UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

ANÁLISE E EXTRAÇÃO DE CARACTERÍSTICAS ESTRUTURAIS E COMPORTAMENTAIS PARA PERFIS DE MALWARE

DANIEL MENDES CALDAS

ORIENTADOR: ROBSON DE OLIVEIRA ALBUQUERQUE
CO-ORIENTADOR: FLÁVIO ELIAS GOMES DE DEUS

DISSERTAÇÃO DE MESTRADO EM ENGENHARIA ELÉTRICA
ÁREA DE CONCENTRAÇÃO INFORMÁTICA FORENSE E SEGURANÇA DA INFORMAÇÃO

PUBLICAÇÃO: PPGENE.DM - 622 A/16

BRASÍLIA / DF: DEZEMBRO/2016

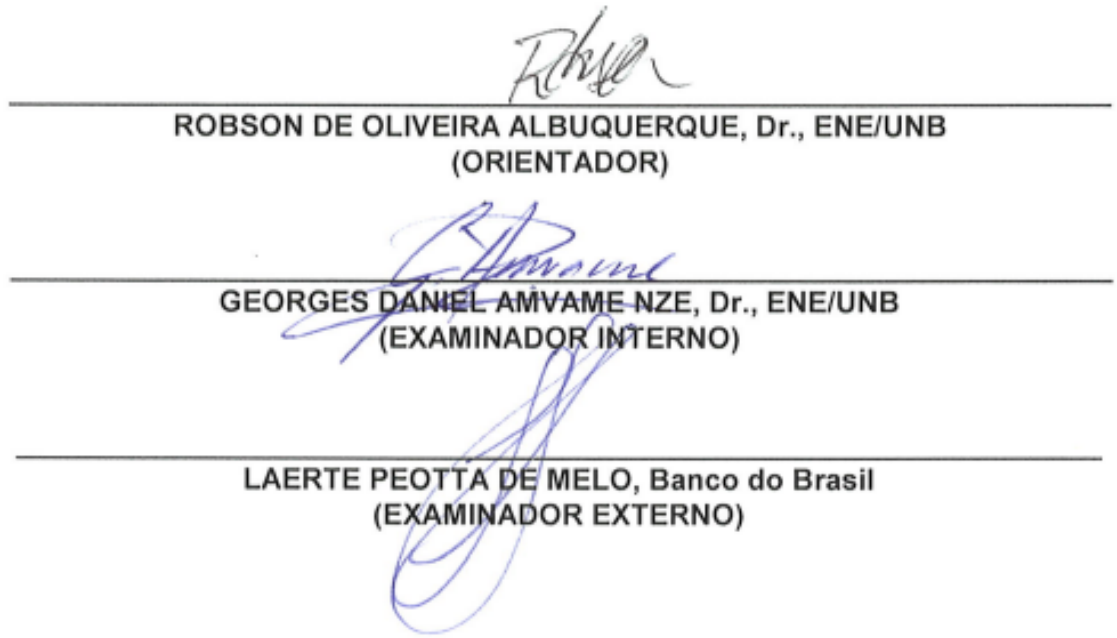
UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

ANÁLISE E EXTRAÇÃO DE CARACTERÍSTICAS ESTRUTURAIS E COMPORTAMENTAIS PARA PERFIS DE MALWARE

DANIEL MENDES CALDAS

DISSERTAÇÃO DE MESTRADO PROFISSIONAL SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE.

APROVADA POR:



ROBSON DE OLIVEIRA ALBUQUERQUE, Dr., ENE/UNB
(ORIENTADOR)

GEORGES DANIEL AMVAME NZE, Dr., ENE/UNB
(EXAMINADOR INTERNO)

LAERTE PEOTTA DE MELO, Banco do Brasil
(EXAMINADOR EXTERNO)

BRASÍLIA/DF, 05 DE DEZEMBRO DE 2016.

FICHA CATALOGRÁFICA

CALDAS, DANIEL
ANÁLISE E EXTRAÇÃO DE CARACTERÍSTICAS ESTRUTURAIS E COMPORTAMENTAIS PARA PERFIS DE MALWARE [Distrito Federal] 2016.
xxiii, 82p., 297 mm (ENE/FT/UnB, Mestre, Engenharia Elétrica, 2016).

Dissertação de Mestrado – Universidade de Brasília, Faculdade de Tecnologia. Departamento de Engenharia Elétrica.

1. Análise de Malware 2. Classificação
3. Atributos chave

I. ENE/FT/UnB. II. Título (Série)

REFERÊNCIA BIBLIOGRÁFICA

CALDAS, D. M. (2016). ANÁLISE E EXTRAÇÃO DE CARACTERÍSTICAS ESTRUTURAIS E COMPORTAMENTAIS PARA PERFIS DE MALWARE. Dissertação de Mestrado, Publicação PPGENE.DM - 622 A/16, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 82p.

CESSÃO DE DIREITOS

NOME DO AUTOR: Daniel Mendes Caldas
TÍTULO DA DISSERTAÇÃO: ANÁLISE E EXTRAÇÃO DE CARACTERÍSTICAS ESTRUTURAIS E COMPORTAMENTAIS PARA PERFIS DE MALWARE.
GRAU/ANO: Mestre/2016.

Daniel Mendes Caldas
QE 46, Área Especial 3. Bloco D, Apt 501 Guara II - DF
CEP 71070-649 – Brasília – DF - Brasil

Aos meus pais e amigos.

AGRADECIMENTOS

Aos professores Prof. Dr. Robson de Oliveira Albuquerque, orientador deste trabalho, e Prof. Dr. Flávio Elias Gomes de Deus, co-orientador deste trabalho, pelo constante apoio, incentivo, e dedicação, essenciais para o desenvolvimento deste trabalho e para o meu desenvolvimento como pesquisador.

Aos professores e demais servidores que contribuíram com o curso de Pós-Graduação em Engenharia Elétrica do Departamento de Engenharia Elétrica.

Ao colega de curso, Fábio Luiz da Rocha Moraes, pelo apoio no decorrer do curso e no desenvolvimento do trabalho.

A todos, os meus sinceros agradecimentos.

O presente trabalho foi realizado com o apoio do Departamento Polícia Federal – DPF e do Instituto de Criminalística da Polícia Civil do Distrito Federal, com recursos do Programa Nacional de Segurança Pública com Cidadania – PRONASCI, do Ministério da Justiça.

RESUMO

ANÁLISE E EXTRAÇÃO DE CARACTERÍSTICAS ESTRUTURAIS E COMPORTAMENTAIS PARA PERFIS DE MALWARE

Autor: Daniel Mendes Caldas

Orientador: Robson de Oliveira Albuquerque

Programa de Pós-graduação em Engenharia Elétrica

Brasília, dezembro de 2016

Grande parte das ações criminosas na rede são praticadas com uso de softwares maliciosos, comumente denominados malwares. Malwares são programas desenvolvidos com o intuito de se infiltrar em um sistema de computador alheio de forma ilícita, para causar danos, alterações ou roubo de informações (confidenciais ou não). Uma resposta eficiente do Estado para um crime cibernético requer o entendimento do meio utilizado para o cometimento da ação criminosa. No ambiente jurídico, a análise de malware deve prover evidências necessárias para a materialização do fato e a determinação da autoria, visando a condenação do agente malicioso, e a fornecer meios para inocentar os erroneamente acusados. Existem duas abordagens fundamentais para o problema da análise de malware, a análise estática, que envolve examinar o malware sem executá-lo observando suas características estruturais, e a análise dinâmica, que envolve analisar o malware durante sua execução, observando suas características comportamentais.

A maior parte da pesquisa feita hoje na área de análise de malware tem seu foco na detecção, como forma de prevenção e de evitar ataques. A pesquisa na área de classificação de malware ainda se encontra em estágios iniciais, e as técnicas desenvolvidas até então demonstram serventia limitada no dia a dia do investigador forense. O objetivo deste trabalho é realizar uma contribuição na área de classificação de malware. Tal contribuição é feita na forma de um estudo do panorama atual da área, na indústria de antivírus e na literatura. O estudo apresenta as principais classificações existentes até então, e suas limitações. É apresentada uma nova proposta de classificação, definida como profiling de malware, baseada no profiling criminal. Além da definição de perfis, este trabalho sugere características, comportamentais e estruturais, que podem representar comportamentos que reflitam os perfis, e apresenta uma estratégia eficiente para extração das características, utilizando a ferramenta Cuckoo Sandbox. Por fim faz-se uma análise das características extraídas, que revela que sete das onze características extraídas se mostraram promissoras para o profiling de malware.

ABSTRACT

ANALYSIS AND EXTRACTION OF STRUCTURAL AND BEHAVIOURAL FEATURES FOR MALWARE PROFILING

Author: Daniel Mendes Caldas

Supervisor: Robson de Oliveira Albuquerque

Programa de Pós-graduação em Engenharia Elétrica

Brasília, december of 2016

Most of the criminal actions in the network are practiced with the use of malicious software, commonly denominated malwares. Malwares are programs designed to infiltrate a computer system unlawfully, to cause damage, alteration or theft of information (confidential or not). An efficient response of the state to a cyber crime requires understanding the means used to commit the criminal action. In the legal environment, malware analysis should provide evidence necessary for the materialization of the fact and the determination of authorship, aiming at condemning the malicious agent, and providing means to clear the wrongly accused. There are two fundamental approaches to the problem of malware analysis, the static analysis, that involves examining the malware without executing it, observing its structural features, and the dynamic analysis, that involves analyzing the malware during its execution, observing its behavioral features.

Most of the research done today in the area of malware analysis has its focus on malware detection, as a way of preventing attacks. The research in the area of malware classification is still in the early stages, and the techniques developed until then show limited use in the day-to-day of the forensic investigator. The goal of this work is to make a contribution in the area of malware classification. This contribution is made with a study of the current panorama of malware classification, in the antivirus industry and in the literature. The study presents the main classes definitions until now, and their limitations. A new classification strategy based on criminal profiling is presented, named as malware profiling. In addition to the definition of profiles, this work suggests features, behavioral and structural, that represent behaviors of the profiles, and presents an efficient strategy to extract these features from malware samples, using the Cuckoo Sandbox tool. Finally, an analysis of the extracted features is performed, which reveals that seven of the eleven extracted features have shown promise for malware profiling.

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE SÍMBOLOS, NOMENCLATURA E ABREVIAÇÕES

API	Application program interface
APK	Android Application Package
CARO	Computer Anti-virus Researchers Organization
CD	Compact Disc
CERT.BR	Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil
CPU	Central Processing Unit
CSS	Cascading Style Sheets
DLL	Dynamic-link library
DoS	Denial of Service
DDoS	Distributed Denial of Service
FTP	File Transfer Protocol
GNU GPL	GNU General Public License
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IDA Pro	Interactive Disassembler
IM	Instant Message
IP	Internet Protocol
IRC	Internet Relay Chat
JSON	JavaScript Object Notation
MD5	Message Digest Algorithm 5
P2P	Peer-2-Peer
PeID	Portable Executable Identificator
QEMU	Quick Emulator
SHA	Secure Hash Algorithm
VM	Virtual Machine
XML	Extensible Markup Language

1. INTRODUÇÃO

A análise de malware é uma das principais ferramentas da informática forense para investigação de crimes cibernéticos. Seu objetivo é ganhar a compreensão de como funciona uma parte específica de um malware, de modo a permitir a interpretação dos vestígios do crime e a geração de evidências. Neste capítulo é apresentada uma introdução ao tema, com a motivação que levou à sua escolha, a justificativa para os caminhos seguidos e os objetivos gerais e específicos deste trabalho.

A Tecnologia da Informação avançou rapidamente em pouco tempo (STALLINS, 2002). As instituições, tanto públicas quanto privadas, estão utilizando estes avanços tecnológicos para melhorar seus processos, as operações empresariais e o seu valor de mercado. Hoje em dia é possível pagar contas on-line, comprar desde livros a mantimentos, realizar transações financeiras diversas, e trabalhar à distância, inclusive lidando com dados de extrema sensibilidade (MARCIANO, 2006). Tal praticidade trouxe também um ponto frágil para as instituições. Uma vasta quantia de importantes e sensíveis dados ficam armazenados, concentrados, e fluem ao redor da internet, muitas vezes desprotegidos. Estatísticas para o ano de 2015 mostram que uma nova vulnerabilidade foi descoberta a cada semana (SYMANTEC, 2016). Acompanhando o surgimento de fragilidades nos sistemas de informática, é natural que o número de agentes maliciosos tenda a crescer. A Figura 1.1 traz o número de novos malwares detectados, por trimestre, pela empresa McAfee Labs. O gráfico apresentado na Figura 1.1 mostra um aumento considerável no número de de malware detectados, desde o primeiro trimestre do ano de 2013. Pode-se notar também que, entre o primeiro e o segundo trimestre do ano de 2014 há uma queda no número de novas amostras de malware dectadas, porém o relatório não explica se essa queda se deu devido à ineficência da ferramenta de detecção, ou se realmente houve um menor surgimento de malware na rede.

De acordo com o relatório do McAfee labs sobre ameaças, de fevereiro de 2015, a quantidade de novos itens de malware detectados não para de crescer. A cada minuto, são detectadas 387 novas ameaças. O total de amostras armazenadas nos laboratórios da McAfee aumentou 17% do terceiro trimestre para o quarto trimestre de 2014 (MCAFEE LABS, 2015).

Fonte: Retirado de (MCAFEE LABS, 2015)

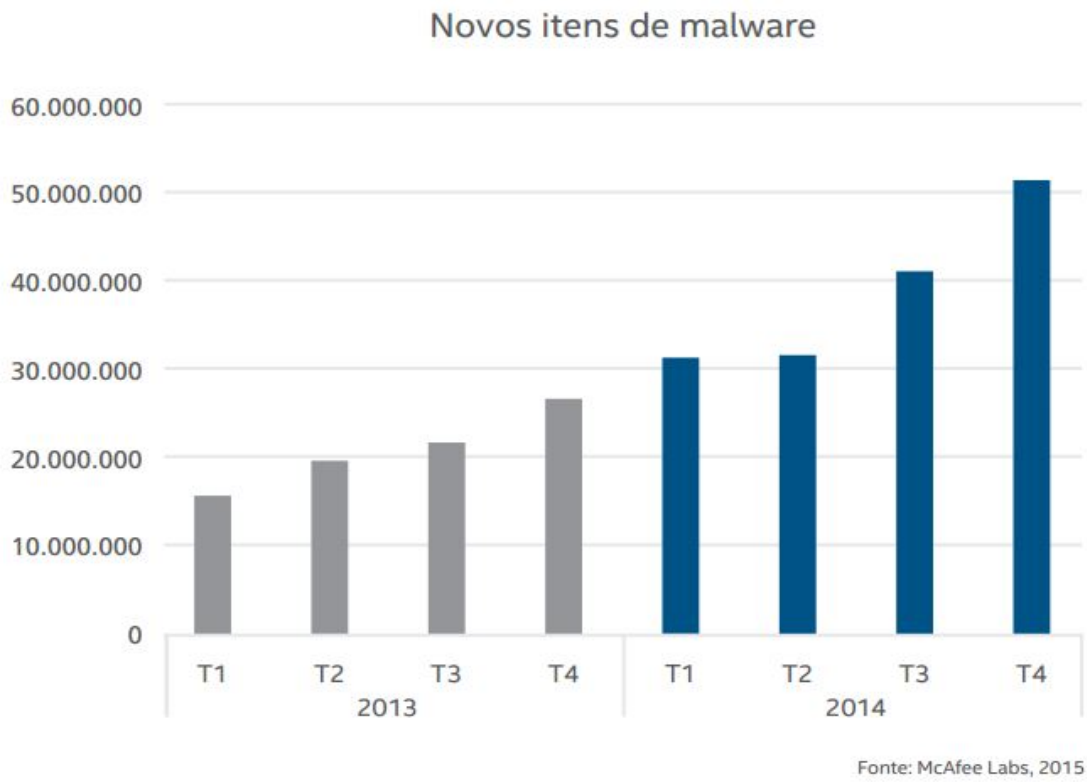


Figura 1.1: Novos itens de malware

A segurança pública é dever do Estado, direito e responsabilidade de todos, nos termos do artigo 144 da Constituição da República (BRASIL, 1988). Quando uma pessoa comete uma infração penal, atingindo um bem alheio (seja ele real ou virtual), cabe ao estado exercer o seu poder de polícia, com ações repressivo-investigativas. Tais ações se manifestam, no Brasil, através da polícia judiciária (Polícia Civil e Polícia Federal).

Para poder proteger os seus cidadãos, reprimindo e punindo os agentes maliciosos, o Estado precisa estar preparado para atuar com eficácia e rapidez no caso de ações delituosas no meio virtual, e tal preparo se traduz em profissionais capacitados e práticas efetivas de investigação nas polícias judiciárias.

1.1. MOTIVAÇÃO

Grande parte das ações criminosas na rede são praticadas com uso de softwares maliciosos, comumente denominados malwares. Malwares são programas desenvolvidos com o intuito de se infiltrar em um sistema de computador alheio de forma ilícita, para causar danos, alterações ou

roubo de informações (confidenciais ou não). São considerados malwares os vírus de computador, worms, trojan horses (cavalos de troia), spywares, e outros (SIKORSKI; HONIG, 2012).

Enquanto a Segurança da Informação tem como objetivo prevenir e detectar falhas na segurança (MARCIANO, 2006), a Informática Forense fornece uma análise post-facto de uma brecha no sistema. Ela não tem como objetivo recuperar o dano causado, mas visa o entendimento dos meios utilizados para a prática do crime e fornecer inteligência para elucidar o fato. No ambiente jurídico, a análise de malware deve prover evidências necessárias para a materialização do fato e a determinação da autoria, visando a condenação do agente malicioso, e a fornecer meios para inocentar os erroneamente acusados.

Quando se está investigando um crime, questões básicas vem à tona. O investigador precisa descobrir onde e como o crime foi cometido, quais os meios utilizados pelo agente criminoso, quem é esse agente e quais são suas motivações. No caso de um crime cibernético, essas questões só podem ser completamente respondidas com o entendimento do meio utilizado (malware), que permite aos investigadores a interpretação dos vestígios deixados nas ações criminosas, para a síntese de evidências, como pode ser visto na Figura 1.2.

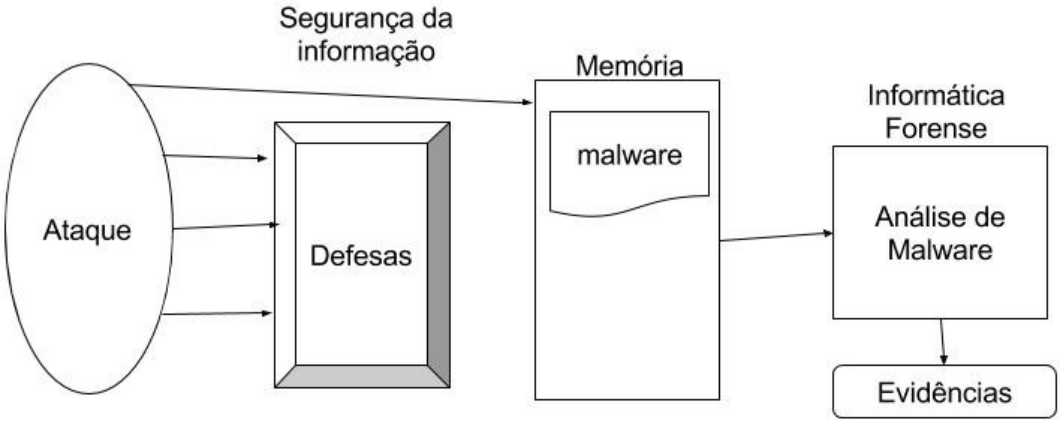


Figura 1.2: Análise de Malware na Informática Forense

Apesar de possuir estrutura legal satisfatória, a legislação penal para crimes cibernéticos no Brasil ainda é recente. O alargamento das relações entre as pessoas – decorrente das mudanças geradas pelo avanço tecnológico – implica na criação de novos fatores criminógenos, os quais decorrem justamente do estabelecimento de condutas humanas anteriormente inexistentes, bem como do surgimento de novas ferramentas que se consubstanciam como objeto, ou mesmo meio, para a prática de novos delitos. Assim, a Lei nº 12.737, de 30 de novembro de 2012, trouxe novos tipos e modalidades de crimes cibernéticos. Diante deste cenário, a tendência é que cada vez mais as condutas virtuais reprovadas pelos legisladores, sejam tratadas como ilícitas na sociedade.

O novo código penal deverá conter um título específico para tratar de crimes cibernéticos (BRASIL, 2012), e as mudanças contidas no dispositivo trazem uma legislação mais específica. Dessa forma, somente o entendimento completo da ação criminosa cibernética permitirá aos operadores do direito a correta aplicação da legislação penal, e tal entendimento depende fortemente de uma análise eficiente do malware utilizado por parte dos investigadores forenses.

1.2. JUSTIFICATIVA

A maior parte da pesquisa feita hoje na área de análise de malware tem seu foco na detecção, como forma de prevenção e de evitar ataques. A pesquisa na área de classificação de malware ainda se encontra em estágios iniciais, e as tecnicas desenvolvidas até então demonstram serventia limitada no dia a dia do investigador forense. Uma definição de perfis de malware pode ser muito útil quando o tempo é curto, e as consequências são críticas. Esforços feitos por governos e instituições policiais por todo o mundo têm ajudado a combater os crimes cibernéticos (BROADHURST et al., 2014). Ações focadas no entendimento dos aspectos das ferramentas utilizadas para perpetrar o crime, como por exemplo: identificação e decifragem de malware, reconhecimento e combate a botnets e a outros meios utilizados pelos criminosos devem ser aprimoradas. Ao mesmo tempo, a falta de informações sobre os criminosos, seu modus operandi, e suas motivações geram dificuldades no combate ao crime cibernético.

Como não existe defesa perfeita, é necessária a interpretação dos vestígios deixados pelo ataque com o uso das técnicas da Informática Forense. As técnicas de análise de malware, como engenharia reversa e interpretação de vestígios, nos permitem entender o funcionamento do malware e extrair evidências importantes dos vestígios deixados pelo ataque, utilizando dumps de memória, imagens de disco, pacotes de rede capturados, entre outros. Extrair informações desses vestígios permite identificar informações importantes sobre a ação criminosa, seus objetivos e autoria.

Nesse panorama, a análise forense de crimes cibernéticos se traduz em combinar ferramentas e técnicas específicas, para extrair, interpretar e combinar os vestígios, obtendo informação útil para a correta persecução penal. As técnicas que envolvem análise de malware ainda necessitam de muito trabalho manual. São necessários, então, maiores esforços no sentido de automatizar e tornar mais eficiente o trabalho dos investigadores.

1.3. OBJETIVOS

O objetivo geral deste trabalho é realizar um estudo do panorama atual da classificação de malware, na indústria de antivirus e na literatura. O estudo apresenta as principais classificações existentes até então, e suas limitações.

A ideia é contribuir para o desenvolvimento de uma metodologia automatizada, que permita a classificação de malwares em grupos (perfis), utilizando características estruturais e comportamentais desses.

Objetivos específicos deste trabalho:

- Apresentação de uma lista de perfis de malware
- Realização de análise e seleção de características relevantes para o processo de definição de perfis de malware
- Desenvolvimento de uma técnica eficiente para extração das características selecionadas de arquivos binários de malware
- Extração das características com a técnica utilizada e análise dos resultados

1.4. ESTRUTURA DO TRABALHO

O conteúdo dessa dissertação está estruturado da seguinte forma:

Capítulo dois introduz os conceitos e fundamentos que são essencias para o entendimento do trabalho. É feita uma breve apresentação do conceito de malware, seus objetivos e modos de proliferação. O conceito de análise de malware é explanado e seus principais objetivos e desafios são delineados. Traz também um panorama da área de classificação de malware. Mostra as abordagens que vêm sido usadas para tratar o problema de classificação de malware na indústria e na literatura, e suas limitações. Apresenta trabalhos que compõe os últimos avanços na área, e trabalhos que serviram de referência para o desenvolvimento desta dissertação.

Capítulo três traz a ideia de perfis de malware. Apresenta-se o conceito de profiling, e como ele pode ser utilizado para malware. São definidos os perfis e é realizada uma discussão a respeito das características estruturais e comportamentais de malware, para a seleção das características consideradas mais relevantes para o processo de definição de perfis de malware.

O capítulo quatro apresenta uma metodologia para extração das características selecionadas. A ferramenta utilizada é explicada, e explana-se como ela foi utilizada para a extração das

características. Traz também uma análise das características extraídas, discutindo sua relevância no processo de definição de perfis de malware.

Finalmente, o capítulo cinco traz um sumário do trabalho desenvolvido na dissertação, as conclusões, limitações, problemas da abordagem adotada, e possíveis trabalhos futuros.

2. ANÁLISE E CLASSIFICAÇÃO DE MALWARE

Neste capítulo são apresentados os conceitos e fundamentos essencias para o entendimento do trabalho. É feita uma apresentação do conceito de malware, do conceito de análise de malware, e do panorama da área de classificação de malware, na indústria e na literatura. Esse panorama mostra as abordagens que vêm sido usadas para tratar o problema de classificação de malware em ambas as áreas, e suas limitações. Por fim são apresentados trabalhos que compõe os últimos avanços na área, e trabalhos que serviram de referência para o desenvolvimento desta dissertação.

2.1. SOFTWARE MALICIOSO (MALWARE)

Malware, uma abreviação para software malicioso, são programas desenvolvidos com o objetivo de alterar sistemas computacionais, obter informações, acesso, privilégios ou exibir propaganda, de forma não autorizada e/ou desejada pelo usuário (AYCOCK, 2006). O conceito é basicamente definido pelo objetivo malicioso, agindo contra a vontade do usuário do sistema, e não inclui programas que causam danos não intencionais, resultantes de alguma falha não prevista por seus criadores.

Malware é um conceito abrangente, que pode ser dividido em vários grupos, incluindo virus, worms, cavalos de troia, rootkits, entre outros (SIKORSKI; HONIG, 2012). São binários executáveis, scripts, ou até mesmo fragmentos de código malicioso, inclusos em código benigno de forma escusa, infectando aplicações, processos do sistema operacional, ou até mesmo código de boot do sistema.

As origens dos malwares são as mais diversas. Podem ser criados por agentes motivados por interesse financeiro, por ideologias ou até mesmo por diversão. Existem casos de malwares encontrados em software fornecido oficialmente por companhias de tecnologia. Em 2005 foram descobertos rootkits em CDs vendidos pela empresa Sony, que se instalavam silenciosamente e se escondiam nos computadores dos compradores dos CDs. Esses rootkits tinham como objetivo prevenir a cópia ilícita dos CDs, e fornecer dados para a empresa sobre os hábitos musicais dos compradores. A instalação desses rootkits criou ainda vulnerabilidades não intencionais nos sistemas, que foram usadas por malwares criados por outras pessoas (MICROSOFT, 2005)

Eles exploram falhas de segurança na concepção do sistema operacional, em aplicações (como navegadores, por exemplo), ou em versões vulneráveis de plugins do navegador, como Adobe Flash Player, Acrobat Reader, e Java.

Um exemplo de vulnerabilidade comum é a exploração de saturação de buffer, onde o software projetado para armazenar dados em uma região específica de memória não impede que mais dados do que o buffer pode acomodar sejam fornecidos. O malware então fornece dados que transbordam o buffer, com código executável malicioso, que quando acessado, pode realizar ações sem a autorização do usuário. Um usuário pode ser infectado de diversas formas, entre elas executando arquivos, acessando links na internet, abrindo e-mails infectados, entre outros.

2.2. ANÁLISE DE MALWARE

A análise de malware é o estudo de amostras de malware, com o objetivo de determinar sua funcionalidade, e extrair a maior quantidade de informações possíveis. É a arte de dissecar o programa malicioso para entender como ele funciona, descobrir como o identificar, e como elimina-lo.

Durante a análise de malware, é comum trabalhar em um arquivo binário, que só pode ser lido em linguagem de máquina (assembly). Para o entendimento do binário, deve-se lançar mão de uma variedade de técnicas e ferramentas, e assim ter uma visão mais abrangente de seu funcionamento, cada uma revelando pequenas quantidades de informação do quebra-cabeça.

Existem duas abordagens fundamentais para o problema da análise de malware, análise estática e análise dinâmica. A análise estática envolve examinar o malware sem executá-lo, observando suas características estruturais, e análise dinâmica envolve analisar o malware durante sua execução, observando suas características comportamentais.

2.2.1. Análise estática

É o conjunto de técnicas e ferramentas utilizadas para analisar o malware sem executá-lo. Pode ser aplicada no código fonte ou no arquivo binário, gerado após a compilação do programa. Durante o processo de compilação de um código fonte, algumas informações podem ser perdidas, o que pode complicar a tarefa de analisar o código (SECLAB, 2012)

Nos dias de hoje, o processo de inspeção de um binário sem executá-lo é feita principalmente de forma manual e, apesar de mais segura, pode ser extremamente demorada e requer grande expertise do investigador. As principais técnicas utilizadas na análise estática podem ser vistas na Tabela 2.1.

Tabela 2.1: Técnicas de análise estática.

Técnica	Descrição
Fingerprinting	Envolve operações em nível de arquivo, como calcular hashs criptográficos do arquivo binário (MD5, SHA-1, etc.), com o objetivo de identificar e distinguir o malware de outros arquivos, e verificar que o arquivo não foi modificado.
Extração de strings	Programas normalmente exibem mensagens de texto, que podem acabar inseridas no código binário como cadeias de caracteres de texto. Examinar tais cadeias de caracteres (strings) permite ao investigadaor tecer conclusões a respeito do malware, orientando os próximos passos da análise.
Detecção de Packers	A maior parte dos malwares nos dias de hoje são encontrados de forma compactada ou encriptada. Isso pode ser feito com o uso de packers, programas feitos para disfarçar o malware e dificultar sua análise pelos investigadores. Programas como o PeiD (ALDEID, 2016) detectam se o arquivo binário está compactado, e podem indicar qual packer foi utilizado, podendo assim serem úteis para o processo de análise estática.
Disassembly	A maior parte da análise estática se dá através da engenharia reversa do código, por meio do disassembly de seu binário. É feita utilizando ferramentas capazes de converter o código de máquina em linguagem assembly. Com o código assembly reconstruído, o investigador pode inspecionar a lógica do programa, descobrindo sua função e como ele trabalha. Ferramentas normalmente usadas nessa etapa incluem IDA Pro (HEX-RAYS, 2016) e OllyDbg (YUSCHUK, 2014).
Descompiladores	Ferramentas cujo objetivo é reverter o binário em código fonte, fazendo o caminho reverso do processo de compilação. Tal recuperação não é possível na maioria das arquiteturas e linguagens, mas pode ser útil em casos específicos.

2.2.2. Análise dinâmica

Consiste em executar uma determinada amostra de malware, em um ambiente controlado, e monitorar suas ações, com o objetivo de analisar seu comportamento. O monitoramento é feito através da coleta de informações do sistema, como logs, arquivos alterados, processos criados, bibliotecas utilizadas pelo malware, entre outros. Envolve também o uso de um depurador, que permite ao investigador observar o fluxo de execução do programa durante para uma determinada entrada.

Uma grande vantagem da análise dinâmica é que durante a execução na memória, o malware não está compactado nem ofuscado, evitando assim algumas técnicas de ofuscação comumente utilizadas para impedir a análise estática.

Uma desvantagem em relação à análise estática é que apenas um caminho de execução é observado, o que acaba gerando uma cobertura incompleta do código. Além disso malwares comumente são programados para alterar seu comportamento ou até mesmo parar de executar caso detectem que estão sendo executados em um ambiente controlado. Além disso, a análise dinâmica possui maior dificuldade para ser realizada de forma automática, impossibilitando assim análise de amostras em larga escala.

2.3. ANÁLISE AUTOMÁTICA DE MALWARE

Realizar análise automática de malware é um processo que consiste em utilizar técnicas com pouca ou nenhuma intervenção humana para obter informações acerca de características funcionais do malware, através da sua execução em um ambiente controlado.

Essa análise é obtida através de, principalmente, duas metodologias (WILLEMS; HOLZ; FREILING, 2007) :

- Monitoramento de ações em tempo de execução;
- Comparação do estado do sistema operacional antes e depois da execução da amostra de malware

A comparação de estados do sistema operacional traz as alterações realizadas pelo binário de executado. Gera-se uma visão geral das funcionalidades da amostra de malware, como arquivos criados e deletados, entre outros. Essa metodologia, no entanto, é limitada, pois não captura as mudanças dinâmicas intermediárias que ocorrem no decorrer da execução. Mudanças como chamadas de APIs, arquivos criados e removidos antes do final da execução, entre outas, não são detectadas por essa análise.

Já o monitoramento em tempo real realiza a captura de dados relacionados à execução da amostra durante a execução. A implementação normalmente é feita através de hooks e DLL injections (SIKORSKI; HONIG, 2012), e apesar de mais complexa, traz bons resultados quando técnicas de polimorfismo e ofuscação são utilizadas pelas amostras de malware.

Máquinas virtuais e ambientes virtualizados são muito úteis para o investigador forense, quando o objetivo é realizar análises automatizadas de amostras de malware. Essas técnicas permitem a execução de uma amostra de malware em ambiente controlado de forma prática. Configurações e

aplicativos de máquinas virtuais podem ser facilmente alteradas, e diversas máquinas virtuais com caracteristicas ligeiramente diferentes podem ser utilizadas ao mesmo tempo.

2.3.1. Máquina virtual

Na computação, uma máquina virtual (VM) é uma emulação de um sistema computacional. Máquinas virtuais são baseadas em arquiteturas de computador e fornecem a funcionalidade de um computador físico, emulado dentro de um outro sistema computacional. Suas implementações podem envolver hardware especializado, software ou uma combinação desses.

Existem diferentes tipos de máquinas virtuais (SMITH; NAIR, 2005) , cada uma com funções diferentes:

- As máquinas virtuais de sistema (também conhecidas como VMs de virtualização completas) fornecem um substituto para uma máquina real. Eles fornecem a funcionalidade necessária para executar sistemas operacionais inteiros. Um hypervisor usa a execução nativa para compartilhar e gerenciar hardware, permitindo que vários ambientes que estão isolados uns dos outros, coexistam na mesma máquina física, denominada host. Os hypervisors modernos usam virtualização assistida por hardware, principalmente a partir das CPUs do host (GOLDEN, 2011).
- Máquinas virtuais de processo são projetadas para executar programas de computador em um ambiente independente de plataforma.

Algumas máquinas virtuais, como a QEMU, são projetadas para também emular arquiteturas diferentes e permitir a execução de aplicativos de software e sistemas operacionais escritos para outra CPU com diferentes arquiteturas. A virtualização no nível do sistema operacional permite que os recursos de um computador sejam particionados por meio do suporte do kernel para várias instâncias isoladas do espaço do usuário, que geralmente são chamadas de contêineres e podem parecer e se sentir como máquinas reais para os usuários finais (BELLARD, 2005).

2.3.2. Sandboxing para análise de malware

Um sandbox é um mecanismo de segurança para separar os programas em execução. É frequentemente usado para executar programas não confiáveis, possivelmente de terceiros não verificados ou não confiáveis, fornecedores, usuários ou websites, sem risco de danos à máquina host ou ao sistema operacional (GOLDBERG et al., 1996).

Pesquisadores da área de segurança dependem fortemente das tecnologias de sandboxing para analisar o comportamento de malware. Ao criar um ambiente que imita ou replica os desktops

direcionados, os pesquisadores podem avaliar como o malware infecta e compromete um host de destino. Vários serviços de análise de malware são baseados na tecnologia sandboxing (YEE et al., 2009).

Um sandbox normalmente fornece um conjunto controlado de recursos para programas a serem executados, como espaço no disco e na memória. O acesso à rede, a capacidade de inspecionar o sistema host ou ler a partir de dispositivos de entrada são geralmente desativados ou fortemente restringidos.

No sentido de fornecer um ambiente altamente controlado, um sandbox pode ser visto como um exemplo específico de virtualização. Sandboxes podem ser interpretados como ambientes virtualizados que possuem um nível superior de controle e segurança.

O avanço das técnicas de virtualização contribuiram para o surgimento de uma série de novas ferramentas de análise de malware que utilizam o conceito de sandboxing, como Cuckoo Sandbox (CUCKOO FOUNDATION, 2016), Anubis (ISECLAB, 2015), entre outros. Essas ferramentas permitem configurações personalizadas para análise automatizada de arquivos binários de malware. Fazem a execução de arquivos binários em ambientes controlados, gerando um relatório com informações sobre as ações realizadas pela amostra de malware.

Ferramentas de análise de malware em sandbox permitem o monitoramento de, entre outros:

- Dados trafegados pela rede
- Processos criados
- Áreas de memória acessadas
- DLLs carregadas
- Chamadas de APIs feitas ou importadas
- Arquivos acessados
- Modificação de registro do sistema operacional

Um fator a ser levado em consideração pelo investigador forense ao escolher uma ferramenta de análise de malware em sandbox é o tipo de resultado esperado. Os relatórios podem variar muito de ferramenta para ferramenta, pois dependem das particularidades de cada uma e da implementação das mesmas.

Outro fator muito importante é a transparência. Muitas ferramentas de sandboxing são comerciais. Dessa forma, o perito não tem acesso ao código fonte, e não pode realizar estudos ou modificações

nesse código. Além disso, a confirmação pelo perito de que o resultado é válido fica limitada quando se utiliza ferramentas comerciais.

Considerando-se esses fatores, foi escolhida a ferramenta Cuckoo Sandbox para uso neste trabalho, que é modular, altamente configurável e personalizável, e está sobre licença GNU GPL.

A tabela x traz uma lista com algumas das principais ferramentas de sandboxing utilizadas para análise de malware existentes hoje, com suas principais características.

Tabela 2.2: Ferramentas de sandboxing para análise de malware.

Ferramenta	Descrição
Anubis	Anubis é um popular serviço online para analisar arquivos executáveis do Windows desconhecidos. Quatro formatos de relatório (HTML, XML, PDF e Texto) são apresentados como resultado, assim que a análise é concluída. Um dos pontos positivos do Anubis é o resumo encontrado na parte superior do relatório, que interpreta os resultados informando o que os arquivos fazem, em vez de apenas mostrar informações técnicas sobre as atividades do arquivo. É uma ferramenta de código fechado e não tem possibilidade de customização.
ThreatAnalyzer	Anteriormente conhecido como CWSandbox, o ThreatAnalyzer é um serviço on-line que executa o arquivo enviado realizando uma análise automatizada de malware. Permite submeter amostras de malware ou arquivos compactados, com diversas amostras, através de um simples campo no navegador. Depois disso, cria uma fila de submissão e a executa, realizando uma série de testes. Também possui código fechado e não tem possibilidade de customização.
Comodo	Comodo Instant Malware Analysis é outro serviço de análise de malware online por sandoboxing, sendo um dos mais simples de usar e entender. Basta procurar o arquivo que você deseja analisar, marcar a caixa para concordar com seus termos e clicar no botão. O arquivo será então analisado em tempo real e a página do relatório será atualizada continuamente até que a análise seja concluída. Também é uma ferramenta de código fechado e não traz possibilidades de customização.
Akana	Akana é um serviço on-line para análise de software feito para android (APK), que combina alguns plugins para verificar se o arquivo submetida para análise se trata de um malware ou não.

2.4. CLASSIFICAÇÃO DE MALWARE

Neste tópico é apresentado um panorama da área de classificação de malware, na indústria e na literatura. Mostra os últimos trabalhos na área, com as abordagens que vêm sido usadas para tratar o problema, e suas limitações.

2.4.1. Definição de classes de malware na indústria de antivírus

O problema de classificação de malware é uma realidade há muitos anos. Até os anos 90, o conceito de malware era comumente utilizado para descrever todos os softwares maliciosos. No ano de 1991

a CARO (Computer Anti-virus Researchers Organization), organização formada por grupo de especialistas em antivírus, realizou um encontro, de onde surgiu a primeira convenção para nomeação de malware (CARO, 2016). Desde então diversos esquemas e convenções diferentes vêm sendo adotados pelas empresas que fazem parte da indústria de antivírus.

Uma prática comum é criar um nome para cada amostra de malware diferente que é analisada. O processo de nomeação é altamente subjetivo, e a maioria das empresas adotam estratégias diferentes para a nomeação de suas amostras, o que resulta em muitas amostras de malware iguais sendo referenciados com nomes diferentes por pesquisadores e profissionais da indústria de antivírus.

Fonte: Retirada do website (VIRUS TOTAL, 2016).

Fortinet	W32/Filecoder.EM!tr
GData	Gen:Variant.Zusy.128574
Ikarus	Trojan-Ransom.TeslaCrypt
Jiangmin	Trojan/Bitman.a
K7AntiVirus	Trojan (004bcd7c1)
K7GW	Trojan (004bcd7c1)
Kaspersky	Trojan-Ransom.Win32.Bitman.d
Malwarebytes	Trojan.CryptoLocker
McAfee	Ransom-FYG!209A288C6820
McAfee-GW-Edition	BehavesLike.Win32.Dropper.dh
eScan	Gen:Variant.Zusy.128574
Microsoft	Ransom:Win32/Tescrypt.A

Figura 2.1: Análise de uma amostra de malware

A Figura 2.1 mostra o resultado parcial da análise de uma amostra de malware, realizada no dia 13 de novembro de 2016, no website https://www.virustotal.com/. Nota-se que diversos nomes são atribuídos à mesma amostra. O software antivírus da empresa Microsoft atribui o nome Tescript.A para amostra, enquanto a Kaspersky atribui o nome Bitman.d. Outros nomes como TeslaCrypt e Zusy também são atribuídos por outras empresas.

Além disso, as empresas também adotam esquemas de classificação diferentes e independentes. Utilizam rols de classes diferentes, e não existe consenso e documentação clara e acessível que indique o que é levado em conta no processo de classificação.

No website da empresa Kaspersky (KASPERSKY, 2016) é encontrada uma definição de classes, que pode ser vista na Tabela 2.3.

Tabela 2.3: Definição de classes da empresa Kaspersky.

Classe	Definição
Virus	Programas que infectam outros programas adicionando-lhes um código de vírus ao seu processo de inicialização. Esta definição simples revela a ação principal de um vírus: infecção. A velocidade de propagação dos vírus é menor do que a dos worms.
Worm	Este tipo de Malware usa recursos de rede para propagação. Esta classe foi chamada de worm por causa de sua peculiar característica de "fluência" de computador para computador usando a rede, e-mails e outros canais informativos. Graças à essa fluência a velocidade de espalhamento de worms é muito alta. Worms invadem seu computador, calculam endereços de rede de outros computadores e enviam para esses endereços suas cópias. Além dos endereços de rede, os dados dos catálogos de endereços dos clientes de correio também são usados. Os representantes deste tipo de Malware às vezes criam arquivos em discos do sistema, mas não usam outros recursos do computador (exceto a memória operacional).
Trojan	Programas que executam em computadores infectados sem autorização por ações do usuário. Dependendo das condições podem deletar informações no disco, congelar o sistema, roubar informações, etc. Este tipo de malware não é um virus na definição tradicional, ou seja, não infecta outros programas ou dados. Os cavalos de Tróia não podem invadir o PC por si mesmos, são difundidos por violadores como software útil e necessário. E ainda, danos causados por Trojans são maiores do que os causado por um ataque de vírus tradicional.
Spyware	Software que permite coletar dados sobre um usuário ou organização específica, que não estão cientes disso. Você talvez nem saiba que está infectado por um spyware em seu computador. Como regra geral, o objetivo de um spyware é: Rastrear as ações do usuário no computador Coletar informações sobre o conteúdo do disco rígido; Muitas vezes significa digitalizar algumas pastas e registros do sistema para fazer uma lista de softwares instalados no computador. Coletar informações sobre a qualidade da conexão, modo de conexão, velocidade do modem, etc. Coletar informações não é a principal função desses programas, eles também ameaçam a segurança. No mínimo dois programas conhecidos - Gator e eZula – permitem ao violador não só coletar informações, mas também controlar o computador. Outro exemplo de spyware são programas incorporados no navegador instalado no computador para retransferir o tráfego. Você definitivamente já se deparou com esses programas, se ao acessar um endereço de um website, outro website foi aberto. Um dos spywares é o phishing-delivery
Phishing	São e-mails cujo objetivo é obter informações confidenciais do usuário, como regra geral. O Phishing é uma forma de engenharia social, caracterizada por tentativas fraudulentas de adquirir informações sensíveis, como senhas e detalhes de cartão de crédito, mascarando-se como uma pessoa ou negócio confiável em uma comunicação eletrônica aparentemente oficial, como um e-mail ou uma mensagem instantânea. As mensagens contêm um link para um website deliberadamente falso onde o usuário é instruído a inserir o número do seu cartão de crédito e outras informações confidenciais.
Adware	Código de programa incorporado a software sem que o usuário esteja ciente disso, para mostrar publicidade. Como regra um adware é incorporado a softwares que são distribuídos gratuitamente. A propaganda está na interface de trabalho. O Adware

Classe	Definição
	geralmente reúne e transfere para o seu distribuidor informações pessoais do usuário.
Riskware	Este software não é um vírus, mas representa uma ameaça em potencial. Por algumas condições a presença de tais riskware no seu PC coloca seus dados em risco. Estão incluídos nesta classe de software utilitários de administração remota, programas que usam conexão discada, e alguns outros que se conectam com websites da internet que oferecem servios de pay-per-minute.
Jokes	Software que não prejudica o computador, mas exibe mensagens de que dano já foi causado ou que será causado em algumas condições. Este software frequentemente adverte o utilizador sobre perigo não existente, por exemplom, exibe mensagens sobre formatação de disco rígido (embora nenhuma formatação esteja realmente acontecendo), detectar vírus em arquivos não infectados e etc.
Rootkit	São utilitários usados para ocultar atividades maliciosas. Eles disfarçam o Malware, para evitar que sejam detectados pelos aplicativos antivírus. Os rootkits também podem modificar o sistema operacional no computador e substituir suas principais funções para disfarçar sua presença e ações que o violador faz no computador infectado.
Spam	Mensagems de e-mail anônimas e em massa, de caráter indesejável. O spam pode ser propaganda política, comercial, ou e-mails que pedem para ajuda para alguém. Outra categoria de spam são mensagens que sugerem que você invista uma grande soma de dinheiro, ou convidando você para pirâmides financeiras, e-mails que roubam senhas e número de cartão de crédito, mensagens sugerindo para enviá-los para seus amigos (mensagens de felicidade), etc. Spam sobrecarrega servidores de correio e aumenta o risco de perder informações importantes para o usuário.
Outros	Outros programas diferentes que foram desenvolvidos para criar outros Malware, organizando ataques DoS em servidores remotos, invadindo computadores, etc. Hack Tools, construtores de vírus e outros referem-se a tais programas.

Fonte: Adaptado de (KASPERSKY, 2016).

Já no website da empresa F-Secure (F-SECURE, 2016) encontramos uma definição de classes que divide as classes de malware em 3 categorias: Malware, Spyware e Riskware.

De acordo com as informações fornecidas em seu website, programas da categoria Malware representam um risco de segurança significativo para o sistema do usuário e/ou suas informações. Incluem vírus, worms e trojans, entre outras ameaças. Essas ameaças podem executar ações prejudiciais, como roubar dados pessoais ou de programas, manipular secretamente o dispositivo ou programas instalados, ou impedir completamente o usuário de usar o dispositivo. A Tabela 2.4 traz a definição da categoria Malware apresentada pela empresa F-Secure.

Tabela 2.4: Definição de tipos de malware da empresa F-Secure.

Classe	Definição
Virus	Integra seu próprio código em programas ou arquivos de dados e se espalha integrando-se em mais arquivos cada vez que um arquivo afetado é executado.
Worm	Usa recursos de computador ou de rede para fazer cópias completas de si mesmo e distribuí-las para outras vítimas. Pode incluir código ou outro malware para danificar tanto

	o sistema quanto a rede. Worms também podem ser desenvolvidos mais especificamente com base no tipo de rede que eles usam para se espalhar: • Net-Worm: através de uma rede local ou da Internet • Email-Worm: via e-mails, contidos no próprio e-mail ou como anexos de arquivo • P2P-Worm: em arquivos enviados por redes peer-to-peer (redes P2P) • IM-Worm: sobre redes de mensagens instantâneas • (IM)IRC-Worm: através de canais de conversa por internet (IRC) • Bluetooth-Worm: difusão via Bluetooth
Rootkit	Esconde a si mesmo, ou outros arquivos, dos programas de segurança do dispositivo. Pode ser usado por usuários remotos para manipular o dispositivo.
Backdoor	Permite que usuários remotos manipulem um programa, computador ou rede.
Trojan	Permite que usuários remotos manipulem um programa, computador ou rede. Cavalos de Tróia utilizam desorientação, desinformação, omissão ou fraude para enganar o usuário na instalação ou execução, para que ele possa executar ações potencialmente indesejadas / prejudiciais. Não se reproduz. Trojans podem ser desenvolvidos mais especificamente com base no tipo de ações que desejam executar: • Trojan-Spy: instala programas de espionagem como keyloggers • Trojan-PWS: rouba senhas e outras informações confidenciais • Trojan-Downloader: baixa programas de um servidor remoto, instala e executa • Trojan-Dropper: carrega pelo menos um programa, que instala e executa • Trojan-Proxy: permite que usuários remotos conectem o sistema infectado a um servidor proxy anonimamente • Trojan-Dialer: conecta-se à Internet através de linhas telefônicas premium. Também pode levar a websites não solicitados ou inapropriados.
Rogue	Utiliza mensagens de alta gravidade, mensagens enganosas ou fraudes diretas para pressionar os usuários a comprarem software antivírus que podem não funcionar conforme informado.
Exploit	Aproveita-se de uma vulnerabilidade em um programa ou sistema operacional para obter acesso ou executar ações além do que é normalmente permitido.
Classe	**Definição**
Packed	Compactado para um tamanho menor usando um programa packer, conhecido por ser usado por outros malwares.
Construtor	Um programa utilitário usado para construir malware.

Fonte: Adaptado de (F-SECURE, 2016)

Já a categoria Spyware é definida por programas que introduzem um risco de segurança que pode afetar os dados pessoais do usuário. Programas na categoria Spyware incluem trackware e adware. Esses programas podem oferecer um serviço útil em troca de ter permissão para coletar informações sobre o usuário.

O tipo de informação recolhida por estes programas varia, e pode incluir itens tais como detalhes do sistema ou programas instalados; Comportamento de navegação na web e histórico; E-mails importantes, e detalhes pessoais. As implicações legais também podem surgir com base em onde e

como o programa é usado, e como a informação é coletada, transmitida e armazenada. A Tabela 2.5 traz a definição da categoria Spyware apresentada pela empresa F-Secure.

Tabela 2.5: Definição de tipos de spyware da empresa F-Secure.

Classe	Definição
Spyware	Coleta informações sobre o comportamento de navegação na web do usuário ou aplicativos preferenciais. Os dados coletados podem ser armazenados localmente ou enviados.
Trackware	Permite que um terceiro identifique o usuário ou seu dispositivo, geralmente com um identificador exclusivo. O trackware mais comum é o rastreamento de cookies.
Adware	Fornece conteúdo publicitário, no navegador da Web, no Desktop de um PC ou em um aplicativo.

Fonte: Adaptado de (F-SECURE, 2016)

Programas da categoria Riskware são considerados pela empresa como seguros quando usados por uma pessoa autorizada em uma situação apropriada. Se for mal utilizado, ou usado por um invasor, o programa pode representar um risco de segurança

Por exemplo, os keyloggers são utilitários que podem ser usados pelos administradores do sistema no decurso do seu trabalho autorizado, mas também podem ser maliciosamente usados para monitorar secretamente os usuários. A Tabela 2.6 traz a definição da categoria Riskware apresentada pela empresa F-Secure.

Tabela 2.6: Definição de tipos de riskware da empresa F-Secure.

Classe	Definição
Monitoring-Tool	Monitora e registra ações de um usuário em um dispositivo
Hack-Tool	Ignora as restrições de acesso ou mecanismos de segurança para dar acesso aos usuários ou a capacidade de executar ações além do que é normalmente permitido
Application	Introduz um risco de segurança se usado de forma errada ou maliciosa

Fonte: Adaptado de (F-SECURE, 2016)

Analisando as duas definições de classes apresentadas pelas empresas, pode-se observar diversas inconsistências. Primeiramente, comparando-as entre si, notam-se várias diferenças. A definição utilizada pela empresa Kaspersky propõe um rol com as classes Virus, Worms, Trojans, Spyware, Phishing, Adware, Riskware, Jokes, Rootkit e Spam. Já a definição utilizada pela empresa F-Secure apresenta três categorias, Malware, Spyware e Riskware, com as subclasses Virus, Worm, Rootkit, Backdoor, Trojan, Rogue, Exploit, Packed, Constructor, Spyware, Trackware Adware, Monitoring-Tool, Hack-Tool, e Application.

Percebe-se que além das duas definições possuírem classes que estão presentes em apenas uma delas (Spam, Trackware, Monitoring-Tool, Hack-tool, etc.), a metodologia adotada para a divisão de classes é diferente, sendo que a metodologia adotada pela empresa Kaspersky define apenas um grupo de classes, e a adotada pela empresa F-Secure divide as classes de malware em 3 grupos de subclasses.

Além das inconsistências encontradas comparando as definições entre si, as duas definições apresentam inconsistências entre suas próprias classes. Os fatores utilizados para definir as classes variam, o que pode gerar confusão no momento da classificação ou permitir que uma amostra de malware possa ser classificada em diversas classes.

Por exemplo, na definição de classes da empresa Kaspersky, é classificado como Worm um malware que usa a rede para se propagar (classificado utilizando modo de propagação), como Virus um malware que infecta um software legítimo com fragmento de código malicioso (característica estrutural), como Spyware um malware que visa roubo de informações (objetivo do malware) e como Rootkit um malware que esconde sua atividade do sistema operacional (funcionalidade implementada).

Utilizando essa definição de classes uma amostra apenas de malware pode ser facilmente classificada como um Virus Worm Spyware Rootkit, por exemplo, o que traz confusão no processo de classificação.

Inconsistências também são encontradas na definição de classes da empresa F-Secure, que, por exemplo, define as subclasses Trojan-Spy, Trojan-PWS, e Spyware. Apesar de terem sido separadas em 3 subclasses diferentes, sua característica principal para classificação é a mesma, objetivo do malware de roubar informações.

Ainda no website da empresa Kaspersky, encontramos a Figura 2.2, que traz uma nova definição de classes de malware, diferente da definição encontrada no website da empresa F-Secure, e também diferente da definição de classes encontrada no website da própria empresa Kaspersky (KASPERSKY, 2016). Essa figura traz novas classes, que estão organizadas de acordo com o nível de ameaça. As classes que se encontram na parte superior são consideradas ameaças mais graves.

Fonte: Retirado de (KASPERSKY, 2016)

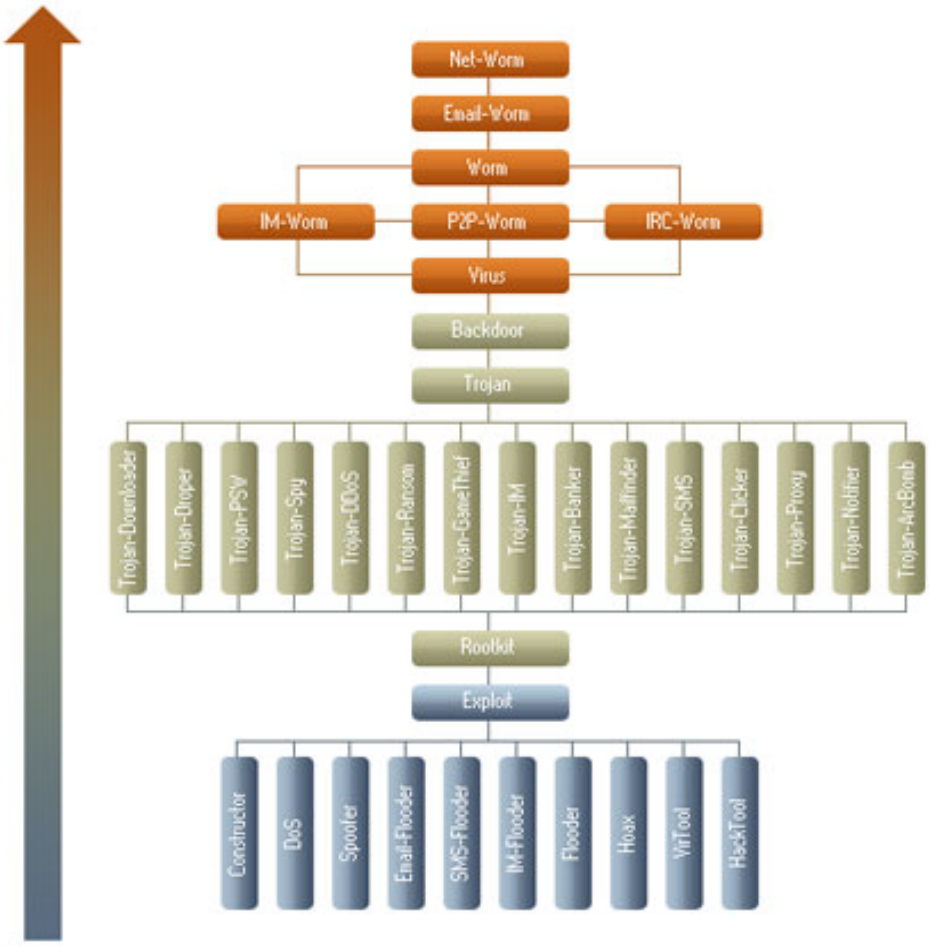


Figura 2.2: Árvore de classificação de malware

Nota-se então que a classificação de malware na indústria de antivírus ainda não possui uma base bem estabelecida. O objetivo de uma empresa produtora de antivírus é gerar um produto melhor que o produto feito pelas empresas concorrentes, capaz de detectar a presença de malware e previnir que o sistema de seus clientes seja infectado.

Apesar de utilizar a classificação de malware nos seus softwares antivírus (RIECK et al., 2008) é natural que as empresas mantenham seu foco na detecção e prevenção, além de manter a tecnologia desenvolvida em segredo.

2.4.2. Classes de malware na literatura

Também existem divergências entre os pesquisadores quanto a definição de classes de malware. Em (AYCOCK, 2006) é apresentada uma divisão de malwares entre classes, baseada no método de operação do malware. O autor utiliza três características principais associadas às amostras de malware para compor sua classificação:

1. Auto-replicante: Malware que tenta se propagar criando novas cópias, ou instâncias, de si mesmo.
2. Crescimento populacional: Característica que descreve a mudança no número de instâncias de malware feita através da auto-replicação.
3. Parasita: Malware que necessita infectar outro código executável para exsitir.

Utilizando essas três características, o autor propõe uma definição com dez classes de malware. Segundo a definição, os malwares devem ser divididos entre Logic Bomb, Trojan, Virus, Backdoor, Worm, Rabbit, Spyware, Adware, e Outros. Aqui, observa-se uma inconsistência na definição de classes adotada pelo autor, e na estratégia exposta para classificação. Classes como Spyware e Adware possuem exatamente os mesmos comportamentos, analisando as três características elencadas pelo autor, porém são consideradas classes diferentes, utilizando como fator de classificação o objetivo do malware. Esse fator de classificação não faz parte das características elencadas, e não é levado em conta em todas as classes da definição.

Em seu trabalho, (FILIOL, 2010) traz uma definição de classes consistente, criada e aprimorada na década de 80 por (COHEN, 1987) e (ADLEMAN, 1998). Essa definição divide malwares entre quatro classes, Logic Bomb, Trojan, Virus e Worm, e sua estrategia de classificação obedece o esquema que pode ser visto na Figura 2.3

Fonte: Adaptada da definição apresentada em (FILIOL, 2010)

Figura 2.3: Ilustração de definição de classes presente na literatura.

Apesar de não apresentar inconsistências, a definição apresentada é antiga e abrangente, e perdeu muito de sua utilidade à medida que novos malwares com funcionalidades diferentes foram criados. É impossível diferenciar nessa definição de classes, por exemplo, um virus que é criado com o objetivo de roubar informações de um virus que é criado com o objetivo de encriptar os arquivos do

sistema alvo e exigir uma recompensa, duas funcionalidades implementadas por malwares que fazem parte da realidade das ameaças presentes nos dias de hoje.

Em (SIKORSKI; HONIG, 2012) é apresentada uma outra definição de classes, com semelhanças à definição encontrada em (AYCOCK, 2006). Essa definição é composta por nove classes, sendo elas Backdoor, Botnet, Downloader, Spyware, Launcher, Rootkit, Scareware, SPAM e Worm/Virus.

No website do Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil (CERT.br) (CERT.BR, 2016), uma organização mantida pelo governo brasileiro, e que tem como objetivo publicar diretrizes para o tratamento de incidentes de segurança da informação no país, é encontrada a definição de classes de malware que pode ser vista na Tabela 2.7.

Tabela 2.7: Definição de classes encontrada em (CERT.BR, 2016).

Classe	Definição
Virus	Vírus é um programa ou parte de um programa de computador, normalmente malicioso, que se propaga inserindo cópias de si mesmo e se tornando parte de outros programas e arquivos. Para que possa se tornar ativo e dar continuidade ao processo de infecção, o vírus depende da execução do programa ou arquivo hospedeiro, ou seja, para que o seu computador seja infectado é preciso que um programa já infectado seja executado. O principal meio de propagação de vírus costumava ser os disquetes. Com o tempo, porém, estas mídias caíram em desuso e começaram a surgir novas maneiras, como o envio de e-mail. Atualmente, as mídias removíveis tornaram-se novamente o principal meio de propagação, não mais por disquetes, mas, principalmente, pelo uso de pen-drives. Há diferentes tipos de vírus. Alguns procuram permanecer ocultos, infectando arquivos do disco e executando uma série de atividades sem o conhecimento do usuário. Há outros que permanecem inativos durante certos períodos, entrando em atividade apenas em datas específicas. Alguns dos tipos de vírus mais comuns são: • Vírus propagado por e-mail: recebido como um arquivo anexo a um e-mail cujo conteúdo tenta induzir o usuário a clicar sobre este arquivo, fazendo com que seja executado. Quando entra em ação, infecta arquivos e programas e envia cópias de si mesmo para os e-mails encontrados nas listas de contatos gravadas no computador. • Vírus de script: escrito em linguagem de script, como VBScript e JavaScript, e recebido ao acessar uma página Web ou por e-mail, como um arquivo anexo ou como parte do próprio e-mail escrito em formato HTML. Pode ser automaticamente executado, dependendo da configuração do navegador Web e do programa leitor de e-mails do usuário. • Vírus de macro: tipo específico de vírus de script, escrito em linguagem de macro, que tenta infectar arquivos manipulados por aplicativos que utilizam esta linguagem como, por exemplo, os que compõe o Microsoft Office (Excel, Word e PowerPoint, entre outros). • Vírus de telefone celular: vírus que se propaga de celular para celular por meio da tecnologia bluetooth ou de mensagens MMS (Multimedia Message Service). A infecção ocorre quando um usuário permite o recebimento de um arquivo infectado e o executa. Após infectar o celular, o vírus pode destruir ou sobrescrever arquivos, remover ou transmitir contatos da agenda, efetuar ligações telefônicas e drenar a carga da bateria, além de tentar se propagar para outros celulares.
Worm	Worm é um programa capaz de se propagar automaticamente pelas redes, enviando cópias de

Classe	Definição
	si mesmo de computador para computador. Diferente do vírus, o worm não se propaga por meio da inclusão de cópias de si mesmo em outros programas ou arquivos, mas sim pela execução direta de suas cópias ou pela exploração automática de vulnerabilidades existentes em programas instalados em computadores. Worms são notadamente responsáveis por consumir muitos recursos, devido à grande quantidade de cópias de si mesmo que costumam propagar e, como consequência, podem afetar o desempenho de redes e a utilização de computadores.
Bot	Bot é um programa que dispõe de mecanismos de comunicação com o invasor que permitem que ele seja controlado remotamente. Possui processo de infecção e propagação similar ao do worm, ou seja, é capaz de se propagar automaticamente, explorando vulnerabilidades existentes em programas instalados em computadores. A comunicação entre o invasor e o computador infectado pelo bot pode ocorrer via canais de IRC, servidores Web e redes do tipo P2P, entre outros meios. Ao se comunicar, o invasor pode enviar instruções para que ações maliciosas sejam executadas, como desferir ataques, furtar dados do computador infectado e enviar spam. Um computador infectado por um bot costuma ser chamado de zumbi (zombie computer), pois pode ser controlado remotamente, sem o conhecimento do seu dono. Também pode ser chamado de spam zombie quando o bot instalado o transforma em um servidor de e-mails e o utiliza para o envio de spam. Botnet é uma rede formada por centenas ou milhares de computadores zumbis e que permite potencializar as ações danosas executadas pelos bots. Quanto mais zumbis participarem da botnet mais potente ela será. O atacante que a controlar, além de usá-la para seus próprios ataques, também pode alugá-la para outras pessoas ou grupos que desejem que uma ação maliciosa específica seja executada. Algumas das ações maliciosas que costumam ser executadas por intermédio de botnets são: ataques de negação de serviço, propagação de códigos maliciosos (inclusive do próprio bot), coleta de informações de um grande número de computadores, envio de spam e camuflagem da identidade do atacante (com o uso de proxies instalados nos zumbis).
Trojan	Cavalo de troia, trojan ou trojan-horse, é um programa que, além de executar as funções para as quais foi aparentemente projetado, também executa outras funções, normalmente maliciosas, e sem o conhecimento do usuário. Exemplos de trojans são programas que você recebe ou obtém de websites na Internet e que parecem ser apenas cartões virtuais animados, álbuns de fotos, jogos e protetores de tela, entre outros. Estes programas, geralmente, consistem de um único arquivo e necessitam ser explicitamente executados para que sejam instalados no computador. Trojans também podem ser instalados por atacantes que, após invadirem um computador, alteram programas já existentes para que, além de continuarem a desempenhar as funções originais, também executem ações maliciosas. Há diferentes tipos de trojans, classificados2 de acordo com as ações maliciosas que costumam executar ao infectar um computador. Alguns destes tipos são: Trojan Downloader: instala outros códigos maliciosos, obtidos de websites na Internet. Trojan Dropper: instala outros códigos maliciosos, embutidos no próprio código do trojan. Trojan Backdoor: inclui backdoors, possibilitando o acesso remoto do atacante ao computador. Trojan DoS: instala ferramentas de negação de serviço e as utiliza para desferir ataques. Trojan Destrutivo: altera/apaga arquivos e diretórios, formata o disco rígido e pode deixar o computador fora de operação. Trojan Clicker: redireciona a navegação do usuário para websites específicos, com o objetivo de aumentar a quantidade de acessos a estes websites ou apresentar propagandas. Trojan Proxy: instala um servidor de proxy, possibilitando que o computador seja utilizado

Classe	Definição
	para navegação anônima e para envio de spam. Trojan Spy: instala programas spyware e os utiliza para coletar informações sensíveis, como senhas e números de cartão de crédito, e enviá-las ao atacante. Trojan Banker ou Bancos: coleta dados bancários do usuário, através da instalação de programas spyware que são ativados quando websites de Internet Banking são acessados. É similar ao Trojan Spy porém com objetivos mais específicos.
Spyware	Spyware é um programa projetado para monitorar as atividades de um sistema e enviar as informações coletadas para terceiros. Pode ser usado tanto de forma legítima quanto maliciosa, dependendo de como é instalado, das ações realizadas, do tipo de informação monitorada e do uso que é feito por quem recebe as informações coletadas. Pode ser considerado de uso: Legítimo: quando instalado em um computador pessoal, pelo próprio dono ou com consentimento deste, com o objetivo de verificar se outras pessoas o estão utilizando de modo abusivo ou não autorizado. Malicioso: quando executa ações que podem comprometer a privacidade do usuário e a segurança do computador, como monitorar e capturar informações referentes à navegação do usuário ou inseridas em outros programas (por exemplo, conta de usuário e senha). Alguns tipos específicos de programas spyware são: Keylogger: capaz de capturar e armazenar as teclas digitadas pelo usuário no teclado do computador. Sua ativação, em muitos casos, é condicionada a uma ação prévia do usuário, como o acesso a um website específico de comércio eletrônico ou de Internet Banking. Screenlogger: similar ao keylogger, capaz de armazenar a posição do cursor e a tela apresentada no monitor, nos momentos em que o mouse é clicado, ou a região que circunda a posição onde o mouse é clicado. É bastante utilizado por atacantes para capturar as teclas digitadas pelos usuários em teclados virtuais, disponíveis principalmente em websites de Internet Banking. Adware: projetado especificamente para apresentar propagandas. Pode ser usado para fins legítimos, quando incorporado a programas e serviços, como forma de patrocínio ou retorno financeiro para quem desenvolve programas livres ou presta serviços gratuitos. Também pode ser usado para fins maliciosos, quando as propagandas apresentadas são direcionadas, de acordo com a navegação do usuário e sem que este saiba que tal monitoramento está sendo feito.
Backdoor	Backdoor é um programa que permite o retorno de um invasor a um computador comprometido, por meio da inclusão de serviços criados ou modificados para este fim. Pode ser incluído pela ação de outros códigos maliciosos, que tenham previamente infectado o computador, ou por atacantes, que exploram vulnerabilidades existentes nos programas instalados no computador para invadi-lo.
Rootkit	Rootkit é um conjunto de programas e técnicas que permite esconder e assegurar a presença de um invasor ou de outro código malicioso em um computador comprometido.

Fonte: Retirado de (CERT.BR, 2016)

A divergência entre as definições de classes aparecem em praticamente todos os trabalhos encontrados na literatura.

Outras definições de classes diferentes são encontradas em (VERACODE, 2016), (PANDA SECURITY, 2016), (DUMMIES, 2016), (EASTLAKE 3RD; JONES, 2001)(ESET, 2016).

2.4.3. Resumo das definições de classes encontradas

No geral, pode-se sintetizar uma definição composta pelas classes mais recorrentes na maioria dos trabalhos estudados e nas definições apresentadas pelas empresas de antivírus. Essa definição é apresentada na Tabela 2.8.

Tabela 2.8: Definição de classes sintetizada a partir de revisão bibliográfica.

Classe	Definição
Vírus	Malware que se auto-replica, e necessita de um software hospedeiro.
Worm	Malware que se auto-replica, e não necessita de software hospedeiro.
Botnet	Malware que fornece o controle do computador infectado ao criminoso, e se replica automaticamente.
Backdoor	Malware que fornece o controle do computador infectado ao criminoso, e não se replica automaticamente.
Spyware	Malware cujo objetivo é o roubo de informações.
Trojan	Malware que se disfarça de um programa benigno.
Rootkit	Malware que disfarça a execução de código, não permitindo a detecção dessa execução por meios normais de diagnóstico.

Analisando essa definição de classes, nota-se que ao menos quatro fatores são levados em consideração no processo de classificação: Características estruturais, Objetivo, Funcionalidades implementadas, Modo de infecção. O maior problema decorrente das definições de classes que seguem essa abordagem é que os fatores de classificação não são levados em conta para todas as classes. Essa definição leva à um processo de classificação confuso e ambíguo.

Além disso, como acontece na indústria de antivírus, a divergência entre os diferentes esquemas de classificação pode causar confusão, e dificulta o desenvolvimento de uma metodologia de classificação automatizada que seja útil para pesquisadores e investigadores forenses.

2.4.4. Famílias de malware

A grande maioria dos trabalhos publicados na área de classificação de malware hoje visa classificar amostras de malware entre famílias. O conceito de família de malware difere do conceito de classe de malware. Originalmente definido em (CARO, 2016), uma família de malware agrupa amostras de malwares que implementam as mesmas funcionalidades em seu código fonte, ou possuem semelhanças estruturais. Essas amostras são chamadas variantes do malware.

Uma variante de malware pode ser criada inserindo-se instruções aleatórias em partes arbitrárias do código binário de uma amostra de malware. Como exemplo, o trecho de código binário visto na Figura 2.4:

```
0000000000400450 <.init>:
 400450:    sub    $0x8,%rsp
 400454:    mov    0x200b9d(%rip),%rax
 400460:    movl   %rax,%rbx
 40045e:    je     400465 <iopl@plt-0x1b>
 400460:    callq  4004b0 <__gmon_start__@plt>
```

Figura 2.4: Código assembly antes da transformação

Pode ser transformado no código binário visto na Figura 2.5:

```
0000000000400450 <.init>:
 400450:    sub    $0x8,%rsp
 400454:    mov    0x200b9d(%rip),%rax
 40045b:    add    *random_value*,%rax   #adiciona valor aleatório ao registrador %rax
 40045e:    sub    *same_value*,%rax     #subtrai o mesmo valor do registrador %rax
 400460:    movl   %rax,%rbx
 400465:    je     400465 <iopl@plt-0x1b>
 400469:    callq  4004b0 <__gmon_start__@plt>
```

Figura 2.5: Código assembly após a transformação

Essa transformação altera a assinatura do arquivo binário, porém mantendo a mesma funcionalidade. O segundo binário seria então uma variante do primeiro, e os dois pertenceriam à mesma família. Os fragmentos de código vistos nas figuras 2.4 e 2.5 são simples, e não tem nenhuma funcionalidade, sendo criados apenas para mostrar o exemplo de inserção de código. Técnicas baseadas nesse princípio são utilizadas para se realizar o polimorfismo, que consiste em alterar uma amostra de malware a ponto de dificultar, ou até mesmo impedir, a análise do código binário, principalmente nos tipos de análise feitas nos antivírus comerciais.

Outras técnicas que são utilizadas para realizar polimorfismo de código malicioso incluem multiplicação de instruções, técnicas de compressão e criptografia (YOU; YIM, 2010).

Variantes de um malware podem surgir também com uma simples alteração do código do malware pelo seu autor, de forma natural, seja para implementar novas funcionalidades, corrigir erros de implementação, entre outros. A Figura 2.6 ilustra o processo de surgimento de variantes de malware.

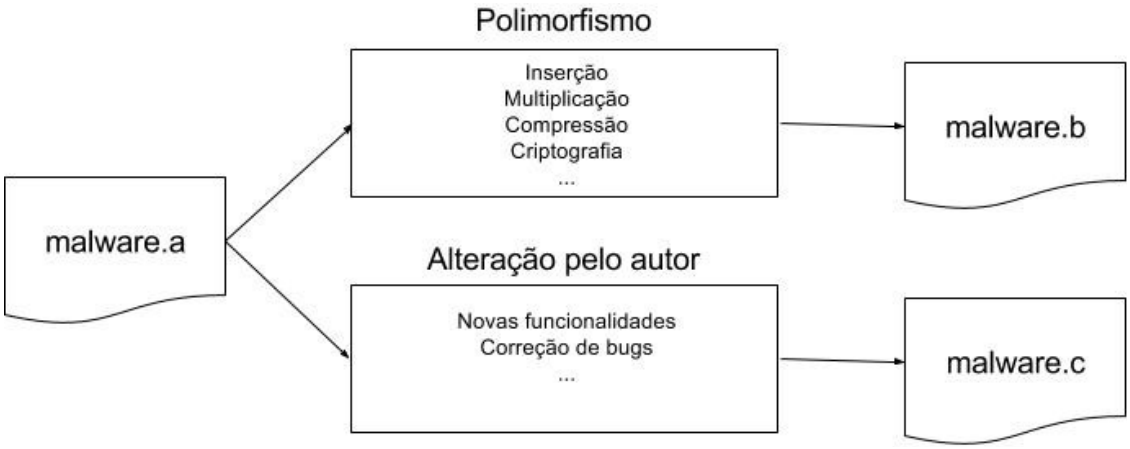


Figura 2.6: Surgimento de variantes de malware.

Nessa figura vemos que uma variante de malware, **malware.a**, pode se transformar em outra variante de malware de duas formas diferentes, por polimorfismo (**malware.b**) ou por uma alteração feita pelo autor (**malware.c**).

2.4.5. Detecção de variantes de malware utilizando controle de fluxo

Em (CESARE; XIANG; ZHOU, 2013) é proposto um sistema que faz classificação de variantes de malware em famílias, utilizando conjuntos de grafos de controle de fluxo. A partir da interpretação do controle de fluxo extraído do arquivo binário de uma amostra de malware, é gerado um grafo que funciona como uma assinatura do malware. Técnicas de medição são empregadas, e um limiar é definido. A amostra questionada é então comparada com as variantes presentes na base de dados, e caso o limiar definido seja atingido, a amostra é classificada como membro da família.

A metodologia proposta obteve sucesso na classificação de variantes de malware de uma mesma família, porém, por se tratar de uma técnica de análise estática, é dependente de um processo de unpacking automático, o que nem sempre é possível. Não existe programa ou metodologia de unpacking genérica que funcione para todas as amostras de malware existentes.

2.4.6. Análise de similaridades e classificação de variantes de malware através de chamadas de apis

Um dos aspectos explorados para a classificação e análise de similaridade de malwares foi a sequência de chamadas de funções do sistema operacional (chamadas de APIs) feitas pelo código do malware. O objetivo de uma API é definir um conjunto de funções que podem ser utilizadas para controlar o sistema operacional. Um malware, assim como os programas benignos, necessita de chamadas de API para realizar suas ações maliciosas.

(CHO et al., 2014) utiliza a sequência de chamadas de APIs feita durante a execução de uma amostra malware para verificar sua similaridade com uma certa amostra de malware já conhecida. A metodologia proposta captura a sequência de chamadas de API feitas pela amostra, e a compara com a sequência de chamadas feitas pela amostra de malware conhecida. Dessa comparação é gerado um número que representa o nível de similaridade entre as amostras. Apesar de apresentar experimentos limitados, trabalhando apenas com 8 famílias de malware, os autores mostram que obtiveram sucesso em sua análise de similaridade. O nível de similaridade obtido comparando variantes de malware da mesma família é significativamente maior do que o obtido comparando variantes de malware de famílias diferentes.

Chamadas de API também já foram utilizadas para a classificação de malware entre classes. Em (JANG et al., 2014) é apresentada uma ferramenta que faz a construção de grafos a partir das chamadas de API feitas durante a execução de um determinado malware, e utiliza ideias baseadas na análise de redes sociais para classifica-lo entre as classes Adware, Trojan ou Worm. Os autores utilizam a ferramenta UCINET 6 (BORGATTI; EVERETT; FREEMAN, 2002) para realizar a análise dos grafos gerados. O sistema obtém uma taxa de sucesso de classificação maior que 96%. Apesar de apresentar resultados satisfatórios ao que se propõe, a ferramenta faz utiliza uma definição de classes muito abrangente. Acreditamos que uma definição de classes mais restritiva, baseada no tipo de ação pretendida pelo criador do malware, possa ser mais útil para o investigador forense.

Outra limitação da estratégia utilizada nesses trabalhos é que tratam-se de análises exclusivamente dinâmicas, podendo perder comportamentos importantes não adotados pela amostra de malware durante a execução. Comportamento essencial para a materialização da ação criminosa e pistas da autoria podem ser deixados de lado, e amostras de malware mais complexas podem ser classificadas de forma errônea. Apesar das limitações, as metodologias fornecem fortes indícios que chamadas de API realizadas por uma amostra de malware podem fornecer pistas importantes para o processo classificação.

2.4.7. Análise e classificação de malware utilizando Fuzzy Hashes

Fuzzy hashes já foram utilizados em algoritmos de análise de malware e classificação entre famílias. Um fuzzy hash é um hash que leva em conta o contexto do arquivo. Ao contrário de outros algoritmos de hash tradicionais, um fuzzy hash não resulta no tipo de saída que é declarativo sobre o arquivo, serve mais como um indicador sobre seu conteúdo, onde são necessários estudos ou dados adicionais para se obter um resultado declarativo.

É um conceito que envolve a capacidade de comparar dois itens diferentes, e determinar um nível fundamental de similaridade (expressa em porcentagem) entre os dois. Os fuzzy hashes podem ajudar a identificar dois arquivos de imagem similares, sendo assim usados para inspeção em possíveis violações de direitos autorais.

Em uma comparação entre dois fuzzy hashes, quanto maior a porcentagem reportada, mais semelhantes são os dois códigos. Assim, um resultado de 100% indica que todo o código em um arquivo também é encontrado no outro, enquanto uma semelhança de 50% implica que apenas metade do código encontrado em ambas as amostras é considerado semelhante.

Há uma grande variedade de possíveis aplicações para hashes fuzzy, e uma delas é análise e classificação de malware. Em (DIGITALNINJA, 2007) é realizado um estudo para analisar a similaridade entre variantes de malware da mesma família, de famílias diferentes, e arquivos binários que foram ofuscados com diferentes packers, utilizando fuzzy hashes.

Os resultados encontrados pelos autores mostram que as técnicas de fuzzy hashing são mais eficientes do que algoritmos convencionais de hashing, para determinar a similaridade entre variantes de malware de uma mesma família. Porém, o uso de packers diminui a eficiência do algoritmo, tornando a técnica falha nesses casos.

No trabalho (HUANG et al., 2011) fuzzy hashs são aplicados em conjunto com métodos de inteligência computacional, com o objetivo de criar um sistema de deteção de malware baseado em características comportamentais. Os autores utilizam hashs fuzzy, dados de rede, mudanças em arquivos, e outros, para decidir se um determinado arquivo analisado é ou não malware. Os resultados obtidos mostram que a metodologia é viável para a detecção de malware e representa mais uma ferramenta para proteger usuários.

2.4.8. Classificação de malware utilizando aprendizado de máquinas

Em (SMUTZ; STAVROU, 2012), características estruturais são utilizadas para realizar a detecção de PDFs infectados com malware. Esse trabalho apresenta um processo de detecção que utiliza técnicas de aprendizado de máquinas, e que se mostrou adequado para a detecção de PDFs infectados, e promissor, pois funcionou também com malwares novos, diferentes dos utilizados na etapa de treinamento do sistema.

O trabalho demonstra que a técnica de classificação de florestas aleatórias (Random Forests) se mostrou a mais eficiente para classificar os PDFs como maliciosos ou benignos. Com uma

população de mais de 100 mil documentos benignos e 5 mil documentos infectados, a taxa de sucesso ficou acima de 99%.

O estudo reforça a ideia de que características estruturais dos arquivos podem ser úteis na tarefa de separar amostras de malware em classes. Sugere uma gama de características a serem utilziados no processo de classificação, e dá pistas da importância de cada uma na tarefa de calssificar os PDFs.

Aprendizado de máquinas também foi utilizado em (RIECK et al., 2008), com o intuito de realizar a classificação de variantes de malware entre famílias. Nesse trabalho, os autores utilizam características comportamentais compartilhadas entre variantes da mesma família de malware, para compor um classificador capaz de determinar à qual família a variante pertence. O método é dividido em três etapas: monitoração do comportamento da variante, construção e treinamento de um classificador e uso desse classificador para as decisões de classificação.

Apesar de muitos trabalhos obterem boas taxas de sucesso, a classificação de variantes de malware entre famílias tem utilidade limitada para o investigador forense. Primeiramente, para que seja útil, é necessário que o classificador tenha de antemão conhecimento acerca da família a qual a variante pertence, o que nem sempre acontece.

Além disso, a classificação entre famílias é muito restritiva. Diversas famílias apresentam objetivos e características semelhantes, e suas variantes poderiam ser agrupadas em classes.

Outra limitação dessa abordagem é que, apesar de funcionar em novas variantes da mesma família, ela não trata o surgimento de novas famílias de malware. Nota-se que a classificação de malwares em famílias tem maior sucesso numa análise de similaridade entre variantes do que num processo amplo de classificação.

3. PROFILING

Profiling consiste em inferir traços de indivíduos através de características específicas, com o objetivo de determinar se o indivíduo se encaixa em um determinado grupo. Neste capítulo apresenta-se o conceito de profiling criminal, alguns usos fora da esfera criminal, e como ele pode ser utilizado para classificação de malware. São definidos os perfis e é realizada uma discussão a respeito das características estruturais e comportamentais de malware, para a seleção das características consideradas mais relevantes para o processo de definição de perfis de malware.

3.1. PROFILING CRIMINAL

Profiling na área criminal refere-se a qualquer meio de inferir os traços de indivíduos responsáveis por cometer atos criminosos. O grupo de profissionais envolvidos na prática de profiling criminal inclui historicamente psicologos, cientistas sociais, cientistas forenses e investigadores criminais.

Geralmente, métodos de profiling criminal incluem avaliações diagnósticas dos suspeitos, análise comportamentais dos vestígios na cena do crime, análise investigativa criminal e psicologia investigativa.

Esses métodos têm sido utilizados para ajudar a identificar criminosos, filtrar listas de suspeitos, auxiliar na conexão de casos, desenvolver pistas e até mesmo conceber estratégias de investigação (TURVEY, 2011). No tribunal, especialistas em profiling criminal comumente prestam testemunho sobre temas relacionados à sua análise, tais como motivação do crime, modus operandi, e conexão de casos.

Num sentido amplo, o profiling criminal contribui para a identificação de suspeitos, direta ou indiretamente. A criação de um perfil criminal para uma determinada infração pode restringir o rol de suspeitos e facilitar a prisão dos infratores. Uma análise das características demográficas e psicológicas de um suspeito também podem ser aplicadas no processo de interrogação (HOLMES; HOLMES, 2008). Ele pode sugerir se a interrogação deve seguir de forma coercitiva ou persuasiva.

Características comumente utilizadas para compor perfis criminais incluem sexo, idade, religião, estado civil, nível de escolaridade, assim como uma análise relacional e comportamental do suspeito. Restringir o rol de suspeitos significa tornar mais eficiente o trabalho de investigação, resultando assim numa economia de orçamento. O perfil criminal, além disso, pode ser construído com base nas respostas que os suspeitos dão durante o processo de interrogação pelos investigadores criminais.

3.1.1. Uso do profiling criminal em outras áreas

Além de ter um papel importante na persecução penal, ideias baseadas em profiling criminal já foram utilizadas com sucesso em diversas áreas. Em (LEE, 2015) é apresentada uma proposta baseada em profiling criminal para prever e prevenir o vazamento de segredos industriais. Os autores apresentam uma abordagem composta de duas etapas.

Na etapa de análise comportamental, apenas informações críticas são levadas em consideração, e a experiência e intuição do analista são essenciais para o processo de profiling.

Já a etapa de investigação de dados passa por um processo de mineração de uma grande quantidade de dados, utiliza técnicas de aprendizado de máquinas, e é útil principalmente para prever novos relacionamentos do sujeito, a partir de dados não relacionados anteriormente.

O estudo apresentado em (ELIASON, 2013) relaciona o profiling criminal com o processo de profiling realizado por guardas florestais para prever o comportamento de caçadores ilegais nas áreas de preservação americanas.

O estudo revela que os guardas utilizam características como modo de operação, características comportamentais, área de atuação e experiência de caça. Mostra também que as características são utilizadas para prever o nível de dificuldade para a apreensão dos caçadores ilegais.

3.2. DEFINIÇÃO DE PERFIS DE MALWARE

O problema de classificação de malware consiste em, dada uma amostra de malware, determinar em qual classe ela se encaixa, dentre um rol de classes definidas de antemão. Deseja-se criar uma metodologia de classificação útil, de forma a auxiliar o trabalho do investigador forense, acelerando a busca por vestígios da ação criminosa.

À medida que o acesso à internet fica cada vez mais popular, o número de agentes criminosos cibernéticos cresce. Esse aumento levou à diversificação e sofisticação das técnicas de produção de malware.

Os hackers jovens e habilidosos das décadas de 80 e 90, que testavam as defesas dos sistemas computacionais, procurando os limites e possibilidades das máquinas, deram lugar a cibercriminosos experientes, que usam suas habilidades com o intuito de obter vantagem financeira ilegal (CAO; LU, 2011).

Essa transição gerou grupos de criminosos cibernéticos especializados, que se especializam em áreas de atuação, e alugam suas habilidades para o crime organizado (BROADHURST et al., 2014).

Recentemente foram descobertos malwares cujo desenvolvimento foi atribuído a nações com objetivos estratégicos. O Stuxnet, descoberto em 2010, tinha como alvo as usinas nucleares iranianas, e foi desenvolvido para causar danos físicos no hardware das usinas (BUSINESS INSIDER, 2013). Flamer, um malware descoberto em 2012, atacava computadores com o sistema operacional Microsoft Windows, e tinha como objetivo realizar ações de espionagem no oriente médio (SYMANTEC, 2012).

Ainda em 2012, a empresa Kaspersky Lab considerou o Flame como o malware mais sofisticado já encontrado até então. Um outro malware também descoberto em 2012, chamado de DuQu, também tinha como objetivo espionagem entre potências mundiais (DALZIEL, 2013).

Dados contidos em sistemas computacionais afetados por malware (informações bancárias, cartões de crédito, etc) são roubados para serem vendidos no mercado negro. Por fim existem aqueles que usam a internet como meio para aplicar suas fraudes, como ofertas de investimento fictícias, sequestro de dados e chantagem, e outras praticas fraudulentas.

O rápido conhecimento dos autores dos ataques, suas motivações, e os meios que foram utilizados para realizar a ação criminosa são ferramentas essenciais para a repressão ao crime cibernético. Uma informação adquirida rapidamente pode ser usada para direcionar o processo de análise, poupando tempo precioso dos investigadores forenses. Dessa forma, uma definição de perfis de malware automatizada pode ter grande valor no processo de análise de malware.

A maioria dos malwares descobertos nos dias de hoje são códigos reciclados, que utilizam funcionalidades conhecidas, e não são escritos a partir do zero.

Um estudo publicado pela empresa Symantec (SYMANTEC, 2016) mostra que o a criação de famílias de malware genuinamente originais tem diminuído nos últimos anos, o que indica que os criadores de malware estão trabalhando para aperfeiçoar os códigos já existentes, e reutilizam funcionalidades já consagradas.

Isso corrobora com o fato de que a descoberta de variantes de malwares já existentes tem crescido. Em 2010 foram descobertas mais de 286 milhões de variantes de malwares já detectados anteriormente.

A similaridade de botnets como Kelihos, Storm, Nuwar, entre outros, sugere que esses códigos foram desenvolvidos pelos mesmos criadores (BUREAU, 2011). Acredita-se que as similaridades entre os malwares seja mais um fator que contribua para o sucesso de uma abordagem baseada em profiling aplicada a amostras de malware.

Nesse trabalho é apresentada uma abordagem de classificação de malware baseada no profiling criminal. Visando evitar a confusão gerada pelas inconsistências presentes nas definições de classes de malware usadas até então, é proposta uma nova definição de classes de malware, agora chamadas de perfis de malware.

Também é proposto um conjunto de características, estruturais e comportamentais, que demonstram-se promissoras para compor uma estratégia de classificação, baseando-se em trabalhos da literatura de análise de malware e em observações feitas durante o processo de análise de amostras de malware pelos autores deste trabalho.

Essa nova estratégia de classificação tem o objetivo de contribuir para a construção de uma metodologia de classificação que permita definição de perfis de malware de forma automatizada.

3.3. LISTA DE PERFIS

O primeiro passo para se alcançar a definição de perfis malware é a criação da listagem dos perfis que serão usados para agrupar as amostras analisadas. Observando as diversas definições de classes presentes na indústria e na literatura, nota-se que um dos principais motivos para a confusão gerada na classificação de malware atualmente é a falta de um critério claro de classificação, que seja válido para todas as classes.

Requisições feitas aos Peritos Criminais da Seção de Perícias de Informática do Instituto de Criminalística da Polícia Civil do Distrito Federal, em casos envolvendo malware, comumente envolvem quesitos acerca da possibilidade da existência de um malware capaz de atingir um objetivo específico em um sistema computacional. Já foram recebidos pedidos para verificar a presença de malware capaz de enviar mensagems no nome de outras pessoas, roubar informações, encriptar dados de usuário, entre outros.

Dessa forma, a escolha do objetivo do malware como fator de classificação torna a definição de perfis simples, permitindo a criação de uma estratégia de classificação concisa, e cria uma classificação interessante do ponto de vista pericial. Além disso, descobrir o objetivo de um malware rapidamente pode trazer pistas valiosas sobre o autor do crime em tempo reduzido.

Considerando-se as amostras de malware estudadas durante o desenvolvimento do trabalho, e encontradas com maior frequência na casuística do trabalho pericial, cria-se a lista de perfis, baseando-se no objetivo pretendido pelo criador da amostra de malware. Para cada perfil, elenca-se quais comportamentos mais comuns apresentados por amostras de malware que apresentam esse

objetivo final. Esses comportamentos serão utilizados para nortear a escolha de características das amostras de malware, estruturais e comportamentais, a serem extraídas e analisadas.

3.3.1. Banker

Malwares que tem como objetivo obter informações bancárias de seus alvos, para fornecer vantagem financeira ilícita ao criminoso. O crescimento dos serviços bancários online fornecidos pelas instituições financeiras têm levado os cibercriminosos a investir em ataques de furto de dados bancários. Uma vez instalado no computador da vítima, o programa recolhe automaticamente dados de pagamento online, transações financeiras, cartões de crédito, entre outros. Permite assim ao criminoso realizar compras e transações financeiras em nome da vítima, sem o seu consentimento. Existem malwares bancários dirigidos a clientes e bancos específicos, e também os malwares bancários multi-alvos, porém os dois serão agrupados no mesmo perfil.

São comumente enviados às vítimas através de e-mails de phishing, que imitam e-mails reais de bancos para chamar a atenção dos clientes. Utilizam falhas de segurança nos sistemas operacionais para se instalarem nos computadores das vítimas.

Entre as técnicas utilizadas pelos malwares desse tipo estão

- Captura de telas que contenham dados financeiros dos usuários
- Monitoração de teclas digitadas no teclado.
- Alterar a conexão com os servidores, agindo com o um homem no meio, ou então alterar arquivos de proxy do navegador, redirecionando os usuários a websites falsos.
- Controlar o uso de teclados virtuais.

3.3.2. DoS Bot

Um ataque de negação de serviço (também conhecido como DoS Attack, um acrônimo em inglês para Denial of Service), é uma tentativa de tornar os recursos de um sistema indisponíveis para os seus utilizadores. Alvos típicos são servidores web, e o atacante procura tornar as páginas hospedadas indisponíveis na internet. Não se trata de uma invasão do sistema, mas sim da sua invalidação por sobrecarga. É um ataque onde o criminoso pode ter sob seu comando até milhares de computadores Zumbis. Nesse caso, as tarefas de ataque de negação de serviço são distribuídas a um grupo de máquinas escravizadas, infectadas com o malware (CERT.BR, 2016).

O ataque consiste em fazer com que os zumbis acessarem um determinado recurso, num determinado servidor, no mesmo momento. Passada essa fase, na determinada hora, todos os

zumbis (ligados e conectados à rede) acessam ao mesmo tempo o recurso do mesmo servidor. Como servidores web possuem um número limitado de utilizadores que pode atender simultaneamente, o grande e repentino número de requisições de acesso esgota esse número, fazendo com que o servidor não seja capaz de atender a mais nenhum pedido. Dependendo do recurso atacado, o servidor pode chegar a reiniciar ou até mesmo congelar totalmente.

Estão incluídos nesse perfil malwares que tenham como objetivo realizar ataques de negação de serviço, ou tornar o sistema infectado parte de uma rede de zumbis que tenha esse objetivo. Em outras palavras, malwares que utilizam o computador infectado para fazer parte de ataques DoS ou DDoS.

A principal característica de malwares que se encaixam nesse perfil é o envio de um grande número de pacotes de rede, para endereços diferentes, em um pequeno intervalo de tempo.

3.3.3. Ransomware

Malware cujo objetivo é restringir o acesso ao sistema infectado, e exigir o pagamento de uma quantia em dinheiro para que o acesso seja reestabelecido, como forma de resgate. Agem bloqueando arquivos ou partes essenciais do sistema operacional infectado. Normalmente fazem uso de algoritmos criptográficos para encriptar os arquivos, o que muitas vezes torna o acesso aos arquivos impossível de ser recuperado, sem a chave criptográfica.

O uso de ransomware trata-se de um golpe, considerado uma ação de extorsão. Além disso nada garante que o criminoso que utiliza o malware forneça a chave para decriptar os arquivos, mesmo após o pagamento da quantia requisitada.

Características comumente detectadas em malwares que se encaixam nesse perfil incluem:

- Acesso a um grande número de arquivos, em um pequeno intervalo de tempo.
- Criação de arquivos criptografados de tamanho excessivo.

3.3.4. Damageware

Malware que tem como objetivo inviabilizar e/ou danificar o funcionamento do sistema alvo. Através de manipulações feitas no sistema infectado, um criminoso pode realizar ações danosas contra o sistema operacional, ou contra hardwares diversos conectados a ele. O Stuxnet (BUSINESS INSIDER, 2013) foi desenvolvido com esse objetivo, e causou danos à instalações nucleares iranianas.

Malwares que se encaixam nesse perfil normalmente realizam deleção de arquivos essenciais do sistema, ou alteração de arquivos que façam o sistema se comportar de forma indevida.

3.3.5. Backdoor

Malware que tem como objetivo fornecer acesso remoto ao sistema infectado ao criminoso, explorando falhas não documentadas existentes em programas instalados e sistemas operacionais desatualizados. Normalmente utilizam serviços criados ou modificados para este fim.

3.3.6. Downloader

Malware que tem como objetivo realizar o download de outros arquivos de malware no sistema infectado. A principal característica desse tipo de malware é o acesso a endereços da internet e o download de arquivos contendo malware.

3.3.7. Adware

Malware que tem como objetivo exibir anúncios e propaganda sem a permissão do usuário. Alguns adware analisam os endereços de internet visitados e pesquisas realizadas para apresentar ao usuário publicidade pertinente aos tipos de bens ou serviços que costuma consumir.

3.3.8. Spyware

Malware que tem como objetivo obter informação confidencial através da internet para propósitos maliciosos. Se tornou um dos principais objetivos de desenvolvedores de malware nos dias de hoje. As informações são enviadas de volta através de diversos canais, utilizando protocolos comuns da rede, como HTTP, FTP, E-mail ou IRC. Com o desenvolvimento de técnicas de criptografia e esteganografia, esses dados passam a maioria das vezes despercebidos pela rede. Se encaixam nesse perfil programas que são utilizados para roubar informações como dados pessoais de pessoas importantes (Registros de Identidade, Endereços, informações sobre suas rotinas, etc.) credenciais de E-mail e de redes sociais, segredos comerciais, propriedade intelectual, entre outros. Uma vez que dados pessoais sensíveis foram roubados, o criminoso pode até mesmo se passar pela pessoa para cometer crimes e obter outras informações.

3.3.9. Lista de perfis proposta

Com uma lista de perfis consistente, que leva como fator de classificação apenas o objetivo da amostra de malware a ser classificada, basta encontrar pistas que indiquem que a amostra tenta alcançar esse objetivo.

A Figura 3.1 ilustra a lista de perfis proposta, assim como o objetivo a ser detectado para cada perfil. Nessa figura, uma amostra de malware pode ser analisada e ter seu objetivo descoberto. A partir do objetivo, é determinado então o perfil.

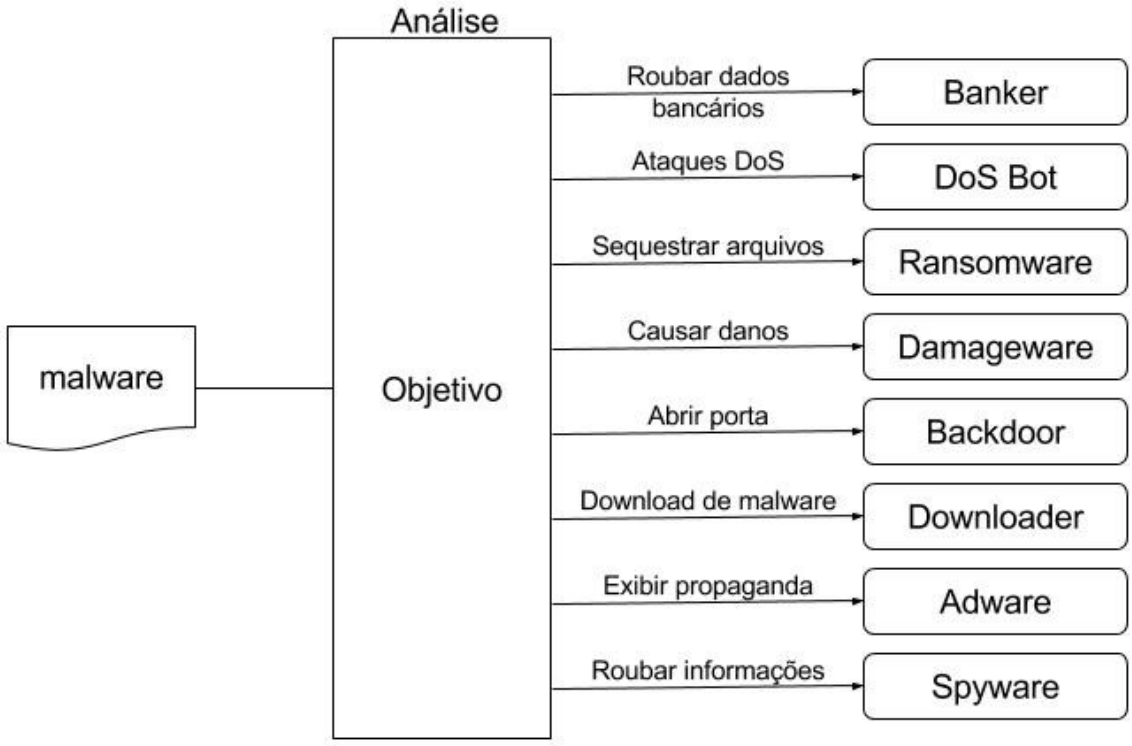


Figura 3.1: Perfis de malware

3.4. SELEÇÃO DE CARACTERÍSTICAS

Para alcançar uma estratégia de classificação capaz de realizar a definição de perfis de malware, deve-se selecionar características a serem observadas nas amostras de malware, que reflitam os comportamentos adotados pelas amostras para alcançar cada objetivo.

Neste trabalho é apresentada uma seleção de características, comportamentais e estruturais, a serem observadas nas amostras de malware analisadas. Essas características são extraídas, e analisadas individualmente, com o objetivo de inferir acerca de sua relevância no processo de definição de perfis de malware.

3.4.1. Chamadas de APIs

Chamadas de APIs fornecem informações importantes sobre o comportamento de um software, e outros trabalhos já as utilizaram para realizar análise de malware (AHMADI et al., 2016), (CHO et al., 2014). Malwares normalmente utilizam as APIs SetWindowsHookEx, GetForegroundWindow e

GetAsyncKeyState para capturar dados digitados pelo alvo (SIKORSKI; HONIG, 2012), técnica conhecida como keylogging. Escolhe-se então considerar o uso ou não de APIs dessa natureza como uma das características. Chamadas de APIs também podem ser usadas para criar serviços (MICROSOFT, 2016) que permitam ao criminoso ter acesso ao sistema infectado. Captura-se então as chamadas de APIs de rede da biblioteca Winsock, bind e listen, que podem ser utilizadas para abrir uma porta de entrada ao sistema operacional, e seu uso ou não pela amostra foi selecionado como uma característica. Chamadas se APIs durante a execução são consideradas características comportamentais, e a presença de APIs de interesse na lista de APIs importadas pelo binário são consideradas características estruturais.

3.4.2. Características Estruturais

Características estruturais são características que podem ser extraídas diretamente do arquivo binário da amostra de malware, sem executá-la.

O header de um arquivo binário pode trazer pistas importantes sobre seu funcionamento e origem. Ele contem um cabeçalho seguido de um número arbitrário de seções. Os nomes dessas seções variam à depender do compilador utilizado, mas normalmente obedecem um padrão. Anomalias nos nomes dessas seções são comuns em arquivos binários de malware (SIKORSKI; HONIG, 2012). Os nomes comumente encontrados em binários benignos são ".text", ".rdata", ".data", ".idata", ".edata", ".pdata", ".rsrc", e ".reloc". O número de seções com nome fora desse padrão foi selecionado como uma das características estruturais selecionadas

Muitos tipos de malware realizam alterações de arquivos no sistema infectado. Acreditamos que a forma com que essas alterações são feitas pode trazer pistas sobre os objetivos do malware. Assim o número de arquivos acessados no sistema infectado durante a execução de uma amostra foi selecionado como uma das características comportamentais.

Além dessas características, foram selecionadas características já utilizadas na literatura para detecção e classificação de malware. Em (AHMADI et al., 2016) o tamanho do arquivo foi identificado como uma das características mais expressivas no seu classificador.

A tabela 3.1 apresenta as características estruturais selecionadas para extração e análise.

Tabela 3.1: Características estruturais selecionadas para extração

ID	Característica
C1	Tamanho do arquivo binário
C2	APIs de keylogging
C3	APIs de rede
C4	Número de seções com nome desconhecido

3.4.3. Características Comportamentais

Características comportamentais são características que podem ser observadas durante a execução de uma amostra de malware.

O comportamento do malware na rede pode fornecer informações importantes sobre seu objetivo. Amostras de malware que tem o objetivo de praticar ataques de DDoS normalmente realizam um número muito maior de envios de pacote de rede que recebem. Desse modo, a porcentagem de pacotes de saída (pacotes cuja origem é a máquina infectada), em relação ao total de pacotes que trafegam na rede, foi selecionada como uma das características.

A instalação no Autorun do sistema operacional foi utilizada em (LU et al., 2010) como característica para detecção de malware.

Técnicas que realizam polimorfismo são frequentemente utilizadas em malwares para evitar a detecção por softwares antivírus (RAD; MASROM; IBRAHIM, 2012). A criação ou não de variantes polimórficas da amostra de malware durante sua execução foi selecionada como característica comportamental a ser observada.

Em estudos realizados durante o desenvolvimento deste trabalho, outros comportamentos suspeitos foram observados durante a execução de algumas amostras de malware. Algumas amostras se conectam e trocam mensagens com endereços de IP que deixam de responder rapidamente. Serviços da internet legítimos normalmente se mantêm online para receber novas requisições. A deleção do arquivo binário original após este ter sido executado, também foi um comportamento suspeito observado. Esses dois comportamentos foram selecionados como características comportamentais a serem observadas.

Por fim, tentativas de acesso a arquivos relacionados ao algoritmo de Bitcoin também foram observadas. A Tabela 3.2 apresenta as características comportamentais selecionadas para extração e análise.

Tabela 3.2: Características comportamentais selecionadas para extração

ID	Característica
C2	APIs de keylogging
C3	APIs de rede
C5	Número de arquivos acessados
C6	Porcentagem de pacotes de saída
C7	Instalação no Autorun
C8	Cria variantes polimórficas
C9	Se conecta a hosts suspeitos
C10	Deleta o binário original
C11	Acessa dados de bitcoin

As características C2 e C3 são consideradas estruturais e comportamentais, já que as APIs mencionadas são checadas tanto na lista de APIs importadas pelo binário, quanto na lista de chamadas de APIs feitas durante a execução da amostra.

Para realizar a extração das características comportamentais e estruturais das amostras, foi utilizada a ferramenta Cuckoo Sandbox. O funcionamento da ferramenta, assim como o processo de extração e a análise das características extraídas serão explanados no próximo capítulo.

4. EXTRAÇÃO E ANÁLISE DE CARACTERÍSTICAS

Para se alcançar o processo de classificação necessário para a definição de perfis de malware, comportamentos específicos de cada perfil devem ser verificados em uma determinada amostra de malware.

Neste trabalho, é realizada a extração das características selecionadas, promissoras para representar comportamentos em um possível processo de classificação. Essas características são extraídas utilizando scripts, desenvolvidos pelo autor na linguagem python, que trabalham com a ferramenta Cuckoo Sandbox.

Para a extração das características, as amostras de malware foram executadas em uma máquina virtual VirtualBox, doravante denominada de guest, com o sistema operacional Windows XP. A execução da amostra é controlada pela máquina física, doravante denominada host, com o sistema operacional Linux Mint. O host controla a execução com a ferramenta Cuckoo Sandbox, e as informações entre as duas máquinas são trocadas através de uma rede virtual, utilizando adaptadores de rede virtuais. A Figura 4.1 ilustra o ambiente montado para execução das amostras e extração de características.

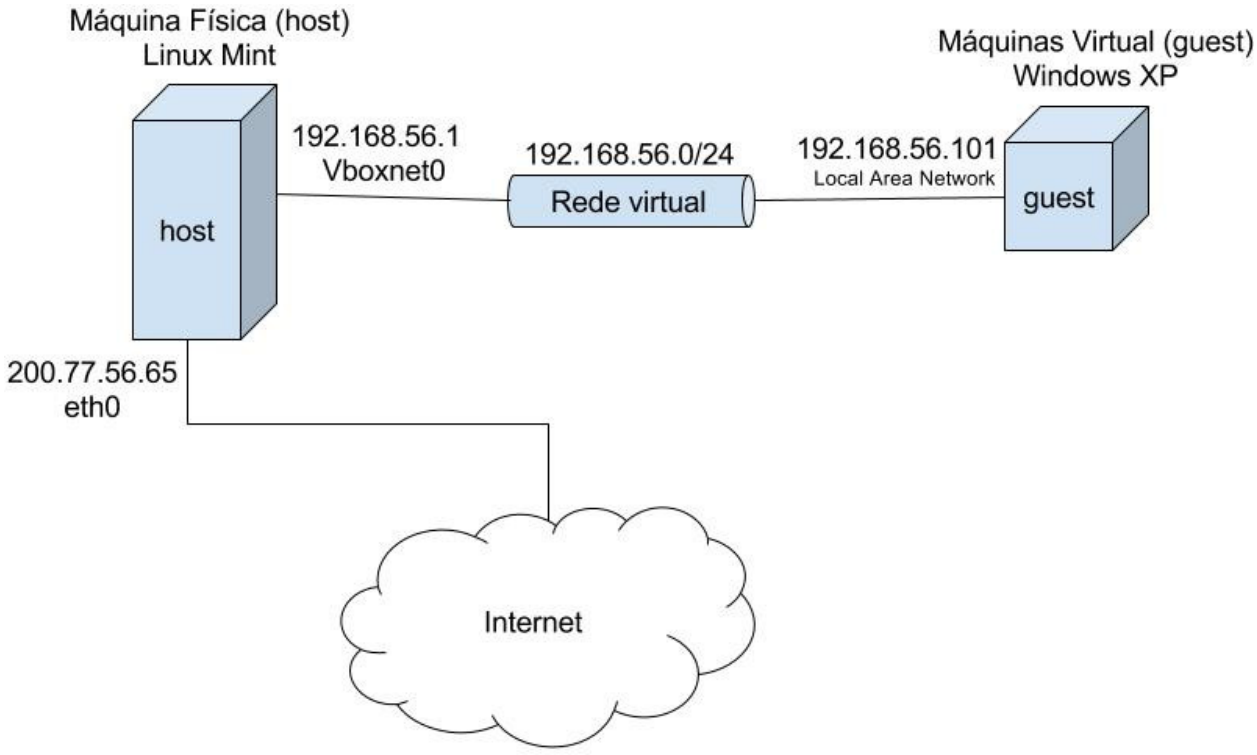


Figura 4.1: Ambiente montado para execução das amostras

O acesso à internet pela máquina guest é utilizando um adaptador de rede virtual da máquina host, chamado Vboxnet0, que age como Gateway, redirecionando os pacotes destinados à máquina guest.

Após a extração, é feita uma análise de cada característica, acerca de sua relevância no processo de definição de perfis de malware.

4.1. A FERRAMENTA CUCKOO SANDBOX

Cuckoo Sandbox é uma ferramenta de código aberto que tem o objetivo de permitir análise de malware de forma automática. Construída utilizando a linguagem de programação Python (PYTHON SOFTWARE FOUNDATION, 2016), é altamente modular, e faz o uso de diversos componentes para monitorar o comportamento de uma amostra de malware durante sua execução em um ambiente controlado.

Sandbox é definido como um mecanismo de segurança que tem o objetivo de separar programas em execução (GOLDBERG et al., 1996). São comumente utilizados para executar códigos não testados ou suspeitos, de origem não confiável. O conceito de sandboxing aplica-se à análise dinâmica de malware feita em máquinas virtuais. Tal execução permite, na maioria dos casos, a execução de um arquivo não confiável (malware) em um ambiente isolado e controlado, para obter pistas sobre o comportamento desse arquivo.

Apesar de muito útil, essa abordagem também possui desvantagens. Amostras de malware podem detectar que estão sendo executadas em um ambiente virtualizado e abortarem a execução. Portanto é essencial que seja combinada com técnicas de análise estática.

Cuckoo é uma ferramenta que permite ao investigador forense realizar esse tipo de execução em ambiente controlado. Além de fornecer um ambiente seguro, possui módulos configuráveis que permitem a criação de diversos tipos de análises compreensivas que podem trazer e processar informações obtidas durante a execução da amostra de malware.

Dentre os tipos de resultados que podem ser obtidos em análises utilizando o Cuckoo Sandbox estão:

- Chamadas de APIs
- Arquivos acessados e baixados pela amostra
- Dumps de memória volátil da máquina virtual
- Tráfego de rede
- Capturas de tela

A ferramenta consiste de uma central, que gerencia a execução da amostra e realiza as análises, e de um grupo de uma ou mais máquinas virtuais, que executam as amostras. Para cada execução é iniciada uma instância da máquina virtual limpa e isolada, gerada a partir de um snapshot criado antes da execução de qualquer malware. A Figura 4.2 ilustra uma rede virtual criada para ser utilizada com a ferramenta Cuckoo Sandbox.

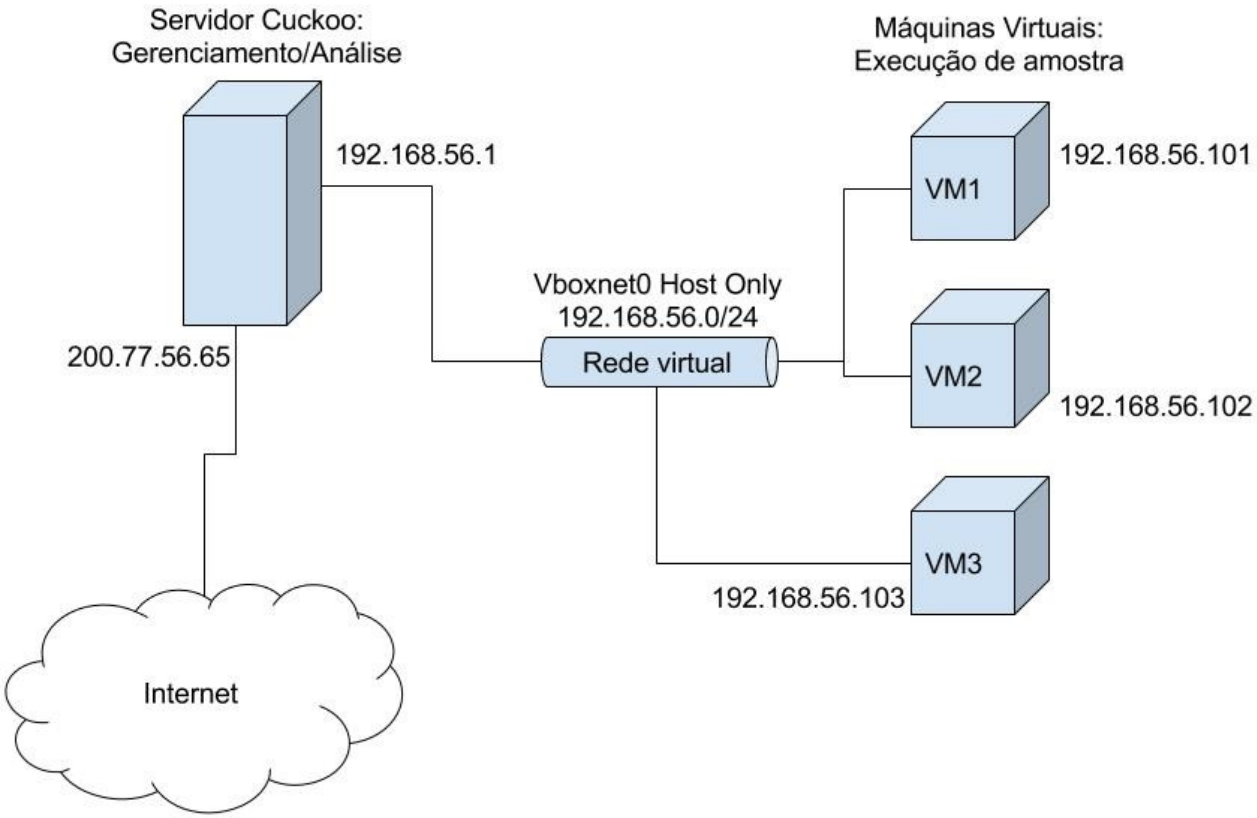


Figura 4.2: Configuração de uma rede virtual Cuckoo Sandbox

Além de trabalhar com máquinas virtuais, a ferramenta Cuckoo Sandbox também pode trabalhar com máquinas físicas, utilizando imagens dessas máquinas para criar instâncias limpas do sistema.

Os módulos são componentes essenciais da ferramenta Cuckoo Sandbox. São estruturados em classes Python que determinam como as informações devem ser coletadas, processadas e apresentadas. Além dos módulos padrão fornecidos em conjunto com a ferramenta, a estrutura modular permite a fácil criação de módulos específicos para atender demandas do usuário. Os três principais tipos de módulos são processamento, assinatura e relatório.

4.1.1. O container global de resultados

O container global é uma instância de uma estrutura de dados da linguagem Python, denominada dicionário, que contêm resultados produzidos por todos os módulos da ferramenta, indexados por chaves definidas nos módulos. Dicionários são tipos compostos de dados, como strings, listas e tuplas, porém permitem o uso de qualquer tipo de dados como índice. Também permitem armazenar qualquer tipo de dados.

O fragmento de estrutura de dados apresentado na Figura 4.3, resultante de uma análise de malware utilizando a ferramenta Cuckoo Sandbox, e produzido pelo módulo de processamento **analysisinfo.py**, é um exemplo de uso de dicionário:

```
01.  {
02.      "info": {
03.          "category": "file",
04.          "score": 6.8,
05.          "package": "",
06.          "started": 1477938433.054216,
07.          "route": "none",
08.          "custom": "",
09.          "machine": {
10.              "status": "stopped",
11.              "name": "XPSP3x32",
12.              "label": "XPSP3x32",
13.              "manager": "VirtualBox",
14.              "started_on": "2016-10-31 18:27:13",
15.              "shutdown_on": "2016-10-31 18:30:04"
16.          },
17.          "ended": 1477938605.074598,
18.          "version": "2.0-dev",
19.          "platform": "",
20.          "owner": "",
21.          "options": "",
22.          "id": 81,
23.          "duration": 172
24.      },
```

Figura 4.3: Estrutura de dados resultante do módulo **analysisinfo.py**

Nesse exemplo, assumindo uma estrutura de dados de dicionário, com o nome de results, uma chamada a **results[“info”][“category”]** retornaria a string **“file”**, e uma chamada a **results[“info”][“duration”]** retornaria o inteiro **172**.

4.1.2. Módulos de processamento

Os módulos de processamento são scripts Python que definem como serão extraídos e processados os dados presentes na máquina virtual. Permitem definir maneiras personalizadas de analisar os resultados brutos gerados. As informações resultados desse processamento são inseridas em uma estrutura de dados chamada de container global, que será usado posteriormente pelos módulos de assinatura e relatório.

Os módulos de processamento devem ser inseridos na pasta **/modules/processing**, e devem ser habilitados para execução no arquivo de configuração **/conf/processing.conf**. Os módulos são habilitados inserindo-se uma entrada no arquivo de configuração, conforme Figura 4.4:

```
[nome-do-modulo]
enabled = yes
```

Figura 4.4: Modelo de entrada no arquivo de configuração **processing.conf**

A ferramenta executa automaticamente todos os módulos habilitados e presentes na pasta **/modules/processing**.

Módulos criados pelo usuário devem seguir uma estrutura predefinida. A Figura 4.5 ilustra um exemplo com a estrutura mínima de um módulo de processamento:

```
01.    from lib.cuckoo.common.abstracts import Processing
02.
03.          class ExemploProcessamento(Processing):
04.
05.              def run(self):
06.                  self.key = "index"
07.                  results = do_something()
08.              return results
```

Figura 4.5: Estrutura mínima de um módulo de processamento

Nesse exemplo, a função **run( )** é executada durante a execução da amostra, e gera resultados que são armazenados na variável **results.** Essa variável pode tomar a forma de qualquer tipo de dados, string, lista, dicionário, etc. Esses resultados serão inseridos no container global, em uma entrada indexada pela chave **"index"**.

Os módulos de processamento têm acesso a um conjunto de variáveis que contem os caminhos dos arquivos gerados pela análise, com os dados brutos capturados da máquina virtual. As variáveis podem ser vistas na Tabela 4.1.

Tabela 4.1: Variáveis de ambiente disponíveis para módulos de processamento.

Variável	Conteúdo
self.analysis_path	caminho para a pasta que contém os resultados (por exemplo, **storage/analysis/1**).
self.log_path	caminho para o arquivo **analysis.log.**
self.conf_path	caminho para o arquivo **analyze.conf**.
self.file_path	caminho para o arquivo analisado.
self.dropped_path	caminho para a pasta que contém os arquivos baixados.
self.logs_path	caminho para a pasta que contém os logs comportamentais brutos.
self.shots_path	caminho para a pasta que contém as capturas de tela.
self.pcap_path	caminho para o dump de rede, no formato pcap.
self.memory_path	caminho para o arquivo de dump de memória.
self.pmemory_path	caminho para o arquivo de dump de memória de processos.

Com essas variáveis, o investigador forense pode acessar os dados brutos colhidos pela ferramenta e gerar informações a serem tratadas pelos módulos de assinatura.

4.1.3. Módulos de assinatura

Os módulos de assinatura são utilizados para tratar os resultados gerados pelos módulos de processamento. Permitem a criação de assinaturas que podem identificar padrões predefinidos, que representam comportamentos específicos de amostras de malware. São usados neste trabalho para a extração de características comportamentais que representam comportamentos indicativos de ações específicas, promissoras para um processo de classificação na definição de perfis de malware.

Exemplos do que os módulos de assinatura podem detectar incluem:

- Identificar modificações feitas pela amostra de malware no sistema, como instalação de drivers.
- Tratar chamadas de APIs, buscando chamadas de APIs específicas.

- Acessar os arquivos criados e acessados por uma amostra de malware.
- Acessar os processos criados ou relacionados a uma amostra de malware.

A ferramenta Cuckoo Sandbox traz uma série de assinaturas criadas pelo time de desenvolvimento, e algumas dessas assinaturas são utilizadas nesse trabalho para extrair as características comportamentais de interesse.

Os módulos de assinatura devem ser inseridos na pasta **/modules/signatures.** Os módulos presentes nessa pasta são executados automaticamente pela ferramenta. Os resultados gerados pelos módulos de assinatura são inseridos container global de resultados.

Assim como os módulos de processamento, módulos de assinatura também devem seguir uma estrutura mínima. A Figura 4.6 apresenta um exemplo de uma assinatura criada para este trabalho:

```
01.  from lib.cuckoo.common.abstracts import Signature
02.
03.      class AccessManyFiles(Signature):
04.
05.          name = "many_files"
06.          description = "This malware access %d files"
07.          severity = 2
08.          categories = ["ransom"]
09.          authors = ["caldas"]
10.          minimum = "2.0"
11.          nfiles = 0
12.
13.          def on_complete(self):
14.              self.nfiles = sum(1 for _ in self.get_files())
15.              self.description = self.description % self.nfiles
16.              self.mark(nfiles=self.nfiles)
17.              return True
```

Figura 4.6: Módulo de assinatura **manyfiles.py,** desenvolvido neste trabalho

Essa assinatura conta o número de arquivos acessados pela amostra de malware durante a execução, e insere o resultado como uma marca no container global de dados. O resultado dessa assinatura é uma estrutura de dados similar à encontrada na Figura 4.7:

```
01.          {
02.               "markcount": 1,
03.               "families": [],
04.               "description": "This malware access 9 files",
05.               "severity": 2,
06.               "marks": [
07.                   {
08.                       "nfiles": 9,
09.                       "type": "generic"
10.                   }
11.               ],
12.               "references": [],
13.               "name": "many_files"
14.           },
```

Figura 4.7: Resultado do módulo de assinatura **manyfiles.py**

Assinaturas possuem atributos iniciais, que são definidos no início do código (name, severity, authors, etc) e funções a serem executadas. Na assinatura em questão o código da função **on_complete( )** será executado ao fim da execução da assinatura. As funções **on_call( ), on_process( )** e **on_thread( )** também pode ser usadas para executar comandos durante a execução da amostra de malware.

Além de possuir acesso ao container global de resultados, as assinaturas possuem funções pré-definidas com processamentos úteis que podem ser realizados durante a execução da amostra. Essas funções são chamadas de helpers.

4.1.4. Módulos de relatório

Módulos de relatório são os módulos responsáveis por formatar e apresentar os resultados contidos no container global. Após o processamento dos dados extraídos da máquina virtual pelos módulos de processamento e assinatura, o container global de dados é passado para todos módulos de relatório disponíveis, que tem a tarefa de transformá-lo em informação acessível em diferentes formatos.

Os módulos de relatório são localizados na pasta **/modules/reporting**, e assim como os módulos de processamento devem ser habilitados para execução, no arquivo de configuração **/conf/reporting.conf**. Os módulos são habilitados inserindo-se uma entrada no arquivo de configuração conforme Figura 4.8

```
[nome-do-modulo]
enabled = on
```

Figura 4.8: Modelo de entrada no arquivo de configuração **reporting.conf**

Após o término da execução da amostra, a ferramenta executa automaticamente todos os módulos habilitados e presentes na pasta **/modules/reporting**.

Além do container global, os módulos de relatório tem acesso à variáveis que contem informações relevantes acerca da análise realizada. Algumas dessas variáveis podem ser vistas na Tabela 4.2.

Tabela 4.2: Variáveis de ambiente disponíveis para módulos de processamento.

Variável	Conteúdo
self.analysis_path	caminho para a pasta que contém os arquivos de resultado da análise (storage/analyses/1/, por exemplo).
self.reports_path	caminho para a pasta onde os relatórios devem ser gravados (storage/analyses/1/reports/, por exemplo).
self.conf_path	caminho para a pasta que contém o arquivo analysis.conf da análise em questão (storage/analyses/1/analysis.conf, por exemplo).
self.options	um dicionário de dados contendo as opções especificadas pelo usuário em conf/reporting.conf.

A ferramenta Cuckoo Sandbox traz módulos de relatório criados pelo time de desenvolvimento, que são capazes de gerar relatórios no formato JSON (JSON, 2016), além de popularem uma base de dados MongoDB (MONGODB, 2016), que é utilizada para gerar uma página HTML para apresentar os resultados. A Figura 4.9 ilustra um exemplo de relatório HTML gerado pela ferramenta.

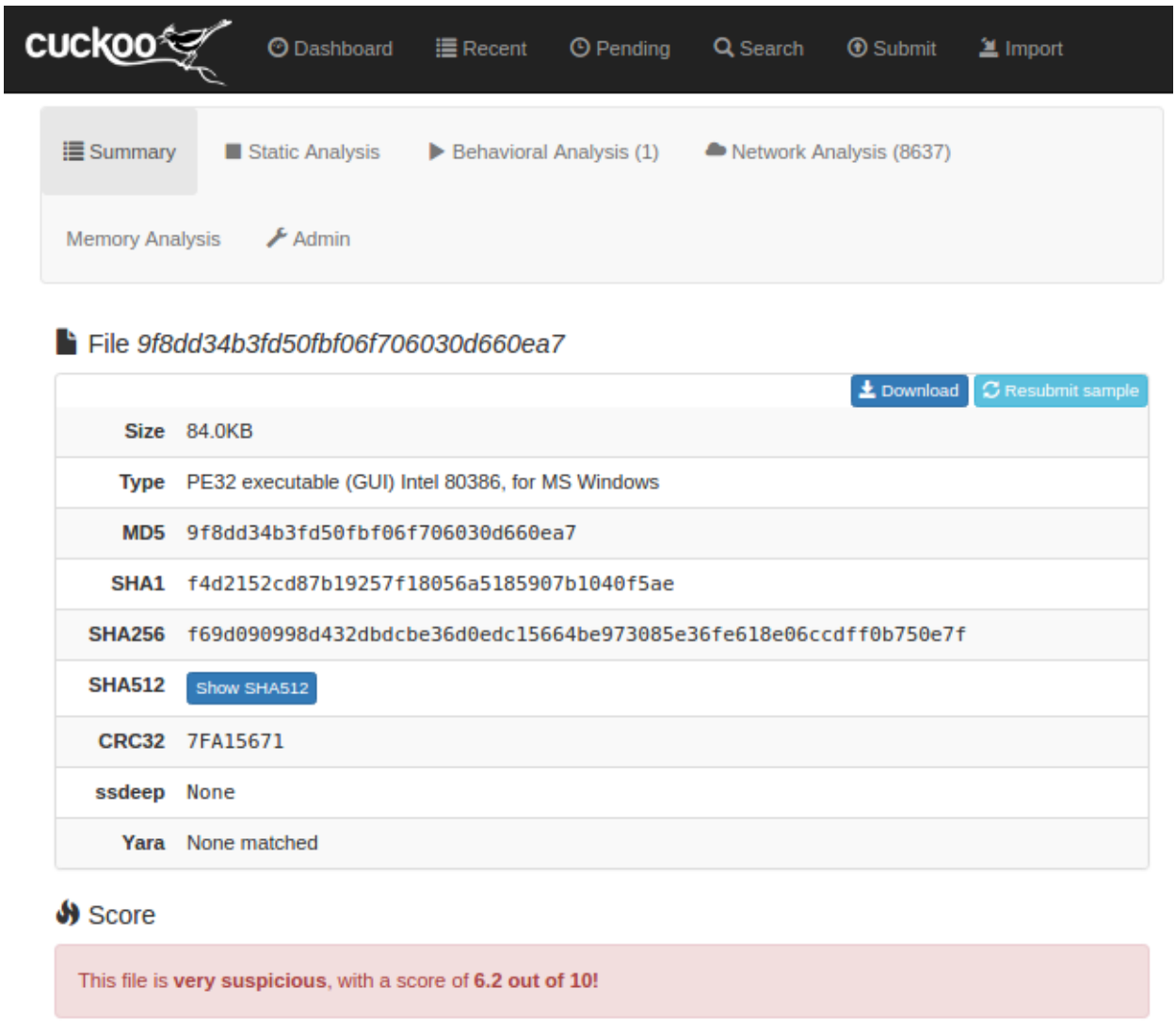


Figura 4.9: Exemplo de relatório HTML gerado pela ferramenta Cuckoo Sandbox

Visando criar uma extração automatizada de características, foi criado um módulo de relatório para o desenvolvimento deste trabalho. Este módulo obtém características de interesse e as armazena em um arquivo de texto, para posterior análise.

4.2. EXTRAÇÃO DE CARACTERÍSTICAS

Neste item, é apresentada a estratégia utilizada para extração das características de interesse das amostras de malware executadas, com auxílio da ferramenta Cuckoo Sandbox. Traz também as classes Python desenvolvidas para cada característica. A Figura 4.10 ilustra as características extraídas, separadas por tipo:

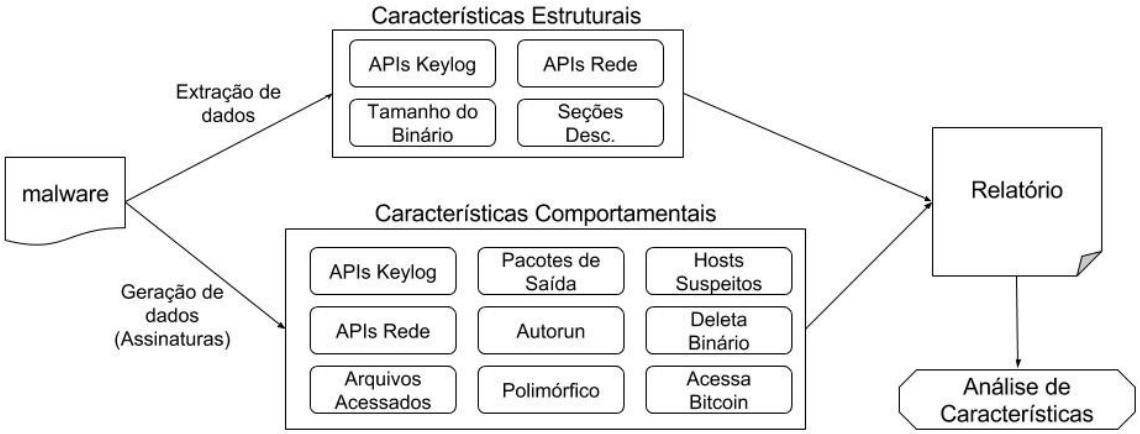


Figura 4.10: Resumo das características a serem extraídas

As características são extraídas utilizando módulos de processamento e de assinatura, desenvolvidos neste trabalho. As características estruturais são extraídas com módulos de processamento, obtidas diretamente do arquivo binário. As características comportamentais são extraídas analisando dados contidos no container global, gerados pelos módulos de processamento. Esses dados são tratados, e para cada característica é feita a busca de um comportamento específico, que a represente satisfatoriamente. Além dos módulos desenvolvidos, também foram utilizados módulos fornecidos com conjunto com a ferramenta Cuckoo Sandbox. A Figura 4.11 ilustra o processo de extração de características.

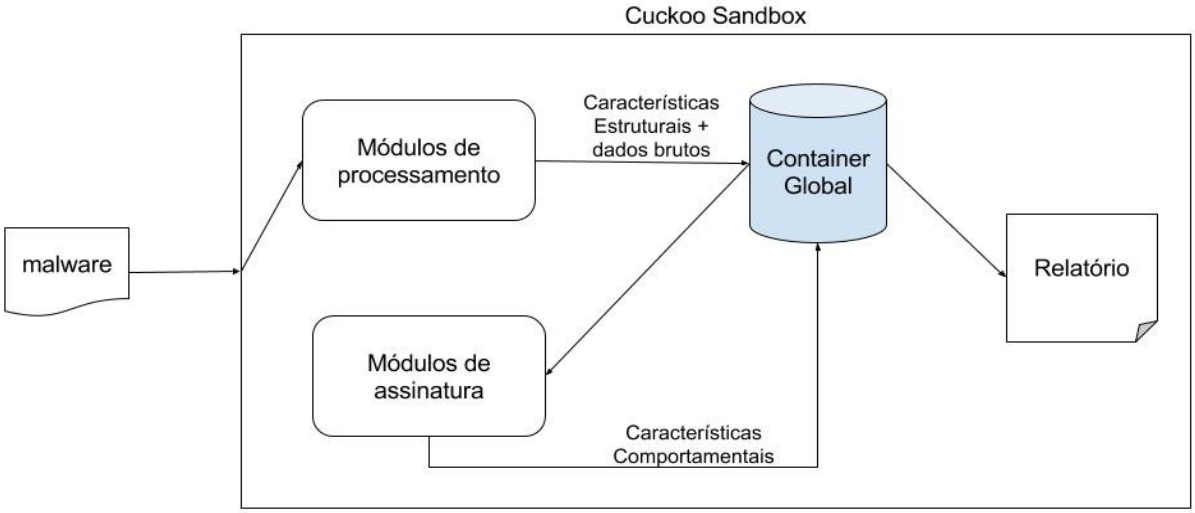


Figura 4.11: Processo de extração de características

4.3. EXTRAÇÃO DE CARACTERÍSTICAS ESTRUTURAIS

As características estruturais são extraídas diretamente do arquivo binário, por módulos de processamento que acompanham a ferramenta.

O tamanho do arquivo é obtido no módulo de processamento **targetinfo.py.** Parte desse módulo pode ser vista na Figura 4.12.

Fonte: Parte da ferramenta Cuckoo Sandbox

```
01.    class TargetInfo(Processing):
02.        """General information about a file."""
03.        def run(self):
04.            """Run file information gathering.
05.            @return: information dict.
06.            """
07.            self.key = "target"
08.            if not self.task:
09.                return {"category": "unknown", "file": {"name": "unknown"}}
10.
11.            target_info = {"category": self.task["category"]}
12.
13.            # We have to deal with file or URL targets.
14.            if self.task["category"] == "file":
15.                target_info["file"] = {}
16.                # et's try to get as much information as possible, i.e., the
17.                # filename if the file is not available anymore.
18.                if os.path.exists(self.file_path):
19.                    target_info["file"] = File(self.file_path).get_all()
```

Figura 4.12: Módulo de processamento **targetinfo.py**

Já os nomes das seções e as APIs importadas pela amostra de malware são extraídas no módulo de processamento **static.py**. Este fragmento de código presente no arquivo **static.py** faz a extração dos nomes das seções, e pode ser visto na Figura 4.13.

Fonte: Parte da ferramenta Cuckoo Sandbox

```
01.    # Copyright (C) 2010-2013 Claudio Guarnieri.
02.    # Copyright (C) 2014-2016 Cuckoo Foundation.
03.    # This file is part of Cuckoo Sandbox - http://www.cuckoosandbox.org
04.    # See the file 'docs/LICENSE' for copying permission.
05.    def _get_sections(self):
06.        """Gets sections.
07.        @return: sections dict or None.
08.        """
09.        sections = []
10.
11.        for entry in self.pe.sections:
12.            try:
13.                section = {}
14.                section["name"] =
15.             convert_to_printable(entry.Name.strip("\x00"))
16.                section["virtual_address"] =
17.             "0x{0:08x}".format(entry.VirtualAddress)
18.                section["virtual_size"] =
19.             "0x{0:08x}".format(entry.Misc_VirtualSize)
20.                section["size_of_data"] =
21.             "0x{0:08x}".format(entry.SizeOfRawData)
22.                section["entropy"] = entry.get_entropy()
23.                sections.append(section)
24.            except:
25.                continue
26.
27.        return sections
```

Figura 4.13: Módulo de processamento **static.py**

Durante a geração do relatório, os nomes dessas seções passam por um processo de filtragem, e caso seja encontrado um nome incomum, soma-se um no contador de nomes de seções desconhecidos.

São considerados nomes conhecidos os nomes de seção ".text", ".rdata", ".data", ".idata", ".edata", ".pdata", ".rsrc", e ".reloc". Esse fragmento de código retirado do módulo de relatório **caldasdump.py**, visto na Figura 4.14 e desenvolvido para este trabalho, representa essa operação:

```
01.    # Number of unknown sections
02.
03.    known_sections = (".text", ".rdata", ".data", ".idata", ".edata", ".pdata",
       ".rsrc", ".reloc")
04.
05.    unknown_sections = 0
06.    for pe_section in report["static"]["pe_sections"]:
07.        if pe_section["name"] not in known_sections:
08.            unknown_sections += 1
09.    file_results.write(str(unknown_sections))
```

Figura 4.14: Fragmento do módulo de relatório **caldasdump.py**

Apesar de serem extraídas no módulo de processamento **static.py** APIs importadas passam por um filtro no modulo de assinatura desenvolvido neste trabalho, que será explicada na próxima seção.

4.4. EXTRAÇÃO DE CARACTERÍSTICAS COMPORTAMENTAIS

As características comportamentais são extraídas utilizando módulos de assinatura específicos para cada característica. As execuções das amostras foram feitas em uma máquina virtual VirtualBox instalada com o sistema operacional Windows XP. Cada amostra foi executada individualmente em uma instância limpa da máquina virtual.

4.4.1. APIs de keylogging

As chamadas de APIs de keylogging são extraídas no modulo de assinatura **keylogAPIs.py**. Define-se uma função **on_call( )** que é chamada a cada chamada de API feita pela amostra de malware, ou por processos criados a partir dessa. Um fragmento desse módulo pode ser visto na Figura 4.15.

```
01.    def on_call(self, call, process):
02.        if call["api"] == "SetWindowsHookEx":
03.            self.mark_call()
04.            self.mask = 1
```

Figura 4.15: Fragmento do módulo de assinatura **keylogAPIs.py**

Nessa função, é verificado se a API chamada faz parte do grupo de APIs suspeitas, utilizadas por programas de keylogging. Esse grupo é composto pelas APIs: "SetWindowsHookEx",

"SetWindowsHookExW", "UnhookWindowsHookEx", "UnhookWindowsHookExW", "GetAsyncKeyState", "GetAsyncKeyStateW", "GetForegroundWindow", e "GetForegroundWindowW"

Também nesse módulo está presente um filtro que trata as APIs importadas no binário da amostra de malware. Na função **on_complete( )** dessa assinatura, executa-se o código visto na Figura 4.16:

```
01.    def on_complete(self):
02.
03.        report = self.get_results("static", {})
04.        for pe_import in report["pe_imports"]:
05.            for importedAPI in pe_import["imports"]:
06.                if importedAPI["name"] in api_names
07.                    self.mark(name=importedAPI["name"])
08.                    self.mask = 1
```

Figura 4.16: Fragmento do módulo de assinatura **keylogAPIs.py**

Esse código recupera os resultados contidos no container global sob a chave “static”, e compara cada API importada, verificando se pertencem ao grupo suspeito.

4.4.2. APIs de rede

Chamadas às APIs de rede selecionadas são verificadas utilizando o módulo de assinatura **caldasbind.py.** Esse módulo trabalha de maneira similar ao módulo **keylogAPIs.py,** usando a função **on_call( )** para verificar se a API chamada faz parte do grupo de APIs suspeitas. O fragmento do código da assinatura, visto na Figura 4.17, mostra seu funcionamento:

```
01.    def on_call(self, call, process):
02.        if call["api"] == "bind":
03.            self.mark_call()
04.            self.mask |= 1
```

Figura 4.17: Fragmento do módulo de relatório **caldasbind.py**

Este código é repetido para cada uma das três APIs de rede verificadas, “bind", "listen", "accept". Caso quaisquer duas das três chamadas suspeitas sejam feitas pela amostra de malware, uma marca é inserida no container global com o nome de “network_bind”.

4.4.3. Arquivos acessados

Os arquivos acessados são verificados no módulo de assinatura **manyfiles.py.** Um fragmento desse módulo pode ser visto na Figura 4.18.

```
01.    def on_complete(self):
02.        self.nfiles = sum(1 for _ in self.get_files())
03.        self.description = self.description % self.nfiles
04.        self.mark(nfiles=self.nfiles)
05.        return True
```

Figura 4.18: Fragmento do módulo de relatório **manyfiles.py**

Esse módulo usa a função **on_complete( )** para contar o número de arquivos acessados pela amostra de malware e o insere no container global.

4.4.4. Porcentagem de pacotes de saída

A porcentagem de pacotes de saída é calculada utilizando informações geradas no módulo de processamento **caldasprocessing.py**. Esse módulo, entre outras coisas, faz o tratamento de um arquivo do arquivo **dump.pcap.** O arquivo **dump.pcap** contém um dump com todos os pacotes de rede que trafegaram pela rede virtual durante a execução da amostra de malware. O módulo faz um tratamento, e verifica em cada pacote, se o IP de origem do pacote é igual ao IP da máquina virtual que está executando a amostra. Caso seja igual, ele soma uma unidade a um contador de pacotes de saída.

O total de pacotes e o número de pacotes de saída são inseridos no container global para serem tratados posteriormente pelo módulo de relatório desenvolvido neste trabalho.

4.4.5. Instalação no Autorun

A constatação de instalação no autorun do sistema operacional pela amostra de malware é feita pelo módulo de assinatura **persistence_autorun.py.** Esse módulo é fornecido em conjunto com a ferramenta Cuckoo Sandbox. Utiliza a função **on_complete( )** para verificar as chaves de registro do windows que controlam o que é executado automaticamente quando o sistema é inicializado. Caso um desses registros tenha sido alterado durante a execução da amostra de malware, uma marca é inserida no container global com o nome de “persistence_autorun”.

4.4.6. Comportamento polimórfico

O comportamento polimórfico é detectado pelo módulo de assinatura **packer_polymorphic.py.** Esse módulo é fornecido em conjunto com a ferramenta Cuckoo Sandbox. Ele calcula um hash SHA1 (EASTLAKE 3RD; JONES, 2001) do arquivo binário da amostra de malware, e verifica os arquivos criados pela amostra de malware que possuem tamanho similar ao da amostra. Caso algum arquivo similar à amostra possua hash SHA1 diferente uma marca é inserida no container global com o nome de "packer_polymorphic".

4.4.7. Conexão com hosts suspeitos

O contato com endereços de IP que deixam de responder é verificado no módulo de assinatura **dead_hosts.py**. Esse módulo é fornecido em conjunto com a ferramenta Cuckoo Sandbox. Ele utiliza dados resultantes do módulo de processamento **network.py.** O módulo, **network.py**, entre outras coisas, considera como host suspeito combinações de IP/porta que não responderam ao mínimo de 3 tentativas, após ter sido contatado com sucesso. Caso algum host suspeito tenha sido detectado, uma marca é inserida no container global com o nome de "dead_host".

4.4.8. Deleção do binário original

O módulo de assinatura **deletes_self.py** é usado para verificar se a amostra de malware deleta o arquivo binário original após o início da execução na máquina virtual. Esse módulo é fornecido em conjunto com a ferramenta Cuckoo Sandbox. Ele varre a lista de arquivos deletados pelo malware, contida no container global, e verifica se o arquivo binário original faz parte dessa lista. Caso o arquivo binário original faça parte da lista de arquivos deletados, uma marca é inserida no container global com o nome de "deletes_self".

4.4.9. Acessa dados de bitcoin

O módulo de assinatura **infostealer_bitcoin.py** é usado para verificar se a amostra de malware acessa arquivos usados em algoritmos de bitcoin após o início da execução na máquina virtual. Esse módulo é fornecido em conjunto com a ferramenta Cuckoo Sandbox. Ele cria uma lista de arquivos comumente associados a algoritmos de bitcoin, e varre a lista de arquivos acessados pela amostra de malware, contida no container global. Caso algum arquivo da lista tenha sido acessado, uma marca é inserida no container global com o nome de "infostealer_bitcoin".

4.5. ANÁLISE DA EXTRAÇÃO

Foram coletadas 41 amostras de malware para execução utilizando a metodologia descrita. As amostras foram coletadas dos depósitos de malware online Das Malwerk (CERT.BR, 2016), MalShare (THE MALSHARE PROJECT., 2016), the Zoo (NATIV, 2016) e de um honeypot genérico.

Dessas amostras, 14 apresentaram algum tipo de técnica anti-virtualização e não tiveram suas características extraídas. As amostras executadas, acompanhadas de um identificador (ID), o hash md5 dos arquivos, e o perfil no qual elas se encaixam podem ser vistas na Tabela 4.3.

Tabela 4.3: Amostras de malware executadas

Malware	ID	Hash MD5	Perfil
Emotet	1	8baa9b809b591a11af423824f4d9726a	Banker
Androm.dsdcle	2	b3962f61a4819593233aa5893421c4d1	Banker
Banload.BEK	3	f86ba7479b96c7fc8870feafcfa9e22e	Banker
Bestafera.ecus	4	1ec29c745a1f740412e925894cd5f2d0	Banker
Bfestafera.bmh	5	38aef00d10665331222ddf5c625fa846	Banker
Nivdort	6	ed2cd14a28ff2d00a5cefcf6a074af8d	Banker
Bestafera.bpv	7	a3c2a7fe9f34aee069588bf7dcc7aadc	Banker
TeslaCrypt.b	8	209a288c68207d57e0ce6e60ebf60729	Ransomware
TeslaCrypt.a	9	6e080aa085293bb9fbdcc9015337d309	Ransomware
Santana	10	46bfd4f1d581d7c0121d2b19a005d3df	Ransomware
Radamant	11	6152709e741c4d5a5d793d35817b4c3d	Ransomware
TeslaCrypt.c	12	6d3d62a4cff19b4f2cc7ce9027c33be8	Ransomware
Locky	13	b06d9dd17c69ed2ae75d9e40b2631b42	Ransomware
Cryptowall	14	47363b94cee907e2b8926c1be61150c7	Ransomware
Cryptolocker	15	04fb36199787f2e3e2135611a38321eb	Ransomware
Allaple-5	16	0a278f8d72e4d3d2d44485764398c84d	DoS Bot
Allaple-307	17	0bf61fd80666b81fdda155af542b8ae4	DoS Bot
Allaple-246	18	0fe98a3f214d87be0f51a818b346b2a7	DoS Bot
Allaple-125	19	0feb6d3191d732ff2cad1039da5909ec	DoS Bot
Graftor.D3C30C	20	118f75d523892f21ab040a4679a7a199	Downloader
Upatre.a	21	7a1f26753d6e70076f15149feffbe233	Downloader
Kazy	22	ebefee9de7d429fe00593a1f6203cd6a	Downloader
Upatre.c	23	4d6c045c4cca49f8e556a7fb96e28635	Downloader
Agent.cuxe!c	24	2a4bd255bb4357c48e8f40c3869aad48	Downloader
ZeroAccess	25	a2611095f689fadffd3068e0d4e3e7ed	Downloader
Rozena.A	26	0dd87c12aa01f2d38f2b1b6ac236c907	Downloader
Ardamax	27	e33af9e602cbb7ac3634c2608150dd18	Spyware

O perfil de cada amostra foi determinado com base em análises manuais, estáticas e dinâmicas, e pesquisas na internet e na literatura, que foram realizadas pelo autor com o objetivo de detectar

comportamentos que representem o objetivo do malware.

Um exemplo de análise manual é a execução da amostra de malware TeslaCrypt.a, que após sua execução na máquina guest, exibe a tela vista na Figura 4.19 :

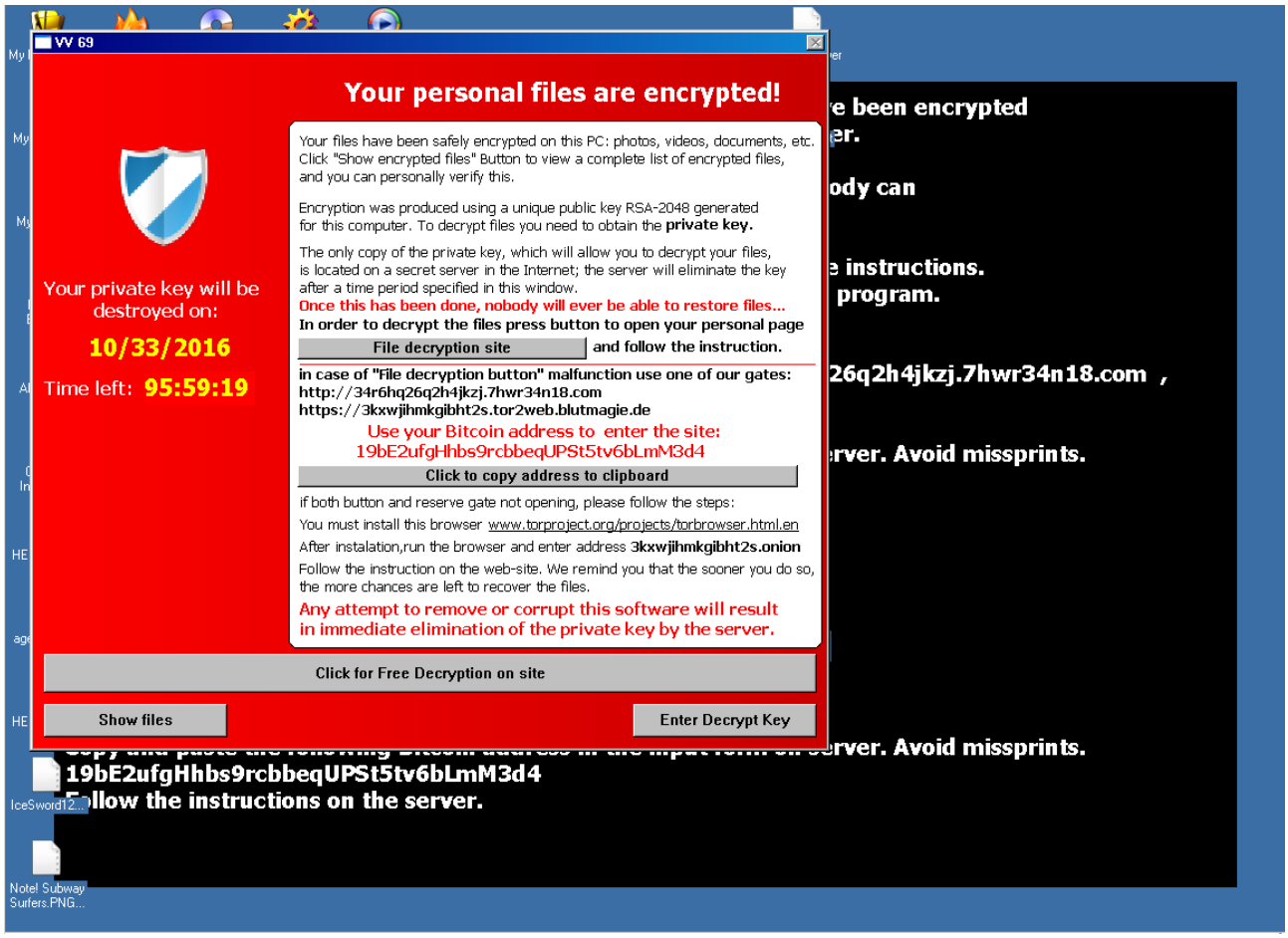


Figura 4.19: Tela exibida após a execução do malware TeslaCrypt.a

Na tela apresentada na Figura 4.19 é exibida uma mensagem que cobra um valor como forma de resgate para arquivos do usuário, encriptados, o que classifica a amostra TeslaCrypt.a no perfil Ramsomware,

Outro exemplo é a análise manual da amostra de malware Allaple-5. Através da análise manual do tráfego de rede realizado pela máquina guest durante a execução da amostra detectou-se que a máquina realizava o envio de um grande número de pacotes para diversos endereços IPs diferentes, apenas com a string “Babcdefghijklmnopqrstuvwabcdefghi” caracterizando um ataque de negação de serviço, sendo assim classificada como um DoS Bot. A Figura 4.20 mostra um extrato do dump de tráfego de rede, analisado utilizando a ferramenta WireShark (OREBAUGH; RAMIREZ; BEALE, 2006). Nessa figura, em seu quadrante superior, a coluna Time traz o tempo, contado em segundos a partir do início da captura dos pacotes, e a coluna Source traz o IP da máquina que deu

origem ao pacote. Nota-se que diversos pacotes são enviados a partir da máquina virtual que está executando o malware, com um intervalo de milésimos de segundo. Além disso, no quadrante inferior pode-se ver o conteúdo de um desses pacotes. A string “ Babcdefghijklmnopqrstuvwabcdefghi” é, na verdade, uma sequência de caracteres qualquer (lixo), o que mostra que a amostra de malware está enviando mensagens com o intuito de promover um ataque de negação de serviço.

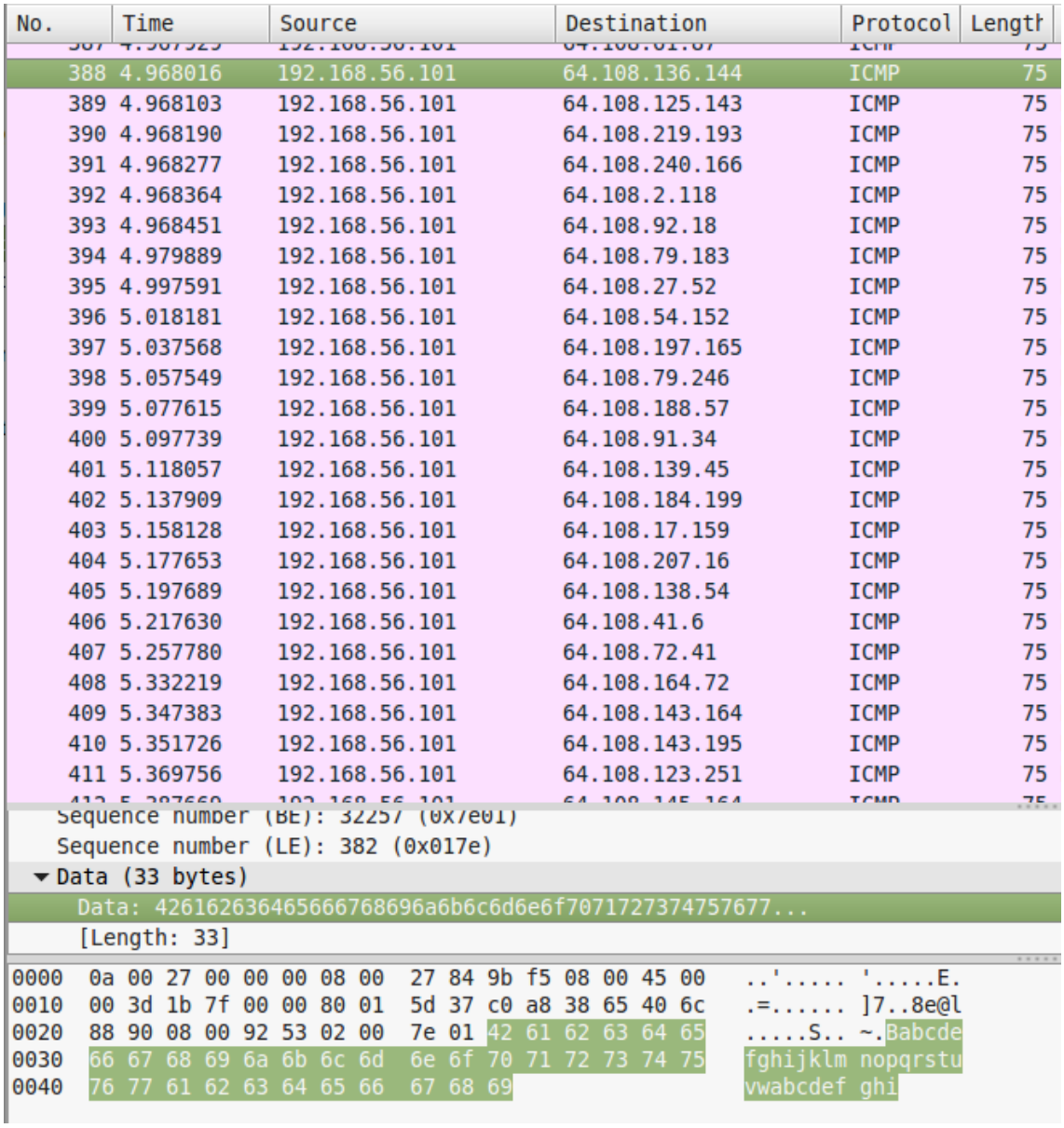


Figura 4.20: Análise de tráfego de rede durante a execução da amostra Allaple-5

As tabelas 4.4, 4.5, 4.6, 4.7, 4.8, 4.9, 4.10, 4.11, 4.12, 4.13, 4.14 e 4.15 trazem os resultados obtidos da extração das características selecionadas, separados por perfil. As características são nomeadas e apresentadas conforme definidas nas tabelas 3.1 e 3.2, presentes na seção 3.4 deste trabalho.

A característica Tamanho de arquivo (C1) é medida em bytes. A característica Arquivos acessados (C5) é apresentada em número de arquivos. A característica Porcentagem de pacotes de saída (C6) é apresentada em porcentagem de pacotes de saída, em relação ao total de pacotes. As demais características são apresentadas como um indicador binário, sendo que 1 significa que a amostra apresentou o comportamento observado, e 0 que a amostra não apresentou o comportamento.

4.5.1. Resultados para o perfil Banker

As tabelas 4.4 e 4.5 trazem os resultados obtidos da extração para o perfil Banker:

Tabela 4.4: Resultados da extração para amostras do perfil Banker (C1 a C5)

Banker	Tam Arq (C1)	APIsKeylg (C2)	APIs Rede (C3)	Seções Desc (C4)	Files MOD (C5)
1	96535	1	0	0	21
2	258560	0	0	0	38
3	1232896	1	0	4	12
4	8244224	1	0	4	28
5	750080	1	0	3	15
6	892416	1	1	0	44
7	8287232	1	0	4	28

Na Tabela 4.5 nota-se que não foi detectado, nas amostras de ID 1 e 3, o comportamento de instalação no Autorun. Isso não significa que as amostras não usam de técnicas para tornar sua execução automática, assim que é iniciado o sistema operacional. Significa apenas que elas não utilizaram as técnicas escolhidas para observação. Outras técnicas, como o uso de rootkits, podem ser utilizadas para esconder sua inicialização no sistema. Estudos detalhados de cada amostra devem ser realizados de forma a identificar qual técnica representa o comportamento adotado.

Tabela 4.5: Resultados da extração para amostras do perfil Banker (C6 a C11)

Banker	% Pact (C6)	AutoRun (C7)	Poly (C8)	Hosts Susp (C9)	Deleta Bin (C10)	BitCoin (C11)
1	53,85	0	0	1	0	0
2	48,35	1	0	0	0	0
3	50,00	0	0	1	0	0
4	50,00	1	0	0	0	0
5	87,50	1	0	1	0	0
6	82,82	1	0	0	0	0
7	45,45	1	0	0	0	0

4.5.2. Resultados para o perfil Ransomware

As tabelas 4.6 e 4.7 trazem os resultados obtidos da extração para o perfil Ransomware:

Tabela 4.6: Resultados da extração para amostras do perfil Ransomware (C1 a C5)

Ransomware	Tam Arq (C1)	APIsKeylg (C2)	APIs Rede (C3)	Seções Desc (C4)	Files MOD (C5)
8	290816	1	0	0	11378
9	263680	1	0	4	9663
10	50861	1	0	0	1443
11	53248	0	0	2	16
12	267278	1	0	4	9698
13	184320	0	0	0	20
14	246272	0	0	0	6
15	346112	1	0	0	11

Tabela 4.7: Resultados da extração para amostras do perfil Ransomware (C6 a C11)

Ransomware	% Pact (C6)	AutoRun (C7)	Poly (C8)	Hosts Susp (C9)	Deleta Bin (C10)	BitCoin (C11)
8	50,96	1	0	0	0	0
9	51,69	1	0	0	0	0
10	51,85	1	0	1	0	0
11	78,79	1	0	1	0	0
12	51,11	1	0	0	0	0
13	81,67	0	0	1	0	0
14	80,00	0	0	1	0	0
15	53,81	1	0	0	1	0

4.5.3. Resultados para o perfil DoS Bot

As tabelas 4.8 e 4.9, trazem os resultados obtidos da extração para o perfil DoS Bot:

Tabela 4.8: Resultados da extração para amostras do perfil DoS Bot (C1 a C5)

DoS Bot	Tam Arq (C1)	APIsKeylg (C2)	APIs Rede (C3)	Seções Desc (C4)	Files MOD (C5)
16	57344	0	0	1	9
17	57856	0	0	1	25
18	67584	0	0	1	25
19	85504	0	0	2	928

Tabela 4.9: Resultados da extração para amostras do perfil DoS Bot (C6 a C11)

DoS Bot	% Pact (C6)	AutoRun (C7)	Poly (C8)	Hosts Susp (C9)	Deleta Bin (C10)	BitCoin (C11)
16	98,66	0	0	0	0	0
17	100	0	0	0	1	0
18	77,94	0	0	0	1	0
19	90,49	0	0	0	0	0

4.5.4. Resultados para o perfil Downloader

As tabelas 4.10 e 4.11 trazem os resultados obtidos da extração para o perfil Downloader:

Tabela 4.10: Resultados da extração para amostras do perfil Downloader (C1 a C5)

Downloader	Tam Arq (C1)	APIsKeylg (C2)	APIs Rede (C3)	Seções Desc (C4)	Files MOD (C5)
20	32768	1	1	0	1791
21	28160	0	0	0	23
22	56224	0	0	2	43
23	49152	0	0	3	24
24	94208	0	0	0	24
25	163840	1	1	0	54

Tabela 4.11: Resultados da extração para amostras do perfil Downloader (C6 a C11)

Downloader	% Pact (C6)	AutoRun (C7)	Poly (C8)	Hosts Susp (C9)	Deleta Bin (C10)	BitCoin (C11)
20	42,63	0	0	1	0	0
21	55,97	0	0	1	1	0
22	20	0	0	0	0	0
23	51,69	0	0	1	0	0
24	99,9	1	0	1	0	0
25	92,41	0	0	0	0	0

4.5.5. Resultados para o perfil Backdoor

As tabelas 4.12 e 4.13 trazem os resultados obtidos da extração para o perfil Backdoor:

Tabela 4.12: Resultados da extração para amostras do perfil Backdoor (C1 a C5)

Backdoor	Tam Arq (C1)	APIsKeylg (C2)	APIs Rede (C3)	Seções Desc (C4)	Files MOD (C5)
26	73802	0	1		0

Tabela 4.13: Resultados da extração para amostras do perfil Backdoor (C6 a C11)

Backdoor	% Pact (C6)	AutoRun (C7)	Poly (C8)	Hosts Susp (C9)	Deleta Bin (C10)	BitCoin (C11)
26	0	0	0	0	0	0

4.5.6. Resultados para o perfil Spyware

As tabelas 4.14 e 4.15 trazem os resultados obtidos da extração para o perfil Spyware:

Tabela 4.14: Resultados da extração para amostras do perfil Spyware (C1 a C5)

Spyware	Tam Arq (C1)	APIsKeylg (C2)	APIs Rede (C3)	Seções Desc (C4)	Files MOD (C5)
27	802724	1	0	0	157

Tabela 4.15: Resultados da extração para amostras do perfil Spyware (C6 a C11)

Spyware	% Pact (C6)	AutoRun (C7)	Poly (C8)	Hosts Susp (C9)	Deleta Bin (C10)	BitCoin (C11)
27	50,48	1	0	1	0	0

4.5.7. Síntese dos resultados

Um sumário com os resultados organizados por perfil pode ser encontrado na Tabela 4.16. As características C1, C4 C5 e C6 são apresentadas como média aritmética dos resultados obtidos das amostras de cada perfil. As outras características são mostradas como porcentagem da frequência em que aparecem na população de cada perfil.

Tabela 4.16: Sumário dos resultados, separados por perfil

Perfil	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11
Banker	2823135	85,71	14,29	2	27	59,71	71,43	0	42,86	0	0
Ransomware	212823	62,50	0	1	4029	62,49	75,00	0	50,00	12,50	0
DoS Bot	67072	0	0	1	247	91,77	0	0	0	50,00	0
Downloader	70725	33,33	33,33	1	327	60,43	16,67	0	66,67	16,67	0
Backdoor	73802	0	100,00	0	0	0	0	0	0	0	0
Spyware	802724	100,00	0	0	157	50,48	100,00	0	100,00	0	0

Analisando os resultados da extração, algumas considerações podem ser feitas. As amostras do perfil Banker apresentaram um tamanho de arquivo (C1) significativamente maior do que o apresentado pelas outras amostras. Além disso, 85% dessas amostras fizeram uso das APIs de keylogging rastreadas (C2), o que mostra que essas características podem ser interessantes para classificar amostras nesse perfil. Essas conclusões podem ser observadas nas Figuras 4.21 e4.22.

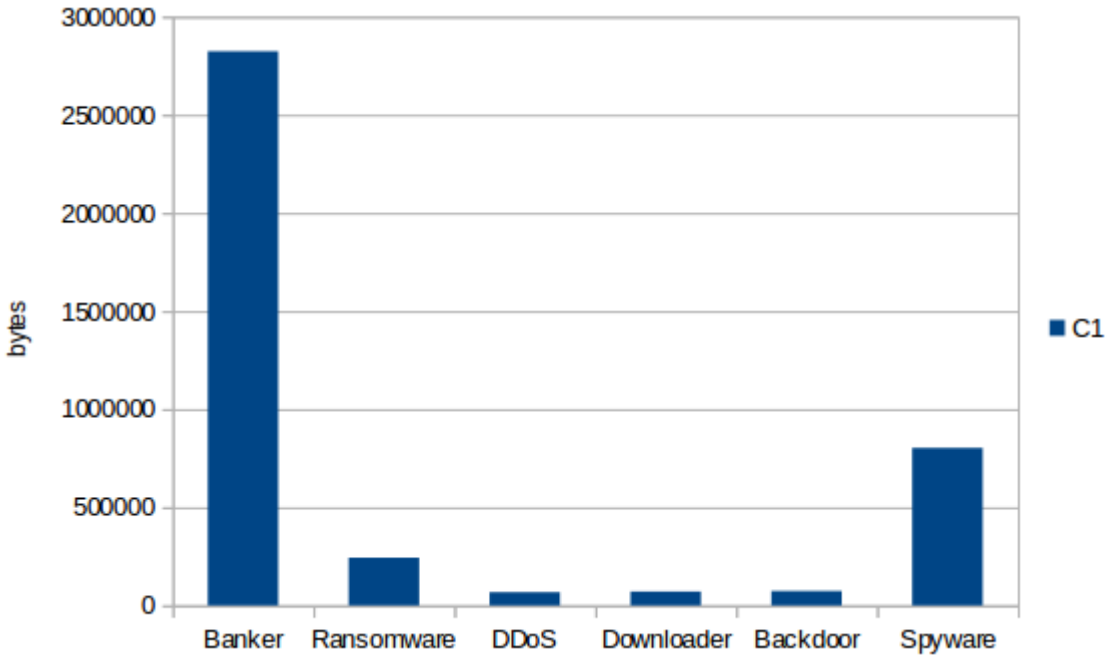


Figura 4.21: Gráfico do tamanho de arquivo por perfil

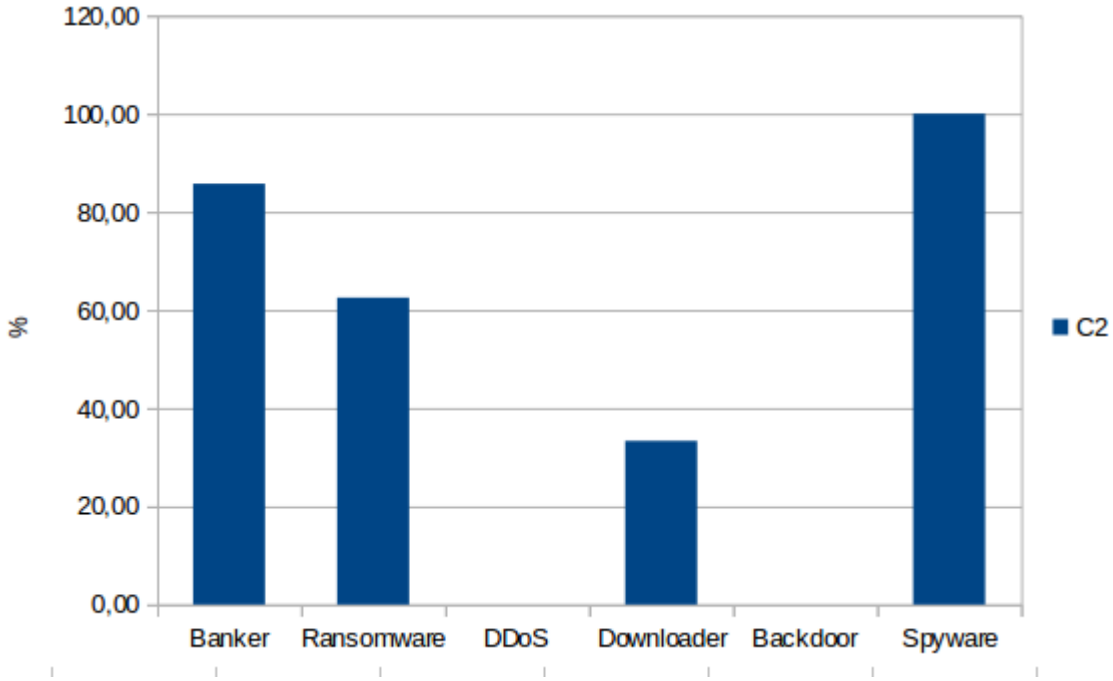


Figura 4.22: Gráfico das APIs de keylogging por perfil

Somente as amostra dos perfis Backdoor, Downloader e Banker utilizaram as APIs de rede (C3) escolhidas para observação.

Como esperado, as amostras do perfil DoS Bot fizeram um uso abusivo da rede, enviando muito mais pacotes (C6) do que receberam. A média de procentagem de pacotes de rede enviados ficou acima de 90% para o perfil DoS Bot. Esse comportamento pode ser melhor observado na Figura 4.23.

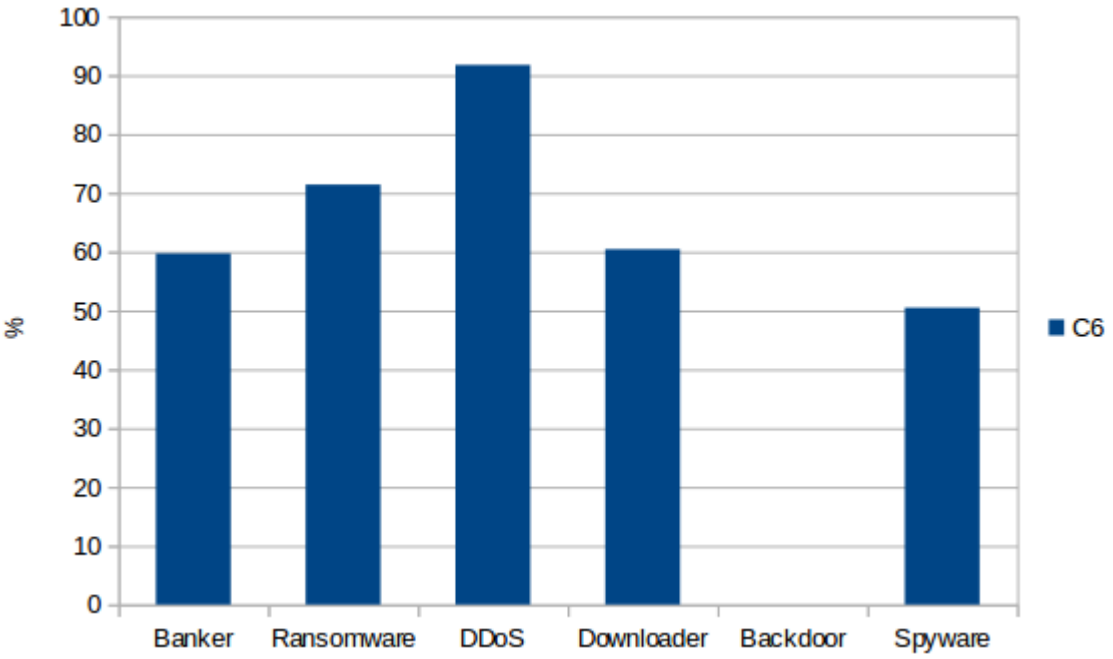


Figura 4.23: Gráfico da porcentagem de pacotes de saída por perfil

As amostras do perfil Ransomware apresentarem um número de arquivos acessados (C5) expressivamente maior que as amostras dos demais perfil, acredita-se que essa característica reflita bem funcionalidades implementadas nos malwares desse perfil, sendo assim uma característica promissora para a classificação.

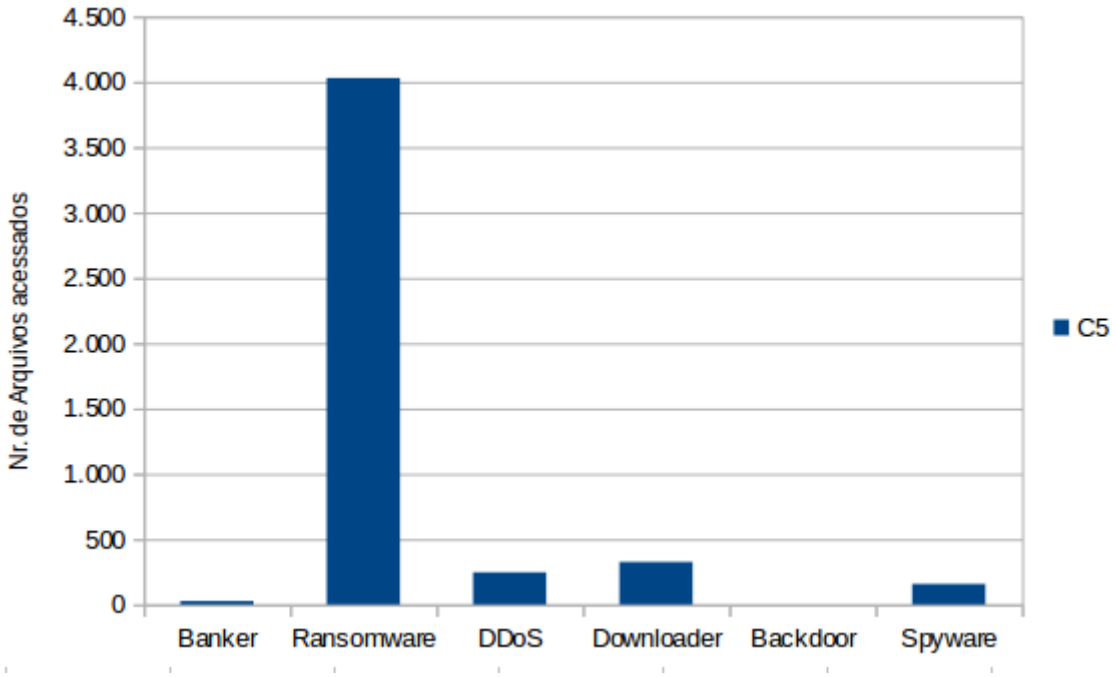


Figura 4.24: Gráfico do número de arquivos acessados por perfil

Apesar de ser um comportamento característico de Ransomware, na Tabela 4.10 nota-se que a amostra de ID 20 realiza um número de acesso a arquivos muito alto. Isso se dá devido à características estruturais da amostra, que infecta diversos arquivos no sistema operacional, e não significa que ela se trata de um Ransomware.

Poucas amostras do perfil Downloader, e nenhuma amostra do perfil DoS Bot apresentou o comportamento instalação no Autorun (C7).

As amostras do perfil DoS Bot não realizaram conexão com hosts suspeitos (C9), o que significa que a ausência desse comportamento pode ser relevante para esse perfil.

As características que representavam os comportamentos de polimorfismo (C8) e acesso a arquivos de bitcoin (C11) não foram observadas em nenhuma das amostras analisadas.

As características Número de seções desconhecidas (C4) e Deleção de arquivos binários (C10) não apresentaram resultados promissores.

Apesar de detectar algumas características promissoras, esses resultados precisam ser confirmados com a extração das características de um número maior de amostras. Poucas amostras para cada perfil foram executadas. Os perfis Spyware e Backdoor tiveram apenas uma amostra analisada com sucesso.

5. CONCLUSÃO

Neste trabalho foi apresentada a problemática da classificação de malware, e um panorama da área, na indústria e na literatura. Foi demonstrada a necessidade de avanços, delineando-se fragilidades e limitações das abordagens conhecidas, e proposta uma nova abordagem de classificação de malware, a definição de perfis de malware.

A definição de perfis de malware apresentada no trabalho baseia suas ideias no profiling criminal. A abordagem de classificação proposta utiliza uma definição de classes baseada no objetivo à ser alcançado pela amostra de malware classificada, e pode ser muito útil para agilizar o trabalho do investigador forense. Um estudo da literatura sugeriu ainda características, estruturais e comportamentais, a serem extraídas e estudadas, que mostraram relevância para classificação nessa abordagem.

Foi apresentada a ferramenta Cuckoo Sandbox, que permite a criação de scripts Python personalizados, para executar amostras de malware em máquinas virtuais e físicas, e extrair informações acerca do comportamento dessas amostras. Foi desenvolvida uma estratégia para extração das características selecionadas, baseada em scripts desenvolvidos pelos autores e fornecidos em conjunto com a ferramenta, para extração. O funcionamento de cada script, em cada característica, foi explicado.

A estratégia de extração se mostrou eficiente, e vinte e sete amostras foram executadas individualmente conforme a metodologia apresentada, e tiveram suas características extraídas.

Isso mostra que a estratégia desenvolvida pode ser utilizada com sucesso em um processo de análise automática de malware, trazendo assim uma contribuição positiva na área de análise de malware automática e classificação.

Das onze características selecionadas para extração, sete se mostraram promissoras para sofrerem maiores estudos e serem usadas em um processo de definição de perfis automatizado.

Acredita-se que as características promissoras representam alguns comportamentos adotados pelos perfis de malware para alcançarem os objetivos. Além disso os resultados das características que não se mostraram promissoras são úteis pois indicam que não se deve gastar mais tempo analisando tais características no processo de classificação de malware. Em outras palavras, esses resultados podem ser usados num próximo passo, para refinar o rol de características.

5.1. TRABALHOS FUTUROS

Como trabalhos futuros espera-se utilizar a estratégia desenvolvida em um processo de classificação automatizado de malware. Esse processo deve visar a classificação de uma amostra de malware entre um dos perfis elencados na lista de perifs apresentada.

Com esse resultado consegue-se assim uma análise automatizada e classificação de malware, criando uma ferramenta útil para tornar mais eficiente o trabalho do investigador forense.

Limitações da abordagem apresentada incluem o número limitado de amostras analisadas, e a ausência de um processo de classificação automatizado, baseado nas características extraídas.

Possíveis trabalhos futuros:

- Aumentar o número de amostras analisadas.
- Aumentar e refinar o rol de características.
- Utilizar fuzzy hashes como característica estrutural.
- Desenvolvimento de ferramentas para classificação automatizadas.

5.2. RESULTADOS ACADÊMICOS

Algumas dessas contribuições foram aceitas para publicação em congressos científicos, como o caso do artigo Análise e extração de características comportamentais e estruturais para profiling de malware, aceito para publicação na *14ª Conferência Ibero Americana WWW/Internet 2016* e do artigo Analysis and Extraction of Behavioral and Static Features for Malware Profiling, aceito para publicação no *II Congresso Brasileiro de Jovens Pesquisadores em Matemática Pura e Aplicada.*

REFERÊNCIAS BIBLIOGRÁFICAS

ADLEMAN, M. **An abstract theory of computer viruses**. . In: ADVANCES IN CRYPTO. 1998

AHMADI, M. et al. **Novel feature extraction, selection and fusion for effective malware family classification**. Proceedings. **Anais**... In: SIXTH ACM CONFERENCE ON DATA AND APPLICATION SECURITY AND PRIVACY. New Orleans, USA: 2016

AYCOCK, J. **Computer viruses and malware**. [s.l.] Springer Science & Business Media, 2006. v. 22

BELLARD, F. **QEMU, a fast and portable dynamic translator.** FREENIX Track. **Anais**... In: USENIX ANNUAL TECHNICAL CONFERENCE. abr. 2005

BORGATTI, S.; EVERETT, M.; FREEMAN, L. Ucinet for Windows: Software for social network analysis. 2002.

BRASIL. Constituição da República Federativa do Brasil. . 1988.

BRASIL, S. F. Projeto de Lei do Senado nº 236, de 2012. Institui novo Código Penal. . 2012.

BROADHURST, R. et al. An Analysis of the Nature of Groups Engaged in Cyber Crime. **International Journal of Cyber Criminology**, p. 1–20, fev. 2014.

BUREAU, P. **Same botnet, same guys, new code**. . In: VIRUS BULLETIN INTERNATIONAL CONFERENCE. Barcelona, Spain: 2011

BUSINESS INSIDER. **The Stuxnet Attack On Iran's Nuclear Plant Was "Far More Dangerous" Than Previously Thought**. Disponível em: <http://www.businessinsider.com/stuxnet-was-far-more-dangerous-than-previous-thought-2013-11>. Acesso em: 23 nov. 2016.

CAO, X.; LU, Y. Social Network Analysis of a Criminal Hacker Community. **Security and Privacy Assurance in Advancing Tech-nologies: New Developments**, p. 160–173, 2011.

CARO. **The CARO Website**. Disponível em: <http://www.caro.org/index.html>. Acesso em: 23 nov. 2016.

CERT.BR. **Códigos maliciosos (Malware)**. Disponível em: <http://cartilha.cert.br/malware/>. Acesso em: 23 nov. 2016.

CESARE, S.; XIANG, Y.; ZHOU, W. Control Flow-based Malware Variant Detection. **IEEE Transactions on Dependable and Secure Computing**, p. 307–317, set. 2013.

CHO, I. et al. Malware similarity analysis using API sequence alignments. **Journal of Internet Services and Information Security**, 2014.

COHEN, F. Computer viruses: theory and experiments. **Computers & security**, p. 22–35, 1987.

CUCKOO FOUNDATION. **Cuckoo Sandbox**. Disponível em: <https://www.cuckoosandbox.org/>. Acesso em: 23 nov. 2016.

DALZIEL, H. **The Four Amigos: Stuxnet, Flame, Gauss, and DuQu.** Disponível em: <https://www.concise-courses.com/stuxnet-flame-gauss-duqu/>. Acesso em: 23 nov. 2016.

DIGITALNINJA. **Fuzzy Clarity - Using Fuzzy Hashing Techniques to Identify Malicious Code**. Disponível em: <http://www.shadowserver.org/wiki/uploads/Information/FuzzyHashing.pdf>. Acesso em: 13 dez. 2016.

DUMMIES. **Know the different types of malware**. Disponível em: <http://www.dummies.com/computers/pcs/know-the-different-types-of-malware/>. Acesso em: 23 nov. 2016.

EASTLAKE 3RD, D.; JONES, P. **US secure hash algorithm 1 (SHA1). (No. RFC 3174).** Disponível em: <https://www.rfc-editor.org/info/rfc3174>. Acesso em: 23 nov. 2016.

ELIASON, S. Policing the Poachers in the United States: A Qualitative Analysis of Game Wardens and Profiling. **International Journal of Criminal Justice Sciences**, p. 235–247, dez. 2013.

ESET. **Viruses, malware and remote attacks defined**. Disponível em: <http://support.eset.com/kb186/?locale=en_US>. Acesso em: 23 nov. 2016.

FILIOL, E. Viruses and malware. In: **Handbook of Information and Communication Security**. [s.l.] Springer Berlin Heidelberg, 2010. p. 747–769.

F-SECURE. **Classification - How F-Secure classifies threats**. Disponível em: <https://www.f-secure.com/en/web/labs_global/classification>. Acesso em: 23 nov. 2016.

GOLDBERG, I. et al. **A secure environment for untrusted helper applications: Confining the wily hacker**. Proceedings. **Anais**... In: 6TH CONFERENCE ON USENIX SECURITY SYMPOSIUM, FOCUSING ON APPLICATIONS OF CRYPTOGRAPHY. jul. 1996

GOLDEN, B. **Virtualization For Dummies**. [s.l.] John Wiley & Sons, 2011.

HEX-RAYS. **IDA: About**. Disponível em: <https://www.hex-rays.com/products/ida/>. Acesso em: 23 nov. 2016.

HOLMES, R.; HOLMES, S. **Profiling violent crimes: An investigative tool**. [s.l.] Sage publications., 2008.

HUANG, H.-D. et al. **Applying FML and Fuzzy Ontologies to Malware Behavioural Analysis**. . In: IEEE INTERNATIONAL CONFERENCE ON FUZZY SYSTEMS. Taiwan: jun. 2011

ISECLAB. **Anubis: Analyzing Unknown Binaries**. Disponível em: <http://analysis.iseclab.org/>. Acesso em: 23 nov. 2016.

JANG, J. et al. **Mal-netminer: malware classification based on social network analysis of call graph**. Proceedings. **Anais**... In: 23RD INTERNATIONAL CONFERENCE ON WORLD WIDE WEB. Seoul, South Korea: 2014

JSON. **Introducing JSON**. Disponível em: <http://www.json.org/>. Acesso em: 23 nov. 2016.

KASPERSKY. **Safety 101: Types of known threats**. Disponível em: <https://support.kaspersky.com/viruses/general/614>. Acesso em: 23 nov. 2016.

LEE, C. Criminal profiling and industrial security. **Multimedia Tools and Applications**, p. 1689–1696, 2015.

LU, Y. et al. Using multi-feature and classifier ensembles to improve malware detection. **Journal of CCIT**, p. 57–72, 2010.

MARCIANO, J. L. **Segurança da Informação - uma abordagem social**. Tese de Doutorado em Ciência da Informação—Brasília: Universidade de Brasília, 2006.

MCAFEE LABS. **Relatório do McAfee Labs sobre ameaças**. [s.l: s.n.]. Disponível em: <http://www.mcafee.com/br/resources/reports/rp-quarterly-threat-q4-2014.pdf>. Acesso em: 27 nov. 2016.

MICROSOFT. **Sony, Rootkits and Digital Rights Management Gone Too Far**. Disponível em: <https://blogs.technet.microsoft.com/markrussinovich/2005/10/31/sony-rootkits-and-digital-rights-management-gone-too-far/>. Acesso em: 23 nov. 2016.

MICROSOFT. **Binding a Socket**. Disponível em: <https://msdn.microsoft.com/en-us/library/windows/desktop/ms737548(v=vs.85).aspx>. Acesso em: 23 nov. 2016.

MONGODB. **MongoDB**. Disponível em: <https://www.mongodb.com/>. Acesso em: 23 nov. 2016.

NATIV, Y. **theZoo Malware DB**. Disponível em: <http://thezoo.morirt.com/>. Acesso em: 23 nov. 2016.

OREBAUGH, A.; RAMIREZ, G.; BEALE, J. **Wireshark & Ethereal network protocol analyzer toolkit**. [s.l.] Syngress, 2006.

PANDA SECURITY. **Types of Malware**. Disponível em: <http://www.pandasecurity.com/homeusers/security-info/types-malware/>. Acesso em: 23 nov. 2016.

PYTHON SOFTWARE FOUNDATION. **About Python**. Disponível em: <https://www.python.org/about/>. Acesso em: 23 nov. 2016.

RAD, B.; MASROM, M.; IBRAHIM, S. Camouflage in malware: from encryption to metamorphism. **International Journal of Computer Science and Network Security**, p. 74–83, 2012.

RIECK, K. et al. **Learning and classification of malware behavior**. . In: INTERNATIONAL CONFERENCE ON DETECTION OF INTRUSIONS AND MALWARE, AND VULNERABILITY ASSESSMENT. Springer Berlin Heidelberg., jul. 2008

SECLAB. **A Survey on Automated Dynamic Malware Analysis Techniques and Tools**. Disponível em: <http://www.seclab.tuwien.ac.at/papers/malware_survey.pdf>. Acesso em: 23 nov. 2016.

SIKORSKI, M.; HONIG, A. **Practical Malware Analysis**. [s.l.] No Starch Press, 2012.

SMITH, J.; NAIR, R. The Architecture of Virtual Machines. **IEEE Computer Society**, p. 32–38, 2005.

SMUTZ, C.; STAVROU, A. **Malicious PDF Detection using Metadata and Structural**. . In: 28TH ANNUAL COMPUTER SECURITY APPLICATIONS CONFERENCE. Orlando, USA: dez. 2012

STALLINS, W. **Arquitetura e Organização de Computadores**. [s.l.] Makron Books, 2002.

SYMANTEC. **Flamer: Highly Sophisticated and Discreet Threat Targets the Middle East**. Disponível em: <https://www.symantec.com/connect/blogs/flamer-highly-sophisticated-and-discreet-threat-targets-middle-east>. Acesso em: 23 nov. 2016.

SYMANTEC. **Internet Security Threat Report**. [s.l: s.n.]. Disponível em: <https://www.symantec.com/security-center/threat-report>.

THE MALSHARE PROJECT. **MalShare**. Disponível em: <http://malshare.com/>. Acesso em: 23 nov. 2016.

TURVEY, B. **Criminal profiling: An introduction to behavioral evidence analysis**. [s.l.] Academic press, 2011.

VERACODE. **Common Malware Types: Cybersecurity 101**. Disponível em: <https://www.veracode.com/blog/2012/10/common-malware-types-cybersecurity-101>. Acesso em: 23 nov. 2016.

VIRUS TOTAL. **Análise de amostra no website Virus Total**. Disponível em: <https://www.virustotal.com/>. Acesso em: 23 nov. 2016.

WILLEMS, C.; HOLZ, T.; FREILING, F. Toward Automated Dynamic Malware Analysis Using CWSandbox. **IEEE Security and Privacy**, p. 32–39, mar. 2007.

YEE, B. et al. **Native Client: A Sandbox for Portable, Untrusted x86 Native Code**. . In: IEEE SYMPOSIUM ON SECURITY AND PRIVACY. 2009

YOU, I.; YIM, K. **Malware Obfuscation Techniques: A Brief Survey**. . In: BWCCA: BROADBAND AND WIRELESS COMPUTING, COMMUNICATION AND APPLICATIONS. Fukuoka, Japan: nov. 2010

YUSCHUK, O. **OllyDbg**. Disponível em: <http://ollydbg.de/>. Acesso em: 23 nov. 2016.

ANEXOS

A – CÓDIGO FONTE DESENVOLVIDO NESTE TRABALHO

caldasdump.py

```
from lib.cuckoo.common.abstracts import Report
class CaldasDump(Report):
    results_path = "/home/caldaz/workspace/cuckoo modules/results"
    def run(self, results):
        """Writes My Results on file.
        @param results: Cuckoo results dict."""
        file_results = open(self.results_path, 'a')
        known_sections = (".text", ".rdata", ".data", ".idata",
".edata", ".pdata", ".rsrc", ".reloc")
        unknown_sections = 0
        signature_vector = [0, 0, 0, 0, 0, 0, 0, 0, 0]

        report = dict(results)
        # Analyses ID
        file_results.write(str(report["info"]["id"]))
        file_results.write('\t')
        # File name and size
        if report["target"]:
            file_results.write(report["target"]["file"]["name"])
            file_results.write('\t')
            file_results.write(str(report["target"]["file"]
["size"]))
            file_results.write('\t')
        # Signatures
        for signature in report["signatures"]:
            # AutoRun
            if signature["name"] == "persistence_autorun":
                signature_vector[0] = 1
            # APIs de rede (bind, listen, accept)
            if signature["name"] == "network_bind":
                signature_vector[1] = 1
            # Files MOD
            if signature["name"] == "many_files":
                signature_vector[2] = signature["marks"][0]
["nfiles"]
            # Calls or imports Keylogging APIs
            if signature["name"] == "keylogging_api":
                signature_vector[3] = 1
            # Creates a slightly different version of itself
            if signature["name"] == "packer_polymorphic":
                signature_vector[4] = 1
            # Try to access bitcoin wallets
            if signature["name"] == "infostealer_bitcoin":
                signature_vector[5] = 1
            # Connects to dead hosts
            if signature["name"] == "dead_host":
                signature_vector[6] = 1
```

```
            # Creates Exe files on the filesystem
            if signature["name"] == "creates_exe":
                signature_vector[7] = 1
            # Deletes original binary
            if signature["name"] == "deletes_self":
                signature_vector[8] = 1
        for value in signature_vector:
            file_results.write(str(value))
            file_results.write('\t')
        # out/in % (packages)
        if report["caldasnetwork"]["srcself"]:
            percent_self = float(report["caldasnetwork"]
["srcself"] * 100) \
                           / report["caldasnetwork"]["totalpckgs"]
            print(percent_self)
            print("{:.2f}".format(percent_self))
            file_results.write("{:.2f}".format(percent_self))
            file_results.write("\t")
        # Number of unknown sections
        for pe_section in report["static"]["pe_sections"]:
            if pe_section["name"] not in known_sections:
                unknown_sections += 1
        file_results.write(str(unknown_sections))
        file_results.write("\t")
        file_results.write("\n")
```

caldasprocessing.py

```
import hashlib
import logging
import json
import os
import re
import socket
import struct
import tempfile
import urlparse
from lib.cuckoo.common.abstracts import Processing
from lib.cuckoo.common.config import Config
from lib.cuckoo.common.dns import resolve
from lib.cuckoo.common.irc import ircMessage
from lib.cuckoo.common.objects import File
from lib.cuckoo.common.exceptions import CuckooProcessingError
try:
    import dpkt
    HAVE_DPKT = True
except ImportError:
    HAVE_DPKT = False
try:
    import httpreplay
    import httpreplay.cut
```

```
    # Be less verbose about httpreplay logging messages.
    logging.getLogger("httpreplay").setLevel(logging.CRITICAL)
    HAVE_HTTPREPLAY = True
except ImportError:
    HAVE_HTTPREPLAY = False

import heapq
from itertools import islice
from collections import namedtuple
Keyed = namedtuple("Keyed", ["key", "obj"])
Packet = namedtuple("Packet", ["raw", "ts"])
log = logging.getLogger(__name__)
cfg = Config()

_v = getattr(httpreplay, "__version__", None) if HAVE_HTTPREPLAY
else None

try:
    import dpkt
    HAVE_DPKT = True
except ImportError:
    HAVE_DPKT = False
import heapq
from itertools import islice
from collections import namedtuple
Keyed = namedtuple("Keyed", ["key", "obj"])
Packet = namedtuple("Packet", ["raw", "ts"])
log = logging.getLogger(__name__)
cfg = Config()
class Pcap(object):
    """Reads network data from PCAP file."""
    ssl_ports = 443,
    self_ip = "192.168.56.101"
    notified_dpkt = False
    def __init__(self, filepath):
        """Creates a new instance.
        @param filepath: path to PCAP file
        """
        self.filepath = filepath
        # List of all packages.
        self.packages = []
        # List of all IPs in packages.
        self.ips = []
        # Dictionary containing all the results of this
processing.
        self.results = {}
    def run(self):
        """Process PCAP.
        @return: dict with network analysis data.
        """
        try:
```

```
            file = open(self.filepath, "rb")
        except (IOError, OSError):
            log.error("Unable to open %s" % self.filepath)
            return self.results
        try:
            pcap = dpkt.pcap.Reader(file)
        except dpkt.dpkt.NeedData:
            log.error("Unable to read PCAP file at path \"%s\".",
                      self.filepath)
            return self.results
        except ValueError:
            log.error("Unable to read PCAP file at path \"%s\".
File is "
                      "corrupted or wrong format." %
self.filepath)
            return self.results
        offset = file.tell()
        first_ts = None
        srcself = 0
        totalpckgs = 0
        for ts, buf in pcap:
            if not first_ts:
                first_ts = ts
            try:
                ip = iplayer_from_raw(buf, pcap.datalink())
                package = {}
                if isinstance(ip, dpkt.ip.IP):
                    package["src"] = socket.inet_ntoa(ip.src)
                    package["dst"] = socket.inet_ntoa(ip.dst)
                elif isinstance(ip, dpkt.ip6.IP6):
                    package["src"] =
socket.inet_ntop(socket.AF_INET6,
                                                     ip.src)
                    package["dst"] =
socket.inet_ntop(socket.AF_INET6,
                                                     ip.dst)
                else:
                    offset = file.tell()
                    continue
                if package["src"] == self.self_ip:
                    srcself += 1
                totalpckgs += 1
                self.packages.append(package)
                offset = file.tell()
            except AttributeError:
                continue
            except dpkt.dpkt.NeedData:
                continue
            except Exception as e:
                log.exception("Failed to process packet: %s", e)
        self.results["packages"] = self.packages
```

```
        self.results["srcself"] = srcself
        self.results["totalpckgs"] = totalpckgs
        file.close()
        return self.results
class CaldasNetworkAnalysis(Processing):
    """Network analysis."""
    order = 2
    key = "caldasnetwork"
    def run(self):
        results = {}
        if not os.path.exists(self.pcap_path):
            log.warning("The PCAP file does not exist at
path \"%s\".",
                        self.pcap_path)
            return results
        if os.path.getsize(self.pcap_path) == 0:
            log.error("The PCAP file at path \"%s\" is empty." %
self.pcap_path)
            return results
        if not HAVE_DPKT and not HAVE_HTTPREPLAY:
            log.error("Both Python HTTPReplay and Python DPKT are
not "
                      "installed, no PCAP analysis possible.")
            return results
        pcap_path = self.pcap_path
        if HAVE_DPKT:
            results.update(Pcap(pcap_path).run())
        return results
def iplayer_from_raw(raw, linktype=1):

    if linktype == 1:  # ethernet
        try:
            pkt = dpkt.ethernet.Ethernet(raw)
            return pkt.data
        except dpkt.NeedData:
            pass
    elif linktype == 101:  # raw
        return dpkt.ip.IP(raw)
    else:
        raise CuckooProcessingError("unknown PCAP linktype")
```

manyfiles.py

```
# Copyright (C) 2016 Caldas.
from lib.cuckoo.common.abstracts import Signature
class AccessManyFiles(Signature):
    name = "many_files"
    description = "This malware access %d files"
    severity = 2
    categories = ["ransom"]
    authors = ["caldas"]
```

```
    minimum = "2.0"
    nfiles = 0
    def on_complete(self):
        self.nfiles = sum(1 for _ in self.get_files())
        self.description = self.description % self.nfiles
        self.mark(nfiles=self.nfiles)
        return True
```

keylogsAPIs.py

```
#
# Copyright (C) 2016 Caldas.
#
from lib.cuckoo.common.abstracts import Signature
class KeyloggingAPI(Signature):
    name = "keylogging_api"
    description = "This program calls or imports known keylogging
APIs"
    severity = 2
    categories = ["keylogger"]
    authors = ["caldas"]
    minimum = "2.0"
    api_names = ("SetWindowsHookEx", "SetWindowsHookExW",
"UnhookWindowsHookEx", \
                        "UnhookWindowsHookExW", "GetAsyncKeyState",
"GetAsyncKeyStateW", \
                        "GetForegroundWindow",
"GetForegroundWindowW")
    filter_apinames = "SetWindowsHookEx", "SetWindowsHookExW",
"UnhookWindowsHookEx", \
                        "UnhookWindowsHookExW", "GetAsyncKeyState",
"GetAsyncKeyStateW", \
                        "GetForegroundWindow",
"GetForegroundWindowW"
    def init(self):
        self.mask = 0
    def on_call(self, call, process):
        if call["api"] == "SetWindowsHookEx":
            self.mark_call()
            self.mask = 1
        if call["api"] == "SetWindowsHookExW":
            self.mark_call()
            self.mask = 1
        if call["api"] == "UnhookWindowsHookEx":
            self.mark_call()
            self.mask = 1
        if call["api"] == "UnhookWindowsHookExW":
            self.mark_call()
            self.mask = 1
        if call["api"] == "GetAsyncKeyState":
            self.mark_call()
```

```
            self.mask = 1
        if call["api"] == "GetAsyncKeyStateW":
            self.mark_call()
            self.mask = 1
        if call["api"] == "GetForegroundWindow":
            self.mark_call()
            self.mask = 1
        if call["api"] == "GetForegroundWindowW":
            self.mark_call()
            self.mask = 1
    def on_complete(self):
        report = self.get_results("static", {})
        for pe_import in report["pe_imports"]:
            for importedAPI in pe_import["imports"]:
                if importedAPI["name"] in ("SetWindowsHookEx",
"SetWindowsHookExW", "UnhookWindowsHookEx", \
                        "UnhookWindowsHookExW", "GetAsyncKeyState",
"GetAsyncKeyStateW", \
                        "GetForegroundWindow",
"GetForegroundWindowW"):
                    self.mark(name=importedAPI["name"])
                    self.mask = 1
        return self.mask
```

caldasbind.py

```
from lib.cuckoo.common.abstracts import Signature
class NetworkBIND(Signature):
    name = "network_bind"
    description = "Starts servers listening"
    severity = 2
    categories = ["bind"]
    authors = ["caldas"]
    minimum = "2.0"
    filter_apinames = "bind", "listen", "accept"
    def init(self):
        self.mask = 0
    def on_call(self, call, process):
        if call["api"] == "bind":
            self.mark_call()
            self.mask |= 1
        if call["api"] == "listen":
            self.mark_call()
            self.mask |= 2
        if call["api"] == "accept":
            self.mark_call()
            self.mask |= 4
    def on_complete(self):
        return self.mask >= 3
```