para que, no final, todos os *hashes* sejam concatenados, gerando assim o *Fuzzy Hash* do arquivo. Para gerá-lo, é possível utilizar os seguintes tipos de métodos: *Context-Triggered Piecewise Hashing* (CTPH), *Statistically Improbable Features* (SIF), *Block-Based Hashing* (BBH) e *Block-Based Rebuilding* (BBR). A Figura 2 exemplifica o processo de geração do *Fuzzy Hash* de um arquivo.

Figura 2 – Processo de geração de *Fuzzy Hash*



Fonte: Naik *et al.* (2020)

Baseando-se nessas duas técnicas de análise de *Malware*, os autores Naik *et al.* (2020) propõem duas novas técnicas. A primeira, chamada de YARA *Enhanced* (YARA aprimorada), busca integrar as duas técnicas previamente apresentadas de uma forma que sejam complementares. O intuito dessa integração é que o índice de similaridade entre amostras, obtido através de um método de *Fuzzy Hashing*, seja utilizado como parâmetro adicional aos *matches* de regras YARA no momento do veredito final da análise, especialmente em casos em que não houve nenhum *match* de regra YARA. Ao final dos testes, combinando ambas as técnicas e variando os métodos de *Fuzzy Hashing*, obteve-se um F1-Score de 79.08% utilizando regras YARA aprimoradas baseadas em SSDEEP, que representa ganho de 3.59% de efetividade de detecção quando comparado com a aplicação exclusiva de regras YARA.

A segunda técnica proposta pelos autores Naik *et al.* (2020) é chamada de *Embedded YARA* (YARA integrada). Essa técnica visa utilizar lógica de Fuzzy (ou lógica difusa) para complementar as condições de *match* convencional de *strings* com condições de *match* de *Fuzzy Hash*. Segundo os mesmos autores, já que a combinação dessas duas condições produz múltiplos resultados, é possível criar regras de lógica difusa para otimizar os resultados. Um exemplo seria criar regras como "Se a quantidade de *matches* YARA é média e o índice de similaridade de *Fuzzy Hash* é baixo, então a amostra é provavelmente *Malware*", "Se a quantidade de *matches* YARA é alta e o índice de similaridade de *Fuzzy Hash* é baixo, então a amostra é muito provavelmente *Malware*" ou "Se a quantidade de *matches* YARA é baixa e o índice de similaridade de *Fuzzy Hash* é baixo, então a amostra tem pouca chance de ser um *Malware*".

De acordo com testes executados pelos autores Naik *et al.* (2020), é possível notar a melhoria do F1-Score a cada nova técnica apresentada. Enquanto as regras YARA básicas possuem F1-Score de 75.49%, as regras YARA aprimoradas apresentam 79.08%, e as regras YARA integradas chegam em um resultado de 83.48%. Apesar dos resultados promissores, é necessário atentar-se a algumas principais limitações como a dependência clara do detalhamento prévio de regra(s) YARA, a falta de capacidade de análise comportamental ou semântica no processo de *Fuzzy Hashing* e a inconsistência da interpretação de resultados de índice de semelhança obtidos por *Fuzzy Hashing*, já que o limiar definido vai variar de acordo com a configuração realizada pelo analista.

2.2 BOTDET: A SYSTEM FOR REAL TIME BOTNET COMMAND AND CONTROL TRAFFIC DETECTION

Os autores Ghafir *et al.* (2018) desenvolveram uma aplicação chamada BotDet que é capaz de atuar na detecção de tráfego de rede malicioso, especialmente *Command and Control* (C&C), de forma semelhante a um *Intrusion Detection System* (IDS). O seu funcionamento ocorre através de duas fases: detecção de técnicas e correlação de alertas. A primeira consiste em utilizar quatro métodos de detecção de técnicas de ataque usadas por *botnets* e comunicações de C&C, enquanto a segunda busca reduzir o número de casos de falso-positivo.

O primeiro módulo de detecção de técnicas executadas por atacantes, utilizado por Ghafir *et al.* (2018), se chama detecção de endereços IP maliciosos (MIPD). O seu intuito é, dentro do fluxo de comunicações de rede, identificar endereços IP e servidores de C&C que já foram identificados previamente como maliciosos por *feeds* de inteligência de terceiros, como organizações especializadas em inteligência de ameaças. Tendo identificado alguma comunicação suspeita que teve *match* na *blacklist*, é feita também uma verificação de histórico para identificar se é algo recorrente, para então poder gerar os alertas necessários e acionar a equipe de segurança.

O segundo módulo implementado por Ghafir *et al.* (2018) se chama detecção de certificado SSL malicioso (MSSLD). Devido às dificuldades de análise de tráfego de rede criptografado, é importante que o próprio certificado SSL seja validado, para garantir que o atacante não está utilizando a criptografia como forma de burlar mecanismos de defesa. Semelhante ao módulo anterior, é feita também uma validação de *match* com *blacklist*. Se foi identificada alguma conexão utilizando um certificado caracterizado como malicioso, é gerado um alerta e

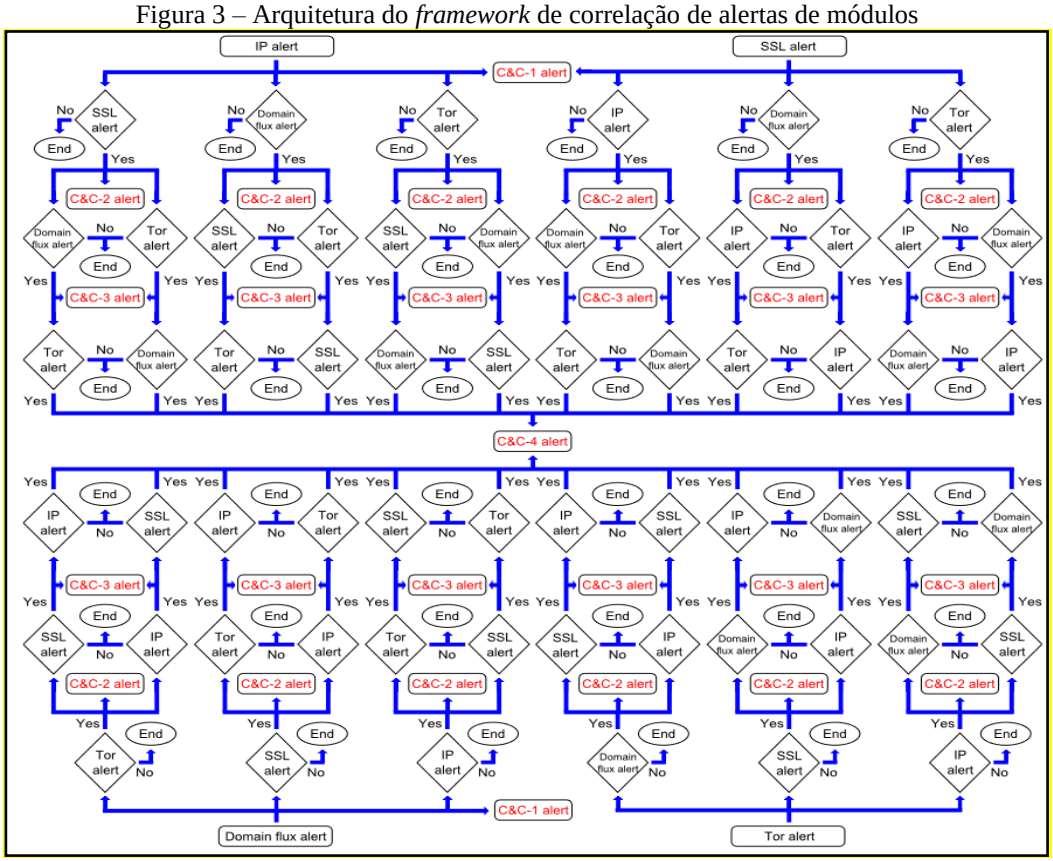
adicionado o *hash* do certificado SSL e o endereço IP a uma tabela que é usada para não gerar eventos recorrentes em um período de 24 horas.

O terceiro módulo utilizado pelos autores Ghafir *et al.* (2018) se chama detecção de fluxo de domínio (DFD). O seu intuito é lidar com a técnica de fluxo de domínio que utiliza um algoritmo de geração de domínios (DGA) em máquinas infectadas para criar queries DNS e conectar a diversos nomes de domínio gerados, que eventualmente redirecionam para os servidores de C&C do atacante, o que dificulta o controle por *blacklist*. Para lidar com isso, o DFD busca realizar uma busca por falhas de *queries* DNS, que normalmente estão presentes em alta quantidade em dispositivos infectados. Com a detecção de uma quantidade especificada destas falhas, é feita uma análise dos eventos e então é gerado um alerta a respeito do evento malicioso.

O quarto e último módulo desenvolvido por Ghafir *et al.* (2018) se chama detecção de conexão Tor (TorCD). Buscando privacidade e confidencialidade, é comum que atacantes utilizem a rede Tor para redirecionar o tráfego de rede da máquina infectada através de um circuito de retransmissores, garantindo que nenhuma parte do fluxo conhecerá o caminho completo até o servidor do atacante. O intuito é que o módulo TorCD detecte qualquer comunicação com redes Tor através de uma lista pública de servidores Tor. Como a rede Tor praticamente não é utilizada por aplicações convencionais, o módulo busca gerar os alertas assim que obtém um *match* de conexão com rede Tor.

Além dos módulos para detecção de técnicas de ataque previamente apresentados, Ghafir *et al.* (2018) desenvolveram também um *framework* de correlação (CF), que, dentro de um período definido de 24 horas, consegue gerar diferentes tipos de alertas, baseando-se na quantidade de ocorrências deste período. Se forem identificados três alertas de um mesmo *host*, por exemplo, será gerado um alerta C&C-3. Para evitar falsos-positivos do módulo TorCD, a única exceção de geração de alerta definida é para os casos de alerta C&C-1 (1 evento), que não considera um único evento de conexão Tor como algo malicioso, já que utilizar rede Tor não é um comportamento malicioso por natureza e pode representar um serviço legítimo. A lógica de correlação dos alertas gerados pelos módulos e a exceção aplicada a alertas C&C-1 do módulo TorCD podem ser visualizadas na figura 3.

Figura 3 – Arquitetura do *framework* de correlação de alertas de módulos



Fonte: Ghafir *et al.* (2018)

Para realizar os testes da aplicação BotDet, desenvolvida por Ghafir *et al.* (2018), foram usados três cenários. No primeiro, são utilizados arquivos PCAP (agregados de pacotes capturados em uma rede) de terceiros, que contém diversos casos de comunicação de *Malwares*, além de comunicação legítima. Após serem fornecidos para a aplicação, foi observado que a melhor relação entre casos positivos e casos negativos foi obtida através da correlação de dois módulos de detecção, totalizando 82.3% de taxa positiva verdadeira (TPR) e 13.6% de taxa negativa verdadeira (FPR). No segundo caso, foi construída uma rede virtual conectada à Internet e, em seguida,

foram injetadas amostras de *Malware*. Após a captura do tráfego e utilização da aplicação BotDet, nota-se um resultado semelhante ao anterior com um TPR de 79% e FPR de 16.8%. O terceiro caso consiste na análise do tráfego de um *campus* pelo período de um mês, apesar de que não são informadas as taxas de detecção.

Com esses estudos práticos, é possível notar que a aplicação possui uma taxa de sucesso eficiente que é capaz de auxiliar as equipes de segurança da informação na priorização de alertas de possível tráfego malicioso. Apesar disso, é necessário destacar a dependência quase absoluta da aplicação de ter *feeds* de inteligência de terceiros que trazem as *blacklists* e informações de reputação de servidores, o que significa que a eficácia está atrelada à qualidade das informações obtidas e que há a necessidade de a base da aplicação ser constantemente atualizada.

2.3 DESIGNING IN-VM-ASSISTED LIGHTWEIGHT AGENT-BASED MALWARE DETECTION FRAMEWORK FOR SECURING VIRTUAL MACHINES IN CLOUD COMPUTING

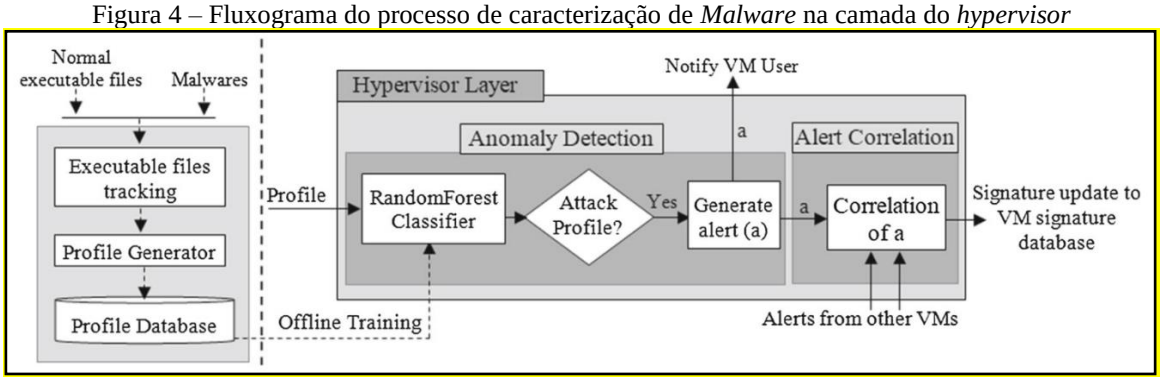
Buscando desenvolver um *framework* de detecção de *Malware* capaz de se adequar às demandas geradas pela crescente adoção de infraestruturas em nuvem, os autores Patil *et al.* (2019) optaram por utilizar uma estratégia composta por dois principais componentes: um agente presente na própria máquina virtual e um módulo de detecção de anomalia a nível de *hypervisor*. A ideia principal é que o agente faça a análise de assinaturas maliciosas no sistema operacional e o módulo de segurança baseado no *hypervisor* fará o monitoramento de múltiplas máquinas virtuais e a correlação de eventos.

No desenvolvimento do agente, Patil *et al.* (2019) reforçam que, devido às diferenças de uso de cada máquina virtual e o constante desenvolvimento de novas ameaças por parte de atacantes, há a necessidade de que o banco de assinaturas do agente seja constantemente atualizado. Como o agente focará exclusivamente na análise estática dos arquivos, ou seja, não terá testes executados através da execução do arquivo em questão, é de extrema importância que o maior escopo possível de assinaturas esteja disponível. Com isso, portanto, é possível garantir que o *Malware* não conseguirá executar nenhuma ação maliciosa enquanto está sendo analisado, especialmente em casos de *Ransomware*.

O agente que é instalado nas máquinas virtuais consiste em três submódulos: um rastreador de arquivos executáveis, um módulo de detecção baseado em assinaturas e um gerador de perfis. A utilização dessas funcionalidades, definidas por Patil *et al.* (2019), pode ser representada como um fluxo de análise. Inicialmente, o rastreador busca identificar a existência ou entrada de qualquer arquivo executável na máquina virtual. A partir da identificação, o executável é enviado para a *Application Programming Interface* (API) da plataforma VirusTotal para que seja analisado por um banco de assinaturas maliciosas de diversas fontes de inteligência. Quando há um *match* do arquivo com a assinatura de algum programa malicioso já identificado, o agente o considera como *Malware* e gera um alerta para o usuário da máquina virtual e para o *hypervisor*, que fará a correlação de outros eventos para identificar possíveis ataques distribuídos na rede. Nos casos em que não é identificado como *Malware*, o executável é processado através do gerador de perfis, que utiliza a ferramenta Pefile, cuja funcionalidade é desmontar o arquivo para que sejam extraídas as características como cabeçalhos e seções, que serão então usados para montar um perfil do executável.

Além da correlação de eventos de múltiplas máquinas virtuais para buscar identificar um possível ataque de maior escopo ou movimento lateral do atacante na rede, os autores Patil *et al.* (2019) buscam detectar anomalias através dos perfis previamente gerados por cada máquina virtual. Esse processo consiste em utilizar um classificador Random Forest (floresta aleatória) que foi treinando anteriormente tanto com amostras de executáveis legítimos quanto *Malwares* já conhecidos. Nos casos em que a previsão do classificador indica que é um executável malicioso, é gerada uma assinatura que é incluída na base de dados consultada por todas as máquinas virtuais e é enviado um alerta para o submódulo de correlação de eventos. O fluxo do sistema de detecção de *Malware* proposto pode ser observado na Figura 4.

Figura 4 – Fluxograma do processo de caracterização de *Malware* na camada do *hypervisor*



Fonte: Patil *et al.* (2019)

Para testar o *framework* proposto, os autores Patil *et al.* (2019) utilizaram três servidores físicos, onde cada um possui um *hypervisor* VMware Esxi que é usado para virtualizar três máquinas virtuais com Windows 7 ou Ubuntu 14.04. Após a etapa de *deployment* dos *softwares* necessários para a execução do *framework*, são implantadas diversas amostras de *Malware* (Trojan, vírus, *backdoor, adware, worm, ransomware, riskware, rootkit, spyware,* entre outros) e executáveis legítimos nas máquinas virtuais para que sejam gerados os perfis no classificador Random Forest. Para a etapa de treinamento anterior do classificador, foram utilizados *datasets* de *Malware* recentes disponíveis na plataforma Kaggle. Baseando-se nesta estrutura de testes, o *framework* foi capaz de chegar em uma acurácia de classificação de *Malware* de mais de 98%, sendo necessários, aproximadamente, apenas 8.5 microssegundos para analisar um perfil de executável. Esse resultado muito positivo, em conjunto à facilidade de instalação e utilização prática dos componentes do *framework*, são pontos que comprovam a eficácia das técnicas propostas para cenários reais de *deployment* de soluções de segurança para computação em nuvem. Apesar do resultado, contudo, ainda há dificuldade de analisar *Malwares* mais complexos que são descriptografados apenas quando estão em execução, como alguns exemplos de *Ransomware*. Para esses casos, seria necessário implementar uma outra camada focada em técnicas de análise dinâmica como monitoramento de chamadas de sistema operacional.

3 PROPOSTA DO SOFTWARE

A seguir é apresentada a justificativa para a elaboração desse trabalho, os principais requisitos do sistema que será desenvolvido e a metodologia a ser empregada. Também serão aprestadas revisões bibliográficas para entendimento fundamental do trabalho.

3.1 JUSTIFICATIVA

O Quadro 1 busca identificar as características presentes nos trabalhos correlatos e como elas se diferem. As linhas representam as características, enquanto cada coluna corresponde a um trabalho correlato.

Quadro 1 - Comparativo dos trabalhos correlatos

Trabalhos Correlatos / Características	Naik *et al.* (2020)	Ghafir *et al.* (2018)	Patil *et al.* (2019)
Objeto analisado	Arquivo / binário	Tráfego de rede	Arquivo / binário
Tipo de análise	Estática	Dinâmica	Estática
Instalação de agente	Não necessário	Necessário	Necessário
O que é proposto	Técnica	Aplicação final	Aplicação final
Objeto correlacionado	Detecções YARA e índice de similaridade	Alertas de conexão suspeita	Perfil do arquivo
Uso de aprendizado de máquina	Nenhum	Nenhum	Random Forest
Taxa de detecção correta	83.48%	82.3%	98%

Fonte: elaborado pelo autor.

Conforme as informações do Quadro 1, nota-se que os trabalhos de Naik *et al.* (2020) e Patil *et al.* (2019) focam na identificação de indícios maliciosos em arquivos, enquanto o trabalho de Ghafir *et al.* (2018) realiza as análises a nível de tráfego de rede. Devido a essas características, é possível perceber que os escopos de análise também diferem da mesma forma entre os trabalhos. Os que realizam a detecção de arquivos maliciosos conseguem produzir a análise apenas sem a execução do binário (análise estática), enquanto o trabalho focado na análise de tráfego de rede pode utilizar tráfego que já ocorreu ou tráfego em tempo real (análise dinâmica).

Outra característica relevante é que os trabalhos dos autores Ghafir *et al.* (2018) e Patil *et al.* (2019) representam uma aplicação/ferramenta completa e implementada e, devido às estratégias de coleta de informação que foram adotadas em cada trabalho, ambas requerem que seja instalado um agente ou cliente que será responsável pela identificação de indícios maliciosos e pré-processamento do evento. O trabalho de Naik *et al.* (2020), por outro lado, não chega a desenvolver um agente ou aplicação em si, já que foca no detalhamento específico das técnicas propostas que podem ser utilizadas para detectar *Malwares*.

Uma semelhança comum a todos os trabalhos correlatos apresentados é o fato de utilizarem técnicas de correlação de eventos ou alertas como forma de aumentar sua eficácia e assertividade, uma vez que, ao correlacionar, é possível identificar se um possível ataque abrange um escopo maior ou se houve movimentação lateral na rede por parte do atacante. Apesar de compartilharem essa característica, cada grupo de autores opta por utilizar uma técnica diferente. Os autores Naik *et al.* (2020) buscam correlacionar a assertividade das detecções de regras YARA com o índice de semelhança entre amostras, obtendo assim, através da utilização de lógica difusa, um entendimento melhor de risco do arquivo em questão. Outra estratégia, que foi utilizada por Ghafir *et al.*

(2018), é considerar a quantidade de detecções de técnicas possivelmente maliciosas em um mesmo *host* nas últimas 24 horas e gerar um alerta com criticidade correspondente. Por fim, os autores Patil *et al.* (2019) buscam inicialmente obter o perfil dos arquivos que não tiveram *match* com nenhuma assinatura maliciosa. Já com a estrutura do perfil do arquivo pronta, composta por elementos como cabeçalhos e seções, é feita a classificação desse perfil através de um classificador já preparado, que utiliza o algoritmo de aprendizado de máquina chamado de Random Forest (floresta aleatória), o que é um diferencial desse trabalho e contribui para a sua alta eficácia.

Nos testes realizados, todos os trabalhos conseguiram obter um bom desempenho e detectar as ameaças com eficácia. Utilizando as técnicas propostas por Naik *et al.* (2020) no cenário de testes correspondente, foi obtida uma taxa de detecção de 83.48%. Os autores Ghafir *et al.* (2018) conseguiram uma taxa semelhante na análise dinâmica de tráfego de rede, alcançando 82.3%. O destaque, porém, vai para Patil *et al.* (2019), que conseguiram atingir uma taxa de 98% ao combinar as assinaturas de *Malware* já conhecidos com o seu modelo baseado em Random Forest para as análises de arquivos desconhecidos.

A partir das características descritas acima e da comparação dos estudos mencionados, este trabalho mostra-se relevante pois pretende desenvolver uma aplicação para detecção de *Malware* que combina elementos da análise estática com possibilidade de utilização de um agente/cliente para análise dinâmica em um ambiente controlado. A aplicação será construída em um modelo cliente-servidor para interface com usuário que permitirá a checagem de arquivos por assinaturas e outras técnicas de análise estática, além de possibilitar controle sobre as bases de assinaturas utilizadas e disponibilizar interface para integração com inteligência de terceiros. Para o processamento em *backend*, será montada uma infraestrutura de microsserviços. A aplicação também permitirá a coleta de informações de tráfego de rede e ações executadas pelo *Malware* a partir do agente previamente mencionado. Essas informações obtidas em *runtime* também serão correlacionadas e utilizadas como critério para a classificação final do binário.

Com isso, espera-se que a aplicação final desenvolvida seja capaz de obter uma boa acurácia e permita que o usuário ou a equipe de analistas de cibersegurança tenha maior segurança ao acreditar no veredito do arquivo apresentado, contribuindo assim para a segurança do usuário ou do ambiente empresarial como um todo. A possibilidade de combinar análise estática e dinâmica, juntamente com a flexibilidade de personalização do banco de assinaturas a ser utilizado e a integração com fontes de inteligência externas, busca fortalecer a capacidade de identificação, de acordo com o tipo de análise optada e a quantidade/qualidade das fontes de inteligência de ameaças escolhidas.

3.2 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Os requisitos do trabalho estão divididos em duas partes principais: requisitos da plataforma principal e requisitos do agente de análise dinâmica.

A plataforma principal deverá:

a) manter o cadastro de usuários (Requisito Funcional – RF);
b) manter o registro de integrações realizadas com inteligência de ameaças de terceiros (RF);
c) manter o registro de histórico das análises realizadas (RF);
d) manter o repositório de assinaturas atualizado (RF);
e) integrar com os microsserviços que processarão as etapas de análise estática (RF);
f) permitir conexão e integração com o agente de análise dinâmica (RF);
g) permitir conexão com um serviço de TAXII Feed, responsável pela obtenção de inteligência de ameaças de terceiros (RF);
h) utilizar as melhores práticas de segurança no desenvolvimento (Requisito Não Funcional – RNF);
i) utilizar a linguagem Javascript/Typescript no *framework* Node.js para desenvolvimento (RNF);
j) utilizar o ambiente de desenvolvimento Visual Studio Code (RNF);
k) armazenar os dados em um banco de dados SQLite (RNF);

O agente de análise dinâmica deverá:

a) comunicar-se com a plataforma principal para repasse das informações coletadas (RF);
b) fazer *sniffing* de rede para identificação de indícios maliciosos (RF);
c) analisar comportamentos maliciosos no ambiente em que está instalado (RF);
d) manter o registro de integrações realizadas com inteligência de ameaças de terceiros (RF);
e) utilizar a linguagem C# para desenvolvimento da interface (RNF);
f) utilizar a linguagem C++ para realizar as análises necessárias (RNF);
g) utilizar o ambiente de desenvolvimento Visual Studio (RNF);

3.3 METODOLOGIA

O trabalho será desenvolvido observando as seguintes etapas:

a) levantamento bibliográfico: buscar fontes bibliográficas relacionadas ao intuito desse trabalho, com foco especial nas técnicas de detecção de *Malware* em análise dinâmica;
b) elicitação de requisitos: através das novas informações obtidas na etapa anterior, avaliar a necessidade de atualização dos requisitos propostos;
c) especificação: utilizar uma ferramenta de diagramação para elaborar modelos de caso de uso e a arquitetura de microsserviços;
d) desenvolvimento inicial da aplicação Web: iniciar o desenvolvimento da aplicação principal com foco na possibilidade de envio de arquivo para análise;
e) desenvolvimento de microsserviços de análise estática: implementar os microsserviços, como aplicador de regras YARA, calculador de índice de similaridade por *Fuzzy Hash*, analisador de características do arquivo, entre outros, que serão utilizados pela aplicação principal para fazer a análise estática a partir do arquivo recebido;
f) implementação do atualizador de assinaturas de *Malware*: desenvolver atualização automática do repositório de inteligência de ameaças da aplicação;
g) desenvolvimento do serviço de integração com TAXII Feed: implementar um cliente TAXII;
h) desenvolvimento da interface e conexão do agente: implementar a interface simples do agente e criar métodos de comunicação com a aplicação principal;
i) desenvolvimento do módulo de *sniffing* de rede do agente: criar lógica de captura de tráfego de rede do *host* e análise de indícios maliciosos;
j) desenvolvimento do módulo do agente de análise de comportamento suspeito: implementar técnicas de identificação de comportamentos de *Malware*;
k) testes de integração: validar se as comunicações entre a plataforma principal e o agente estão funcionando conforme esperado;
l) aprimoramento de interface da aplicação principal: tornar a interface mais amigável ao usuário e com funcionalidades mais completas;
m) testes de detecção: validar acurácia de detecção de tipos diversos de *Malware*, variando a quantidade de inteligência de ameaça de terceiros utilizada e a utilização do agente para análise dinâmica;

As etapas serão realizadas nos períodos relacionados no Quadro 2.

Quadro 2 - Cronograma

	2024									
	fev.		mar.		abr.		maio		jun.	
etapas / quinzenas	1	2	1	2	1	2	1	2	1	2
levantamento bibliográfico										
elicitação de requisitos										
especificação										
desenvolvimento inicial da aplicação Web										
desenvolvimento de microsserviços de análise estática										
implementação do atualizador de assinaturas de *Malware*										
desenvolvimento do serviço de integração com TAXII Feed										
desenvolvimento da interface e conexão do agente										
desenvolvimento do módulo de *sniffing* de rede do agente										
desenvolvimento do módulo do agente de análise de comportamento suspeito										
testes de integração										
aprimoramento de interface da aplicação principal										
testes de detecção										

Fonte: elaborado pelo autor.

4 REVISÃO BIBLIOGRÁFICA

Este capítulo descreve brevemente sobre os assuntos que fundamentarão o estudo a ser realizado: tipos de *Malware*, compartilhamento de inteligência de ameaças e *sniffing* de rede.

Segundo Tahir (2018), *Malwares*, ou *softwares* maliciosos, existem de diversas formas e podem ser categorizados em classes, apesar de não pertencerem exclusivamente a apenas uma. Inicialmente tem-se os vírus, que são capazes de infectar computadores e outros arquivos. Sua propagação é feita através de sua anexação a executáveis e aplicações legítimas e busca se espalhar por arquivos e outros *hosts* em uma rede. A partir do momento que se tem uma replicação automática e o código malicioso é independente, o *Malware* passa a ser categorizado como *Worm*. Esse tipo de ameaça costuma se propagar automaticamente por dispositivos de armazenamento ou e-mails. Uma outra estratégica de infecção de *Malwares* é adotada pela classe de *Trojan Horse*, que busca se mascarar como um programa legítimo útil, mas com ações maliciosas embutidas. Uma outra categoria que ganha destaque pelo seu risco se chama *Rootkit*. Esse tipo de *Malware* é capaz de tomar controle de todo o

sistema operacional e, devido aos privilégios obtidos, consegue burlar o antivírus e criar um ambiente seguro para outros tipos de ameaças, o que significa que, muitas vezes, é necessário utilizar um dispositivo externo para analisar o disco de *boot* e remover um *Rootkit*. Além disso, existem categorias que buscam apenas coletar informações sobre o *host* ou usuário, como *Spyware*, que busca roubar informações pessoais e dados de atividades, *Sniffers*, que buscam roubar informações sobre o tráfego de rede e *Keyloggers*, que visam coletar todas as teclas pressionadas por um usuário, em busca de dados sensíveis. Entre as categorias de *Malware*, a que mais representa risco para indústrias é *Ransomware*. Esse tipo de *Malware* toma controle do *host* e busca se espalhar o máximo possível na rede e, em cada dispositivo infectado, todos os dados são criptografados, o que impede o uso convencional e costuma impedir o funcionamento do negócio. Esse senso de urgência é o que torna essa categoria extremamente rentável, já que quem é infectado pode optar por pagar um resgate dos dados através da obtenção de uma chave de descriptografia, apesar de não ser garantindo que o malfeitor devolverá o acesso ao conteúdo criptografado após receber o pagamento.

Wagner *et al.* (2019) definem que o objetivo do compartilhamento de inteligência de ameaças, ou Cyber Threat Intelligence (CTI), é criar conhecimento situacional entre as partes interessadas para que estejam preparadas para eventuais ataques atuais e futuros. A partir do momento em que as organizações passam a se preocupar em agir de forma proativa ao invés de reativa, há interesse claro em estar recebendo as informações de novas ameaças, vulnerabilidades e ataques para que as remediações já sejam aplicadas. Esse tipo de informação normalmente é compartilhado por empresas fabricantes de soluções de segurança, pesquisadores independentes e até empresas convencionais após terem sido vítimas de algum ataque. As informações que são compartilhadas podem conter, por exemplo, descrição do atacante, campanhas de *Malware* que ocorreram ou ainda ocorrem, motivações de grupos atacantes e indicadores de compromisso (IoC), que podem abranger aspectos como endereços IP maliciosos que estiverem envolvidos em um ataque ou *hashes* de arquivos que fazem parte do ataque.

Segundo Sikos (2020), *sniffing* de pacotes é um método de capturar uma cópia de pacotes presentes em um tráfego de comunicação de rede. Esses dados de rede podem ser analisados e segregados utilizando *softwares* dedicados, como *sniffers* de pacote ou *sniffers* de rede. A eficácia desse tipo de ferramenta se dá, pois, pacotes de rede são mais do que apenas dados de comunicação e metadados, uma vez que, ao serem reestruturados, podem representar um arquivo completo que foi trafegado. Essa possibilidade, em conjunto à análise de conteúdo de pacotes e seus protocolos, é uma das principais ferramentas utilizadas para rastreio de informações e arquivos em investigações a nível de rede.

REFERÊNCIAS

GHAFIR, Ibrahim; PRENOSIL, Vaclav; HAMMOUDEH, Mohammad; BAKER, Thar; JABBAR, Sohail; KHALID, Shehzad; & JAF, Sardar (2018). **BotDet: A System for Real Time Botnet Command and Control Traffic Detection**. IEEE Access, 1–1. doi:10.1109/access.2018.2846740

LIU, Xiang; AHMAD, Sayed F.; ANSER, Muhammad K.; KE, Jingying; IRSHAD, Muhammad; UL-HAQ, Jabbar; ABBAS, Shujaat. (2022). **Cyber security threats: A never-ending challenge for e-commerce**. Front. Psychol. 13:927398. DOI: 10.3389/fpsyg.2022.927398

MUGHAL, Arif A. **The Art of Cybersecurity: Defense in Depth Strategy for Robust Protection.** International Journal of Intelligent Automation and Computing, V. 1, n. 1, p. 1–20, 2018. Disponível em: https://research.tensorgate.org/index.php/IJIAC/article/view/19. Acesso em: 30 sep. 2023.

NAIK, Nitin; JENKINS, Paul; SAVAGE, Nick; YANG, Longzhi; BOONGOEN, Tossapon; IAM-ON, Nnatthakan; NAIK, Kshirasagar; SONG, Jingping (2020). **Embedded YARA rules: strengthening YARA rules utilising fuzzy hashing and fuzzy rules for malware analysis**. Complex & Intelligent Systems. DOI:10.1007/s40747-020-00233-5

PATIL, Rajedra; DEDEJA, Harsha; & MOD, Chirag (2019). **Designing in-VM-assisted lightweight agent-based malware detection framework for securing virtual machines in cloud computing**. International Journal of Information Security. doi:10.1007/s10207-019-00447-w

SIKOS, Leslie F. (2020). **Packet analysis for network forensics: A comprehensive survey**. Forensic Science International: Digital Investigation, 32, 200892. doi:10.1016/j.fsidi.2019.200892

TAHIR, Rubia (2018). **A Study on Malware and Malware Detection Techniques**. Modern Education and Computer Science Press: I.J. Education and Management Engineering, 2018, 2, 20-30. DOI: 10.5815/ijeme.2018.02.03

WAGNER, Thomas D.; MAHBUB, Khaled; PALOMAR, Esther; & ABDALLAH, Ali. E. (2019). **Cyber threat intelligence sharing: Survey and research directions**. Computers & Security, 87, 101589. DOI: 10.1016/j.cose.2019.101589