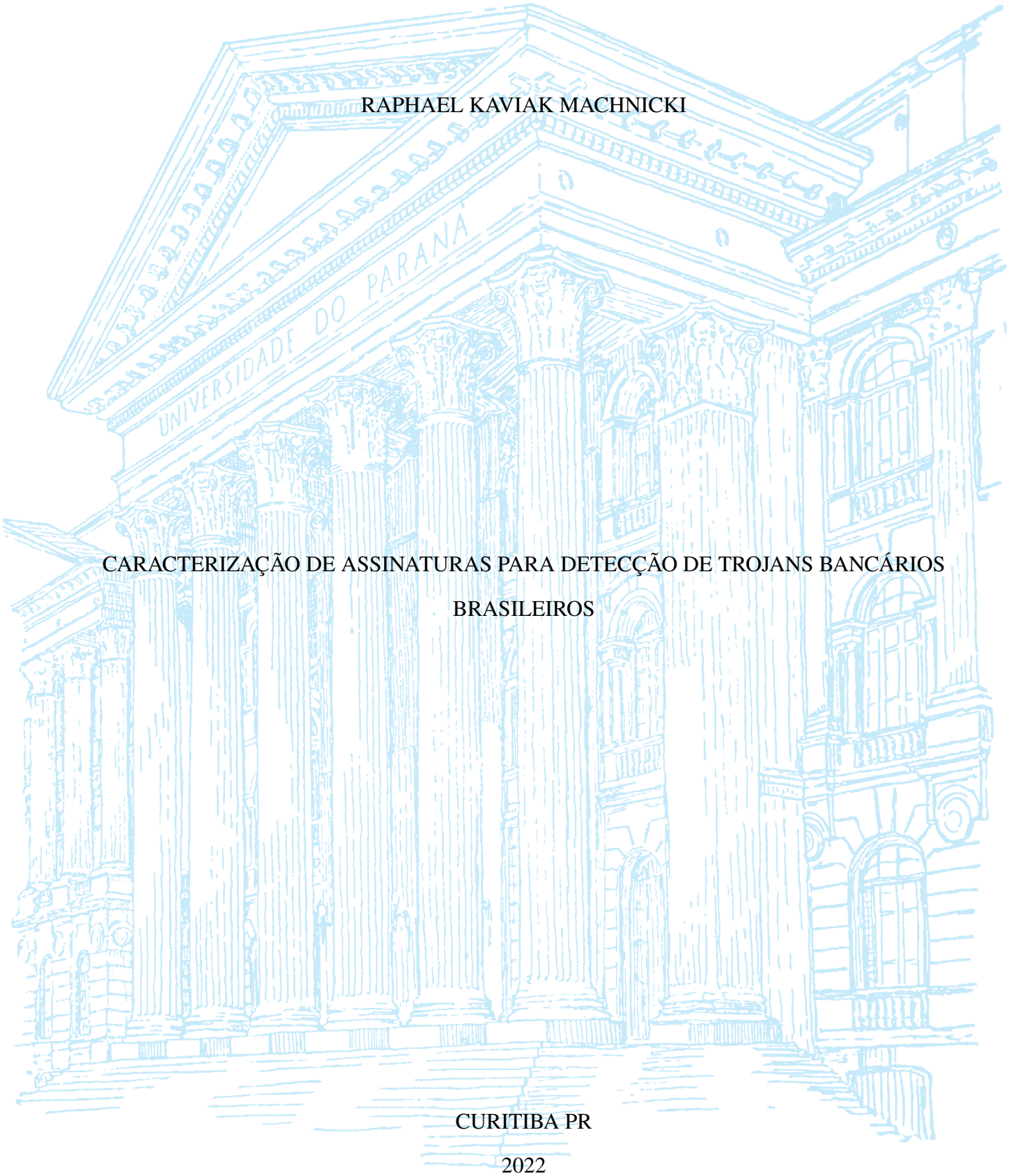
UNIVERSIDADE FEDERAL DO PARANÁ

RAPHAEL KAVIAK MACHNICKI




# CARACTERIZAÇÃO DE ASSINATURAS PARA DETECÇÃO DE TROJANS BANCÁRIOS BRASILEIROS

CURITIBA PR

2022

RAPHAEL KAVIAK MACHNICKI

# CARACTERIZAÇÃO DE ASSINATURAS PARA DETECÇÃO DE TROJANS BANCÁRIOS BRASILEIROS

Trabalho apresentado como requisito parcial à conclusão do Curso de Bacharelado em Ciência da Computação, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: André Ricardo Abed Grégio.

CURITIBA PR

2022

*"O amor vale mais que a própria vida."*
*(Clóvis de Barros Filho)*

## AGRADECIMENTOS

Agradeço primeiramente aos meus pais, Ana e André, por todo apoio, carinho e ensinamentos ao longo desses anos até hoje. Agradeço ao meu orientador André Grégio, pelas oportunidades, apoio e aprendizado, o que contribuiu de forma significativa para a minha formação acadêmica e pessoal. Agradeço aos integrantes do Laboratório Secret por todos os conhecimentos compartilhados e por toda ajuda. Agradeço aos meus amigos pela amizade ao longo desses anos de graduação e que levarei por toda minha vida. Agradeço aos professores do Departamento de Informática pelos ensinamentos passados.

## RESUMO

O Brasil figura há bastante tempo nas posições mais altas entre os países atacados por *Trojans* bancários. Com o intuito de minimizar perdas financeiras e roubo de credenciais dos usuários, surge a necessidade de se estudar técnicas e ferramentas para detectar e prevenir essas ameaças. Para cumprir esse objetivo, pode-se fazer o uso de sistemas de detecção de intrusão em redes (NIDS) que, usando detecção por assinatura, são capazes de reconhecer padrões de tráfego gerado por diferentes famílias de malware associadas a *Trojans* bancários. Apesar desse tipo de malware ser comum no cenário nacional, não existe uma base de dados das características principais a se adotar pelas assinaturas que se propõem a detectá-los. Nesse contexto, esta monografia define os principais aspectos que devem compor as assinaturas para quatro famílias diretamente envolvidas com ataques a usuários de *Internet Banking*: AutoIt, Banload, Banbra e Mekotio. Corroborando a necessidade de caracterização de assinatura de malware em cenário nacional, foram feitos testes com o antivírus VIPRE e observou-se que suas assinaturas não eram capazes de detectar como malicioso nenhum tráfego gerado por executáveis das famílias citadas. Ressalta-se que os exemplares de malware analisados estabelecem comunicação com um servidor *Command and Control*, com o intuito de enviar credenciais e informações de sistema, e receber comandos específicos, dependendo do exemplar de malware. Assim, a partir dessa comunicação, foi possível construir as assinaturas desenvolvidas neste trabalho, as quais serão mostradas e discutidas em detalhes.



Palavras-chave: Sistemas de detecção de intrusão em redes, *Trojans* bancários, antivírus

## ABSTRACT

Brazil has long been in the highest positions among the countries attacked by Banking Trojans. In order to minimize financial losses and theft of credentials from users, there is a need to study techniques and tools to detect and prevent these threats. To achieve this goal, network intrusion detection systems (NIDS), using signature-based detection, can detect traffic patterns generated by different banking trojan malware families. Despite this kind of malware being common on the national scene, there is not a database of the main characteristics to be adopted by the signatures proposed to detect them. In this context, this monography defines the main aspects that must compose the signatures to detect four families of malware related to attacks on Internet Banking users: AutoIt, Banload, Banbra and Mekotio. Corroborating the need for malware signature characterization in a national scenario, tests were carried out with the VIPRE antivirus and it was observed that its signatures were not able to detect as malicious any of the traffic generated by executables of the mentioned families. It is noteworthy that analyzed malware samples establish communication with a Command & Control server, with the purpose to send credentials and system information, and receive specific commands, depending on the malware sample. Based on this exchange of messages, it was possible to build the rules developed in this monograph, which will be shown and discussed in detail.



Keywords: Network intrusion detection systems, banking Trojans, antiviruses

## LISTA DE FIGURAS

## LISTA DE TABELAS

## LISTA DE ACRÔNIMOS

| | |
|---|---|
| C&C | *Command and Control* |
| CPF | Comprovante de Pessoa Física |
| URL | *Uniform Resource Locator* |
| URI | *Uniform Resource Identifier* |
| IP | *Internet Protocol* |
| HTTP | *Hypertext Transfer Protocol* |
| HTML | *HyperText Markup Language* |
| DNS | *Domain Name System* |
| FTP | *File Tranfer Protocol* |
| TCP | *Transmission Control Protocol* |
| UDP | *User Datagram Protocol* |
| DHCP | *Dynamic Host Configuration Protocol* |
| SMTP | *Simple Mail Transfer Protocol* |
| PCAP | *Packet Capture* |
| NIDS | *Network Intrusion Detection System* |
| NIPS | *Network Intrusion Prevention System* |
| SSL | *Secure Sockets Layer* |
| CSV | *Comma-Separated Values* |
| IDA | *Interactive Disassembler* |
| DOM | *Document Object Model* |
| MITM | *Man In The Middle* |
| DLL | *Dynamic-link library* |
| JSON | *JavaScript Object Notation* |
| AM | Aprendizado de Máquina |
| AV | Antivírus |
| RAT | *Remote Access Trojans* |

**SUMÁRIO**

# 1 INTRODUÇÃO

Autores de malware usam servidores comprometidos para armazenar dados roubados de usuários e enviar comandos para máquinas infectadas, com o objetivo de ter acesso as mais variadas informações, por exemplo: Qual é o antivírus instalado na máquina afetada? Quais sites o usuário possui aberto no momento? Qual é o identificador da máquina no grupo definido pelo sistema operacional? A técnica usada pelos exemplares de malware com esse tipo de comportamento é chamada de *command and control (C&C)*. Com base nessas informações, o servidor *C&C* é capaz de definir qual é a melhor estratégia de realizar um ataque contra cada vítima, que geralmente tem por finalidade o roubo de credenciais, ou até mesmo a falsificação de documentos e arquivos (Gardiner et al., 2014).

Com o intuito de detectar e/ou evitar a comunicação entre o malware instalado na máquina afetada e o servidor de *C&C*, pode-se fazer o uso de sistemas de detecção/prevenção de intrusão em redes (NIDS/NIPS). Esse tipo de solução tem como principal função o reconhecimento de padrões de intrusão utilizados para realizar um tipo específico de ataque, e funciona inspecionando o *payload* dos pacotes que compõem o tráfego de rede em uma determinada interface.

O Brasil é um dos países com o maior envolvimento em ataques cibernéticos, sendo vítima e também os realizando (Diniz et al., 2014). Uma boa parcela de todo o malware distribuído no país corresponde a ataques contra usuários de *Internet Banking*, o que reflete a grande massa de pessoas usando esse tipo de serviço, dado que proximadamente 79% dos internautas o usam, principalmente para realizar transações (Sgarioni, 2021). Esse número elevado de ataques envolvendo aplicações de bancos tem a ver não só com o alto número de malware com esse propósito, mas com o próprio desenvolvimento desse tipo de aplicação, tendo-se exemplos de aplicações bancárias que armazenam informações de cartão de crédito em bancos de dados locais, não realizam fixação de certificado (falta de *pinned certificate*, resultando em ataques *MITM - Man In The Middle*), além de não ter controle sobre a presença de *debuggers* e dispositivos rootados (Botacin et al., 2019).

Somando-se problemas com o desenvolvimento das aplicações de *Internet Banking* e a grande quantidade de tentativas de ataque, tem-se no Brasil uma grande variedade de *Trojans* bancários, que geralmente usam técnicas de *C&C* para infecção da máquina atacada. Existem dezenas de malware com essas carcterísticas, entre eles podemos citar *Dridex, BrazKing, BRata, Ghimob, Bizarro, Agent Tesla, Zeus, Danabot, Ursnif, Retefe*, todos eles sendo classificados também como *RAT - Remote Access Trojans*.

A Tabela 1.1 mostra as TOP 10 famílias de malware usadas por atacantes no Brasil em agosto de 2020 (InforChannel, 2020). Destaca-se que três deles são responsáveis pelo download e execução de trojans bancários (*Emotet*, *Trickbot* e *Glupteba*), e sete envolvem técnicas de C&C (*Emotet*, *Lucifer*, *Trickbot*, *Pykspa*, *Formbook*, *Remcos* e *Glupteba*). Além disso, notou-se que todos eles têm um alto potencial de disseminação, podendo até mesmo gerar e enviar *e-mails* com links maliciosos.

Dado a presença de *Trojans* bancários classificados como *RATs*, surge a necessidade de encontrar maneiras de detectá-los, tanto durante sua execução, no ambiente de usuário do sistema operacional, quanto a partir do tráfego gerado por eles, fazendo o uso de sistemas de detecção de intrusão. Usando essa última abordagem, tem-se que não existe um conjunto fixo de regras de detectores de instrusão em redes capazes de detectar com precisão de qual família de malware corresponde o tráfego analisado, devido as limitações da detecção por assinatura, e as contantes melhorias aplicadas nas técnicas de camuflagem de malware, que serão discutidas neste trabalho.

| **Nome** | **Objetivo** | **Trojan Bancário** | ***C&C*** |
|---|---|---|---|
| Emotet | Credenciais | Sim | Sim |
| XMRig | Criptomoedas | Não | Não |
| Lucifer | Criptomoedas | Não | Sim |
| Trickbot | Credenciais | Sim | Sim |
| RigEK | Exploit kit | - | - |
| Pykspa | online meeting | Não | Sim |
| Formbook | Credenciais | Não | Sim |
| Remcos | Credenciais | Não | Sim |
| Glupteba | Credenciais | Sim | Sim |
| Ramnit | Disseminação | - | Não |

Tabela 1.1: TOP 10 famílias de malware detectadas em 2020 no Brasil.

A Seção 3.1 mostra como foram obtidos os arquivos de malware e tráfego utilizados nesta monografia. Mostra-se que as famílias de malware mais populares no `dataset` correspondiam a *Trojans* bancários, e também a requisições feitas a partir de scripts AutoIt, uma linguagem que permite automações no sistema operacional Windows. Com o objetivo de definir um conjunto de regras capazes de detectar 3 famílias distintas de trojan populares no `dataset`: (1) Banload: trojan.downloader, (2) Banbra: trojan.banker e (3) Mekotio: trojan.banker, serão estudados uma série de tráfegos gerados por executáveis dessas respectivas famílias, com o objetivo de entender o seu comportamento com relação a comunicação via rede (requisições e respostas HTTP, *queries* DNS, etc), para que seja possível, por meio de um conjunto de regras desenvolvido nesta monografia, determinar sinais de pós e pré infecção. Além das famílias supracitadas, foram também estudadas regras para detecção de requisições maliciosas via scripts AutoIt. Foi descoberto que, apesar de ser possível estudar o comportamento genérico das famílias e definir semelhanças entre o tráfego gerado pelas execuções e as regras usadas para fazer a detecção, não é possível estabelecer um conjunto imutável de regras que certamente detecta e continuará detectando as 3 famílias, devido principalmente às características intrínsecas da detecção usando assinatura de malwares, que é sensível a mudanças determinadas pelo autor do mesmo.

Além dos objetivos já mencionados, este trabalho mostrará como os módulos de antivírus que se propõe a fazer detecção de intrusão em rede funcionam, bem como investigações em mais detalhes das regras do antivírus VIPRE (VIPRE, 2022), em sua forma *freeware*. Essas regras foram extraídas no artigo *AntiViruses under the microscope: A hands-on perspective* (Botacin et al., 2022), cujo autor da presente monagrafia foi coautor. No artigo, foi mostrada somente a composição e quantidade das regras de NIDS usadas pelo antivírus, enquanto que neste trabalho, tais regras foram submetidas contra um conjunto de tráfego sabidamente malicioso (gerado pelas famílias Banload, Banbra e Mekotio). O resultado foi que nenhuma das 466 regras presentes na versão do VIPRE utilizada gerou algum alerta, devido principalmente a falta de rotinas de varredura contra *C&C* e *queries* DNS para domínios gratuitos.

## 1.1 JUSTIFICATIVA

Identificou-se que nos últimos anos, houve grande incidência de ataques de malware do tipo RAT contra usuário brasileiros, principalmente aqueles de *Internet Banking*. Para compor um conjunto de regras de NIDS capazes de determinar sinais de pré e pós-infecção contra algumas das famílias que definem esses tipos de malware, pode-se analisar o tráfego gerado por um conjunto de executáveis previamente classificados como pertecentes a essas famílias, com

o intuito de entender o comportamento de requisições feitas por RATs, e assim ser capaz de construir regras que os detectem.

Nesse contexto, um conjunto de assinaturas foi caracterizado para cada uma das famílias, bem como foi detalhado o processo de análise e inspeção de tráfego dos conjuntos de PCAPs que mais continham indícios de comunicação entre malware e servidor. Os resultados obtidos nesse trabalho, principalmente com relação a filtragem do tráfego e criação de assinaturas, visam contribuir com os usuários de sistemas de detecção de intrusão em redes, uma vez que detalha a construção das novas regras, bem como as disponibiliza.

## 1.2 OBJETIVOS

O objetivo desse trabalho é determinar um conjunto de regras de NIDS, bem como mostrar como fazê-lo, para um conjunto de tráfego gerado por executáveis previamente classificados como RATs bancários. Para realizar esse objetivo principal, o trabalho de pesquisa foi subdividido em itens secundários:

1. Captura de tráfego malicioso conhecido.
2. Filtragem do tráfego: relação entre o tráfego gerado por uma execução de arquivo e uma determinada família de malware.
3. Análise inicial de todo o tráfego já filtrado, enfatizando um conjunto de regras escritas para detecção de *C&C* e RATs.
4. Análise detalhada do tráfego que foi alertado pelo NIDS para as regras do passo anterior (3), usando o *Wireshark*.
5. Construção das novas regras com base nos comportamentos e características definidos ao fim do passo anterior (4).

## 1.3 ORGANIZAÇÃO DA MONOGRAFIA

Esta monografia está dividida em cinco capítulos. No Capítulo 2, é realizada uma revisão bibliográfica a fim de contextualizar o leitor sobre os conceitos e tecnologias utilizadas para o desenvolvimento desta pesquisa. No Capítulo 3, é apresentado como será realizado o desenvolvimento das regras, como e qual NIDS será utilizado, e a origem dos arquivos utilizados nesta pesquisa. Na Seção 3.3, tem-se uma breve definição de como são feitas as rotinas de inspeção de pacotes por um antivírus, bem como é feita análise das regras do antivírus VIPRE contra o tráfego gerado por RATs. No Capítulo 4, são apresentadas as regras escritas com base na análise do tráfego. Por fim, no Capítulo 5 são apresentadas as conclusões obtidas, as dificuldades encontradas e propostas de trabalhos futuros.

# 2 FUNDAMENTAÇÃO TEÓRICA E ESTADO DA ARTE

## 2.1 *NIDS - SISTEMAS DE DETECÇÃO DE INTRUSÃO EM REDES*

Os sistemas de detecção de instrusão em redes (*NIDS - Network Instrusion Detection Systems*) são sistemas capazes de fazer análise dos pacotes que trafegam através da rede na qual estão inseridos. Além disso, podem funcionar como *sniffers*, ou seja, capturar e mostrar os pacotes que trafegaram pela rede, ou como *logger de pacotes*, o que significa que é possível armazenar em arquivos de captura de pacotes (PCAPs) o tráfego de interesse que passe pela rede.

Para fazer a análise de tráfego, os NIDS precisam contar com um conjunto de regras que, ao ser carregadas, identificam quais características o sistema deve procurar ao analisar o *payload* e cabeçalhos de um pacote. Existem diferentes NIDS, e consequentemente, as regras suportadas por cada um deles é diferente dos demais. Exemplo disso é que cada empresa de Antivírus (AV) cria um módulo diferente para inspeção de rede, ou seja, rotinas de **varredura dos pacotes que tem como objetivo detectar algum padrão conhecido de ataque ou vulnerabilidade**, que é justamente o propósito principal dos sistemas de detecção em redes. Num estudo mais aprofundado sobre cada módulo que compõe os principais antivírus (Botacin et al., 2022), foi possível perceber que cada um deles usa um sistema de detecção próprio (escrito pelos desenvolvedores do AV), o que implica que as regras suportadas por um não serão necessariamente suportadas por outro. Dos AVs analisados, o único que usa um NIDS baseado em sofware livre (SNORT) é o *VIPRE*, que terá seu conjunto de regras submetido a um conjunto sabidamente malicioso de PCAPs. As conclusões desse experimento serão discutidas na Seção 3.3

Além do Snort, outro NIDS de software livre é o Suricata. O Snort foi lançado oficialmente como NIDS em 1999 e é até hoje o NIDS mais utilizado pela comunidade em geral, considerado pelo *sectools*, atualmente, como a quinta ferramenta de segurança preferida da comunidade (sectools.org, 2011). Já o Suricata foi lançado em 2010, com o objetivo de ser uma alternativa *multi-threading* ao Snort, porém mantendo o máximo de semelhanças possível com o seu antecessor. A abordagem usando paralelelismo do Suricata realmente gerou bons resultados com relação ao desempenho, sendo que tarefa de analisar o *payload* de um conjunto de pacotes a partir de um arquivo de regras parece altamente suscetível a uma abordagem *multi-thread*. O Suricata possui um desempenho superior comparado ao Snort, podendo ser até 20 vezes mais rápido, dependendo das configurações e quantidade de tráfego analisada, além disso, mesmo utilizando o Suricata sem paralelismo (modo *single-thread*),o Snort ainda apresenta um desempenho inferior. Isso ocorre devido ao fato de ter sido desenvolvido para os computadores de 1998, que eram *single-core* e 32 *bits*, e mesmo com melhorias no código com o passar do tempo, o motor do Snort manteve-se *single-threaded* (White et al., 2013).

Neste trabalho, pelos motivos de desempenho apresentados acima, **o NIDS utilizado será o Suricata**, versão 6.0.4. Durante anos, antes do lançamento do *Snort 3*, ambos os NIDS suportavam o mesmo conjunto de regras. A partir dessa nova versão, houve uma mudança na sintaxe das regras, contudo, houve também a disponibilização de um programa que atualiza as regras da versão 2 do Snort (que são suportadas pelo Suricata) para a versão 3, como é descrito em 2.1.1.

### 2.1.1 Regras para Snort 3

As regras escritas neste trabalho seguem a sintaxe utilizada no Suricata. Em versões anteriores do Snort, ambos os sistemas suportavam regras com a mesma sintaxe, porém, com a

atualização mais recente, o Snort mudou a sintaxe de suas regras, que passaram a ser diferentes das regras do Suricata. Para contornar esse problema, os desenvolvedores do Snort criaram um método para conversão das regras com a sintaxe antiga para a nova, usando um script *lua*, é possível converter as regras para serem suportadas pela nova versão. Como dito anteriormente, as regras desse trabalho foram escritas com base na sintaxe do Suricata, caso o leitor queira utilizar alguma regra desse trabalho, e utilize o Snort como NIDS, é necessário que as regras sejam atualizadas. Para isso, é possível usar o comando da Listagem 2.1, onde:

Listing 2.1: Comando para tranformação das assinaturas em regras suportadas pelo Snort 3

```
snort2lua -c regras_trabalho.rules -r
   regras_trabalho_atualizadas.rules
```

- snort2lua: Script Lua, tranforma regras no formato Suricata/Snort2 para Snort 3
- regras_trabalho.rules: Regras elaboradas neste trabalho (elencadas no apêndice C)
- regras_trabalho_atualizadas.rules: Regras atualizadas para o Snort 3

## 2.2 DETECÇÃO POR ASSINATURA E REGRAS DE NIDS

As regras são parte fundamental do funcionamento dos NIDS, são elas que vão de fato garantir que o NIDS seja capaz de alertar algum tráfego de interesse passando pela rede. As regras do Suricata funcionam por meio de detecção por assinatura, isso significa que as regras são escritas de maneira a identificar algum tipo de padrão específico, que pode ser definido por aquela assinatura em questão. Para enriquecer essa definição, é possível desenvolver um exemplo de assinatura que detecta tranferência de arquivo usando o protocolo *FTP - File Tranfer Protocol*, em que o conteúdo desse arquivo contenha um ou mais números de CPF. Com esse contexto, fica fácil determinar como será a assinatura desenvolvida.

Pode-se, a partir disso, tentar criar uma regra do Suricata (baseada em assinatura) capaz de detectar o tráfego especificado acima, mas antes disso, define-se na Listagem 2.2 alguns detalhes fundamentais dos elementos que compõe uma regra:

Listing 2.2: Partes que compõe uma regra de NIDS

```
ação protocolo endereço porta -> endereço porta opções
```

A **ação** deve ser um dos elementos de *alert, pass, drop e reject*, e fica responsável por gerar um alerta no arquivo de log, ignorar o resto das regras para esse pacote, derrubar um pacote silenciosamente e gerar um alerta e derrubar um pacote e gerar um alerta, com retorno de um *RST/ICMP unreach error* para o endereço origem do pacote, respectivamente.

O **protocolo** indica qual é o protocolo esperado para que um pacote sofra a ação definida anteriormente. Em seu website, o Suricata divulga uma lista elencando todos os protocolos suportados atualmente (suricata.io, 2019), sendo eles da camada de transporte (TCP, UDP) ou de aplicação (HTTP, DNS, SSH, entre outros).

O **endereço** pode ser um endereço de IP, ou uma variável no arquivo de configuração, contendo um ou mais IPs, e o mesmo aplica-se para **porta**. A **seta** `"->"` representa o destino do tráfego, fazendo com que o primeiro par endereço porta seja a origem do pacote, enquanto o segundo par é o destino.

A parte de **opções** contém a parte mais complexa da regra, e onde está a assinatura em si. Voltando ao exemplo acima, do tráfego FTP com arquivo contendo CPF, é possível criar a regra discriminada na Listagem 2.3.

Listing 2.3: Detecção de tranferência de arquivo contendo texto com CPF via FTP

```
alert ftp $HOME_NET any -> $EXTERNAL_NET 20(
msg:"CPF(texto plano) em arquivo enviado via FTP";
pcre:"/^\d{3}\.\d{3}\.\d{3}\-\d{2}$/";
classtype:policy-violation;sid:1; rev:1;)
```

A regra acima contém cinco atributos importantes na parte de opções: (1) **msg**: é o que vai identificar a assinatura nos arquivos de log, (2) **pcre**: *Pearl Compatible Regular Expressions*, expressões regulares, escritas para casarem com alguma parte do payload de um pacote, (3) **classtype**: qual o tipo de ataque/vulnerabilidade/característica essa regra procura, os mais comuns são *trojan-activity* e *web-application-attack*, (4) **sid**: Identificador único da regra (obrigatório), (5) **rev**: número de revisão da regra (obrigatório).

Agora que os conceitos de regra e assinatura foram definidos, podemos exemplificar na Listagem 2.4 outra regra, dessa vez com o objetivo de detectar o download de um arquivo compactado zip, a partir do navegador *chrome* no sistema operacional Windows 10.

Listing 2.4: Detecção de requisição para arquivo zip a partir do Chrome e Windows 10

```
alert http $HOME_NET any -> $EXTERNAL_NET any
(msg:"Download de arquivo zip a partir do chrome";
flow:to_server,established; http.method; content:"GET";
http.uri; content:".zip"; http.user_agent; content:"Mozilla/5.0
(Windows NT 10.0) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/100.0.4896.127 Safari/537.36"; sid:2; rev:1;)
```

Na Listagem 2.4, fica evidente o uso de diversos campos do cabeçalho HTTP para compor a assinatura, e cada um deles, com a palavra-chave *content* própria. *content* é o termo mais importante e mais utilizado para fazer uma detecção minuciosa, porque analisa cada campo do cabeçalho pelo termo que deve ser casado, e só é gerado um alerta caso todos os *content* da assinatura estejam condizendo com a situação do pacote.

Com o que foi exposto até aqui, é possível concluir que a detecção por assinatura contém vantagens e desvantagens. A principal vantagem é o desempenho: o Suricata, ao analisar um pacote e determinar seu protocolo, pode processar regras definidas para esse protocolo contra esse pacote, não havendo necessidade de processar todas as regras do conjunto para um mesmo pacote. A principal desvantagem da detecção por assinatura é a alta sensibilidade a mudanças, caso alguma parte do padrão mude, a regra inteira deixará de funcionar. Por exemplo, a regra criada acima de detecção de CPFs sobre o FTP só funciona se o CPF for escrito dessa maneira: xxx.xxx.xxx-xx. Caso os CPFs passem a ser escritos sem os pontos e o hífen, a regra não gerará mais alertas, sendo necessário modificar o padrão da expressão regular, ou criar uma nova assinatura. Mais vantagens e desvantagens sobre a detecção por assinaturas serão expostos na Seção 5.1.

## 2.3 SURICATA - MODO INLINE/NIPS E NIDS

Usando o Suricata, é possível determinar como o tráfego chegará para a posterior análise dos *payloads* dos pacotes. Existem duas maneiras de posicionar o sistema para fazer essa detecção: usando ou não o modo *inline*. O modo *inline*, como o nome sugere, permite que o sistema fique antes da rede interna, e necessariamente todo o tráfego passará pelo Suricata antes de atingir a rede. Com isso, há garantia que se caso algum conteúdo malicioso seja detectado em alguns dos pacotes analisados, seja possível tomar ações no mesmo momento contra esse pacote, podendo até mesmo rejeitar ou derrubar o pacote na mesma hora. Esse modo também é

chamado de *NIPS, Network Instrusion Prevention System*, já que além de fazer a detecção, o Suricata pode tomar uma ação imediata, prevenindo alguma potencial infecção. A arquitetura do Suricata em modo NIPS é exibida na Figura 2.1. Esse modo deve ser utilizado quando o objetivo das regras é identificar possíveis sinais de pré infecção de alguma família de malware, impedindo a infecção de fato.

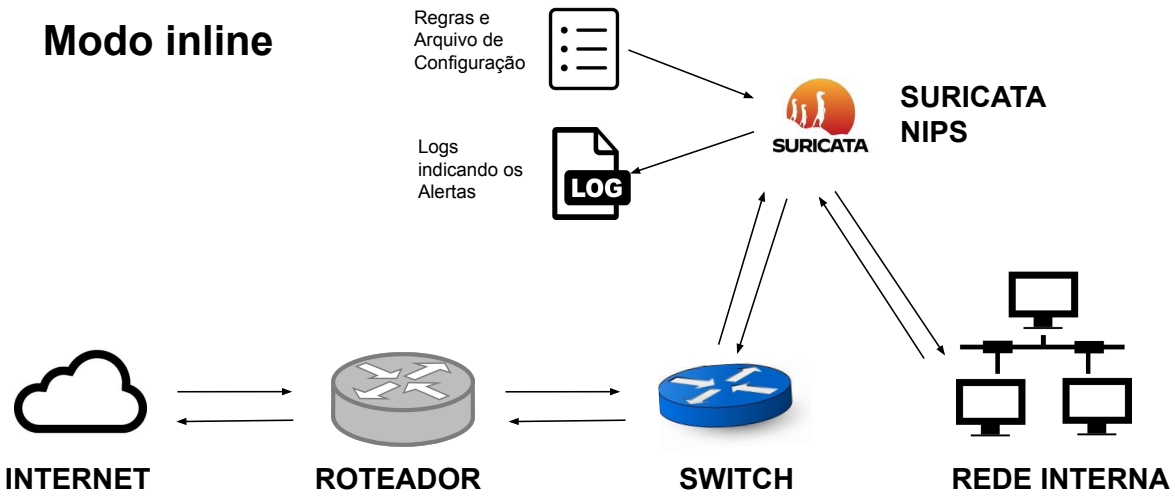



Figura 2.1: Suricata em modo inline/IPS.

O Suricata também permite um funcionamento com modo *inline* desligado, isso significa que o funcionamento será puramente para a detecção de ameaças, sem a opção imediata de recusar um conjunto de pacotes. Como é possível perceber na Figura 2.2, é possível configurar uma opção no *switch* para que um subconjunto do tráfego (ou todo o tráfego) seja duplicado para a porta do Suricata. Com isso, tem-se menor comprometimento de desempenho, pois o tráfego não precisa passar pelo Suricata antes de chegar ate à rede interna. Porém, não é possível recusar um pacote imediatamente, o que é feito é somente a adição de uma nova entrada nos arquivos de log, caso o payload de algum dos pacotes case com alguma das assinaturas presentes na regras. Esse modo deve ser utilizado quando o objetivo das regras é identificar possíveis sinais de pós infecção de alguma família de malware.

## 2.4 SURICATA - MODO *OFFLINE/REPLAY*

Neste trabalho, o Suricata será utilizado em modo *replay*, o que signica que o NIDS não será posicionado na rede para checagem de tráfego, mas fará a análise de um conjunto de PCAPs. Uma maneira interessante de usar o Suricata é posicioná-lo na rede (como IPS ou IDS), e configurá-lo de maneira a gravar o tráfego que gerou os alertas em arquivos PCAP, salvando a assinatura e o nome dos PCAPs nos arquivos de log. Depois, é possível usar alguma ferramenta de mais alto nível para a inspeção de tráfego, como por exemplo o *wireshark*. Pode-se investigar qual a direção do tráfego, qual protocolo gerou o alerta, quais atributos desse protocolo foram analisados para chegar a essa assinatura, qual o domínio ou a URI das requisições, entre outros, e a partir dessas informações e pesquisas sobre o comportamento analisado, tentar escrever regras ainda mais completas, que juntas formem um subconjunto específico para alguma família/variante de malware. Essa abordagem será bastante utilizada no decorrer desse trabalho, exceto pelo fato

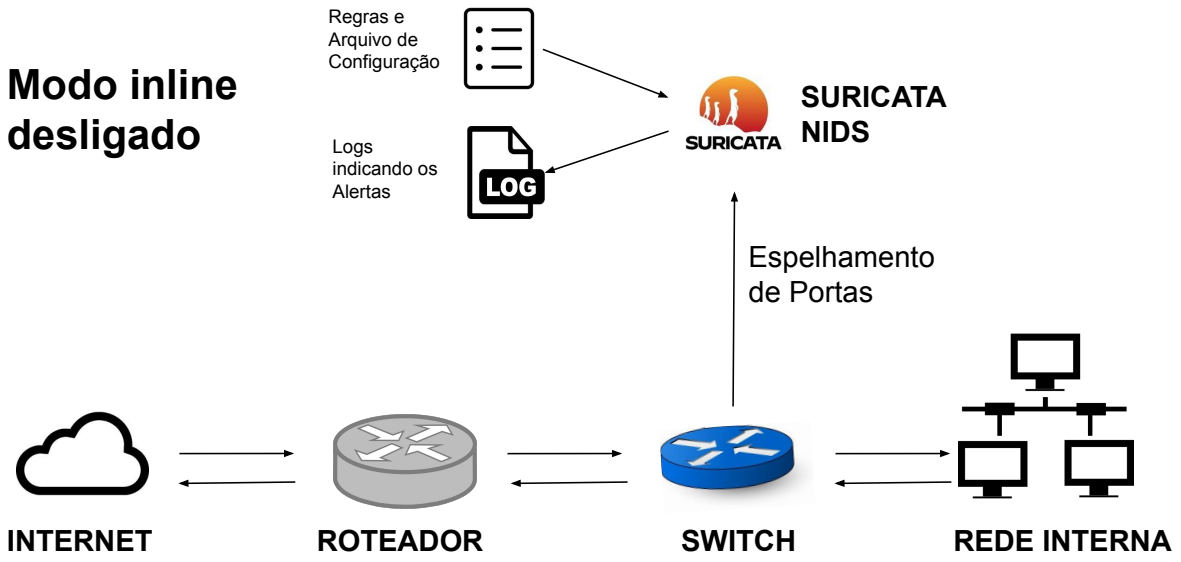



Figura 2.2: Suricata em modo de detecção puro/IDS.

de que o Suricata não será posicionado na rede em momento algum, serão sempre analisados PCAPs (modo *replay*), e depois, para os arquivos que geraram alertas, será feita uma análise mais detalhada com o *wireshark*, a fim de escrever regras mais assertivas e descriminantes.

## 2.5 *C&C - COMAND AND CONTROL*

*Command and Control* é uma categoria de ameaças que têm como principal característica o controle de um computador já comprometido previamente, e que se comunica a partir de um servidor, por meio da rede. O servidor malicioso controla a máquina infectada por meio de comandos enviados via rede, além de receber dados coletados pelo computador infectado. As formas mais comuns de infecção que levam à instalação de malware e ao estabelecimento de uma conexão com um servidor malicioso são *phishing e-mails*, falsas promoções e download de arquivo "*Trojanizados*", por exemplo; o Banbra, um dos malware detalhados no Capítulo 4, usa links falsos com o download imediato de executáveis que se apresentam como licença padrão de um antivírus, ou o episódio inédito de um seriado popular, enquando o Mekotio (também analisado no Capítulo 4) usa e-mails falsos para promover a instalação e divulgação de conteúdo malicioso.

O Tráfego entre servidor malicioso e máquina infectada pode ser detectado usando NIDS, desde que seja conhecida a estrutura dos comandos enviados pelo servidor, e das respostas enviadas pelo computador infectado. Este trabalho contará com um conjunto de regras específicas para detecção de C&C, para as famílias de malware Banload, Banbra e Mekotio.

### 2.5.1 Domínios DNS dinâmicos gratuitos usados por *C&C*

Algumas regras do Suricata são escritas de forma a detectar tráfego sobre o protocolo DNS com *queries* para um conjunto de domínios gratuitos disponibilizados por empresas que provém serviços de DNS dinâmico (noip, 2022). DNS dinâmico é um conceito legítimo e muito útil no contexto atual, no qual os IPs são majoritariamente determinados de maneira dinâmica, e mudam com certa frequência, a partir de endereços definidos pelo protocolo *DHCP - Dynamic Host Configuration Protocol*. Caso haja mudanças nos endereços dos computadores de uma

rede, é preciso que a tabela DNS contendo o nome dos domínios também tenha seus valores de endereço IP alterados. Nesse contexto entra o serviço de DNS dinâmico, que detecta as mudanças e automaticamente atualiza o banco de dados do DNS, para que não haja a necessidade de fazê-lo manualmente.

Como os domínios são disponibilizados gratuitamente, atacantes os usam para configurar um servidor malicioso, que pode servir como *C&C*, ou configuram *endpoints* para fazer download automático de um arquivo malicioso, por isso regras contendo *queries* para domínios gratuitos são comuns, e podem indicar, com base no alerta gerado por outras assinaturas, algum tipo de atividade maliciosa. Uma pesquisa da *Cisco Cloud Web Security (CWS)* (Gundert, 2014) mostra que domínios com DNS dinâmico são bloqueados (adicionados a algum tipo de *blacklist*) com uma taxa maior que todos os outros conjuntos de tráfego web, o que sugere um abuso maior desse tipo de serviço por autores de malware. A Figura 2.3 mostra os top 100 domínios dinâmicos para os quais foram feitas consultas, em um estudo que levou em consideração um banco mais de 23 milhões de exemplares de malware. É possível definir no arquivo de configuração do Suricata uma lista, contendo todos esse domínios, fazendo com que funcione como uma *blacklist*.

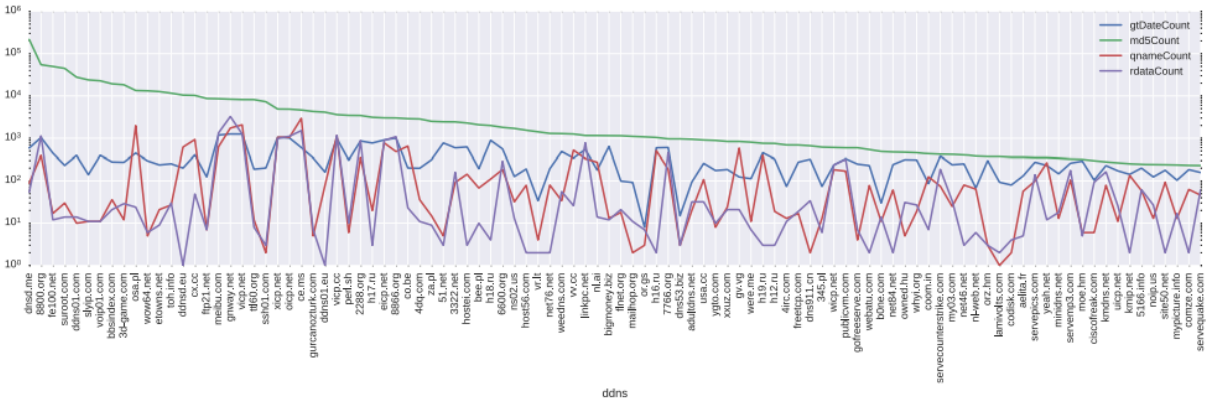



Figura 2.3: Top 100 domínios dinâmicos consultados por malware. Fonte: (Lever et al., 2017)

### 2.5.2 Código Polimórfico

O código de alguns dos mais recentes *Trojans* bancários abastecido por servidores *C&C* é polimórfico, o que significa que pode alterar-se sempre que é descarregado, para evitar a deteção baseada em assinatura. O Emotet, que foi a família de trojans bancários mais difundida no Brasil em 2020 (InforChannel, 2020), é capaz de detectar se está se está sendo executado em uma máquina virtual e fica inativo se detetar um ambiente de *sandbox*, trazendo alto nível de dificuldade para realizar sua detecção em contexto geral, e quase impossibilitando a detecção via NIDS.

## 2.6 DETECÇÃO POR ANOMALIA *VERSUS* DETECÇÃO POR ASSINATURA

Sistemas de instrusão em redes baseado em detecção por anomalia têm como principal função determinar o comportamento da rede. Todo o tráfego que não estiver de acordo com esse comportamento pré definido (também chamado de acordo), gera um alerta (Jyothsna et al., 2011). A construção do comportamento em si ocorre a partir de 3 módulos: (1) detecção estática (cadeias de markov, decisões operacionais, desvio padrão), (2) máquinas de estado (mudanças no input causam transições na máquina) e (3) apresendizado de máquina (redes bayesianas, algoritmos genéticos, redes neurais, detecção de *outliers*).

Esse tipo de detecção (por anomalia) é mais complexa e mais custosa que a detecção por assinatura, uma vez que cabe aos administradores da rede definir qual é o acordo, ou seja, qual é o comportamento esperado da rede para cada um dos protocolos (Figura 2.4). Além disso, o sistema deve ser capaz de analisar cada protocolo separadamente, a fim de incrementar o conjunto de regras. Somado-se a isso, o sistema de intrusão que usa detecção por anomalia conta com diversos módulos, cada um responsável por fazer uma tarefa de alto custo computacional (rodar um algoritmo de aprendizado de máquina, por exemplo), ao passo que o sistema que usa detecção por assinatura, simplesmente precisa carregar e fazer análise sintática das regras, e já pode iniciar a inspeção do tráfego.

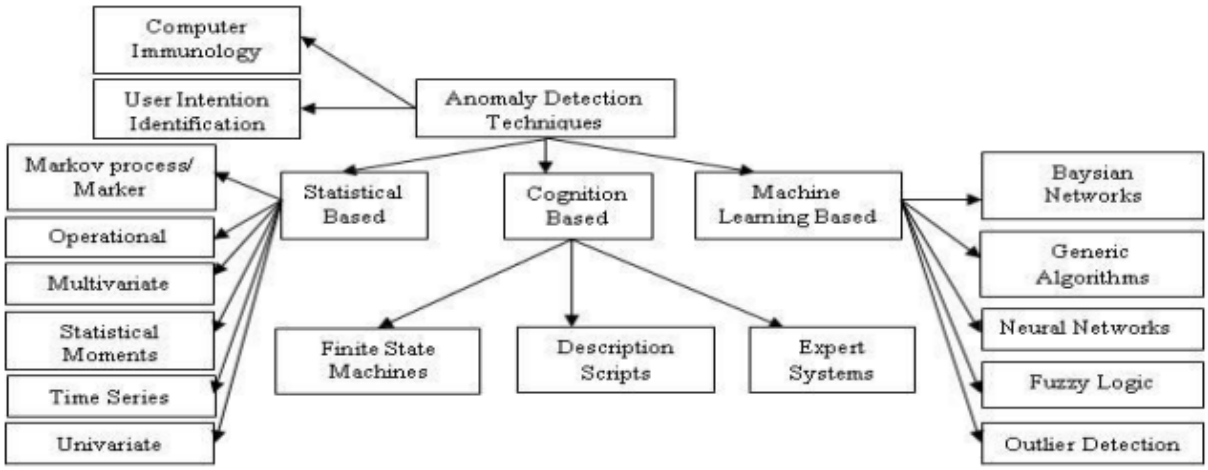



Figura 2.4: Arquitetura de um NIDS baseado em detecção por anomalias. Fonte: (Jyothsna et al., 2011).

## 2.7 REVISÃO DA LITERATURA

(Lever et al., 2017) tiveram como principal objetivo analisar durante um longo período (5 anos) de malware detectado por 3 fontes diferentes. Com a execução dinâmica de mais de 23 milhões de exemplares de malware, eles obtiveram 26,8 milhões de traços de rede. Esse trabalho reporta que para a vasta maioria de exemplares maliciosos, a inspeção do tráfego de rede é a maneira que melhor consegue antecipar a existência de um novo domínio malicioso. Além disso, foi descoberto que domínios maliciosos estavam ativos antes de serem classificados como tal e adicionados a uma *blacklist*, implicando que o tempo de detecção de novos domínios de *C&C* é alto, e até que ele seja tratado a infecção provavelmente já ocorreu. O trabalho também mostra técnicas de filtragem de domínios maliciosos, já que nem todo o tráfego gerado pelo malware é feito para servidores maliciosos. Também mostra-se que um exemplar de malware gera em média requisições para 10 domínios diferentes, e que a consulta para boa parte dos domínios realizada aconteceu somente uma vez. Isso indica alta rotatividade de endereço dos domínios programado pelos autores dos malwares, geralmente usando um algoritmo de geração de domínios, e também que criar uma *blacklist* passa a não ter efetividade.

(Rieck et al., 2010) abordaram uma maneira de realizar a geração das assinaturas de malware automaticamente, com base em um conjunto de tráfego malicioso coletado utizando-se *honeypots*. O malware de maior interesse desse trabalho são os RATs (chamados de *botnets*), e as assinaturas são criadas de forma a capturar o tráfego com origem no sistema infectado, indo para o domínio malicioso, esse tráfego foi chamado de *"Phoning Home"*. Para gerar uma grande quantidade de tráfego *"Phoning Home"*, cada binário (malware) foi executado diversas vezes, e a cada vez mudava-se algum parâmetro do ambiente em que ele estava sendo executado, por

exemplo o IP, sistema operacional, horário do sistema, etc. Também guardou-se tráfego benígno. Para gerar as assinaturas, cada *string* do payload dos pacotes virava um *token*, e com base nas frequencias de cada token nos tráfegos malicioso e benigno, eram estabelecidas as novas regras. Na época, os resultados alcançados com essaa regras detectam mais de 94% do tráfego *"Phoning Home"*.

(Bekerman et al., 2015) utilizaram um algoritmo de aprendizado supervisionado para detecção de comunicação maliciosa, a partir do tráfego entre o malware e o servidor *Command & Control*. A grande contrubuição desse trabalho é a capacidade de detectar (antecipar) malware desconhecido, baseado nos dados colhidos das comunicações feitas por malware já estudados. O primeiro passo é extrair as *features* para o algoritmo de AM (Aprendizado de Máquina), para isso, um dos módulos do extrator é responsável por remontar o tráfego de acordo com as camadas de rede, e retirar de cada uma das camadas, para cada um dos protocolos, algumas *features*, por exemplo, da camada de aplicação são extraídas *features* dos protocolos HTTP, SSL e DNS; para o HTTP: são coletados User-Agent, Cookie, Content Type, Hostname e Referrer, e para cada um dos protocolos de cada uma das diferentes camadas são extraídas outras características. O resultado é um arquivo CSV de características, passados para 3 diferentes algoritmos: Naïve Bayes, Decision Tree e Random Forest, para os quais os rótulos foram classificados usando NIDS convencionais (Suricata e Snort). O resultado mostrou que os rótulos determinados pelos algoritmos eram úteis para separar tráfego de malware dos quais foram ou não extraídos *features*, mostrando a capacidade do sistema de classificar como malicioso até mesmo tráfego de malware inédito.

(Lu et al., 2020) identificaram as características principais do *ransomware "wannacry"*. O trabalho estudou detalhadamente o passo a passo da execução de cada módulo que compõe o malware a partir de engenharia reversa, com a utilização da ferramenta *IDA - Interactive Disassembler*. A partir disso conseguir determinar de que maneira cada módulo usava a rede. Foi descoberto que diversos comportamentos eram detectáveis por NIDS, como varredura de portas, uso do protocolo SMB entre cliente e servidor malicioso para descobrir se uma vulnerabilidade está ativa ou foi consertada. Se a vulnerabilidade estiver aberta, é possível detectar a troca de mensagens entre servidor e malware, contendo comandos e informações do *host* atacado. Na Seção de Resultados, é apresentado um conjunto de características dos módulos do malware responsáveis por explorar a vulnerabilidade *EternalBlue* e *DoublePulsar*, que podem ser tranformadas em regras de NIDS. Geradas as regras, foi mostrado que a abordagem utilizada gerou resultados melhores que as regras padrões e mais recentes do Snort e Suricata.

# 3 METODOLOGIA E EXPERIMENTOS

Neste capítulo, mostra-se o conjunto dos arquivos de tráfego usados na composição da monografia. Além disso, será explicado como o Suricata foi usado para realizar a análise desse tráfego, e quais os resultados da aplicação das regras do antivírus VIPRE contra esses arquivos de captura.

## 3.1 DATASET

O conjunto de arquivos usados para realizar esse trabalho é composto por exemplares de malware submetidos ao sistema Corvus (SECRET, 2019) entre 2019 e 2021. Corvus é um sistema de análise estática e dinâmica de malware, com técnicas de classificação baseadas em aprendizado de máquina. Os arquivos submetidos são executados em uma *sandbox*, permitindo um ambiente controlado e a caracterização do artefato como malicioso ou não, com base em uma série de testes executados no próprio *backend* da Corvus, como também fazendo chamadas de outros serviços (Botacin et al., 2021), como por exemplo o *VirusTotal*.

Os rótulos gerados pelos antivírus são sabidamente ruidosos e até mesmo diferentes entre si (Mohaisen e Alrawi, 2014). O AVClass (Sebastián et al., 2016) é capaz de remover parte desse ruído aplicando técnicas de normalização de rótulo, detecção de tokens genéricos e pseudônimos. A ferramenta recebe uma entrada contendo a rotulação devolvida de um executável por diferentes antivírus, e devolve para qual família de malware há maior probabilidade desse executável pertencer, além do fator de confiança baseado em semelhanças entre as rotulações estabelecidas pelos antivírus.

A partir do relatório fornecido pelo *VirusTotal*, o sistema Corvus usa o *AVClass* para determinar, com base nas respostas dos antivírus, se o arquivo submetido é considerado suspeito, e se sim, a qual família de malware ele pode partencer.

Corvus aceita a submissão de arquivos ELF, APK, PE e PDF, mas não aceita PCAPs, somente os captura se o arquivo submetido usar a rede. Ao perceber que o arquivo submetido tenta fazer alguma requisição via web, o Corvus automaticamente guarda esse tráfego em formato PCAP. Com isso, é possível associar um arquivo executável com o tráfego que ele gera. Todo o conjunto de arquivos submetidos ao Corvus no período analisado foi englobado num arquivo JSON, que contém, para cada arquivo único, representado por um hash MD5, um conjunto de informações apontadas pelo Corvus, desde análises estáticas e dinâmicas, até classificação por meio de inteligência artificial.

Entre 2019 e 2021, foram submetidas ao sistema Corvus 6485 análises, das quais resultaram na coleta de 4612 arquivos de captura de tráfego. Na tabela 3.1 mostra-se as TOP 10 família de malware mais populares no *dataset*, bem como a quantidade de PCAPs. Nota-se que entre as famílias mais populares do *dataset*, destacam-se Banload, AutoIt, Mekotio (em quarto lugar) e Banbra (em quinto lugar), que são as famílias analisadas neste trabalho, e correspondem a exemplares de malware envolvendo *Trojans* bancários. A partir de todos os arquivos submetidos, tem-se por meio do JSON quais deles pertecem às famílias de malware alvos desse trabalho (as famílias que envolvem ataques conhecidos a usuários de *Internet Banking*). A partir dai, foi analisado que a família Banload foi a que mais continha exemplares, seguido de AutoIt, Mekotio e Banbra (o Capítulo 4 mostrará que tanto o Banbra como o Mekotio precisam do Banload para serem executados, por isso a superioridade nas amostras). Então essas foram as famílias

| Família | Quantidade de PCAPs |
|---|---|
| Banload | 1002 |
| AutoIt | 292 |
| Delf | 204 |
| Mekotio/Bestafera | 193 |
| Banbra | 110 |
| Agenttesla | 93 |
| Xorala | 86 |
| Agensla | 68 |
| Razy | 63 |
| Zusy | 54 |

Tabela 3.1: TOP 10 famílias de malware mais populares no *dataset*

escolhidas para a condução das análises, para os quais os resultados serão expostos também no Capítulo 4. A Figura 3.1 mostra o fluxograma das etapas descritas acima.

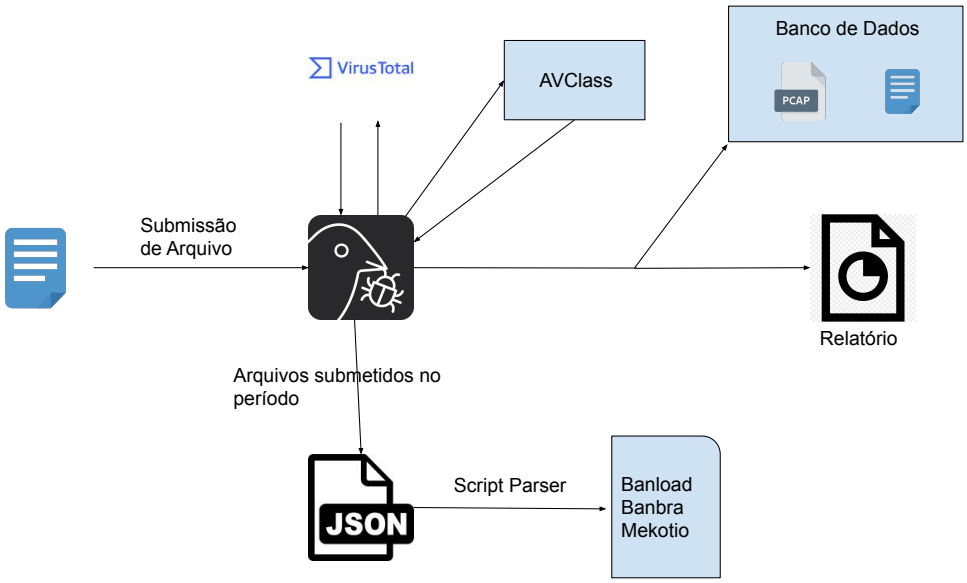



Figura 3.1: Obtenção do dataset através da Corvus.

Após a filtragem dos arquivos, foram obtidos 1597 arquivos de tráfego PCAPs para serem analisados a partir do NIDS, cada um deles associado a um malware.

## 3.2 USO DO SURICATA

Como dito na Seção 2.4, o Suricata foi utilizado em modo *replay*. Dessa maneira, é possível especificar um conjunto de arquivos PCAP (aqueles extraídos de arquivos submetidos na Corvus, detalhados na Seção 3.1) para serem analisados contra um conjunto fixo de regras. Para a primeira rodada de experimentos, o Suricata foi instanciado com um conjunto livre de regras, conhecido como *ET OPEN Ruleset*, onde ET significa *Emerging Threats* (rules.emergingthreats.net, 2022). Essas regras geraram alertas para as famílias analisadas, e foram úteis tanto na construção das regras deselvolvidas nesse trabalho, quanto no entendimento geral do comportamento desses tipos de malware.

### 3.2.1 Regras *Filestore*

Depois de analisar os PCAPs que geraram alertas para uma determinada família, foi possível criar regras ainda mais específicas, alertando se o tráfego em questão fazia alguma requisição com o objetivo de obter (fazer download) de algum arquivo pela rede. Os PCAPs que geravam alertas para essas regras foram submetidos a uma regra do tipo *filestore*. Esse tipo de regra é capaz de extrair os arquivos movimentados pelo arquivo de tráfego. Depois dessa extração, temos 3 artefatos importantes: (1) O arquivo executável, classificado pelo AVClass como malicioso (mais especificamente contra usuários de *Internet Banking*), (2) o PCAP que representa as requisições e respostas que esse executável fez através da rede, e (3) o arquivo que foi movimentado através de rede. A Figura 3.2 indica os *inputs* do Suricata para ter como resultado um conjunto de arquivos extraídos, nota-se que essa metodologia foi aplicada somente a família Banload, pois esse tipo de malware tem como uma de suas principais funções o download e execução de malware outras famílias de Trojans bancários, que serão extraídos a partir do tráfego de exemplares de malware da família Banload. A regra *Filestore* em questão é apresentada na Listagem 3.1, sua função é armazenar em memória secundária qualquer arquivo enviado de qualquer endereço sob qualquer porta, através do procolo HTTP:

Listing 3.1: Regra Filestore Genérica

```
1 alert http any any -> any any (msg:"Regra filestore";
2 filestore; sid:1; rev:1;)
```

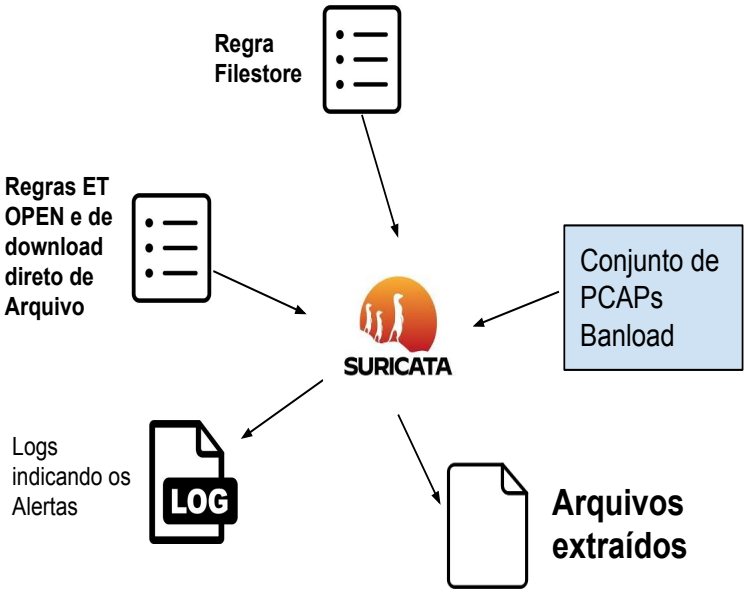



Figura 3.2: Obtenção de arquivos através de regras filestore.

A regra *Filestore* teve como principal função colocar em evidência os PCAPs que faziam requisição e de fato conseguiam fazer o download dos arquivos. Esses PCAPs foram considerados os mais importantes entre os 1002 analisados, e foram de fundamental importância para o desenvolvimento das regras envolvendo a família Banload. Por escassez de tempo para continuação dos experimentos, os arquivos extraídos do tráfego não foram executados nem submetidos ao Corvus, mas serviram como indicativo de pós infecção de malware Banload. A continuidade da análise dos arquivos provenientes do tráfego malicioso é deixada como proposta para trabalhos futuros.

## 3.3 IDENTIFICAÇÃO DE TRÁFEGO MALICIOSO POR ANTIVÍRUS

Os antivirus são a solução mais usada pelas pessoas em contexto geral para detecção e remoção de ameaças (Lévesque et al., 2015). Esse tipo de software é dividido em diversos módulos, como por exemplo sistemas de arquivos, rede, drivers, processos, bibliotecas, bancos de dados, entre outros (Botacin et al., 2022). Dentre as funções de um antivírus, está a inspeção de rede, ou seja, rotinas de varredura dos pacotes que tem como objetivo detectar algum padrão conhecido de ataque ou vulnerabilidade. São três as principais abordagens que um antivírus utiliza para fazê-lo: (1) utilizar alguma extensão do navegador para ler o conteúdo de uma página antes que ele seja de fato executado pelo navegador; (2) usar um servidor `proxy`, que passa a requisição por algum processo interno ao antivírus, antes de ser redirecionada ao cliente e (3) checar as requisições depois que elas foram estabelecidas, por meio de drivers. Essas três abordagens serão expostas com mais detalhes abaixo. A maioria dos antivírus usa a terceira abordagem, ou seja, carregar drivers de Detectores de Instrusão para assegurar políticas de segurança nas conexões. Essas políticas na verdade são regras de NIDS, que podem ser escritas com os mais diversos objetivos, como por exemplo criar um blacklist de endereços, verificar tráfego por protocolos, identificar assinaturas de malware com base em conteúdos e expressões regulares, bem como executar funções de um NIDS, descritas no Capítulo 2. O foco dessa seção é o estudo das regras do NIDS Snort implementadas pela versão *freeware* do antivírus VIPRE, como estão distribuídas e de que maneira detectam e se comportam com fluxos de tráfego sabidamente maliciosos.

### 3.3.1 Extensões do Navegador

Nessa abordagem, há uma extensão do navegador que possui acesso ao *Document Object Model (DOM)* da página para qual foi feita a requisição. Esse documento possui todas as *tags* HTML e seus conteúdos, que podem ser inspecionados pelo próprio antivírus, além disso, é possível que o navegador não permita a conexão, depois de se comunicar com a *engine* do antivírus e considerar que a reputação da URL não é boa, ou que há algum tipo de conteúdo malicioso no DOM.

Essas extensões na verdade obedecem a um modelo cliente servidor, em que a extensão do navegador é um cliente definido pelo antivírus, enquanto a *engine* do antivírus é o servidor. Além de inpecionar a reputação da URL, a *engine* ainda pode ter a função de inspecionar campos de formulários e scripts contidos no DOM. A maior parte desse processo é realizado de maneira autônoma pelo antivírus, sendo a interação com o usuário requerida somente quando é necessário a confirmação que ele realmente deseja acessar o site.

### 3.3.2 Desvio do Tráfego

Os antivírus que realizam essa abordagem desviam o fluxo de pacotes para permitirem que estes sejam analisados, ou seja, realizam uma intercepção *Man in The Middle (MITM)*. Para isso, há um servidor *proxy* que faz de fato o desvio dos pacotes, que passam por um processo disparado pelo antivírus, que por sua vez certifica que o tráfego não levantou suspeitas, ou que de fato pode ser malicioso, gerando uma mensagem do tipo INVALID_CERTIFICATE.

Essa maneira de detecção, apesar de permitir inspeção do tráfego, diminui a vazão da rede, e dobra o número de *handshakes* necessários, uma vez que o servidor deve fazê-lo com o cliente e também com o servidor proxy do antivírus.

### 3.3.3 Drivers de NIDS

A maioria dos antivírus fazem a inspeção dessa maneira, depois da conexão já estar estabelecida. São lançados a partir do Kernel, *drivers* que fazem interação com detectores de instrusão, e transmitem os alertas e ações tomadas para o *engine* do antivírus. Firewalls intrínsecos aos antivírus são implementados justamente via esses drivers, usando NIDS. Grande parte dos antivírus implementa NIDS especiais para seu caso de uso, o único que implementa usando um sistema de código aberto é o VIPRE, que usa o Snort (Botacin et al., 2022).

### 3.3.4 Resultado do teste das regras usadas pelo VIPRE

As regras analisadas nesse capítulo são usadas por uma versão gratuita (demonstração) do antivírus VIPRE (versão: 11.0.4.2; md5: 77a9dbd31ed5ebe490011ffa139afe03), para análise no trabalho *AntiViruses under the Microscope: A Hands-On Perspective* (Botacin et al., 2022), do qual o autor dessa monografia foi coautor.

O antivírus VIPRE usa o Snort como NIDS. As 466 regras estão armazenadas em `\ProgramData\VIPRE\Rules\idsrules.dat` e são classificadas de acordo com a assinatura do malware que tentam detectar (Botacin et al., 2022). Nessa primeira abordagem, vamos usar todo o tráfego que , pela metodologia descrita na Seção 3.1, foi classificado como pertecendo à uma das famílias alvo desse trabalho.

Ao instanciar o Snort em modo *replay* com o PCAPs gerados pela execução dos exemplares de malware das famílias estudadas nesta monografia e as regras do VIPRE, temos como resultados que **nenhuma das 466 regras gerou algum alerta** para os 1597 pcaps, das famílias AutoIt, Banload, Banbra e Mekotio/Bestafera. Esse resultado foi inesperado no sentido que os PCAPs continham sabidamente exemplos de tráfego de malware em arquivos executáveis.

Primeiramente, é interessante ressaltar que as regras usadas pelo antivírus VIPRE possuem três campos a mais que as regras padrão do Snort (e Suricata), um é para controle interno do identificador dessas regras (SBRuleID), o segundo para determinar o risco que o alerta gerado para essa vulnerabilidade representa (SBRiskLevel), e o terceiro é a categoria que a empresa adota para essa intrusão em geral, chamado de SBCategory. Destaca-se que o prefixo 'SB' significa SunBelt, que era como a empresa agora denominada VIPRE chamava-se antes de 2013. A Listagem 3.2 apresenta um exemplo do uso desses campos, todos os outros não mencionados são padrões de regras do Snort.

Listing 3.2: Regra Snort do VIPRE

```
1 alert tcp $EXTERNAL_NET any -> $HOME_NET 445 (SBRuleId:xx;
2 msg:"... - MS08-067"; flags:A+; content:"|1F 00|";
3 content:"|C8 4F ...|"; content:"|00 2E ...|";
4 classtype:attempted-admin; reference:url,www.microsoft.com/
5 technet/security/Bulletin/MS08-067.mspx; rev:x; sid:xxxxx;
6 SBRiskLevel:x; SBCategory:"attempted-admin";)
```

O trabalho anterior concluiu que 6,62% das regras eram específicas para detecção de *Trojans*. Dentre essas regras, nota-se que existem verificações a respeito de tráfego *Command and Control*, como por exemplo o *worm Slappper*, as vulnerabilidades *EthernalBlue* e *DoublePulsar* sob o protocolo SMB, *buffer overflow* no serviço WebDAV da Microsoft, vulnerabilidades no controle de acesso do *spooler* de impressão, permitindo ataques remotos no sistema de arquivos, entre outros. Com essas observações, nota-se que as intrusões detectadas pelas regras do VIPRE *Freeware* são de fato úteis e detectam ataques importantes, mas **falham em identificar ataques de Trojans bancários**, não sendo possível encontrar nenhuma regra que ao menos tente fazê-lo.

No entanto, o que é possível afirmar após essa análise é que o módulo de NIDS para detecção por assinatura do antivírus não é capaz de identificar *Trojans* bancários, mas não pode-se dizer que o antivírus em geral não o faça, pois os outros módulos do antivírus não foram testados com esse propósito nesta monografia. Além disso, também há a possibilidade da versão paga do antivírus possuir um conjunto de regras diferente deste que foi análisado nos trabalhos.

## 3.4 CONCLUSÃO

Esse capítulo mostra que os arquivos de tráfego e malware utilizados nos experimentos desta monografia foram coletados usando o sitema Corvus, com base em análises submetidas entre 2019 e 2021. Ademais, mostrou-se que as submissões foram compiladas em um arquivo JSON, que foi analisado com o objetivo de encontrar as famílias de malware mais significativas que apareceram durante o período. A conclusão foi que famílias de malware com exemplares de *Trojans* bancários tiveram alta popularidade no *dataset*, por isso foram escolhidas. Além disso, esse tipo de malware representar grande ameaça no contexto atual de ataques que acontecem no Brasil.

Ainda, neste Capítilo mostrou-se que o Suricata será usado em modo *offline*, usando inicialmente regras do conjunto *ET OPEN Ruleset*. Detalhou-se o uso de regras *Filestore* para indícios de malware Banload fazendo requisições para o download de outros *Trojans* bancários.

Finaliza-se abordando as maneiras que diferentes antivírus usam para realizar detecção de indícios maliciosos em tráfego de rede. Mostra-se que o VIPRE usa o Snort como NIDS, e que nenhuma das regras analisadas usadas pela versão *freeware* desse AV funcionou para detecção dos *Trojans* bancários analisados neste trabalho. Essa conclusão corrobora com a necessidade de caracterização desse tipo de malware, principalmente no cenário nacional. O Capítulo 4 trará o conjunto de assinaturas que define evidências de pré e pós-infecção para as famílias de *Trojans* bancários abordadas nesta monografia.

# 4 CARACTERIZAÇÃO DAS ASSINATURAS

Apresentam-se neste capítulo os resultados alcançados da aplicação da metodologia mostrada no Capítulo 3 para detectar as seguintes famílias de malware: Autoit, Banload, Banbra e Mekotio. Essas famílias foram escolhidas porque representam malware envolvido em ataques a sistemas de *Internet Banking* brasileiros, além de figurar nas TOP 10 famílias com maior número de exemplares no *dataset*. Na descrição dos rótulos desses exemplares de malware, ficará visível que de fato eles tem alto potencial para realizar ataques contra conhecidos bancos brasileiros, e quais características fazem com que eles sejam usados para esse tipo de ataques.

## 4.1 AUTOIT

Scripts escritos em AutoIt são úteis e poderosos. AutoIt é uma linguagem de programação que permite a automatização de tarefas no sistema operacional Windows (autoitscript.com, 2022), por exemplo, a escrita de um arquivo usando o bloco de notas, ou a instalação de um mesmo programa em todos os computadores de uma organização, em que pode-se escrever um script que ao mesmo tempo baixa, instala e configura esse programa, garantindo que o processo de instalação seja padronizado e rápido. Usando o código da Listagem 4.1, é possível:

- rodar o executável do bloco de notas;
- selecionar a janela que for corresponde à essa nova instância do editor;
- enviar as linhas que irão compor o documento separadas por *Enter* (quebra de linha);
- mandar um sinal para fechar a janela. Nesse momento o comportamento padrão no Windows é: perguntar se há ou não necessidade de salvar o conteúdo;
- confirmar a intenção de salvar;
- definir o caminho em que esse arquivo será salvo;
- fechar o arquivo

Listing 4.1: Exemplo de escrita de um arquivo usando *notepad.exe* com AutoIt

```
1  ; Executar bloco de notas
2  Run("notepad.exe")
3
4  ; Selecionar janela com este título
5  WinWaitActive("Sem título - Bloco de Notas")
6
7  ; Linhas a serem escritas no arquivo
8  Send("import time")
9  Send("{enter}")
10 Send("time.sleep(3)")
11 Send("{enter}")
12 Send("print('Teste')")
13 Send("{enter}")
```

```
14 Send("time.sleep(5)")
15
16 ; Ação de fechar janela, nesse momento abrirá uma janela de '
     Salvar como'
17 WinCLose("*Sem título - Bloco de Notas")
18
19 ; Confirmar intenção de salvar o arquivo
20 WinWaitActive("Bloco de notas", "Salvar")
21 Send("!s")
22
23 ; Selecionar Janela 'Salvar como'
24 WinWaitActive("Salvar como")
25
26 ; Selecionar caixa de texto com o caminho do arquivo a ser
     salvo
27 ControlFocus("Salvar como", "", "Edit1")
28 ; Passar caminho do arquivo
29 ControlSend("Salvar como", "", "Edit1",
30     "C:\Users\Computador\Desktop\autoit\teste.py"
31 )
32
33 ; Salvar arquivo
34 ControlClick("Salvar como", "Sa&lvar", "Button2")
35 WinWaitActive("teste.py - Bloco de Notas")
36
37 ; Fechar a janela
38 WinClose("teste.py - Bloco de Notas")
39
40 ; Passar o caminho do executável do python,
41 ;que executará o arquivo que acaba de ser criado.
42
43 Run("C:\Users\Computador\AppData\Local\Microsoft\WindowsApps\
44 PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\python.exe
     teste.py")
```

Como resultado da execução do script acima, temos a abertura de um terminal que executa o trecho em python que foi criado pelo script (Figura 4.1).

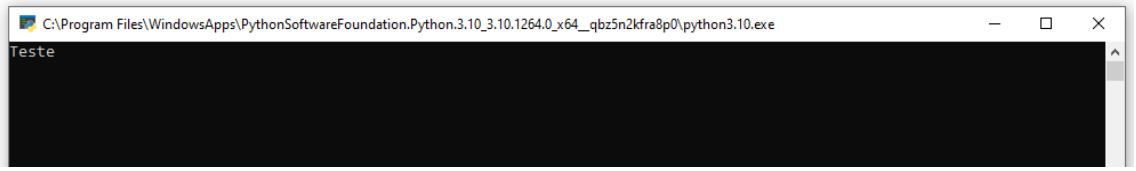



Figura 4.1: Terminal Python executando as ações definidas no script AutoIt.

Na listagem 4.1, é possível perceber que AutoIt é uma linguagem muito útil no contexto de automatizações, e ao mesmo tempo abre brecha pra diversas abordagens maliciosas, como por exemplo, escrita e execução de arquivo em qualquer linguagem de programação de maneira transparente ao usuário, ou seja, programas podem ser compilados e executados sem que o usuário fique sabendo, podendo infectar facilmente o computador atacado.

O intuito do exemplo (Listagem 4.1) acima foi simplesmente mostrar como o AutoIt é de fato uma linguagem de script legítima e útil para automatizações de tarefas no Windows. Na listagem 4.2, tenta-se reproduzir um código potencialmente malicioso usando AutoIt, e em seguida, descreve-se mais detalhes.

Listing 4.2: Exemplo de requisição para servidor comprometido com AutoIt

```
#include <WinAPIFiles.au3>

; Criando um caminho temporário
Local Caminho = _WinAPI_GetTempFileName(@TempDir)

; Baixa um arquivo em Background e salva no caminho temporário
Local Arquivo = InetGet(
  "http://www.sitemalicioso.com.br/malware/malware-setup.exe",
  Caminho,
  INET_DOWNLOADBACKGROUND
)


; Espera o download completo do arquivo
Do
  Sleep(250)
Until InetGetInfo(Arquivo, INET_DOWNLOADCOMPLETE)

; Encerrar Conexão
InetClose(Arquivo)

; Executar
Run(Caminho & "malware-setup.exe")

; Abrirá uma janela para instalação, seguir os passos até
; que a instalação seja totalmente concluída.
```

No código acima, nota-se um comportamento comum entre os exemplares de malware com comportamento *Trojan*: o ato de baixar conteúdo da internet com o intuito de infectar a máquina do usuário, instalando programas maliciosos já disponibilizados na rede anteriormente. Nota-se que uma vez que esse script é colocado em execução, são feitas requisições pela rede, downloads e instalações, tudo de forma transparente ao usuário, ou seja, esse tem a sua máquina infectada e pode não ter ciência desse fato. Além disso, muitas outras funções nativas do AutoIt podem ser utilizadas de maneira a ter uma taxa de sucesso ainda maior ao executar o script. Por exemplo, há a possibilidade de mandar um *ICMP echo request* (PING) até o servidor de onde será baixado o arquivo, pra ver se o mesmo está no ar. Outra possível abordagem é utilizar a função `DllCall('dll.dll', BOOL, 'nome_da_funcao')`, que executa a função `nome_da_funcao`, da biblioteca vinculada dinamicamente `dll.dll`, que também pode ser obtida pela rede.

Até aqui, é possível perceber que há um potencial de abuso evidente em scripts AutoIt, no sentido em que eles podem criar, ler, modificar ou deletar arquivos e pastas, fazer download de arquivo via rede e armazená-los em um caminho específico, carregar uma biblioteca dinamicamente linkada e executar um programa. Todos esses passos são legítimos, mas ao

mesmo tempo permitem a instalação de malware, podendo gerar até mesmo persistência (por exemplo, movendo o executável instalado para a pasta *StartUp*). A seguir, serão descritas maneiras de detectar scripts AutoIt fazendo requisições via rede por meio de NIDS, e explicar o que a detecção desse tipo de tráfego pode signicar.

### 4.1.1 Detecção de malware AutoIt usando NIDS

Nesta seção, mostra-se o passo-a-passo de como detectar malware com o comportamento descrito acima usando regras de um sistema de detecção de intrusão de rede. Primeiramente, é necessário deixar claro que por ser um sistema de detecção de intrusão em redes, o que será feito é uma análise do que o malware faz de fato usando a rede para obtenção de conteúdo malicioso. Então, serão escritas regras que detectem o comportamento desse tipo de malware, para que seja possível detectar as requisições feitas pelo mesmo.

Deve-se atentar que estamos tratando aqui somente da parte de rede, e que o script fará requisições e o servidor mandará respostas usando o protocolo HTTP. Mais do que isso, temos que por padrão as **requisições feitas por um script AutoIT utilizam no User-Agent a própria string "AutoIt"**. Esse é o primeiro passo, e não o único, para identificar esse tipo de comunicação.

Então, até o momento, sabe-se que se em um pacote existe na parte de User-Agent do cabeçalho HTTP a string "AutoIT", essa requisição potencialmente foi feita a partir de um script AutoIt. Importante salientar que essa é de fato a única informação que obtida até agora, não sabemos se a comunicação é legítima ou se está de fato fazendo o download de algum arquivo. Para determinar se essa última opção, pode-se inspecionar novamente o *payload* do pacote, dessa vez procurando por alguma coisa que indique uma extensão de arquivo na URL, que é um bom indicativo de uma tentativa de download de um arquivo desse tipo.

Esses dois fatos já nos concedem uma grande quantidade de informação. Com isso, podemos definir uma regra, apresentada na Listagem 4.3:

Listing 4.3: Detecção de requisição a partir de script AutoIt

```
1 alert http $HOME_NET any -> $EXTERNAL_NET any (
2 msg:"Achado Requisição com User-Agent AutoIt";
3 flow:established,to_server; http.method; content:"GET";
4 http.uri; content:".exe"; nocase; endswith; http.user_agent;
5 content:"AutoIt"; depth:6; endswith; fast_pattern;
6 classtype:trojan-activity; sid:1; rev:1;
```

Essa regra é capaz de detectar uma requisição de um script AutoIt baixando um arquivo com extensão exe. Ela procura por uma requisição HTTP saindo da máquina local em direção ao servidor (linha 1), com o método GET do HTTP (linha 3), contendo no final da URI a string ".exe"(linha 4), e contendo, após 6 caracteres do campo de User-Agent, a string "AutoIt"(linhas 4 e 5). Ao analisar essa regra, é possível perceber que ela pode ser aplicada para qualquer extensão de arquivo, não somente '.exe'.

A pergunta que surge é: Como é possível afirmar com certeza se o que de fato está sendo baixado pela rede é malware? Não poderia ser uma execução legítima baixando alguma dependência externa? E a resposta é que sim, poderia. Sabemos somente o tipo do arquivo sendo baixado, não a legitimidade do mesmo. Essa regra permite descobrir se há algum script AutoIt malicioso sendo executado, mas não garante que não surgirão falsos-positivos. O que pode-se fazer é tentar aumentar o conjunto de regras que está sendo usada para buscar comunicação de malware AutoIt, e é isso o que será tratado a seguir.

### 4.1.2 Conjunto mínimo de regras para potencial malware AutoIt

Como mencionado, a regra mostrada anteriormente é capaz de detectar comunicação de malware AutoIt, como também pode detectar comunicação de um script AutoIt legítimo, como no exemplo citado na introdução da linguagem, em que um script pode fazer o download e instalar um programa específico, garantindo padronização e rapidez. Para tentar ser mais assertivo ao identificar esse tipo de malware, é preciso um conjunto maior de regras que seja capaz de identificar outros momentos antes e depois da requisição principal, para a qual será gerado um alerta com base na regra anterior. Essa seção trata de quais outras características podem ser analisadas para confirmar se de fato havia intenção maliciosa no script AutoIt.

Pode-se então primeiramente definir uma regra que diz se há ou não requisições para download de outros arquivos. Esse é um bom sinal que há um script responsável por fazer essas requisições. A regra resultante é mostrada na Listagem 4.4:

Listing 4.4: Detecção de requisição a partir de script AutoIt para diversas extensões

```
alert http $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (
msg: "Download de arquivo por script AutoIt";
flow:established,to_server;http.method; content:"GET";
http.uri; pcre:"/(\.txt|\.tiff|\.zip|\.dll|\.exe|\.au3)/U";
http.user_agent; content:"AutoIt"; classtype:trojan-activity;
sid:2; rev:1;)
```

Essa regra faz uso de expressões regulares (PCRE - *Pearl Compatible Regular Expressions*) para detectar extensões de arquivo na URI do HTTP. Além disso, checa pela string "AutoIt"no User-Agent, além do método GET. **Quanto mais vezes essa regra gerar um alerta, maior a chance do conteúdo ser malicioso**, pois estão sendo feitas várias requisições para baixar arquivos a partir de um script AutoIt.

Além dessa regra geral para detectar download de arquivos com diversas extensões, há um grande problema na identificação do malware pela rede de maneira genérica. O problema se deve ao fato de não haver mais maneiras de prever o que o script pode fazer, e deve-se fazer uma análise do tráfego de rede para ter mais detalhes sobre o tipo de malware e para onde foram feitas as requisições para o download dos arquivos não desejados, para que seja possível escrever regras específicas contra um caminho específico na internet que disponibiliza esses arquivos. Então, a partir do momento que for notada que pelo menos uma das duas regras acima disparou um alerta, é possível utilizar uma ferramenta de análise de tráfego (o *Wireshark*, por exemplo) para obter mais informações. Isso pode ser feito desde que o NIDS seja configurado de maneira a guardar o tráfego que gerou alertas por meio de PCAPs. A Seção 4.1.3 mostrará como obter informações do wireshark a fim de escrever regras mais completas.

Nas seções 4.2 e 4.3, serão definidas as famílias de malware Mekotio e Banload, que usam o AutoIt para carregar DLLs com códigos maliciosos, além de fazer o download de arquivos específicos para esses tipos de malware. Podemos afirmar então que a família AutoIt, para esse trabalho, funciona como uma técnica auxiliar, que permite o download de conteúdo malicioso via rede, mas também permite sua detecção via NIDS, o que pode ser de grande ajuda em capturar sinais de pós-infecção com os alertas gerados.

### 4.1.3 Análise de tráfego que gerou alerta

A fim de analisar o tráfego associado aos alertas gerados pelo Suricata, deve-se procurar pelo protocolo HTTP. Com isso, localiza-se mais detalhes sobre a requisição do User-Agent

que fez o download do arquivo. Ao se abrir um dos arquivos PCAP que gerou alerta, tem-se as informações dispostas na Listagem 4.5:

Listing 4.5: Análise de tráfego com requisição para arquivo estático no WordPress

```
1 GET /wp-content/plugins/modl.exe HTTP/1.1
2 User-Agent: AutoIt
3 Host: yargiciambalaj.com
```

Nota-se a presença da ferramenta de criação de sites *WordPress*, cujo caminho definido no servidor aponta o executável *mold.exe*, armazenado como um arquivo estático. Essa já é uma característica que podemos utilizar para criar mais regras. Também foi possível notar que essa mesma requisição aconteceu mais de uma vez em um perído curto de tempo, o que aumenta as chances de ter sido feita a partir de um script, que ficou tentando reconectar até que o arquivo tivesse sido obtido com sucesso. A regra é apresentada na Listagem 4.6

Listing 4.6: Regra contra caminho no WordPress

```
1 alert http $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg:
2 "Download de executável por script AutoIt a partir do WP";
3 flow:established,to_server;http.method; content:"GET";
4 http.uri; pcre:"wp-content/.*\.exe/U";
5 http.user_agent; content:"AutoIt"; classtype:trojan-activity;
6 sid:3; rev:1;)
```

Continuando a análise dos PCAPs, temos mais estruturas de requisição que geraram alerta parecidas com a anterior, que podem gerar regras parecidas com a regra acima, mudando somente a opção *content* (linha 1, após o GET, Listagem 4.7).

Listing 4.7: Análise de tráfego com requisição para download de executável

```
1 GET /libraries/domit/css/safehome.exe HTTP/1.1
2 User-Agent: AutoIt
3 Host: www.falco.com.co
```

A partir da análise dos arquivos de captura é possível encontrar várias estruturas, que como essa, também casam com alguma assinatura. Para todas elas, é possível construir uma assinatura específica, fazendo uso do mesmo processo usado para a construção da regra acima. Ainda fazendo análise estática dos PCAPs e das regras que geraram alertas, tem-se que algumas delas envolviam consultas DNS para um domínio dinâmico. Esse tipo de domínio na verdade funciona como um apontador para um IP dinâmico. Existem diversos provedores de serviços de DNS dinâmico, que provém uma ferramenta legítima que automatiza a troca do apontamento de um IP dinâmico para outro, quando ocorre essa mudança. Por exemplo: se o IP com endereço X passa a ter o endereço Y, o DNS dinâmico para de apontar para o endereço X e passa a apontar para o endereço Y. O problema aqui é que o domínio DNS apontado é composto por ".redirectme.net", que é um dos domínios DNS gratuitos (noip, 2022), que como mostrado na Seção 2.5.1, pode ser configurado de maneira a servir como servidor *C&C*. A regra que detecta uma consulta DNS para o domínio dinâmico "redirect.me"está na Listagem 4.8.

Listing 4.8: Detecção de Consulta DNS para domínio dinâmico gratuito

```
1 alert dns $HOME_NET any -> any any (msg:
2 "Query para domínio DynDNS *.redirectme .net";
3 dns.query; content:".redirectme.net"; nocase; endswith;
4 classtype:bad-unknown; sid:4; rev:1;
```

Outros domínios observados nos alertas para o protocolo DNS foram: *.cf, ipapi.co, cld.pt, no-ip, ddns.net, *.ga, *.top. Desses, pode-se destacar o*ipapi*, que retorna informações, como localização, timezone, etc de um IP existente, e o *no-ip*, um vendedor bastente utilizado para obtenção de serviços de DNS dinâmicos, e que contém uma lista de domínios dinâmicos gratuitos (noip, 2022).

Essas regras com detecção do protocolo DNS podem signicar que o script AutoIt está fazendo requisições para domínios gratuitos potencialmente maliciosos (já que o atacante não precisou comprar nenhum domínio apenas para disponibilizar ataques), por isso também funcionam como forte indicativo de requisição para download de malware.

### 4.1.4 O que se pode afirmar sobre detecção de AutoIt com NIDS

O conjunto de regras genéricas que podem ser implementadas é limitado, pois basicamente trata-se de encontrar um User-Agent no cabeçalho HTTP e uma extensão de arquivo na URI. Regras mais detalhadas são específicas e necessitam de inspeção manual do tráfego. Contudo, a presença de alertas gerados por regras AutoIt sugerem outros tipo de famílias de malware, como será visto e discutido nas Seções 4.2 e 4.3.

## 4.2 BANLOAD E BANBRA

O rótulo Banload é genericamente atribuído a um tipo de *Trojan Downloader* que baixa diferentes tipos de arquivos maliciosos de um servidor remoto, os instala e os executa na máquina atacada (F-Secure, 2022a). O Banload, por ter essa função, precisa parecer um software legítimo, e mais do que isso, útil para as pessoas. A Figura 4.2 mostra os nomes dos arquivos avaliados nesse trabalho que na verdade pertenciam a essa família de malware. Percebe-se as mais variadas formas de fazer com que o usuário acredite na legitimidade e de fato baixe e execute o arquivo, por exemplo: comprovante, multa, conversa de whatsapp, currículo, fatura, cupom promocional, etc. No Brasil, é muito comum que esse tipo de malware faça download de outro *Trojan*, conhecido como Banbra. O Banbra, assim como o Banload, também é um *Trojan Downloader* que tem por objetivo baixar e instalar programas e biblotecas maliciosas, contudo, como o nome sugere, o Banbra é focado em roubar informações, principalmente credenciais, envolvendo sites de bancos brasileiros (F-Secure, 2022b). Neste trabalho, o enfoque é analisar o tráfego de rede do Banload, que por sua vez fará o download do Banbra, o qual tem como principal objetivo roubar as credenciais de usuário de *Internet Banking*. Serão analisados dois exemplares de malware Banbra: um mais genérico, que usa técnicas de distração e remoção de logs para evitar sua identificação, e outro mais específico, conhecido como *Telex*, que funciona como um formulário, enviando informações e recebendo comandos do servidor *C&C*. Como ambos fazem comunicação com servidores na fase de pós-infecção, pode-se criar regras de NIDS para identificar esse tráfego, assunto abordado na Seção 4.2.1.

```
raphael@raphael-VirtualBox:~/tcc/tcc_pcaps/banload/banload-rkm/sem_hash$ ls
012-PLANILHA-201524.PDF.exe                        comprovante_17082015.exe                                Doc_CB0502003441989O6_Detalhes_Cópia_de_Segurança_54.exe
01-333130133745743-NF-e-Emitida.exe                comprovante-de-deposito-em-conta-corrente-pdf.exe        doc.curric2017.exe
0311conversawhatsapp.exe                           Comprovante-PDF.exe                                     DocumentoAtual_MeuVivoFaturaOutubro-pdf.exe
101534083000.pdf.................................exe comprovante_XML_.exe                                    DOC_VJGHMIGCCAVPA789188614-Nз789188614.exe
2_36547705_directDownload_true.exe                 Comp_TED-Online.exe                                     Download_03452435539.exe
2.Boleto.pdf.exe                                   CTQPBNKBZFXZ.exe                                        Download_03452435584.exe
2.Via.Boleto-14112014.exe                          Cupom_Promocional.exe                                   FatVivoPen28902.exe
2via-Nota_Fiscal_97213647.exe                      Curriculum_PDF....exe                                   Gtt_6723.exe
AA100833276BR.1.exe                                CurriculumVitae-AnaClaudiaBorges-UNIFESP.exe            MartinsFedExNFEDanfedigitaldtfinal15082017F.exe
AlteraÃÃo_Contratual.exe                           debito.exe                                              MeuVivoFaturaDigital.exe
Baixa-NF-e-23062017001.exe                         Debitos_Cliente_Sky.exe                                 NF-e-110120183080300624655003000124709197268751.exe
b_j1.zip.exe                                       Debitos-Serasa-2016-08-20-COD-2389489023948234.exe      OA757691228BR.exe
Boleto-09861.exe                                   Debitos_Serasa_Experian.exe                             OV872874298427424O231116.exe
Cheque_Copia.exe                                   Deposito.pdf.exe                                        PDF_100987464500.exe
CobranÃa_Anexo.exe                                 Detran_multa879.exe                                     Tele-Cine_Play.exe
COD22092238051120l6.exe                            DiagnosticoBB.exe                                       TJSP-Penal_Processo-OAB-TJSP-819203.exe
```

Figura 4.2: Nomes de arquivos Banload, precedido de seus MD5, disponível na Corvus

### 4.2.1 Detecção de malware Banload e Branba usando NIDS

Primeiramente, a partir de uma documentação (F-Secure, 2022a) do malware, percebe-se que o Banload que faz download do exemplar Banbra tratado aqui é baixado automaticamente ao acessar o endereço **http://www.cad-portal.com/includes/[...].php**, logo pode-se criar uma regra, referenciando essa documentação, mostrada na Listagem 4.9.

Listing 4.9: Detecção de requisição a partir de malware Banload

```
alert http $HOME_NET any -> $EXTERNAL_NET any (
msg:"Possivel Download de Malware pelo Banload";
flow:established,to_server; http.method; content:"GET";
http.uri; pcre:"/includes/.*\.php/U"; http.uri;
content:"cad-portal"; classtype:trojan-activity;
reference:url,f-secure.com/v-descs/trojan-downloader_w32
_banload_fvq.shtml; sid:5; rev:1;)
```

Essa regra já indica que houve contaminação pelo Banload que, se executado, fará download do Banbra e outros dois arquivos de imagem (http://www.paeksan.com/technote/001,2.jpg), não maliciosos, mas que servem como indicativo de alerta do Banbra. Para detectar o evento, pode-se escrever uma nova regra, como mostra a Listagem 4.10.

Listing 4.10: Detecção de requisição a partir de malware Banbra para imagens de distração.

```
alert http $HOME_NET any -> $EXTERNAL_NET any (
msg:"Download de imagens relacionado ao Banbra";
flow:established,to_server; http.method; content:"GET";
http.uri; pcre:"/technote/.*\.jpg/U"; http.uri;
content:"paeksan"; classtype:trojan-activity;
reference:url,f-secure.com/v-descs/trojan-downloader_w32
_banload_fvq.shtml; sid:6; rev:1;)
```

É interessante ressaltar que essas imagens, após a execução do script são renomeadas para o mesmo nome dos executáveis maliciosos, também baixados pelo mesmo script. Com isso, tem-se que a imagem 001.jpg será renomeada para msnmsgsr.exe, e 002.jpg será renomeada para innit226.exe. Ou seja, ocorre o download dos exemplares de malware e imagens, execução dos malware, renomeação das imagens para o mesmo nome dos executáveis, e depois essas imagens são mandadas para o mesmo lugar dos executáveis, que são substituídos por essas imagens. O esquema de infecção é mostrado na Figura 4.4.

Além das regras que detectam a requisição para download das imagens, podemos definir também uma regra para a requisições do executáveis, como na Listagem 4.11.

Listing 4.11: Detecção de requisição a partir de malware Banbra para arquivos executáveis.

```
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:
"Download de executáveis relacionado ao Banload";
flow:established,to_server; http.method; content:"GET";
http.uri; pcre:"/technote/.*\.exe/U"; http.uri;
content:"paeksan"; classtype:trojan-activity;
reference:url,f-secure.com/v-descs/trojan-downloader_w32
_banload_fvq.shtml; sid:7; rev:1;)
```

A partir desse momento, o malware Banbra entra em ação. O seu comportamento inclui usar uma ferramenta legítima de remoção de malware para promover a retirada de alguns

formulários de segurança que os sites de banco usam para autenticação mais segura. Essa ferramenta chama-se Avenger (Swandog46, 2008), e é responsável por remover um conjunto de arquivos (elencados no apêndice A), incluindo o executável do plugin bancário G-Buster Browser Defense, na próxima fez em que o computador for reiniciado. Cada banco tem um nome e versão desse executável (por exemplo: o da Caixa Econômica chama-se GBPCEF.exe e faz uma série de checagens e instalações) como mostrado na Figura 4.3.


Figura 4.3: Plugin Browser Defense da Caixa

Depois de remover essa aplicação, o malware ainda chama um outro executável, que apaga o arquivo de log deixado pelo Avenger. Para de fato roubar o login e senha do usuário, o malware injeta HTML malicioso na página do banco, que consegue capturar quais teclas do teclado foram apertadas em cada campo do site, e por consequência roubar o acesso do usuário. Depois de capturadas, o script manda as credenciais para 2 padrões de e-mails distintos (h3llm45t[...]@vfemail.net, jaodas[...]@inbox.com). Com isso, podemos criar uma nova regra buscando pelo protocolo SMTP, com esses hosts de e-mail, como mostra a Listagem 4.12.

Listing 4.12: Detecção de envio de e-mail por malware Banbra.

```
1 alert tcp $HOME_NET any -> $EXTERNAL_NET 25 (msg:"Email
2 para Destinatário Suspeito-BANBRA";
3 pcre:"/(jaodas.*\@inbox\.com|h3llm45t.*\@vfemail\.net)/i";
4 classtype: trojan-activity ;sid:8 ; rev:1;)
```

Sempre mantendo o objetivo de chamar o Banbra, podemos definir outros padrões de comportamento para o Banload, geralmente relacionado à técnica de engenharia social denominada *phishing*. Esse tipo de tentativa de infecção envolve o oferecimento de alguma coisa que pareça vantajosa ou agradável para o usuário, por exemplo o download do último episódio

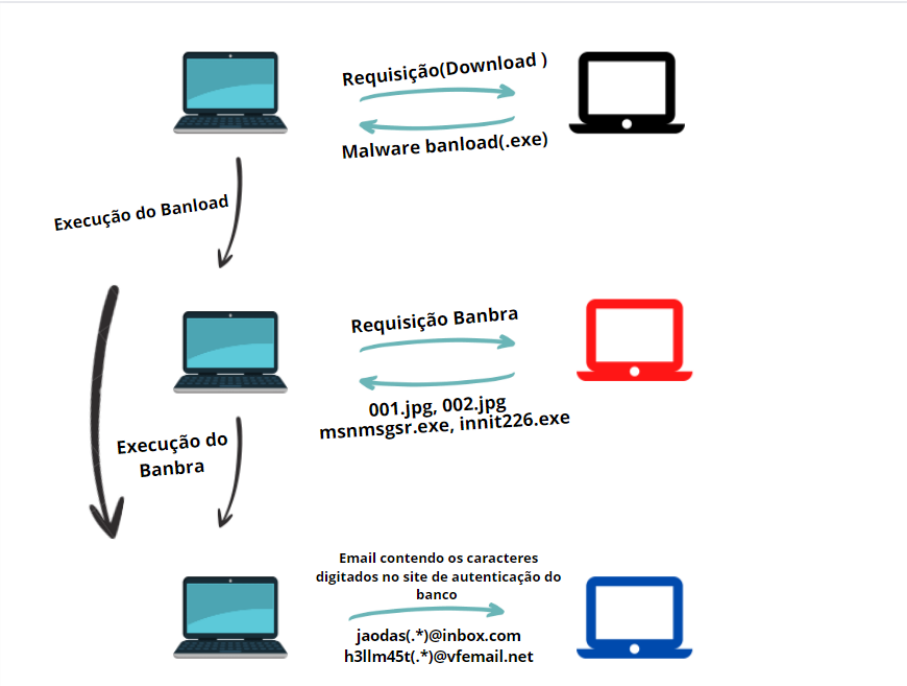



Figura 4.4: Esquema Banload -> Banbra

de uma série popular, ou uma licença gratuita de um antivírus. O usuário então tentado pela aparente ótima oferta, deliberadamente baixa e executa o programa, que na verdade foi escrito por um atacante.

Então, a partir de ofertas interessantes ao usuário, este é instigado a clicar em links, geralmente de URLs encurtadas que fazem o download do Banload que, por sua vez, baixará e invocará o malware Banbra (esquema na Figura 4.4). O malware em destaque agora chama-se *Telex*, e assim como o mencionado anteriormente, tem como principal função o roubo de credenciais de usuário de *Internet Banking*, contudo, usa diferente técnicas para alcançar tal objetivo, as quais serão descritas agora.

Uma técnica muito utilizada, não só por malware da família Banload, é usar um encurtador de URL para esconder o site e link de acesso. Endereços como *bit.ly* e *cl.ly* são usados porque na verdade representam um redirecionamento para uma outra página, onde possivelmente pode ocorrer o download de algum conteúdo malicioso, pode-se ter um exemplo na Listagem 4.13.

Listing 4.13: Exemplo de redirecionamento a partir de URL encurtada.

```
"http": {
    "hostname": "cl.ly",
    "url": "/0O1a0K2A0K3j/download/sys1.rar",
    "http_content_type": "text/html",
    "http_method": "GET",
    "protocol": "HTTP/1.1",
    "status": 301,
    "redirect": "https://api.cld.me/0O1a0K2A0K3j/download/sys1.
       rar",
    "length": 0
  }
```

Nota-se que o endereço 'cl.ly' é na verdade redirecionado para 'api.cld.me'. A Listagem 4.13 representa o log de uma regra do NIDS que alertou para um endereço contento encurtamento de URL, que redireciona para página que faz download de um arquivo, podemos ver essa regra na Listagem 4.14.

Listing 4.14: Regra para detecção de requisição para URL encurtada.

```
alert http $HOME_NET any -> $EXTERNAL_NET any
(msg:"Requisição HTTP para URL encurtada - cl.ly";
flow:established,to_server; http.method; content:"GET";
http.start; content:"HTTP/1.1|0d 0a|Host|3a 20|cl.ly|0d 0a|
Connection|3a 20|Keep-Alive|0d 0a 0d 0a|"; endswith;
   fast_pattern;
classtype:bad-unknown; sid:9; rev:1;)
```

Além do encurtamento de URL, temos outras técnicas utilizadas pelo *Telex* que utilizam a rede, e portanto é possível definir algumas regras. O primeiro evento que acontece após e execução do Banload é a tentativa de download do Telex propriamente dito e de outros arquivos adicionais, a partir de um conjunto pré-definido de URLs. Podemos então criar regras, com o intuito de identificar esses caminhos, como mostra a Listagem 4.15.

Listing 4.15: Regra para detecção do download do Banbra Telex.

```
alert http $HOME_NET any -> $EXTERNAL_NET any
(msg:"Requisição HTTP para caminho suspeito - BANBRA Telex";
flow:established,to_server; http.method; content:"GET";
http.uri; pcre:""/(chackal|bode|toxppxoa|pcpxpaepxo)\.zip/U";
classtype:trojan-activity; sid:10; rev:1;)
```

Vemos na Listagem 4.15 que o malware então tenta fazer o download de 4 arquivos compactados. Esses arquivos contém o Telex propriamente dito, além da biblioteca legítima *sqlite.dll*, arquivos para instalação em sistema de 32 e 64 bits, além de uma cópia do malware Banload (aquele que fez a requisição), usado para que o malware cheque sua própria integridade via hash. Além disso, o malware consegue checar também o *bit-ness* do sistema operacional alvo, e instalar o *rootkit* adequado. Nesse momento, o malware Banload já cumpriu seu papel (como o nome sugere, carregar o malware que de fato faz o roubo de dados) e depois de acertadas algumas configurações, como a linkagem com a DLL já baixada e instalação dos arquivos, o Banbra pode de fato ser executado.

Esse malware possui o seguinte comportamento: é criado um formulário, chamado pelo autor de *Telex*, daí o nome do malware, que estabelece uma conexão com um servidor remoto, e é capaz de executar comandos simples, passados pelo próprio servidor ao receber um conjunto de informações. Ou seja, existe um esquema-cliente servidor bem definido, e mais do que isso, esse é um caso de *CnC - (Comand and Control)*, onde há código malicioso no servidor, que retorna ações para o cliente realizar na máquina atacada. A primeira mensagem que o *Telex* manda ao servidor, é um conjunto de informações a respeito da máquina em que o formulário está. Entre elas, temos:

- ID_MAQUINA - Nome da máquina no grupo do Windows
- VERSAO – Versão do *Trojan* Banbra
- WIN – Sistema Operacional

- NAVEGADOR – Nome do Navegador
- PLUGIN – Nome do *plug-in* (Cada banco possui um)
- AV – Nome do Antivírus instalado

Para cada um dos itens elencados acima, existe um módulo do executável que realiza a ação. Por exemplo, há um módulo responsável somente por checar se há algum antivírus instalado, e outro, que tem objetivo único determinar qual o navegador utilizado como padrão pelo usuário. Nesse momento, é possível definir mais uma regra com base nessa comunicação, mostrada na Listagem 4.16.

Listing 4.16: Regra para detecção dos parâmetros enviados do Telex ao servidor.

```
alert http $HOME_NET any -> $EXTERNAL_NET any
(msg:"Telex BANBRA - CnC - Comunicação com servidor";
flow:established,to_server; http.method; content:"POST";
http.request_body; content:"ID_MAQUINA="; depth:11; nocase;
fast_pattern; content:"&VERSAO="; distance:0; nocase;
content:"&WIN="; distance:0; nocase;
classtype:command-and-control; sid:11; rev:3;
```

Uma vez estabelecida a conexão e enviadas essas primeiras informações, o servidor passa a enviar comandos ao cliente, que tenta executar e retornar a resposta ao servidor. Mais detalhes são apresentados na Tabela 4.1.

Tabela 4.1: Comandos enviados pelo servidor ao *Telex*

| | |
|---|---|
| PING | Checa a conexão |
| Info | Mais informações sobre o sistema operacional atacado |
| Close | Fechar essa conexão |
| reini | Reiniciar sistema operacional |
| REQUESTINFO | Mais informações sobre o antivirus instalado |
| REQUESTKEYBOARD | Sequência de caracteres a ser pressionados |
| HjiopPos | Inicializar posição do Mouse |
| HjiopLD | Click (Apertar botão esquerdo do Mouse) |
| HjiopLU | Desapertar botão esquerdo do Mouse |
| POWT | Colocar a String que acompanhar na janela atual |
| DESMON | WM_SYSCOMMAND: Usa um gerenciados de janelas do windows pra obter informações de mouse, janela ativa, etc. |

Para capturar esse tráfego, é necessário criar regras que casem com esses comandos, usando expressões regulares para que possamos contemplar todos eles numa só regra, mostrada da Listagem 4.17.

Listing 4.17: Regra para detecção dos comandos enviados do servidor ao Telex.

```
alert http $EXTERNAL_NET any -> $HOME_NET any
(msg:"Telex BANBRA - CnC - Servidor enviando comandos";
pcre:"/(PING|Info|Close|reini|REQUESTINFO|REQUESTKEYBOARD|
HjiopPos|HjiopLD|HjiopLU|POWT|DESMON)/U";
classtype:command-and-control; sid:12; rev:1;
```

Ao contrário do AutoIt, esse tipo de malware permite uma detecção com um nível de detalhe maior usando assinatura. Isso acontece porque existe uma troca constante de mensagens padronizadas e sem criptografia entre servidor e cliente (malware instalado na máquina atacada). Como podemos estudar o comportamento do malware com relação a respostas e requisições, temos uma quantidade maior e mais detalhada de regras específicas pra essa família que podem ser utilizadas no NIDS. Mesmo com esse comportamento favorável para usar detecção por assinatura, não se tem garantia que as regras criadas em um momento vão continuar funcionando. Mais detalhes sobre as limitações da detecção com assinaturas serão mostrados na Seção 5.1.

## 4.3 BANLOAD E MEKOTIO

Assim como o Banbra, o Mekotio, também chamado de Bestafera, tem a principal finalidade atacar clientes usando *Internet Banking*. Para fazê-lo, baixa e executa o interpretador AutoIt para rodar um código malicioso, que também é obtido pela rede, no mesmo arquivo compactado. Podemos dizer então que o Makotio usa atributos e técnicas já discutidas nas duas seções anteriores. Assim como o Banbra, o Mekotio também é carregado (baixado, instalado e configurado) por um malware da família Banload, que tem como principal função ser um instrumento que atraia a atenção da vítima, e nesse momento, ser executado na máquina infectada, carregando assim outro tipo de malware que envolve ataques a usuário de *Internet Banking*.

O Mekotio é difundido principalmente via e-mail. No Brasil, até mesmo e-mails contendo falsas campanhas de vacinação contra a COVID-19 foram usados para difundir esse malware (securityreport.com.br, 2021). O fluxograma do ataque é o seguinte: e-mails falsos são spamados para diversos endereços, no corpo desse e-mail há um link para um endereço já programado para fazer o download de um arquivo compactado, geralmente com a extensão zip. Ao descompar este arquivo, tem-se um arquivo *MSI - Microsoft Installer*, esse tipo de arquivo pode instalar, além de modificar ou deletar um executável. Com o MSI, é possível determinar a seção *CustomAction* que pode conter um script ou mais executados junto com o instalador, ou seja, ao mesmo tempo é possivel instalar o malware e rodar um script, que também pode ser malicioso. Além do MSI, há também compactado no arquivo obtido pela rede: um intérprete do AutoIt, um script AutoIt, o código do malware Mekotio, encontrado dentro de um DLL, e também a DLL legítima do *SQlite3*. A Figura 4.5 mostra o fluxograma da execução do Mekotio na máquina atacada.

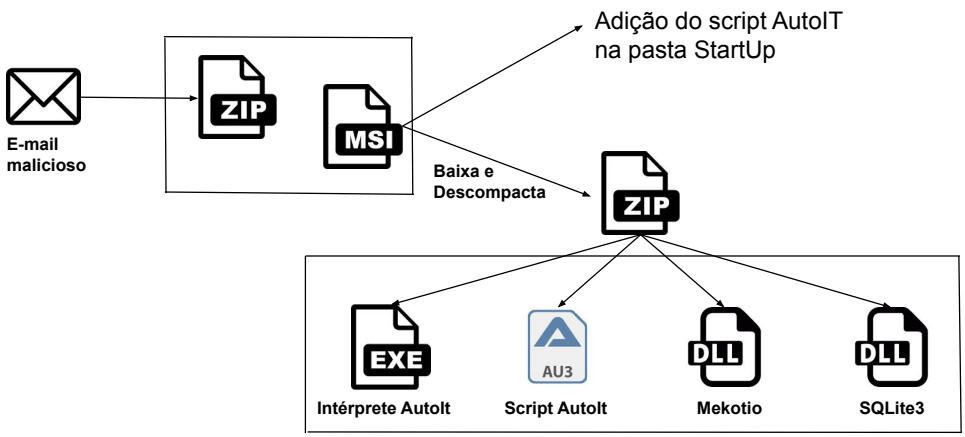



Figura 4.5: Fluxograma de Infecção Mekotio

Depois da extração desses 4 arquivos, ainda há a tentativa de estabelecimento de persistência, gerando uma entrada no mecanismo de inicialização de programas *StartUp*, com o

comando para a execução do script AutoIt, que por irá carregar a DLL que contém o Mekotio. Com tudo instalado, a execução do malware baseia-se em verificar quais os *hosts* acessados pelo navegador na máquina atacada, caso seja de um conjunto pré definido de sites de bancos, será exibida uma tela de autenticação falsa. Depois de armazenados os campos preenchidos pelo usuário, o malware tenta mandar os dados para um servidor remoto. Outra ação bastante realizada é a troca de ambiente de transferência, quando o malware detecta que existe uma string na área de tranferência que se parece com uma conta de banco, ele troca esse valor para outra conta. Se o usuário não perceber, ele acaba fazendo a transação para a conta definida pelo Mekotio.

A partir do momento que as informações do usuário são coletadas, começa a troca de mensagens entre o servidor orquestrando o *C&C* e o Mekotio presente na máquina infectada. As mensagens geralmente podem ser identificadas no payload de um pacote no formato: <|*comando*|>. Há uma lista grande de comandos usados pelo Mekotio, todos eles podendo ser usados para criação de regras de NIDS e indícios de pós-infecção do Mekotio, como mostrado a seguir:

- <|utypzjI|>
- <|IXjzwtR|>
- <|ztUjzwtR|>
- <|SuaykJI|>
- <|SuaykRJ|>
- <|lozyw|>
- <|vhxboj|>
- <|WGSGTNU|>
- <|tksN|>
- <|VOTM|>
- <|LSTU|>
- <|Gpsxi|>
- <|ZKXAKYWQKEHUGZJ|>

Uma possível regra utilizando expressões regulares para detectar os indícios mostrados acima é discriminada na Listagem 4.18.

Listing 4.18: Regra para detecção dos comandos enviados do servidor ao malware Mekotio.

```
1 alert http $EXTERNAL_NET any -> $HOME_NET any
2 (msg:"Mekotio - CnC - Comunicação com Servidor";
3 pcre:"/\<\|utypzjI|IXjzwtR|ztUjzwtR|SuaykJI|SuaykRJ|
4 lozyw|vhxboj|WGSGTNU|tksN|VOTM|LSTU|Gpsxi|
5 ZKXAKYWQKEHUGZJ\|\>/U"; classtype:command-and-control;
6 sid:13; rev:1;
```

Além de roubar credenciais, é sabido que o Mekotio também consegue modificar boletos bancários, trocando o número original do boleto e o código de barras por um gerado pelo atacante. Esse número segue o seguinte padrão: *xxxxxxxxxx-x xxxxxxxxxx-x xxxxxxxxxx-x xxxxxxxxxx-x*, onde cada *x* é um dígito de 0 a 9. Como o número do boleto é enviado para o malware pelo servidor, pode-se criar uma regra de NIDS para a detecção, mostrada na Listagem 4.19.

Listing 4.19: Regra para detecção de número de boleto bancário.

```
alert http $EXTERNAL_NET any -> $HOME_NET any
(msg:"Possível fraude na geração de boleto bancário";
flow:established,from_server;
pcre:"/{^\d{11}\-\d{1}$}4/";
classtype:command-and-control; sid:14; rev:1;
```

## 4.4 CONSIDERAÇÕES FINAIS

Essa seção mostrou um estudo mais aprofundado de alguns arquivos executáveis de 4 principais famílias de malware. As conclusões indicam que regras genéricas envolvendo AutoIt, *Command and Control* e DNS dinâmico são fundamentais para a detecção eficaz das famílias de malware Banbra e Mekotio. Além disso, vários executáveis da família de *Trojans downloaders* bancários Banload foram estudados, indicando seu comportamento: atrair a atenção so usuário e instalar outro tipo de *Trojan*. Além das regras mencionadas, o Mekotio ainda envolve regras para detecção de fraude em boletos, regras com comandos Mekotio, listados acima, e regras contra spams recebidos na caixa de e-mail. Já para a detecção do Banbra devem ser implementadas regras com detecção de arquivos executáveis e de imagem, e-mails com credenciais, URLs encurtadas e comandos e parâmetros específicos do banbra, descritos acima. Um diagrama expressando o conjunto de regras que deve ser implementado para a detecção de cada família pode ser visto na Figura 4.6

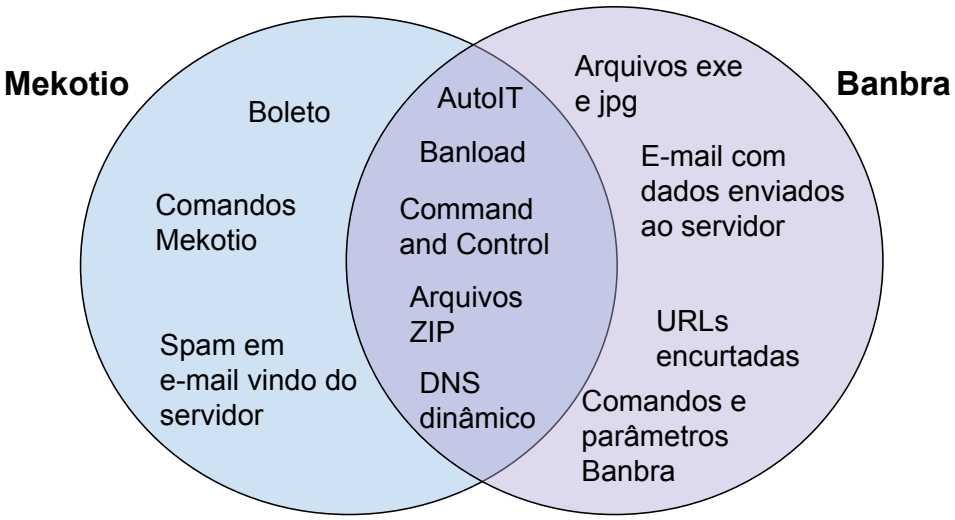



Figura 4.6: Diagrama de Venn com o conjunto de regras para cada família.

Outra conclusão tirada desse estudo, além do conjunto de regras que define cada família, é a pouca eficiência em criar regras contra domínios específicos (contra domínios de DNS dinâmico gratuitos é de extrema importância, como discutido na Seção 4.1.3). O código do malware já vem com algum algoritmo de geração de domínio, em inglês, *DGA - Domain*

*Generation Algorithm*, que tem como função ficar gerando domínios, até que seja descoberto pelo malware que um deles existe, ou seja, que foi configurado pelo atacante. Como cada execução do malware pode fazer requisição para centenas de domínios diferentes, e a cada execução o domínio registrado pode ser diferente, há pouca razão para tentar criar regras contra domínios específicos, e até mesmo a criação de uma *blacklist* pode não ter utilidade, dependendo da frequência que o malware troca seu domínio para *Command and Control*.

# 5 CONCLUSÃO

A monografia apresentou alguns conceitos e ferramentas de soluções referentes a detecção de malware em tráfego de rede, baseando-se no uso de sistemas de detecção de instrusão baseados em assinaturas de malware. Dentre eles destaca-se o funcionamento dos NIDS, construção de regras, comparação entre detecção por assinatura e anomalia e as limitações que a detecção por assinatura empõe, mas também as vantegens que ela pode trazer. O capítulo 2 também mostrou que o Suricata, NIDS escolhido para realizar os experimento conduzidos nesta monografia, pode operar de três modos diferentes: (1) como IDS; (2) como IPS e (3) modo *replay*. Além disso, foram mostrados detalhes da técnica *Command and Control*, abordagem utilizada por autores de malware que foi o principal foco das regras e análises que esta monografia propôs, além de uma revisão da literatura.

Apresentou-se a origem dos exemplares de malware (e do tráfego associado) utilizado nesse trabalho no Capítulo 3. Um fluxograma, desde a submissão do malware até a coleta dos dados usados foi mostrado, indicando como o sistema Corvus funciona. Mostrou-se também o uso das regras *Filestore*, bastante úteis para detectar comportamento de *Trojans downloaders*. Esse trabalho também apresentou, na Seção 3.3, uma breve revisão das maneiras usadas por diferentes antivirus para fazer inspeção do tráfego de rede, bem como fez uma análise das regras do Snort usadas pelo antivírus VIPRE, que foi o único AV que descobriu-se usar uma solução NIDS de software livre, conforme o trabalho *AntiViruses under the Microscope: A Hands-On Perspective*, do qual o autor dessa monografia foi coautor. Com uma análise detalhada das regras, percebeu-se que de fato existem assinaturas para malwares *Trojans RATs*, contudo nenhuma assinatura para *Trojans* bancários brasileiros, e as regras que tentavam detectar comunicações *Command and Control*, também não obtiveram sucesso.

Analisou-se o tráfego de rede gerado pela execução de malwares de quatro famílias diferentes: AutoIt, Banload, Mekotio e Banbra. Foi proposto que tanto a identificação da família Mekotio quanto da Banbra passavam pelas regras usadas para detectar malware AutoIt e Banload, além de regras específicas para cada família, que foram discutidas na Seção 4 deste trabalho. Também na Seção 4 mostrou-se como fazer análise manual dos PCAPs de interesse a fim de escrever novas regras, bem como a conclusão de que, pelo fato dos malware apresentatem algoritmos de geração de nomes de domínio, a criação de regras contra domínios específicos não é eficiente, uma vez que o destino das requisições feitas pelos *Trojans* muda de maneira frequente.

## 5.1 LIMITAÇÕES

A principal vantagem de sistemas que fazem uso de detecção por assinatura é o desempenho, uma vez que as regras são carregadas pelo NIDS, é preciso somente tratar da detecção por meio de reconhecimento do padrão de alguma das regras com o *payload* dos pacotes, não existe mais camadas de processamento, como por exemplo o uso de algoritmos de aprendizado de máquina, usados na detecção por anomalias.

A razão por ser eficiente é a mesma que implica as maiores limitações. Como o conjunto de regras é determinado por meio de uma assinatura fixa, a mínima mudança que o autor do malware fizer com relação a algum parâmetro pode inutilizar uma regra inteira. Por exemplo, um dos parâmetro que o malware *Telex* da família Banbra envia para o servidor *Command and*

*Control* é 'ID_MAQUINA', caso o autor mudasse o nome desse parâmetro para 'maq_id', por exemplo, todas as regras escritas com content:'ID_MAQUINA' tornariam-se obsoletas.

A única maneira de contornar esse problema é trazendo alguma política de atualização constante de regras, de preferência não manual. Alguns dos trabalhos elencados na seção de trabalhos relacionados tentam fazê-lo, uma das alternativas é estabelecer a frequência de um conjunto de strings contido no payload de pacotes de tráfego sabidamente malicioso e a frequencia desse mesmo conjunto em tráfego benigno, caso a frequencia em tráfego malicioso seja maior, cria-se uma regra a partir dessas strings. Essa abordagem não é recente e pareceu funcionar muito bem (Rieck et al., 2010), porém, surgem algumas dúvidas: de quanto em quanto tempo esse processo deve ser repetido para manter as regras sempre atualizadas? onde conseguir fontes recentes de tráfego malicioso e benígno? Além disso, usar essa abordagem gera alertas com base em sinais de pós infecção coletados por terceiros, se algum computador for atacado por malware que ainda não teve seu tráfego considerado malicioso, não haverá nenhuma regra capaz de detectá-lo.

A abordagem mais utilizada para tentar manter as assinaturas em dia é atualizar o arquivo de regras conforme são adicionadas à base *ET Open Ruleset*. Essa abordagem é simples, porque requer apenas um comando para realizar a atualização, e abrange assinaturas para diferentes tipos de malware, sendo a melhor maneira gratuita (e sem fazer análise manual de tráfego) de manter o conjunto de regras atualizado.

## 5.2 TRABALHOS FUTUROS

Uma das principais dificuldades foi de fato compreender o retorno dos scans realizados pelos antivírus, na plataforma VirusTotal. Ao passo que alguns antivírus retornavam uma família específica, por exemplo Banbra, outros simplesmente retornavam *Trojan.Genereric*. O apêndice B mostra a diversidade de resultados, sendo a maiores deles diferentes um dos outros. Para resolver, foi empregado o AVClass, que é capaz de diminuir esse ruído, retornando somente uma família. Contudo, até mesmo alguns retornos do AVClass não eram claros o suficiente para determinar o comportamento de um malware, por exemplo: se o retorno for 'AutoIt', não há como saber se é somente um script AutoIt fazendo uma requisição, ou se é algum tipo de *Trojan* bancário. Além disso, não existe maneira de calcular a proximidade entre resultados do AVClass, por exemplo: quais características diferenciam um Mekotio/Bestafera de um Ursu, sendo que na maioria dos scans Mekotio pelo menos um antivírus também retornava Ursu? Infelizmente, não existe uma base única para definir qual a definição das famílias de malware e a distância entre elas. Uma proposta de trabalho futuro com relação a isso é a criação de um módulo, que pode ser potencialmente acoplado ao AVClass, para indicar qual é a distância entre uma família de malware e outra (que foram apontadas por *scans* do VirusTotal), principalmente quais detalhes do comportamento fazem que a classificação seja da maneira que foi, e quais outras famílias tem abordagens correlatas.

Além dessa, como outra proposta para trabalhos futuros, sugere-se a implementação de indicadores de compromisso, além do conjunto genérico de regras que devem gerar alertas para a identificação do *Trojan* bancário que são informados nesse trabalho. Esse indicadores podem contar informações úteis que além das regras sugerem que algum desses *Trojans* está ou esteve em execução, por exemplo: verificação de criação/atualização/exclusão de arquivos em algum caminho específico, processos AutoIt rodando sem consentimento, checagem de persistência, entre outros.

Deixa-se também como sugestão para trabalhos futuros uma análise dos arquivos gerados a partir de regras *Filestore* para exemplares de malware rotulados como Banload. Já

que a principal função dessa família é fazer o download de outras famílias de *Trojans* bancários, seria interessante um artigo contendo quais famílias são essas, além de mais assinaturas para complementar as desenvolvidas nesta monografia.

# REFERÊNCIAS

autoitscript.com (2022). Autoit. https://www.autoitscript.com/site/. Acessado em 10/05/2022.

Bekerman, D., Shapira, B., Rokach, L. e Bar, A. (2015). Unknown malware detection using network traffic classification. Em *2015 IEEE Conference on Communications and Network Security (CNS)*, páginas 134–142. IEEE.

Botacin, M., Ceschin, F. e Grégio, A. (2021). Corvus: Uma solução sandbox e de threat intelligence para identificação e análise de malware. Em *Anais do XXI Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, páginas 50–57. SBC.

Botacin, M., Domingues, F. D., Ceschin, F., Machnicki, R., Alves, M. A. Z., de Geus, P. L. e Grégio, A. (2022). Antiviruses under the microscope: A hands-on perspective. *Computers & Security*, 112:102500.

Botacin, M., Kalysch, A. e Grégio, A. (2019). The internet banking [in] security spiral.

Diniz, G., Muggah, R. e Glenny, M. (2014). Deconstructing cyber security in brazil. *Strategic Paper*.

F-Secure (2022a). Trojan-downloader:w32/banload.fvq. https://www.f-secure.com/v-descs/trojan-downloader_w32_banload_fvq.shtml. Acessado em 10/05/2022.

F-Secure (2022b). Trojan-spy:w32/banbra.rm. https://www.f-secure.com/v-descs/trojan-spy_w32_banbra_rm.shtml. Acessado em 06/04/2022.

Gardiner, J., Cova, M. e Nagaraja, S. (2014). Command & control: Understanding, denying and detecting-a review of malware c2 techniques, detection and defences. *arXiv preprint arXiv:1408.1136*.

Gundert, L. (2014). Dynamic detection of malicious ddns. https://blogs.cisco.com/security/dynamic-detection-of-malicious-ddns. Acessado em 10/05/2022.

InforChannel (2020). Top malware – agosto de 2020: Nova versão do qbot entra na lista da check point. https://inforchannel.com.br/2020/09/10/top-malware-agosto-de-2020-nova-versao-do-qbot-entra-na-lista-da-check-point/. Acessado em 30/04/2022.

Jyothsna, V., Prasad, R. e Prasad, K. M. (2011). A review of anomaly based intrusion detection systems. *International Journal of Computer Applications*, 28(7):26–35.

Lever, C., Kotzias, P., Balzarotti, D., Caballero, J. e Antonakakis, M. (2017). A lustrum of malware network communication: Evolution and insights. Em *2017 IEEE Symposium on Security and Privacy (SP)*, páginas 788–804. IEEE.

Lévesque, F. L., Somayaji, A., Batchelder, D. e Fernandez, J. M. (2015). Measuring the health of antivirus ecosystems. Em *2015 10th International Conference on Malicious and Unwanted Software (MALWARE)*, páginas 101–109. IEEE.

Lu, G., Liu, Y., Chen, Y., Zhang, C., Gao, Y. e Zhong, G. (2020). A comprehensive detection approach of wannacry: Principles, rules and experiments. Em *2020 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, páginas 41–49. IEEE.

Mohaisen, A. e Alrawi, O. (2014). Av-meter: An evaluation of antivirus scans and labels. Em *International conference on detection of intrusions and malware, and vulnerability assessment*, páginas 112–131. Springer.

noip (2022). Free dynamic dns domains. `https://www.noip.com/support/faq/free-dynamic-dns-domains`. Acessado em 05/04/2022.

Rieck, K., Schwenk, G., Limmer, T., Holz, T. e Laskov, P. (2010). Botzilla: detecting the"phoning home"of malicious software. Em *Proceedings of the 2010 ACM symposium on applied computing*, páginas 1978–1984.

rules.emergingthreats.net (2022). Et open ruleset. `https://rules.emergingthreats.net/open/suricata/rules/`. Acessado em 30/04/2022.

Sebastián, M., Rivera, R., Kotzias, P. e Caballero, J. (2016). Avclass: A tool for massive malware labeling. Em *International symposium on research in attacks, intrusions, and defenses*, páginas 230–253. Springer.

SECRET, L. (2019). Corvus. `https://corvus.inf.ufpr.br/`. Acessado em 06/04/2022.

sectools.org (2011). Sectools. `https://sectools.org/`. Acessado em 06/04/2022.

securityreport.com.br (2021). Falsa campanha de vacinação propaga trojan bancário mekotio no brasil. `https://www.securityreport.com.br/overview/falsa-campanha-de-vacinacao-propaga-trojan-bancario-mekotio-no-brasil/`. Acessado em 06/04/2022.

Sgarioni, M. (2021). 79% dos brasileiros usam apps de bancos para transações financeiras, diz pesquisa tecban/ipsos. `https://www.mobiletime.com.br/noticias/28/07/2021/79-dos-brasileiros-usam-apps-de-bancos-para-transacoes-financeiras-diz-pesquisa-tecban-ipsos/`. Acessado em 30/04/2022.

suricata.io (2019). Suricata rules format. `https://suricata.readthedocs.io/en/suricata-6.0.0/rules/intro.html`. Acessado em 06/04/2022.

Swandog46 (2008). Avenger. `http://swandog46.geekstogo.com/avenger2/avenger2.html`. Acessado em 06/04/2022.

VIPRE (2022). Vipre. `https://www.vipre.com/`. Acessado em 23/05/2022.

White, J. S., Fitzsimmons, T. e Matthews, J. N. (2013). Quantitative analysis of intrusion detection systems: Snort and suricata. Em *Cyber sensing 2013*, volume 8757, páginas 10–21. SPIE.

# APÊNDICE A – ARQUIVOS MODIFICADOS PELO BANBRA

## A.1 ARQUIVOS EXCLUÍDOS:

```
%systemdrive%\Arquivos de programas\GbPlugin\scpsssh2.dll
%systemdrive%\Arquivos de programas\GbPlugin\gbiehuni.dll
%systemdrive%\Arquivos de programas\GbPlugin\gbpdist.dll
%systemdrive%\Arquivos de programas\GbPlugin\isg.gpc
%systemdrive%\Arquivos de programas\GbPlugin\uni.gpc
%systemdrive%\Arquivos de programas\GbPlugin\gbiehisg.dll
%systemdrive%\Arquivos de programas\GbPlugin\GBIEHCEF.DLL
%systemdrive%\Arquivos de programas\GbPlugin\scpVista.exe
%systemdrive%\Arquivos de programas\GbPlugin\gbiehabn.dll
%systemdrive%\Arquivos de programas\GbPlugin\GBIEHABN.DLL
%systemdrive%\Arquivos de programas\GbPlugin\LOGOF.DLL
%systemdrive%\Arquivos de programas\GbPlugin\abn.gpc
%systemdrive%\Arquivos de programas\GbPlugin\AtmCap.ocx
%systemdrive%\Arquivos de programas\GbPlugin\gbpsv.exe
%systemdrive%\Arquivos de programas\GbPlugin\GbpSv.exe
%systemdrive%\Arquivos de programas\GbPlugin\GbpSrv.exe
%systemdrive%\Arquivos de programas\GbPlugin\gbpsrv.exe
%systemdrive%\Arquivos de programas\GbPlugin\gbieh.dll
%systemdrive%\Arquivos de programas\GbPlugin\gbieh.dll
%systemdrive%\Arquivos de programas\GbPlugin\gbieh.gmd
%systemdrive%\Arquivos de programas\GbPlugin\bb.gpc
%systemdrive%\Arquivos de Programas\Scpad\scpMIB.dll
%systemdrive%\program files\Scpad\scpsssh2.dll
%systemdrive%\program files\Scpad\sshib.dll
%systemdrive%\program files\Scpad\scpIBCfg.bin
%systemdrive%\program files\Scpad\scpLIB.dll
%systemdrive%\program files\scpsssh2.dll
%systemdrive%\program files\gbiehuni.dll
%systemdrive%\program files\gbpdist.dll
%systemdrive%\program files\isg.gpc
%systemdrive%\program files\uni.gpc
%systemdrive%\program files\gbiehisg.dll
%systemdrive%\program files\GBIEHCEF.DLL
%systemdrive%\program files\gbiehabn.dll
%systemdrive%\program files\GBIEHABN.DLL
%systemdrive%\program files\LOGOF.DLL
%systemdrive%\program files\abn.gpc
%systemdrive%\program files\AtmCap.ocx
%systemdrive%\program files\gbpsv.exe
%systemdrive%\program files\GbpSv.exe
%systemdrive%\program files\GbpSrv.exe
%systemdrive%\program files\gbpsrv.exe
```

```
%systemdrive%\program files\gbieh.dll
%systemdrive%\program files\gbieh.gmd
%systemdrive%\program files\bb.gpc
%systemdrive%\program files\GbPlugin\Scpad\scpsssh2.dll
%systemdrive%\program files\GbPlugin\Scpad\sshib.dll
%systemdrive%\program files\GbPlugin\Scpad\scpIBCfg.bin
%systemdrive%\program files\GbPlugin\Scpad\scpLIB.dll
%systemdrive%\program files\GbPlugin\scpsssh2.dll
%systemdrive%\program files\GbPlugin\gbiehuni.dll
%systemdrive%\program files\GbPlugin\gbpdist.dll
%systemdrive%\program files\GbPlugin\isg.gpc
%systemdrive%\program files\GbPlugin\uni.gpc
%systemdrive%\program files\GbPlugin\gbiehisg.dll
%systemdrive%\program files\GbPlugin\GBIEHCEF.DLL
%systemdrive%\program files\GbPlugin\gbiehabn.dll
%systemdrive%\program files\GbPlugin\GBIEHABN.DLL
%systemdrive%\program files\GbPlugin\LOGOF.DLL
%systemdrive%\program files\GbPlugin\abn.gpc
%systemdrive%\program files\GbPlugin\AtmCap.ocx
%systemdrive%\program files\GbPlugin\gbpsv.exe
%systemdrive%\program files\GbPlugin\GbpSv.exe
%systemdrive%\program files\GbPlugin\GbpSrv.exe
%systemdrive%\program files\GbPlugin\gbpsrv.exe
%systemdrive%\program files\GbPlugin\gbieh.dll
%systemdrive%\program files\GbPlugin\gbieh.gmd
%systemdrive%\program files\GbPlugin\bb.gpc
%windir%\scpVista.exe
%windir%\gbpsv.exe
%windir%\gbpsrv.exe
%systemdrive%\Arquivos de programas\GbPlugin\GbpSrv.exe
%systemdrive%\Arquivos de programas\GbPlugin\scpVista.exe
%systemdrive%\avenger.txt (Log do Avanger)
```

## A.2 ARQUIVOS CRIADOS:

```
%windir%\msnmsgsr.exe
%windir%\system32\avenger.exe
%windir%\system32\awou.txt
%windir%\system32\Enviou2u2u.txt
%windir%\system32\drivers\workray.sys
%temp%\bloreg222a
%temp%\blokil2a
C:\Documents
Settings\All Users\Start Menu\Programs\Startup\avg.exe
C:\cleanup.bat
C:\cleanup.exe
C:\systemX86.txt
C:\zip.exe
```

# APÊNDICE B – RESULTADOS VIRUSTOTAL

Listing B.1:


```
"Detection": {
    "AVclass": [
        [
            "banbra",
            "1"
        ]
    ],
    "VirusTotal": {
        "md5": "5dbec476e11b42598944717f5b57bbb2",
        "permalink": "https://www.virustotal.com/file/3258668
          c6aeb14f1c3d59cc2d5366da12ca95ef92a53b7d3a93d08fc217bee1
          /analysis/1521826748/",
        "positives": 48,
        "resource": "5dbec476e11b42598944717f5b57bbb2",
        "response_code": 1,
        "scan_date": "2018-03-23 17:39:08",
        "scan_id": "3258668
          c6aeb14f1c3d59cc2d5366da142ca95ef92a53b
           7d3a93d08fc217bee1-1521826748",
        "scans": {
            "ALYac": {
                "detected": true,
                "result": "Gen:Variant.Graftor.176515",
                "update": "20180323",
                "version": "1.1.1.5"
            },
            "AVG": {
                "detected": true,
                "result": "FileRepMetagen [Malware]",
                "update": "20180323",
                "version": "18.2.3827.0"
            },
            "AVware": {
                "detected": true,
                "result": "Trojan.Win32.Generic!BT",
                "update": "20180323",
                "version": "1.5.0.42"
            },
            "Ad-Aware": {
                "detected": true,
                "result": "Gen:Variant.Graftor.176515",
                "update": "20180323",
```

```
                "version": "3.0.3.1010"
            },
            "AegisLab": {
                "detected": true,
                "result": "Troj.Banker.W32.Agent.kqs!c",
                "update": "20180323",
                "version": "4.2"
            },
            "AhnLab-V3": {
                "detected": true,
                "result": "Trojan/Win32.Banbra.R137662",
                "update": "20180323",
                "version": "3.12.0.20130"
            },
            "Antiy-AVL": {
                "detected": true,
                "result": "Trojan[Banker]/Win32.Banbra",
                "update": "20180323",
                "version": "3.0.0.1"
            },
            "Arcabit": {
                "detected": true,
                "result": "Trojan.Graftor.D2B183",
                "update": "20180323",
                "version": "1.0.0.831"
            },
            "Avast": {
                "detected": true,
                "result": "FileRepMetagen [Malware]",
                "update": "20180323",
                "version": "18.2.3827.0"
            },
            "Avast-Mobile": {
                "detected": false,
                "result": null,
                "update": "20180323",
                "version": "180323-02"
            },
            "Avira": {
                "detected": true,
                "result": "TR/Spy.Banker.Gen",
                "update": "20180323",
                "version": "8.3.3.6"
            },
            "Baidu": {
                "detected": false,
                "result": null,
                "update": "20180323",
```

```
                "version": "1.0.0.2"
            },
            "BitDefender": {
                "detected": true,
                "result": "Gen:Variant.Graftor.176515",
                "update": "20180323",
                "version": "7.2"
            },
            "Bkav": {
                "detected": false,
                "result": null,
                "update": "20180322",
                "version": "1.3.0.9466"
            },
            "CAT-QuickHeal": {
                "detected": true,
                "result": "TrojanDownloader.Mavradoi",
                "update": "20180322",
                "version": "14.00"
            },
            "CMC": {
                "detected": false,
                "result": null,
                "update": "20180323",
                "version": "1.1.0.977"
            },
            "ClamAV": {
                "detected": false,
                "result": null,
                "update": "20180323",
                "version": "0.99.2.0"
            },
            "Comodo": {
                "detected": true,
                "result": ".UnclassifiedMalware",
                "update": "20180323",
                "version": "28732"
            },
            "CrowdStrike": {
                "detected": true,
                "result": "malicious_confidence_90% (W)",
                "update": "20170201",
                "version": "1.0"
            },
            "Cybereason": {
                "detected": true,
                "result": "malicious.6e11b4",
                "update": "20180225",
```

```
                "version": "1.2.27"
            },
            "Cylance": {
                "detected": true,
                "result": "Unsafe",
                "update": "20180323",
                "version": "2.3.1.101"
            },
            "Cyren": {
                "detected": true,
                "result": "W32/Trojan.QVEH-0467",
                "update": "20180323",
                "version": "5.4.30.7"
            },
            "DrWeb": {
                "detected": true,
                "result": "Trojan.DownLoader12.63371",
                "update": "20180323",
                "version": "7.0.28.2020"
            },
            "ESET-NOD32": {
                "detected": true,
                "result": "a variant of Win32/Spy.Banbra.OJV",
                "update": "20180323",
                "version": "17106"
            },
            "Emsisoft": {
                "detected": true,
                "result": "Gen:Variant.Graftor.176515 (B)",
                "update": "20180323",
                "version": "4.0.2.899"
            },
            "Endgame": {
                "detected": true,
                "result": "malicious (moderate confidence)",
                "update": "20180316",
                "version": "2.0.5"
            },
            "F-Prot": {
                "detected": false,
                "result": null,
                "update": "20180323",
                "version": "4.7.1.166"
            },
            "F-Secure": {
                "detected": true,
                "result": "Gen:Variant.Graftor.176515",
                "update": "20180323",
```

```
                "version": "11.0.19100.45"
            },
            "Fortinet": {
                "detected": true,
                "result": "W32/Banker.ABRS!tr.spy",
                "update": "20180323",
                "version": "5.4.247.0"
            },
            "GData": {
                "detected": true,
                "result": "Gen:Variant.Graftor.176515",
                "update": "20180323",
                "version": "A:25.16478B:25.11859"
            },
            "Ikarus": {
                "detected": true,
                "result": "Trojan-Spy.Win32.Banker.JU",
                "update": "20180323",
                "version": "0.1.5.2"
            },
            "Invincea": {
                "detected": false,
                "result": null,
                "update": "20180121",
                "version": "6.3.4.26036"
            },
            "Jiangmin": {
                "detected": true,
                "result": "Trojan/Banker.Agent.dmn",
                "update": "20180323",
                "version": "16.0.100"
            },
            "K7AntiVirus": {
                "detected": true,
                "result": "Spyware ( 00495e0a1 )",
                "update": "20180323",
                "version": "10.42.26592"
            },
            "K7GW": {
                "detected": true,
                "result": "Spyware ( 00495e0a1 )",
                "update": "20180323",
                "version": "10.42.26597"
            },
            "Kaspersky": {
                "detected": true,
                "result": "UDS:DangerousObject.Multi.Generic",
                "update": "20180323",
```

```
                "version": "15.0.1.13"
            },
            "Kingsoft": {
                "detected": false,
                "result": null,
                "update": "20180323",
                "version": "2013.8.14.323"
            },
            "MAX": {
                "detected": true,
                "result": "malware (ai score=86)",
                "update": "20180323",
                "version": "2017.11.15.1"
            },
            "Malwarebytes": {
                "detected": false,
                "result": null,
                "update": "20180323",
                "version": "2.1.1.1115"
            },
            "McAfee": {
                "detected": true,
                "result": "Artemis!5DBEC476E11B",
                "update": "20180323",
                "version": "6.0.6.653"
            },
            "McAfee-GW-Edition": {
                "detected": true,
                "result": "BehavesLike.Win32.Downloader.bc",
                "update": "20180323",
                "version": "v2015"
            },
            "MicroWorld-eScan": {
                "detected": true,
                "result": "Gen:Variant.Graftor.176515",
                "update": "20180323",
                "version": "14.0.297.0"
            },
            "Microsoft": {
                "detected": true,
                "result": "TrojanDownloader:Win32/Mavradoi.A",
                "update": "20180323",
                "version": "1.1.14600.4"
            },
            "NANO-Antivirus": {
                "detected": true,
                "result": "Trojan.Win32.Banker.dovzpa",
                "update": "20180323",
```

```
                "version": "1.0.100.22043"
            },
            "Paloalto": {
                "detected": true,
                "result": "generic.ml",
                "update": "20180323",
                "version": "1.0"
            },
            "Panda": {
                "detected": true,
                "result": "Trj/Genetic.gen",
                "update": "20180323",
                "version": "4.6.4.2"
            },
            "Qihoo-360": {
                "detected": false,
                "result": null,
                "update": "20180323",
                "version": "1.0.0.1120"
            },
            "Rising": {
                "detected": true,
                "result": "Malware.Undefined!8.C (TFE:5:
                  TpkRGAeMGc)",
                "update": "20180323",
                "version": "25.0.0.1"
            },
            "SUPERAntiSpyware": {
                "detected": false,
                "result": null,
                "update": "20180323",
                "version": "5.6.0.1032"
            },
            "SentinelOne": {
                "detected": false,
                "result": null,
                "update": "20180225",
                "version": "1.0.15.206"
            },
            "Sophos": {
                "detected": true,
                "result": "Mal/Banspy-I",
                "update": "20180323",
                "version": "4.98.0"
            },
            "Symantec": {
                "detected": true,
                "result": "Trojan.Gen.2",
```

```
                "update": "20180323",
                "version": "1.5.0.0"
            },
            "Tencent": {
                "detected": true,
                "result": "Win32.Trojan-banker.Agent.Szvy",
                "update": "20180323",
                "version": "1.0.0.1"
            },
            "TheHacker": {
                "detected": false,
                "result": null,
                "update": "20180319",
                "version": "6.8.0.5.2551"
            },
            "TotalDefense": {
                "detected": false,
                "result": null,
                "update": "20180323",
                "version": "37.1.62.1"
            },
            "TrendMicro": {
                "detected": true,
                "result": "TROJ_BANBRA.WWS",
                "update": "20180323",
                "version": "9.862.0.1074"
            },
            "TrendMicro-HouseCall": {
                "detected": true,
                "result": "TROJ_BANBRA.WWS",
                "update": "20180323",
                "version": "9.950.0.1006"
            },
            "VBA32": {
                "detected": true,
                "result": "TrojanBanker.Banbra",
                "update": "20180323",
                "version": "3.12.28.0"
            },
            "VIPRE": {
                "detected": true,
                "result": "Trojan.Win32.Generic!BT",
                "update": "20180323",
                "version": "65472"
            },
            "ViRobot": {
                "detected": false,
                "result": null,
```

```
                "update": "20180323",
                "version": "2014.3.20.0"
            },
            "WhiteArmor": {
                "detected": false,
                "result": null,
                "update": "20180223",
                "version": null
            },
            "Yandex": {
                "detected": true,
                "result": "Trojan.PWS.Agent!6ryk7HiAFms",
                "update": "20180323",
                "version": "5.5.1.3"
            },
            "Zillya": {
                "detected": true,
                "result": "Trojan.Agent.Win32.515513",
                "update": "20180323",
                "version": "2.0.0.3519"
            },
            "ZoneAlarm": {
                "detected": true,
                "result": "UDS:DangerousObject.Multi.Generic",
                "update": "20180323",
                "version": "1.0"
            },
            "Zoner": {
                "detected": false,
                "result": null,
                "update": "20180323",
                "version": "1.0"
            },
            "nProtect": {
                "detected": false,
                "result": null,
                "update": "20180323",
                "version": "2018-03-23.02"
            }
        },
        "sha1": "d7cefa1825fc2e2837da9c548ac98cec84c3edae",
        "sha256": "3258668c6aeb14f1c3d59cc2d5366da142ca95
            ef92a53b7d3a93d08fc217bee1",
        "total": 66,
        "verbose_msg": "Scan finished, information embedded"
    }
},
```

# APÊNDICE C – CONJUNTO DE REGRAS DESELVOLVIDO NA MONOGRAFIA

Listing C.1:

```
alert http $HOME_NET any -> $EXTERNAL_NET any (
msg:"Achado Requisição com User-Agent AutoIt";
flow:established,to_server; http.method; content:"GET";
http.uri; content:".exe"; nocase; endswith; http.user_agent;
content:"AutoIt"; depth:6; endswith; fast_pattern;
classtype:trojan-activity; sid:1; rev:1;)
```

Listing C.2:

```
alert http $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg:
"Download de arquivo por script AutoIt";
flow:established,to_server;http.method; content:"GET";
http.uri; pcre:"/(\.txt|\.tiff|\.zip|
\.dll|\.exe|\.au3)/U"; http.user_agent; content:"AutoIt";
classtype:trojan-activity; sid:2; rev:1;)
```

Listing C.3:

```
alert http $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg:
"Download de executável por script AutoIt a partir do WP";
flow:established,to_server;http.method; content:"GET";
http.uri; pcre:"wp-content/.*\.exe/U";
http.user_agent; content:"AutoIt"; classtype:trojan-activity;
sid:3; rev:1;)
```

Listing C.4:

```
alert dns $HOME_NET any -> any any (msg:
"Query para domínio DynDNS *.redirectme .net";
dns.query; content:".redirectme.net"; nocase; endswith;
classtype:bad-unknown; sid:4; rev:1;)
```

Listing C.5:

```
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:
"Possivel Download de Malware pelo Banload";
flow:established,to_server;http.method; content:"GET";
http.uri; pcre:"/includes/.*\.php/U";
http.uri; content:"cad-portal"; classtype:trojan-activity;
reference:url,f-secure.com/v-descs/trojan-downloader_w32
_banload_fvq.shtml; sid:5; rev:1;)
```

Listing C.6:

```
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:
"Download de imagens relacionado ao Banbra";
flow:established,to_server;http.method; content:"GET";
```

```
http.uri; pcre:"/technote/.*\.jpg/U";
http.uri; content:"paeksan"; classtype:trojan-activity;
reference:url,f-secure.com/v-descs/trojan-downloader_w32
_banload_fvq.shtml; sid:6; rev:1;)
```

Listing C.7:

```
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:
"Download de executáveis relacionado ao Banload";
flow:established,to_server;http.method; content:"GET";
http.uri; pcre:"/technote/.*\.exe/U";
http.uri; content:"paeksan"; classtype:trojan-activity;
reference:url,f-secure.com/v-descs/trojan-downloader_w32
_banload_fvq.shtml; sid:7; rev:1;)
```

Listing C.8:

```
alert tcp $HOME_NET any -> $EXTERNAL_NET 25 (msg:"Email
para Destinatário Suspeito-BANBRA";
pcre:"/(jaodas.*\@inbox\.com|h3llm45t.*\@vfemail\.net)/i";
classtype: trojan-activity ;sid:8 ; rev:1;)
```

Listing C.9:

```
alert http $HOME_NET any -> $EXTERNAL_NET any
(msg:"Requisição HTTP para URL encurtada - cl.ly";
flow:established,to_server; http.method; content:"GET";
http.start; content:"HTTP/1.1|0d 0a|Host|3a 20|cl.ly|0d 0a|
Connection|3a 20|Keep-Alive|0d 0a 0d 0a|"; endswith;
   fast_pattern;
classtype:bad-unknown; sid:9; rev:1;)
```

Listing C.10:

```
alert http $HOME_NET any -> $EXTERNAL_NET any
(msg:"Requisição HTTP para caminho suspeito - BANBRA Telex";
flow:established,to_server; http.method; content:"GET";
http.uri; pcre:"/(chackal|bode|toxppxoa|pcpxpaepxo)\.zip/U";
classtype:trojan-activity; sid:10; rev:1;)
```

Listing C.11:

```
alert http $HOME_NET any -> $EXTERNAL_NET any
(msg:"Telex BANBRA - CnC - Comunicação com servidor";
flow:established,to_server; http.method; content:"POST";
http.request_body; content:"ID_MAQUINA="; depth:11; nocase;
fast_pattern; content:"&VERSAO="; distance:0;
nocase;content:"&WIN=";distance:0;nocase;
classtype:command-and-control; sid:11; rev:3;)
```

Listing C.12:

```
alert http $EXTERNAL_NET any -> $HOME_NET any
```

```
(msg:"Telex BANBRA - CnC - Servidor enviando comandos";
pcre:"/(PING|Info|Close|reini|REQUESTINFO|REQUESTKEYBOARD|
HjiopPos|HjiopLD|HjiopLU|POWT|DESMON)/U";
classtype:command-and-control; sid:12; rev:1;)
```

Listing C.13:

```
alert http $EXTERNAL_NET any -> $HOME_NET any
(msg:"Mekotio - CnC - Comunicação com Servidor";
flow:established,from_server;
pcre:"/\<\|utypzjI|IXjzwtR|ztUjzwtR|SuaykJI|SuaykRJ|
lozyw|vhxboj|WGSGTNU|tksN|VOTM|LSTU|Gpsxi|
ZKXAKYWQKEHUGZJ\|\>/U"; classtype:command-and-control;
sid:13; rev:1;)
```

Listing C.14:

```
alert http $EXTERNAL_NET any -> $HOME_NET any
(msg:"Possível fraude na geração de boleto bancário";
flow:established,from_server;
pcre:"/{^\d{11}\-\d{1}$}4/";
classtype:command-and-control; sid:14; rev:1;)
```