

# **Técnicas Ultraleves Para Detecção de Malwares Baseada Em Assinaturas Para Redes de Computadores**

**Lucas Thiago Assumpção Gouvêa**

*Universidade Estadual Paulista Júlio de Mesquita Filho, lucas.thiago@live.com*

**Kelton Augusto Pontara da Costa**

*Universidade Estadual Paulista Júlio de Mesquita Filho, kelton@fc.unesp.br*

## **Resumo**

Desde as origens da Internet, existe uma grande preocupação com a privacidade das comunicações e com a prevenção de danos causados por ataques que se aproveitam de vulnerabilidades nas redes corporativas e públicas. Com o aumento constante do uso de dispositivos conectados à internet e da intensidade e variedade de dados transmitidos entre as pessoas, o número de falhas de segurança em redes ainda é preocupante e tende a continuar subindo. Contudo, os pacotes de alguns programas utilizados nesses ataques, conhecidos como Malware, possuem aspectos que são rastreáveis e que, quando incorporados aos métodos de busca dos sistemas de segurança, podem ajudar a detectar esses programas maliciosos antes que eles causem dano a outros pontos da rede infectada. A esses aspectos damos o nome de assinaturas de Malware e o objeto de estudo deste trabalho são as técnicas de captura de Malware que envolvem assinaturas, novas ferramentas do segmento e as tendências e que estão se apresentando no cenário tecnológico para esse tipo de abordagem. O trabalho propõe também a implementação de um ambiente de aplicação e testagem de assinaturas de Malware sobre um conjunto de programas contendo traços de código malicioso, com o auxílio de ferramentas modernas que também são utilizadas por grandes empresas de segurança digital e desenvolvimento, além de um protótipo de webservice para testagem de assinaturas de Malware e verificação de arquivos online.

**Palavras-chave:** Técnica Ultraleve. Segurança. Malwares. Assinatura.

---

## ***Lightweight Techniques For Signature-Based Malware Detection In Computer Networks***

### ***Abstract***

Since the Internet's origins, there's a great preoccupation with the privacy of communications and with preventing damage caused by attacks that exploit vulnerabilities in corporative and private networks. With the increase in usage of internet connected devices and of the intensity and variety of transmitted data between people, the number of security flaws in networks is still worrisome and tends to keep growing. However, packages of some programs used in those attacks, known as Malware, have traceable aspects that when incorporated to the search methods of security systems, might help detecting them before causing damage to other points of the infected network. To those aspects, we give the name of Malware signatures, and the study object of this work are the Malware capturing techniques involving signatures, new tools in the segment and the trends showing up in the technologic scenario to this kind of approach. The work also proposes the implementation of an environment to apply and test Malware signatures on a set of programs containing traces of malicious code, with the help of modern tools that are also being used by big companies of digital security and development, and also a Web application prototype for online Malware signature testing and file verification.

**Keywords:** *Lightweight Technique. Security. Malwares. Signature.*

---

## 1 Introdução

A presença das redes de computadores no cotidiano vem aumentando de modo muito veloz há anos e apresentou um crescimento em especial com o advento das redes sem fio e da produção massiva e a queda do custo de dispositivos portáteis capazes de se conectarem à internet. De maneira proporcional, aumentaram-se também os riscos relacionados à segurança de redes e à privacidade de informações de diversas espécies que trafegam por entre elas e seus usuários.

Numa proposição inicial, este trabalho define como Malware todo tipo de programa que seja capaz de alterar o comportamento de um software num dispositivo conectado a uma rede qualquer, feito com o desejo de vender informações de usuários, causar desordem, roubar credenciais, enviar spams, alimentar motores de otimização de busca (SEO) com informações falsas sobre visitas a páginas da Web ou ainda de chantagear usuários, como foi o caso do Malware japonês denominado Kenzero, que publicava históricos de navegação juntamente com a identificação de seus usuários, a menos que os mesmos pagassem 1500 ienes para que seus dados fossem retirados do acesso ao público (SAWLE et al., 2014).

De acordo com IDIKA (2007), pode-se classificar os tipos de métodos de detecção de maneira mais rudimentar em duas categorias: detecção baseada em anomalia e detecção baseada em assinatura. A detecção por anomalia é uma técnica que consiste em treinar um algoritmo de detecção para que ele “saiba” o que constitui o comportamento normal de um programa, para que ele possa decidir sobre a periculosidade dos programas a serem inspecionados. Essa mesma abordagem também dá origem a uma técnica muito similar conhecida como detecção baseada em especificações, segundo a qual especificações e conjuntos de regras para que a partir daí defina-se o comportamento de um programa normal. A técnica de detecção por assinaturas, por sua vez, envolve a caracterização do que já seria sabidamente malicioso para encontrar traços semelhantes e encontrar os programas anômalos.

Todas as técnicas citadas acima podem ser aplicadas com um de três modos diferentes, sendo eles análise estática, análise dinâmica e análise híbrida, onde a análise estática num contexto de detecção por assinaturas, verificaria aspectos estruturais de programas inspecionados, como sequências de bytes, e a análise dinâmica buscaria informações de maneira concorrente ao tempo de execução do programa. Em suma, a análise estática procura investigar o programa antes que ele seja executado, e a análise dinâmica atua ou na execução do programa, ou depois que ele já foi executado. As técnicas híbridas são simplesmente combinações dessas duas anteriores, juntando informações estáticas e dinâmicas para que se conclua um parecer sobre a maliciosidade dos códigos analisados.

Os dispositivos móveis atuais ainda não possuem capacidade computacional disponível para que se façam varreduras constantes que utilizem as técnicas de detecção dinâmicas em seus respectivos sistemas à procura de código malicioso, e com a abrangência da internet se tornando cada vez mais expressiva, o dano em potencial que um Malware pode causar a um determinado indivíduo, ou ainda em grupos massivos, também se torna mais preocupante.

Sob a ótica de Szewczyk (2012), as redes de computadores, principalmente sem fio, começam a apresentar problemas de vulnerabilidade a partir do momento da compra do hardware necessário para implementação da rede e da configuração desse equipamento, onde muitas vezes os vendedores destes produtos tentam passar a ideia de que os equipamentos por si são capazes de blindar uma rede contra qualquer intrusão, a fim de obter sucesso comercial valendo-se da ingenuidade de usuários leigos no assunto de segurança de redes. As técnicas de detecção por assinatura, apesar de não serem as mais eficazes à disposição, são muito mais simples de se implementar e resolverem o problema da infecção por *Malwares* conhecidos ou ainda *Malwares* novos com características semelhantes às dos conhecidos.

Com base nas abordagens descritas, o presente trabalho propõe analisar o resultado da ação de diferentes técnicas de detecção de *Malware* baseadas em assinatura em um ambiente de testes para obter métricas relevantes quanto à eficiência nas varreduras realizadas, número de falsos positivos encontrados e se possível determinar qual delas é a mais apropriada para uso em ambientes mobile e de redes de computadores em geral.

O restante do trabalho está organizado da seguinte forma. A Seção 2 encontra-se o referencial teórico que permeia os conceitos em segurança de redes, famílias de *Malware* e demais conceitos relacionados ao projeto. A Seção 3 estão as informações sobre as ferramentas utilizadas na fase de implementação, documentada na Seção 4. A Seção 5 conclui o texto as algumas considerações sobre resultados obtidos e estudos futuros em vários aspectos do projeto.

## 2 Malware

O conceito de *Malware* já foi abordado de maneira mais rudimentar na introdução do trabalho. Porém, como o viés do projeto caminha muito próximo de classificações mais profundas desse tópico, faz-se necessário recorrer a uma bibliografia mais completa sobre os processos relacionados ao *Malware*.

Na pesquisa desenvolvida por SAHEED (2013), pode-se aproveitar muitas informações sobre o patamar do desenvolvimento deste tipo de software. De acordo com pesquisas levantadas pela empresa desenvolvedora de antivírus *McAfee*, esses programas continuam a se tornar mais numerosos dia-a-dia, sendo encontrados novos “indivíduos” aos milhares. Em contrapartida, pesquisadores e laboratórios especializados em segurança aprimoram novos métodos para construir anti-*malware* mais eficiente e mais autônomo.

A seguir, diferenciam-se os tipos de *Malware* com base nas técnicas de criação dos mesmos, separando-os por técnicas de ofuscamento, métodos de invocação, plataforma de atividade, abrangência e técnicas de propagação. Nem todo software malicioso é executado de dentro de máquinas infectadas; atualmente a maioria dos ataques são originários da internet, pois as vulnerabilidades encontradas são mais proeminentes do que em ambientes de rede local, em virtude de a internet ser um espaço, de modo geral, público, beneficiando atacantes que tenham meios de acessar um sistema alvo remotamente. Um sistema de detecção de *Malware* tem majoritariamente duas tarefas: busca e análise, ou seja, saber encontrar material não confiável e saber identificar com precisão que o material em questão realmente possui traços nocivos de código, ou faz com que algum outro programa apresente comportamento considerado anormal.

Para *Malware* baseado em rede bem como para dar início à classificação dos tipos de *Malware*, primeiro observa-se com destaque aqueles que se utilizam da rede como meio de propagação e de atuação. Nesse segmento, é possível identificar o que são os *Adwares*, *Spywares*, *Cookies*, *Backdoors*, *Trojans*, *Sniffers*, *Spam* e *Botnet*.

O *Malware* comum, em contraste com o de rede, é feito para ser executado em ambiente local sem a necessidade de qualquer conexão à rede, pois também pode ser propagado via dispositivos físicos como *pendrives*, HDs externos e periféricos com software malicioso embarcado. Nessa categoria, classificam-se os vírus, *worms* e bombas lógicas.

Dadas essas classificações, pode-se ilustrar, portanto, os tipos de *malware* e suas características através do quadro 1:

**Quadro 1:** Famílias de *Malware* e fatores comparativos

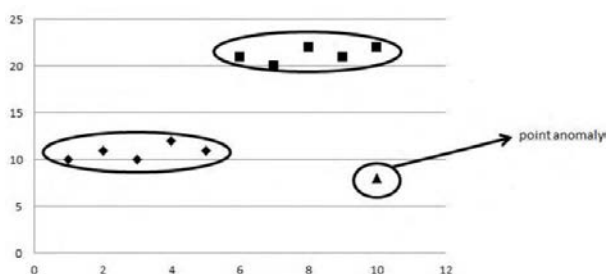
Fatores de comparação		Família do Malware											
		Spyware	Adware	Cookies	Trapdoor	Trojan	Sniffers	Spam	Botnet	Logic bomb	Worm	Virus	
Técnicas de Criação	Padrão	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ofuscado	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Polimórfico	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Kit de Ferramentas(toolkit)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Ambiente de execução	Rede	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗
	Execução remota pela web	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗
	PC	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓
Mídia de propagação	Rede	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Discos removíveis	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Downloads	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Impactos negativos	Brecha de confidencialidade	✓	✗	✓	✗	✓	✓	✗	✗	✗	✗	✗	✗
	Incômodo aos usuários	✗	✓	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗
	Negação de serviços	✗	✗	✗	✓	✗	✗	✓	✓	✓	✓	✓	✓
	Corrupção de dados	✗	✗	✗	✓	✗	✗	✓	✓	✓	✓	✗	✓

Fonte: Saeed (2013)

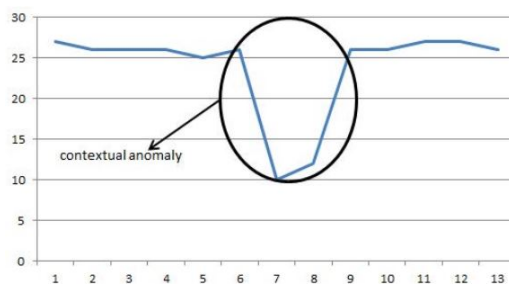
### 3 Detecção de Anomalias

Nesta seção, é exposto o estado atual das tecnologias de detecção e para onde elas estão caminhando no futuro. O trabalho de Baddar et al. (2014) nos mostra uma perspectiva bastante atual do cenário e de como funcionam, de modo geral, os processos de detecção das anomalias. Um sistema de detecção pode agir em ambiente de rede, verificando as características do tráfego da rede, ou em nível de aplicação, onde vai varrer os dados do disco a procura de assinaturas conhecidas de anomalia ou de comportamentos fora do normal. A detecção por comportamento anômalo precisa ser capaz de filtrar casos anômalos isolados, denominados como anomalias pontuais, de casos com suspeita mais palpável de anomalia, que pode-se chamar de anomalia de contexto. Uma anomalia pontual pode ser exemplificada do seguinte modo: uma tentativa de acesso de um usuário num sistema corporativo não é, a princípio, um ato que possa ser classificado como anômalo; porém, se essa mesma tentativa se repete várias vezes em horários não condizentes com o expediente comercial, pode haver alguém suspeito tentando roubar as credenciais desse usuário. As figuras 1 e 2 exemplificam as ideias de anomalia pontual e contextual.

**Figura 1:** Exemplo de anomalia pontual



Fonte: Baddar (2014)

**Figura 2:** Exemplo de anomalia contextual

Fonte: Baddar (2014)

Em detecção baseada em funcionalidade se origina a ideia de detecção baseada em assinaturas e em comportamentos. A detecção baseada em assinaturas compara casos suspeitos com uma relação já conhecida de código comprovadamente nocivo, que fica armazenada geralmente num banco de dados. Numa outra abordagem, a detecção por comportamento vai tentar “aprender” como vive um programa normal dentro do sistema, e como é o ciclo de um programa suspeito, geralmente recorrendo a dados de tempo de execução dos programas que estão sendo processados e classificando essas características com o auxílio de técnicas como aprendizado de máquina e redes neurais (BADDAR, 2014). A maior parte do conjunto recente das aplicações de detecção em nível de rede tem adotado a abordagem do aprendizado de máquinas para detectar software por comportamento, quando em nível de rede. Em nível de aplicação, a grande maioria também procura traçar perfis de comportamento utilizando técnicas de detecção em modo dinâmico como pode-se ver nos quadros 2 e 3.

**Quadro 2:** Tecnologias recentes de detecção em nível de rede

1 <sup>st</sup> Author, Year and Ref.	Approach	NC vs. DC	Scalable?	on/off-line
AbuAitah, 2014 [71]	Machine learning	DC	No	Online
Barani, 2014 [73]	Machine learning	NC	Yes	Online
Salem, 2014 [74]	Machine learning	DC	No	Online
Fiore, 2013 [29]	Machine learning	NC	No	Offline
Guo, 2013 [79]	Machine learning	NC	No	Online
Louvieris, 2013 [81]	Machine learning	NC	No	Offline
Lin, 2012 [83]	Machine learning	NC	No	Offline
Hayes, 2014 [84]	Statistical	DC	Yes	Online
Sun, 2014 [86]	Statistical	NC	No	Offline
Xie, 2014 [90]	Statistical	DC	Yes	Online
Bayraktar, 2014 [91]	Statistical	DC	No	Online
Soltanmohammadi, 2013 [93]	Statistical	DC	No	Online
Limthong, 2013 [95]	Mixed	NC	No	Online
Wang, 2013 [97]	Mixed	NC	No	Offline
Egilmez, 2014 [99]	Spectral	DC	Yes	Online
Cohen, 2014 [101]	Inf. Theoretic	DC	No	Online
Cuadra-Sanchez, 2014 [54]	Inf. Theoretic	NC	No	Offline
Huang, 2014 [102]	Streaming	NC	Yes	Online
Liu, 2012 [104]	Streaming	NC	No	Online

Fonte: Baddar (2014)

**Quadro 3:** Tecnologias recentes de detecção em nível de aplicação

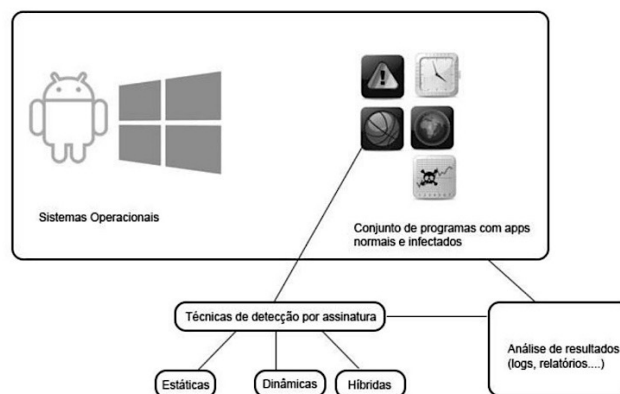
1 <sup>st</sup> Author, Year and Ref.	Stat./Dyn.	on/off-device	Appr.	Scal.?	on/off-line	Sign./Behav.
Damopoulos, 2014 [114]	Dynamic	Mixed	ML	Yes	Online	Behavioral
Shabtai, 2014 [117]	Dynamic	On-device	ML	Yes	Online	Behavioral
Amos, 2013 [120]	Dynamic	Off-device	ML	Yes	Online	Behavioral
Ham, 2013 [35]	Dynamic	Off-device	ML	No	Offline	Behavioral
Wang, 2013 [121]	Dynamic	Off-device	Stat.	Yes	Online	Behavioral
Mas'ud, 2014 [36]	Dynamic	Off-device	ML	No	Offline	Behavioral
Alam, 2013 [123]	Dynamic	Off-device	ML	No	Offline	Behavioral
Wei, 2013 [124]	Dynamic	Off-device	ML	No	Mixed	Behavioral
Shabtai, 2012 [37]	Dynamic	Off-device	ML	No	Online	Behavioral
Yan, 2012 [130]	Dynamic	Off-device	-	No	Offline	Signature
Zheng, 2014 [129]	Dynamic	On-device	-	Yes	Online	Signature
Zhou, 2012 [131]	Dynamic	Off-device	Algo.	No	Online	Mixed
Arp, 2014 [105]	Static	On-device	ML	Yes	Online	Behavioral
Liang, 2014 [109]	Static	Off-device	ML	No	Offline	Behavioral
Cen, 2014 [34]	Static	Off-device	Stat.	No	Offline	Behavioral
Glodek, 2013 [111]	Static	Off-device	ML	No	Offline	Behavioral
Yerima, 2013 [112]	Static	Off-device	ML	No	Offline	Behavioral
Sahs, 2012 [113]	Static	Off-device	ML	No	Offline	Behavioral

Fonte: Baddar (2014)

#### 4 Metodologia

O estudo foi segmentado de acordo com as seguintes etapas: levantamento bibliográfico, análise e desenvolvimento das técnicas de detecção baseadas em assinatura utilizadas atualmente, implementação computacional e análise dos resultados.

A confecção do trabalho ocorreu juntamente com o andamento das demais etapas definidas durante toda a extensão do projeto. A ideia foi utilizar máquinas virtuais e emuladores para implementar em caráter de teste os algoritmos de detecção baseados em assinatura sobre um conjunto de programas contendo alguns programas já infectados com amostras de malware diversas para observar, em termos gerais, a eficiência do tempo de execução dos algoritmos e suas respectivas taxas de sucesso. A figura 3 ilustra o planejamento do trabalho.

**Figura 3:** Estrutura do trabalho

Fonte: elaborada pelos autores (2018)

Para a aplicação deste estudo foi utilizada uma máquina com 4GB de RAM, 1TB de armazenamento e sistema operacional Windows 10 de arquitetura 64 bits.

O fluxo de desenvolvimento do projeto consiste na aplicação de um índice de regras de detecção do Yara, construído a partir da conversão de arquivos contendo bases de assinaturas do Clam AV para arquivos contendo regras de detecção do Yara, sobre o conjunto

de amostras vivas de malware obtidas nos repositórios citados na seção do desenvolvimento do projeto. As varreduras são feitas via linha de comando e para agilizá-las, além do uso de índices para agrupar regras, também pode-se automatizar as varreduras em grupos maiores de arquivos utilizando a linguagem *shell script*. Os resultados das varreduras serão armazenados em arquivos de texto que irão conter todos os atributos analisados a cada arquivo a respeito de uma determinada regra.

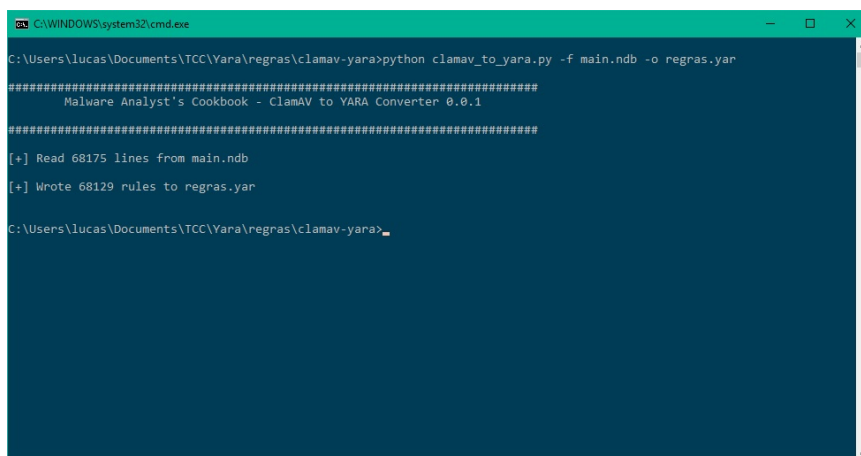
Para melhor compreensão da utilização da ferramenta Yara, é importante ressaltar que a ferramenta é de código aberto e recentemente desenvolvida, cujo intuito é auxiliar desenvolvedores a identificar e classificar amostras de malware. Ela é construída em C, porém, com o auxílio de algumas bibliotecas, há a possibilidade de incorporá-lo em scripts de outras linguagens, como Python. Segundo ALVAREZ (2014), o funcionamento do Yara consiste em comparar regras criadas pelo desenvolvedor que descrevem famílias de malware, com o código de executáveis quaisquer, ou também de arquivos comprimidos em alguns formatos comuns. As regras podem variar bastante em nível de complexidade e mapearem diversas características do comportamento dos executáveis, que serão detalhadas posteriormente. O projeto no momento encontra-se bastante popular e o programa está sendo utilizado como ferramenta de construção de assinatura de malware por desenvolvedores de diversos grupos importantes no ramo, tais como, Kaspersky Lab, Symantec, Trend Micro, *VirusTotal Intelligence*, entre outros.

O Yara funciona interpretando arquivos que contém características que possivelmente apontam para o padrão dos elementos da família do malware que deve ser isolado. As regras são escritas com uma sintaxe simples que remete à construção de uma estrutura de dados em C, contendo *strings* a serem comparadas tanto em formato texto quanto hexadecimal, conjuntos de *strings*, endereços da memória que ficarão sob monitoramento, limitações de tamanho de arquivo, expressões regulares, pontos de entrada de execução e variáveis externas. Todos esses dados do arquivo examinado podem ativar condições, listadas na construção dessas regras, que vão determinar se o arquivo corresponde ou não à família definida nas regras de detecção.

#### 4.1 Experimentos

O desenvolvimento do projeto dividiu-se em quatro etapas: a montagem do ambiente de testes de assinaturas construídas a partir de regras do Yara; a obtenção das regras e construção de um índice de agrupamento das mesmas para agilizar a execução das varreduras; obtenção e escolha de amostras de malware para testagem; bateria de testes final e análise de resultados.

A primeira etapa da implementação computacional do projeto é composta da construção de um índice de regras do Yara compilado a partir das bases de assinaturas do ClamAV e de outros conjuntos avulsos e menores de assinaturas disponibilizados em repositórios de código aberto; em paralelo, também se construiu um conjunto de amostras “vivas” e selecionadas de malware para a testagem do índice sobre arquivos infectados. Foram selecionadas apenas amostras de malware ‘potencialmente inofensivo’, como trojans, que dependem de um comando de execução disparado por um usuário descuidado, para que não houvesse risco de propagação e infecção do próprio ambiente local e de redes com as quais o ambiente local se comunicasse. Por isso, com intuito de prevenção, optou-se pela não realização de testes em, por exemplo, *worms*, que são capazes de propagarem-se de maneira independente pelos ambientes onde chegam. Para converter as assinaturas, foi adaptado um script em Python que interpreta os arquivos que contém as assinaturas do ClamAV, cujo formato é *.cvd*, e reescreve-as na sintaxe amigável de regras do Yara com a utilização de expressões regulares. A figura 4 ilustra o funcionamento do script descrito envolvendo as bases de dados utilizadas na implementação.

**Figura 4:** Funcionamento do script de conversão de bases do ClamAV para regras do Yara


```

C:\WINDOWS\system32\cmd.exe
C:\Users\lucas\Documents\TCC\Yara\regras\clamav-yara>python clamav_to_yara.py -f main.ndb -o regras.yar
#####
Malware Analyst's Cookbook - ClamAV to YARA Converter 0.0.1
#####
[+] Read 68175 lines from main.ndb
[+] Wrote 68129 rules to regras.yar
C:\Users\lucas\Documents\TCC\Yara\regras\clamav-yara>_

```

Fonte: Elaborada pelos autores (2018)

Após a tradução das regras contidas nas bases de dados, é necessário a sua compilação antes de aplicá-las numa varredura real. Um outro script desenvolvido e, *Python* é capaz de realizar tal tarefa, conforme exibido o código-fonte através da figura 5.

**Figura 5:** Script para compilação das regras de detecção

```

1 # Importando biblioteca do Yara
2 import yara;
3 # Invocando método para compilar e armazenar o output buffer na variável
4 rules = yara.compile('./regras.yar');
5 # Produzindo arquivo com resultado
6 rules.save('./regras_compiladas');

```

Fonte: Elaborada pelos autores (2018)

Apenas com a finalidade de atestar o funcionamento da ferramenta estudada no projeto, antes de decidir seguir com seu uso no decorrer da implementação planejada, algumas regras de teste simples foram aplicadas sobre alguns arquivos de texto contendo pedaços de código e um conjunto de *strings* que deveria ser descoberto numa varredura. Os testes foram bem sucedidos nesta etapa por dois motivos; primeiro, as regras escritas eram pequenas e rudimentares, e em segundo lugar também ainda não havia necessidade de conversão de bancos de assinaturas ou montagem de índices mais complexos e adaptação de scripts envolvidos neste processo inicial. Numa tentativa mais condizente com um cenário real de detecção de malware, se fez a construção de um índice com um número maior de regras obtidas nos repositórios já citados previamente. Contudo, nesta segunda etapa ainda não houve tentativa de converter as regras contidas dentro das bases de assinaturas do ClamAV, por motivos de complexidade e de completude dos testes propostos. Nessa mesma fase, as regras foram aplicadas dentro de um conjunto mais numeroso de arquivos, já contendo alguns arquivos de código-fonte e também executáveis de malware misturados em algumas pastas, para verificar como o Yara funcionaria num contexto de varredura mais recursivo. Estas duas primeiras etapas de testes foram bem sucedidas e o Yara acusou corretamente quais arquivos continham as características mapeadas pelas assinaturas do índice de regras aplicado.



## 4.2 Testagem completa e análise de resultados

Para consolidar o trabalho com a ferramenta de uma maneira mais definitiva, um conjunto de aplicativos “saudáveis” foi misturado com as amostras de malware, e depois espalhado arbitrariamente e aleatoriamente dentro de uma estrutura de pastas para que se efetuasse uma varredura recursiva, utilizando um índice de regras mais extenso, agora contendo as regras convertidas e compiladas a partir das bases de dados do ClamAV e das demais regras avulsas encontradas durante a pesquisa desenvolvida. A varredura foi bem sucedida e o Yara não apontou falsos-positivos quando passou pelos arquivos limpos. A única dificuldade encontrada foi quanto à varreduras envolvendo as regras de detecção para malware construído para dispositivos móveis, onde as regras relacionadas ao sistema Android procuravam importar módulos do kit de desenvolvimento do mesmo, que não estava instalado no ambiente do projeto, gerando tal problema que por questões de hardware não foi contornado, pois o carregamento de toda a estrutura do kit sobrecarregava os recursos disponíveis na implementação, tornando inviáveis os testes envolvendo esse grupo de regras. Contudo, na possibilidade de continuar o desenvolvimento do projeto num ambiente com hardware mais generoso e com o uso de um sistema operacional mais apropriado para esse tipo de trabalho, pode-se afirmar com segurança que as regras funcionariam sem nenhum problema.

Enfim, o resultado da varredura recursiva envolvendo um índice de regras completo foi satisfatório e esclarecedor quanto ao funcionamento e à confiabilidade da ferramenta estudada.

## 4.3 Repositório do projeto no GitHub

O projeto foi centralizado e disponibilizado num repositório online dentro do *GitHub* para versionamento de código e também do desenvolvimento do trabalho. Todo o material utilizado na implementação do projeto encontra-se disponível para download na seção de conclusão do estudo.

## 5 Conclusão

A questão do aprimoramento das técnicas de segurança de redes e de informação permanecerá relevante por muito tempo, em virtude do fato de que a sociedade está inserida em um mundo que cada vez mais produz e depende de tecnologia e também apresenta este mesmo comportamento quanto ao volume e à variedade de dados que trafegam pela internet.

Com os estudos levantados e uma análise do cenário atual da computação, pode-se afirmar que as técnicas estudadas e aplicadas no desenvolvimento deste projeto estão, em termos de desempenho, chegando ao seu máximo, pois a manipulação de assinaturas de malware implica características no processo de detecção que limitam uma possível abordagem com o uso de técnicas de programação mais inovadoras. Embora a eficácia das ferramentas atualmente empregadas seja satisfatória e as mesmas apresentem uma boa precisão de detecção, elas estão sempre atuando num contexto do que pode-se chamar de ‘pós-infecção’ ou pelo menos de uma infecção iminente, onde os arquivos maliciosos já estão trafegando por pontos de rede ou até mesmo se encontram latentes dentro dos ambientes vulneráveis infiltrados, seja por estarem à espera de um comando do usuário ou por terem execução agendada. O panorama dos estudos nesse segmento da área de segurança de redes está, em maior parte, tomando um viés para a prevenção de infecções e episódios de vulnerabilidade, onde os métodos de detecção de intrusão por anomalia e comportamento apresentam a vantagem de estarem constantemente procurando se antecipar a possíveis ataques e a novas formas de quebra de segurança, com a utilização de conceitos como redes neurais, grafos, e até mesmo aprendizado de máquina.

No âmbito mais prático do desenvolvimento do projeto, foram obtidos resultados

positivos quanto ao que foi proposto, com a construção de parte de uma aplicação para verificação de arquivos em geral e os testes envolvendo as ferramentas que compõem tal aplicação.

Porém, observou-se que a abertura para a criação e melhoria de técnicas envolvendo assinaturas é relativamente recente, e as ferramentas encontradas e utilizadas para tais fins não são voltadas para o público mais leigo e, talvez, curiosos e entusiastas do assunto, haja visto que mesmo uma ferramenta como o Yara, cujo intuito é tornar essa manipulação de assinaturas mais simplificada, requer uma configuração que exige um razoável conhecimento de utilização de terminais e linhas de comando, e afinidade com alguma linguagem de programação de alto ou médio nível para a construção dos scripts de varredura ou conversão de bases de assinaturas.

Foi pensando nesse aspecto que se desenvolveu a ideia da aplicação online com um sistema capaz de automatizar as tarefas de manuseio de assinaturas de malware, para que as pessoas possam, além de testar a confiabilidade de arquivos encontrados na internet, ter um ponto de partida para conhecer essa área da computação e também tentar colocar um pouco desse conhecimento em prática de uma maneira colaborativa. O código-fonte do protótipo da aplicação encontra-se hospedado em repositório (<https://github.com/ltgouvea/tcc>), aberto para quaisquer desenvolvedores que queiram dar andamento ao projeto caso se interessem.

Já há serviços de varredura de malware gratuita disponíveis pela internet, e pode-se relacionar o estudo desenvolvido neste projeto com outros projetos recentes como o de SUN *et. al* (2012), que propõe a ideia de geração automática de assinaturas de malware incorporando o mapeamento de características nocivas dentro do próprio sistema de armazenamento de arquivos de um sistema operacional, incluindo também um mecanismo de validação das assinaturas geradas para a redução de falsos positivos com a implementação de um sistema de calibração das mesmas.

Assim, o interesse maior no desenvolvimento dessa aplicação é o diferencial que existiria na manipulação de assinaturas de modo colaborativo, possivelmente anônimo e montando bases de dados automaticamente no módulo de conversão de regras descrito no capítulo de desenvolvimento do projeto. O resultado final foi uma aplicação funcional das técnicas propostas no projeto que pode ser utilizada isoladamente, sem o auxílio de outros softwares antivírus quaisquer, e posteriormente ampliada ou incorporada no segmento de uma aplicação online similar à ideia descrita durante o desenvolvimento do projeto.

## REFERÊNCIAS

ALVAREZ, V. **Writing Yara rules**. 2014. Disponível em: <<http://yara.readthedocs.io/en/v3.5.0/writingrules.html>>. Acesso em: 01 jun. 2018.

BADDAR, S. A.-H.; MERLO, A.; MIGLIARDI, M. **Anomaly Detection in Computer Networks: A State-of-the-Art Review**. 2014. Disponível em: <<http://isyu.info/jowua/papers/jowua-v5n4-2.pdf>>. Acesso em: 15 mai. 2018.

IDIKA, N. M. A. **A survey of malware detection techniques**. 2007. Disponível em: <[http://profsandhu.com/cs5323\\_s17/im\\_2007.pdf](http://profsandhu.com/cs5323_s17/im_2007.pdf)>. Acesso em: 12 jul. 2018

SAWLE PRAJAKTA D.; GADICHA, A. **Analysis of malware detection techniques in android**. International Journal of Computer Science and Mobile Computing, 2014. Disponível em: <<https://pdfs.semanticscholar.org/7f33/9156f47345bd102c9b05f45f9bfe4c182720.pdf>>. Acesso em: 22 abr. 2018.

SAEED ALI SELAMAT, A. M. A. A. I. A. **A survey on malware and malware detection systems**. International Journal of Computer Applications (0975 – 8887), 2013. Disponível em:

<<https://pdfs.semanticscholar.org/0645/884497d8be3257531026737b791c6614ddb0.pdf>>. Acesso em: 22 abr. 2018.

SZEWCZYK, P. **A survey of computer and network security support**. Australian Information Security Management Conference, 2012. Disponível em: <<http://ro.ecu.edu.au/cgi/viewcontent.cgi?article=1149&context=ism>>. Acesso em: 15 mai. 2018.