JOÃO MARCELO CERON

**MARS: uma arquitetura para análise de malwares
utilizando SDN**

São Paulo
2018

JOÃO MARCELO CERON

**MARS: uma arquitetura para análise de malwares
utilizando SDN**

Tese apresentada à Escola Politécnica da
Universidade de São Paulo para obtenção
do título de Doutor em Ciências.

São Paulo
2018

JOÃO MARCELO CERON

**MARS: uma arquitetura para análise de malwares
utilizando SDN**

Tese apresentada à Escola Politécnica da
Universidade de São Paulo para obtenção
do título de Doutor em Ciências.

Área de concentração:
Engenharia da Computação

Orientador:
Profa. Dra. Cíntia Borges Margi

Co-Orientador:
Prof. Dr. Lisandro Zambenedetti Granville

São Paulo
2018

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, _____ de _____ de _____

Assinatura do autor: _____

Assinatura do orientador: _____

Catálogo-na-publicação

Ceron, João Marcelo

MARS: uma arquitetura para análise de malwares utilizando SDN / J. M. Ceron -- versão corr. -- São Paulo, 2018.
135 p.

Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo.
Departamento de Engenharia de Computação e Sistemas Digitais.

1.Análise de Malware 2.Redes Definidas por Software 3.Arquitetura de Software 4.Análise Dinâmica 5.Arquitetura e Organização de Computadores I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

AGRADECIMENTOS

Inicialmente gostaria de agradecer à Profa. Cíntia Margi e o professor Lisandro Granville pelo incentivo e orientação. As ideias discutidas foram imprescindíveis para o desenvolvimento deste trabalho. Um agradecimento especial ao Prof. Lisandro por ter me incentivado a seguir com os estudos após o término do mestrado.

Aos colegas de pós-graduação, em especial ao amigo Vladimir Rocha pelas discussões sobre o doutorado e demais temas que tangenciam a vida acadêmica.

Aos colegas de trabalho do Nic.br, em especial aos meus colegas do CERT.br pelo apoio e compreensão. As diversas conversas com o Klaus certamente contribuíram muito para o desenvolvimento deste trabalho.

Um agradecimento a minha amiga Ana Eliana por ter sempre palavras amigas para me auxiliar nas dificuldades encontradas durante esse processo.

Por fim, aos meus pais Hilário e Lourdes, que muito têm me incentivando e apoiando durante toda minha caminhada, e a minha irmã Luciane pelo companheirismo e amizade.

RESUMO

Detectar e analisar *malwares* é um processo essencial para aprimorar os sistemas de segurança. As soluções atuais apresentam limitações no processo de investigação e detecção de códigos maliciosos sofisticados. Mais do que utilizar técnicas para evadir sistemas de análise, *malwares* sofisticados requerem condições específicas no ambiente em que são executados para revelar seu comportamento malicioso. Com o surgimento das Redes Definidas por Software (SDN), notou-se uma oportunidade para aprimorar o processo de investigação de *malware* propondo uma arquitetura flexível apta a detectar variações comportamentais de maneira automática. Esta tese apresenta uma arquitetura especializada para analisar códigos maliciosos que permite controlar de maneira unificada o ambiente de análise, incluindo o *sandbox* e os elementos que o circundam. Dessa maneira, é possível gerenciar regras de contenção, configuração dinâmica de recursos, e manipular o tráfego de rede gerado pelos *malwares*. Para avaliar a arquitetura foi analisado um conjunto de *malwares* em dois cenários de avaliação. No primeiro cenário de avaliação, as funcionalidades descritas pela solução proposta revelaram novos eventos comportamentais em 100% dos *malwares* analisados. Já, no segundo cenários de avaliação, foi analisado um conjunto de *malwares* projetados para dispositivos IoT. Em consequência, foi possível bloquear ataques, monitorar a comunicação do *malware* com seu controlador de *botnet*, e manipular comandos de ataques.

Palavras-chave: Análise de Malware. Redes Definidas por Software. Análise Dinâmica.

ABSTRACT

Mechanisms to detect and analyze malicious software are essential to improve security systems. Current security mechanisms have limited success in detecting sophisticated malicious software. More than to evade analysis system, many malware require specific conditions to activate their actions in the target system. The flexibility of Software-Defined Networking (SDN) provides an opportunity to develop a malware analysis architecture that can detect behavioral deviations in an automated way. This thesis presents a specialized architecture to analyze malware by managing the analysis environment in a centralized way, including to control the sandbox and the elements that surrounds it. The proposed architecture enables to determine the network access policy, to handle the analysis environment resource configuration, and to manipulate the network connections performed by the malware. To evaluate our solution we have analyzed a set of malware in two evaluation scenarios. In the first evaluation scenario, we showed that the mechanisms proposed have increased the number of behavioral events in 100% of the malware analyzed. In the second evaluation scenario, we have analyzed malware designed for IoT devices. As a result, by using the MARS features, it was possible to block attacks, to manipulate attack commands, and to enable the malware communication with the respective botnet controller. The experimental results showed that our solution can improve the dynamic malware analysis process by providing this configuration flexibility to the analysis environment.

Keywords: Malware Analysis, Software-Defined Networking, Dynamic Analysis.

LISTA DE FIGURAS

1	Modelo hierárquico do paradigma SDN.	8
2	Componentes do protocolo OpenFlow.	11
3	Relatório de análise de <i>malware</i>	19
4	Arquitetura simplificada para análise de <i>malware</i>	20
5	Arquitetura de análise utilizando paralelização.	21
6	Arquitetura MARS	30
7	Módulos implementados no controlador SDN	31
8	Protocolo de comunicação com o <i>sandbox</i>	38
9	Automatização do processo de análise.	41
10	Especificação de um processo de análise de <i>malware</i>	41
11	Arquitetura de referência	49
12	Ambiente experimental da arquitetura MARS	50
13	Parâmetros de avaliação do Experimento I.	56
14	Parâmetros de avaliação do Experimento II.	59
15	Bashlite: protocolo de comando e controle.	69
16	Mirai: protocolo de comando e controle.	69
17	Assinatura de propagação do Mirai	70
18	Ambiente experimental da arquitetura MARS no contexto IoT	71
19	Fluxo de execução da análise do <i>malware</i> Bashlite	74
20	Fluxo de execução da análise do <i>malware</i> Mirai	75
21	Quantidade de pacotes gerados pelo Bashlite	77
22	Quantidade de pacotes gerados pelo Mirai	78

23	Comparação relativa a quantidade de pacotes	79
----	---	----

LISTA DE TABELAS

1	Diretivas do protocolo OpenFlow.	12
2	Comparativo das soluções de análise	27
3	Módulos SDN implementados.	31
4	Códigos de instruções de comando.	39
5	Parâmetros de avaliação do <i>malware</i> TR/Dropper.Gen	47
6	Aspectos de configuração dos experimentos.	54
7	Avaliação do <i>malware</i> b5c75c0be343f5acdf05b239331097f456ed928b	55
8	Avaliação do <i>malware</i> 39fdf1bfd3b7f1c11ef67be8d07309446fc41aae	56
9	Metodologia de comparação dos parâmetros de análise.	58
10	Comparação dos parâmetros de avaliação	59
11	Aspectos de configuração dos experimentos IoT.	76

LISTA DE ABREVIACÕES

API	<i>Application Programming Interface</i>
APT	<i>Advanced Persistent Threat</i>
ARP	<i>Address Resolution Protocol</i>
C&C	<i>Command and Control</i>
CDN	<i>Content Delivery Network</i>
CPU	<i>Central Processing Unit</i>
DDoS	<i>Distributed Denial of Service</i>
DHCP	<i>Dynamic Host Configuration Protocol</i>
DNS	<i>Domain Name System</i>
DPI	<i>Deep Packet Inspection</i>
DoS	<i>Denial of Service</i>
FTP	<i>File Transfer Protocol</i>
GRE	<i>Generic Routing Encapsulation</i>
ICMP	<i>Internet Control Message Protocol</i>
IDS	<i>Intrusion Detection System</i>
IP	<i>Internet Protocol</i>
IPv6	<i>Internet Protocol version 6</i>
IoT	<i>Internet of Things</i>
MAC	<i>Media Access Control</i>
MARS	<i>Malware Analysis Architecture based on SDN</i>
NAT	<i>Network Address Translation</i>
NETCONF	<i>Network Configuration Protocol</i>
P2P	<i>Peer to Peer Network</i>

SCADA	<i>Supervisory Control and Data Acquisition</i>
SDN	<i>Software-Defined Networking</i>
SHA-1	<i>Secure Hash Algorithm 1</i>
SMTP	<i>Simple Mail Transfer Protocol</i>
TCP	<i>Transmission Control Protocol</i>
UDP	<i>User Datagram Protocol</i>
XMPP	<i>Extensible Messaging and Presence Protocol</i>

SUMÁRIO

1	Introdução	1
1.1	Motivação e Justificativa	2
1.2	Objetivos	3
1.3	Metodologia	3
1.4	Contribuições	5
1.5	Publicações	5
1.6	Organização da Tese	6
2	Fundamentação Teórica	8
2.1	Redes Definidas por Software	8
2.2	Malware	12
2.3	Honeypots	14
2.4	Análise de Malware	16
2.5	Requisitos Operacionais	19
2.6	Sumário do Capítulo	23
3	Trabalhos Relacionados	24
4	Arquitetura Proposta	29
4.1	Visão Geral	29
4.2	Módulo de Inspeção	32
4.3	Módulo de Contenção	33
4.4	Módulo Gerente de Configuração	36
4.5	Módulo Controle da Análise	37

4.6	Funcionalidades Específicas	39
4.7	Sumário do Capítulo	42
5	Análise Comportamental Utilizando MARS	43
5.1	Contextualização	44
5.2	Metodologia	44
5.3	Implementação da Arquitetura MARS	49
5.3.1	Ambiente Experimental	49
5.3.2	Características de Configuração	52
5.4	Avaliação Experimental	54
5.4.1	Experimento I: Reconfiguração	54
5.4.2	Experimento II: Reconfiguração e Contenção	57
5.5	Discussão	60
5.6	Sumário do Capítulo	61
6	Análise de Malwares para IoT	62
6.1	Contextualização	62
6.2	Trabalhos Relacionados	64
6.3	Malwares IoT	65
6.4	Análise de Malware Adaptativa	68
6.5	Implementação da Arquitetura MARS	71
6.5.1	Ambiente Experimental	71
6.5.2	Características de Configuração	73
6.5.3	Processo de Análise	74
6.6	Avaliação Experimental	76
6.6.1	Experimento I: Contenção de Acessos de Rede	76

6.6.2	Experimento II: Manipulação de Pacotes	79
6.6.3	Experimento III: Encaminhamento de <i>Bot</i>	81
6.7	Discussão	82
6.8	Sumário do Capítulo	83
7	Considerações Finais	84
7.1	Conclusões	84
7.2	Respostas às Perguntas de Pesquisa	86
7.3	Limitações	87
7.4	Trabalhos Futuros	87
	Referências	89
	Glossário	94
	Apêndice A - Malwares Analisados	96
	Apêndice B - Parâmetros de Avaliação	99
	Apêndice C - Malwares IoT Analisados	120

1 INTRODUÇÃO

Malware é um termo genérico para caracterizar os diferentes tipos de códigos maliciosos (e.g. vírus, worms, trojan). Os *malwares* representam um grande risco à segurança pois podem realizar diversas ações nocivas, causando danos aos sistemas computacionais. Para proteger os sistemas dessas ameaças, é importante desenvolver ferramentas que possam identificar comportamentos dos códigos maliciosos e assim aprimorar os mecanismos de defesa. Sistemas de detecção de intrusão (*IDS - Intrusion Detection Systems*) e antivírus precisam conhecer antecipadamente as características comportamentais dos programas para desenvolver assinaturas e identificar ameaças. Neste contexto, as ferramentas de análise de *malware* possuem uma função essencial permitindo que as ações de códigos maliciosos sejam previamente caracterizadas.

O volume de *malwares* diariamente desenvolvidos encoraja o uso de ferramentas automatizadas para análise de *malware*. Uma estratégia popular de investigação consiste em executar um arquivo suspeito em um *sandbox* – i.e., sistema controlado – de modo a mapear as características comportamentais do *malware*. Baseado no comportamento observado, um analista pode desenvolver assinaturas e aprimorar os mecanismos de defesa.

Sabe-se, no entanto, que o resultado da execução do *malware* é dependente da sua interação com outros sistemas. Isso é especialmente observado em *malwares* modernos. Certas famílias de *malwares* não apenas inspecionam o *sandbox* mas também o ambiente que o cerca. *Malwares* como Agobot (MCLAUGHLIN, 2004) podem alterar seu comportamento de execução quando detectam um sistema de análise. Outros *malwares* finalizam a sua execução quando não possuem conectividade com a Internet ou diante da indisponibilidade de certos serviços de rede. A ameaça *Red October*, por exemplo, executa suas funcionalidades exclusivamente em ambientes governamentais ou em redes pertencentes às embaixadas (VIRVILIS; GRITZALIS, 2013). A literatura apresenta inúmeras soluções relacionadas à análise dinâmica de *malware* na qual um arquivo suspeito é executado de modo a investigar suas funcionalidades (FLECK et al., 2013; KIRAT; VIGNA; KRUEGEL, 2014; MOSER; KRUEGEL;

KIRDA, 2007; HOLZ; RAYNAL, 2005; DINABURG et al., 2008). A maioria das soluções, no entanto, foca no aprimoramento de sistemas onde o *malware* é executado, não considerando os demais recursos disponíveis no ambiente de análise. Mais do que permitir acesso à Internet, o ambiente que circunda o *sandbox* provê recursos fundamentais para a execução do *malware*, tais como: resolução de nomes (DNS), endereçamento de rede, serviços de rede, e política de acesso à rede.

1.1 Motivação e Justificativa

As soluções de análise de *malware* negligenciam o controle do ambiente que circunda o *sandbox*, mesmo sabendo que a configuração do ambiente pode afetar as ações desempenhadas pelo *malware*. Em especial, as soluções atuais apresentam limitações ao prover funcionalidades relacionadas com a camada de rede. Algumas soluções abordam a camada de rede no contexto de análise; no entanto, as funcionalidades são implementadas de forma parcial, focando apenas na contenção de tráfego e não na integração dos recursos disponíveis (LAURIANNE, 2004). Diante disso, há uma oportunidade para aprimorar as atuais soluções de análise abordando a seguinte questão:

Problema de pesquisa: *Como investigar malwares avançados que adaptam suas funcionalidades em função dos recursos externos ao sandbox?*

Com o surgimento da tecnologia SDN, notou-se uma oportunidade para desenvolver uma arquitetura para análise de *malwares* na qual a camada de rede é integrada ao processo de execução do *malware*. Sendo possível gerenciar a execução do *malware* e controlar o ambiente que circunda o *sandbox*.

Mais do que utilizar uma nova abordagem para implementar uma arquitetura de análise de *malwares*, esta tese revela que o controle centralizado e a manipulação dos fluxos de rede trazem uma flexibilidade até então não disponível nas ferramentas de investigação.

1.2 Objetivos

Conforme discutido, *malwares* modernos têm a capacidade de adaptar o seu comportamento tendo em vista a disponibilidade de recursos. Para investigar esses *malwares* é necessário contar com soluções flexíveis que permitam adaptar sua configuração segundo as necessidades demandadas pelo *malware*. Diante disso, essa tese tem o seguinte objetivo:

Objetivo: *Desenvolver uma arquitetura especializada para analisar códigos maliciosos que permita controlar a execução do malware no sandbox, bem como os recursos que compõem o ambiente de análise.*

Como resultado, o ambiente de análise pode ser reconfigurado dinamicamente segundo as ações desempenhadas pelo *malware*. Logo, o mesmo ambiente de análise pode assumir diferentes configurações buscando moldar-se às necessidades dos *malwares*.

1.3 Metodologia

Para abordar o objetivo proposto, esta tese apresenta a arquitetura denominada MARS – *MAIwaRe Analysis Architecture based on SDN*. MARS provê um ambiente configurável para executar códigos maliciosos e mapear suas ações num sistema controlado. A solução proposta utiliza SDN (Redes Definidas por Software - *Software-Defined Networking*) para integrar a camada de rede junto ao processo de análise, permitindo controlar as conexões estabelecidas pelo *malware* e interagir com os demais serviços de rede presentes no ambiente de análise.

De modo a investigar a efetividade da solução proposta, foi definido um conjunto de avaliações experimentais onde diferentes *malwares* foram analisados. Com base nas informações resultantes da execução dos *malwares* em determinados cenários de avaliação, buscou-se responder às seguintes perguntas de pesquisa:

P1: *A arquitetura proposta é capaz de detectar malwares que apresentam variações comportamentais tendo em vista a configuração do ambiente que circunda o sandbox?*

A pergunta **P1** é abordada no Capítulo 5, no qual é investigado um conjunto de *malwares* com funcionalidades avançadas utilizando diferentes configurações no ambiente de análise. Como consequência, através da execução de um mesmo *malware* nas múltiplas configurações definidas, torna-se possível identificar variações comportamentais ocasionadas pela alteração das características do ambiente de análise.

Na segunda pergunta de pesquisa, investiga-se a efetividade da arquitetura MARS em diferentes contextos. Para isso, é definida a seguinte questão:

P2: *É possível especificar uma arquitetura flexível de tal forma que possa ser utilizadas em diferentes domínios de análise?*

A pergunta de pesquisa **P2** é respondida no Capítulo 5 e Capítulo 6, nos quais são apresentadas duas análises experimentais: na primeira são analisados *malwares* desenvolvidos para sistemas tradicionais com funcionalidades avançadas (APT); e, na segunda, *malwares* projetados exclusivamente para dispositivos IoT.

Por fim, busca-se investigar as funcionalidades da arquitetura MARS relacionadas com a manipulação de tráfego de rede. Especificamente, como essas funcionalidades podem ser efetivas na investigação dos *malwares* de maneira a revelar novos aspectos comportamentais. Para tal, definiu-se a seguinte pergunta de pesquisa:

P3: *A arquitetura MARS permite que um analista interaja com os fluxos de rede de modo a estimular malwares a revelar detalhes de suas funcionalidades?*

A pergunta de pesquisa **P3** é abordada no Capítulo 6. Nesta investigação, a arquitetura MARS foi customizada para prover um ambiente de análise considerando as particularidades dos *malwares* projetados para IoT. Utilizando assinaturas de rede, foi possível adaptar o ambiente de análise gerenciando os recursos demandados pelo *malwares* e, conseqüentemente, manipular os seus fluxos de rede. Como resultado, foi viável interferir em comandos de ataques revelando assim detalhes comportamentais dos *malwares*.

1.4 Contribuições

As contribuições desta tese de doutorado incluem:

- uma arquitetura que permite aprimorar o processo de análise de *malware* através do controle do ambiente que circunda o *sandbox*;
- projeto e implementação de mecanismos que permitem reconfigurar dinamicamente o ambiente de análise com base nas ações realizadas pelos *malwares*;
- definição e implementação de funcionalidades associadas com o controle da política de acesso à rede de todos os elementos do ambiente de análise, dispensando a utilização de dispositivos intermediários de rede (*middleboxes*);
- projeto e implementação de um sistema que permite que uma mesma infraestrutura de análise possa ser customizada para assumir diferentes configurações de rede. Dessa forma, um *malware* pode ser analisado em distintos cenários de avaliação, possibilitando detectar variações comportamentais de maneira automatizada;
- resultados experimentais que exibem como a configuração dos elementos que circundam o *sandbox* afetam a execução do *malware*.

1.5 Publicações

Ao longo desta pesquisa, os seguintes artigos científicos relacionados aos tópicos analisados foram escritos e publicados, ou submetidos.

1. CERON, J. M.; MARGI, C. B.; GRANVILLE, L. Z. MARS: An SDN-based Malware Analysis Solution. In: IEEE. Computers and Communication (ISCC), 2016 IEEE Symposium on. Messina, Italy, 2016. p. 525-530. DOI: 10.1109/ISCC.2016.7543792.

Esse artigo apresenta resultados preliminares da arquitetura MARS. Para isso, um conjunto de *malwares* foi analisado utilizando distintas políticas de acesso à Internet. Observou-se características comportamentais tendo em vista as modificações implementadas;

2. CERON, J. M.; MARGI, C. B.; GRANVILLE, L. Z. *MARS: From Traffic Containment to Network Reconfiguration in Malware-Analysis Systems*. In *Computer Networks*. ISSN: 1389-1286, 2017. DOI: 10.1016/j.comnet.2017.10.003.

Esse artigo apresenta um estudo de caso onde a arquitetura MARS é utilizada para investigar *malwares* avançados. Investigou-se como a reconfiguração do ambiente de análise e a manipulação das conexões realizadas pelo *malware* permitem identificar características inéditas dos *malwares*;

3. CERON, J. M.; STEDING-JESSEN, K.; HOEPERS, C.; MARGI, C. B.; GRANVILLE, L. Z.; C. *A Responsive Analysis Approach for IoT Malware*. *IEEE Internet of Things Journal*, 2017. [Em avaliação.]

Esse artigo, submetido em 04 de Setembro de 2017, em fase de avaliação, apresenta um estudo de caso onde a arquitetura MARS é utilizada para investigar *malwares* desenvolvidos para dispositivos IoT. Neste trabalho investiga-se as funcionalidades da arquitetura para analisar *malwares* mitigando o tráfego de ataque sem afetar a qualidade dos resultados. Em especial, essa investigação foi desenvolvida em colaboração com o CERT.br - Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil associado ao Nic.br.

Adicionalmente, em resposta ao Edital BR-US 2016 - *CyberSec/Malware detection*, foi desenvolvido um projeto em associação com a Professora Jelena Mirkovic da USC (*University of Southern California*) para participação no referido edital que busca incentivar pesquisas na área de segurança entre Brasil e Estados Unidos. Para tal, participei do desenvolvimento e escrita do projeto, o qual propõe uma infraestrutura de análise de *malware* que incorpora as funcionalidades descritas por MARS. Mesmo sem ter sido contemplado, o projeto passou para etapa final da seleção.

1.6 Organização da Tese

O restante deste documento está organizado da seguinte forma: o Capítulo 2 descreve a fundamentação teórica. O Capítulo 3 apresenta a revisão bibliográfica. O Capítulo 4 discute a arquitetura proposta por esta tese. O Capítulo 5 apresenta uma avaliação experimental na qual é investigado um conjunto de *malwares* modernos. O Capítulo 6 descreve uma segunda avaliação experimental voltada à análise de

malwares projetados para dispositivos IoT. Por fim, no Capítulo 7, apresentam-se as considerações finais e trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Os conceitos apresentados neste capítulo visam facilitar o entendimento da pesquisa de doutorado aqui descrita. Para isso, este capítulo é estruturado em duas seções: na primeira, são apresentados fundamentos da tecnologia SDN e protocolo OpenFlow. Na segunda parte são descritos conceitos relacionados aos códigos maliciosos e ferramentas de análise de *malwares*.

2.1 Redes Definidas por Software

As Redes Definidas por Software constituem uma abordagem na qual se busca abstrair o controle lógico da rede frente aos recursos computacionais dos dispositivos (NUNES et al., 2014). Tal característica permite implementar uma separação lógica entre o processo de encaminhamento de dados e a definição de rotas, provendo uma maior flexibilidade com relação às arquiteturas tradicionais. Para isso, as SDNs implementam uma visão centralizada da rede a partir da qual é possível orquestrar a comunicação entre os diferentes elementos da rede através de interfaces abstraindo particularidades do hardware. Os principais elementos da arquitetura são ilustrados na Figura 1.

Figura 1: Modelo hierárquico do paradigma SDN.

Fonte: Autor.

O modelo descrito pelo conceito de SDN é estruturado em camadas que se comunicam através de uma interface de comunicação bem definida (HALEPLIDIS et al., 2015). As funcionalidades de cada camada são descritas na sequência.

Camada de Infraestrutura - são dispositivos de rede – roteadores e comutadores de pacotes – responsáveis pelo encaminhamento de pacotes. Esses dispositivos possuem uma tabela de encaminhamento utilizada para determinar o repasse de dados. Ao contrário das arquiteturas tradicionais, a consolidação e processamento da tabela de encaminhamento de pacotes é realizada externamente, mais especificamente, na camada de controle.

Camada de Controle - é a parte central da rede responsável por tomar decisões de roteamento e encaminhamento de pacotes. Esta camada tem uma visão unificada da estrutura contendo detalhes do tráfego de rede e os recursos disponíveis de cada dispositivo. Isso permite que algoritmos de encaminhamento de pacotes sejam desenvolvidos de maneira simplificada considerando os diferentes aspectos dos elementos da rede.

Camada de Aplicações - corresponde a programas e aplicações que demandam funcionalidades providas pelo controlador da rede. Através de interfaces, essas aplicações podem gerenciar os recursos dos dispositivos e modificar a configuração da rede.

O modelo hierárquico proposto pelo paradigma SDN é, de fato, bastante flexível e pode ser implementado utilizando diferentes tecnologias. O protocolo OpenFlow é a interface mais utilizada para comunicação entre a camada de controle e camada de infraestrutura, muito embora outros protocolos possam ser utilizados, tais como os protocolos NETCONF e XMPP (MCKEOWN et al., 2008). A comunicação entre a camada de controle e a camada de aplicações está diretamente associada às características do controlador SDN. O controlador SDN é responsável por prover funcionalidades de gerenciamento e controle da rede, recebendo as demandas das aplicações e as traduzindo para os demais componentes. Entre as principais funcionalidades do controlador SDN destacam-se:

- modificar políticas da rede através de software sem a necessidade de configurar os dispositivos segundo as suas particularidades;

- executar ações proativas e reativas diante do estado dos recursos da rede;
- simplificar a implementação de políticas de rede sofisticadas.

Os controladores são implementados através de módulos de software que definem interfaces de modo a controlar os elementos da rede. Existem diferentes controladores SDN disponíveis, cada qual com suas características e escopos definidos: controladores comerciais, controladores de código aberto, controladores descentralizados. Dentre estes, destaca-se o NOX/POX (GUDE et al., 2008), controlador de código aberto muito utilizado em projetos acadêmicos devido a sua flexibilidade e facilidade de prototipação.

NOX é um controlador SDN que disponibiliza uma interface unificada de programação para o desenvolvimento de aplicações de rede. Além dos mecanismos de controle e gerenciamento da rede SDN, o NOX provê um conjunto de componentes que podem ser utilizados no desenvolvimento de aplicações, incluindo: protocolo de reconhecimento de topologia, protocolos de roteamento e serviços de endereçamento. Com isso, o desenvolvimento de aplicações SDN pode valer-se desses componentes para auxiliar a implementação e compor as funcionalidades desejadas.

Posteriormente, o controlador NOX foi portado para a linguagem Python, dando origem ao controlador POX. Assim, além de incorporar os recursos do NOX, a interface de programação utilizada pelo POX foi aprimorada possibilitando um maior nível de abstração das funcionalidades de rede.

É importante salientar que a arquitetura MARS atua na camada de aplicação. Através de interfaces disponibilizadas pela camada de controle (controlador SDN) é possível analisar as informações de rede de modo centralizado, manipular pacotes de dados e modificar os fluxos de rede. Para a comunicação com os dispositivos de rede, a arquitetura MARS utiliza o protocolo OpenFlow, descrito na sequência.

2.1.1 OpenFlow

OpenFlow é um protocolo de comunicação do tipo *southbound* que permite que as ações relacionadas com o encaminhamento de pacote dos dispositivos de rede sejam controladas remotamente (MCKEOWN et al., 2008).

O protocolo OpenFlow possui especificação aberta sendo implementado em diversos equipamentos comerciais de múltiplos fabricantes. Isso permite que equipamentos sejam gerenciados de maneira unificada, abstraindo especificidades de cada hardware, tal como, diferentes versões de sistemas operacionais e particularidades de configuração. Existem diferentes especificações do protocolo OpenFlow. No entanto, existe um conjunto mínimo de funcionalidades que devem ser implementados por todos os equipamentos que seguem o padrão OpenFlow (MCKEOWN et al., 2008).

A Figura 2 ilustra os três componentes básicos definidos pela especificação do protocolo OpenFlow, a saber: controlador, protocolo, e o dispositivo responsável pelo encaminhamento de pacotes (*Switch OpenFlow*).

Figura 2: Visão geral dos componentes do protocolo OpenFlow.

Fonte: Autor.

De maneira geral, o protocolo OpenFlow é utilizado para efetuar a comunicação entre um controlador e o dispositivo de encaminhamento de pacotes (*switch OpenFlow*). Para isso, o controlador conecta-se com cada *switch* através de um canal seguro e troca mensagens de controle para administrar os demais elementos. Com base nas diretivas do protocolo, o controlador pode identificar os fluxos de rede em que uma determinada ação será aplicada. A Tabela 1 apresenta um conjunto parcial de diretivas definidas pelo protocolo OpenFlow utilizada para identificar fluxos de rede.

Tabela 1: Conjunto parcial de diretivas definidos pelo protocolo OpenFlow.

Nome do campo	Descrição
<i>Ingress port</i>	porta física do <i>switch</i>
<i>Ethernet source</i>	endereço MAC do dispositivo de origem
<i>Ethernet destination</i>	endereço MAC do dispositivo de destino
<i>IP source address</i>	endereço IP do dispositivo de origem
<i>IP destination address</i>	endereço IP do dispositivo de destino
<i>IP protocol</i>	protocolo utilizado. E.g.: TCP, UDP, ICMP, etc.
<i>Source port</i>	porta origem (TCP/UDP)
<i>Destination port</i>	porta destino (TCP/UDP)

Fonte: (ONF, 2017).

Essas diretivas são utilizadas para identificar fluxos e definir suas respectivas ações, tais como: encaminhamento, descarte, ou ainda, modificar características dos fluxos. Logo que o *switch* OpenFlow recebe um pacote ele verifica na própria tabela de fluxos quais ações devem ser aplicadas ao fluxo em questão. Caso ainda não exista nenhuma ação em sua tabela, o *switch* consulta o controlador da rede. O controlador, por sua vez, define que ações devem ser aplicadas ao fluxo e orienta o *switch* atualizar a tabela de fluxos com as novas instruções. Como consequência, o controlador SDN pode especificar uma regra na qual os fluxos destinados a um determinado endereço – diretiva *IP destination address* – devem ser descartados pelos dispositivos de encaminhamento de pacotes. Utilizando tabelas de fluxos, os dispositivos de encaminhamento de pacotes gerenciam as múltiplas regras dando suporte a política definida pelo controlador da rede.

2.2 Malware

O termo *malware* refere-se a código malicioso (*malicious code*) que pode ser definido como um software especificamente desenvolvido para causar danos a sistemas computacionais sem o consentimento dos usuários (LIGH et al., 2010). Esses códigos maliciosos são utilizados por atacantes para ter acesso a um sistema e realizar as mais diversas ações no dispositivo hospedeiro. Os *malwares* podem conter funcionalidades relacionadas ao roubo de dados sensíveis – credenciais de acesso e informações financeiras – mas também mecanismos que permitem que o sistema comprometido seja utilizado para efetuar ataques a terceiros. De fato, existem diferentes formas de um *malware* comprometer um sistema: os códigos maliciosos podem embutir-se em programas legítimos, anexar-se em mensagens de e-mail, ou ainda, propagar-se

automaticamente via rede. Por outro lado, existem *malwares* que exploram vulnerabilidades conhecidas e falhas de configuração para ter acesso a um sistema.

2.2.1 Tipos de Malware

De maneira geral, *malware* é um termo genérico para referenciar aos diferentes tipos de códigos maliciosos. No entanto, os *malwares* podem ser classificados em categorias segundo a sua funcionalidade, risco e método de infecção. Na sequência são descritos os principais tipos de *malwares* conforme funcionalidade.

Vírus - os vírus de computadores são programas maliciosos que tem a capacidade de comprometer um sistema e fazer cópias de si mesmos com o objetivo de espalhar-se para outros dispositivos. Para a sua execução, entretanto, os vírus necessitam da ação do usuário.

Worm - o termo *worm* refere-se a um programa malicioso que se propaga automaticamente pela rede. Diferentemente dos vírus, não é necessário uma ação do usuário para que os *worms* propaguem-se e comprometam outros sistemas. Tipicamente, os *worms* espalham-se através de serviços de rede vulneráveis que são explorados para possibilitar acesso ao sistema.

Trojan - ou simplesmente, cavalo de troia, é um programa malicioso que se mascara em um programa legítimo. Um *trojan* busca enganar um usuário disponibilizando funcionalidades desejadas, no entanto, além de realizar tais funcionalidades, o *trojan* contém funções danosas ao sistema que são instanciadas quando o software é executado.

Backdoor - é um programa malicioso que permite acesso a um sistema sem usar os mecanismos legítimos de autenticação. Normalmente são programas utilizados por atacantes para garantir o seu retorno ao sistema comprometido, sem a necessidade de explorar novamente as vulnerabilidades identificadas.

Spyware - é um tipo de código malicioso utilizado para monitorar as atividades de usuários de maneira oculta. Os *spywares* podem ser utilizados para roubar dados digitados (*keylogger*), informações exibidas na tela (*screenlogger*), histórico de navegação Web do usuário, e demais informações do sistema.

Rootkit - define um código malicioso que tem por objetivo ocultar as atividades do invasor no sistema comprometido. Tipicamente esses códigos são utilizados para ocultar processos sendo executados, arquivos criados e programas instalados.

Bot e Botnet - um *bot* – nome originado da palavra *robot* – é um código malicioso que permite que o sistema comprometido seja controlado remotamente através de uma estrutura de administração. Já, uma *botnet* pode ser definida como um conjunto de máquinas infectadas e controladas por uma mesma entidade de administração. Para que os *bots* sejam controlados remotamente utiliza-se uma infraestrutura de controle e comando – *command and control* (C&C) – pela qual o atacante pode encaminhar as instruções de ataque.

Ransomware - é um tipo de código malicioso que torna inacessível certos arquivos do sistema solicitando um pagamento para restabelecer o acesso aos mesmos. Em geral, o *malware* cifra os arquivos, utilizando criptografia, e apenas disponibiliza a chave de decifragem mediante um pagamento (resgate).

As categorias de *malwares* descritos acima não são excludentes. Notam-se, sobretudo em *malwares* modernos, diferentes funcionalidades em um mesmo arquivo malicioso dificultando a sua classificação em categorias. Mais do que as múltiplas funcionalidades implementadas, observa-se a incorporação de mecanismos avançados que envolvem adaptação do comportamento do *malware* com base nas características do sistema infectado.

2.3 Honeypots

Os *honeypots* são recursos computacionais projetados para serem sondados, atacados, ou comprometidos (PROVOS; HOLZ, 2008). De maneira geral, os *honeypots* atuam como uma isca em uma infraestrutura de rede provendo serviços para que o acesso a tais serviços sejam estudados.

Por serem sistemas dedicados nos quais não existe qualquer sistema legítimo sendo executado, todas as conexões destinadas aos *honeypots* constituem um ataque em potencial. A análise das informações coletadas permite identificar características de sondagens recebidas com um baixo nível de falsos positivos. Entre as principais funcionalidades providas pelos *honeypots*, destacam-se:

- coleta de *malwares*;
- identificação de varreduras automatizadas;
- desenvolvimento de assinaturas de ataques;
- correlação de informações com outras fontes de abusos;
- identificação da motivação dos atacantes.

Um *honeypot* pode executar qualquer sistema operacional e disponibilizar diferentes serviços de rede para serem explorados por ataques. A configuração de tais serviços definem o grau de interação dos *honeypots* podendo ser classificados em:

Honeypots de baixa interatividade: utilizam ferramentas para emular serviços/sistemas vulneráveis. Dessa maneira, o atacante interage com sistemas onde um conjunto limitado de funcionalidades está disponível.

Honeypots de alta interatividade: os serviços não são emulados ou restritos, são serviços não emulados utilizados para serem sondados por atacantes.

Os diferentes tipos de *honeypots* fornecem características de ataques em níveis distintos de informações. Os *honeypots* de baixa interatividade, por exemplo, disponibilizam informações mais generalistas, tal como características do tráfego de sondagem e dados quantitativos. Já os *honeypots* de alta interatividade, por serem sistemas não emulados, conseguem mapear todas as atividades do ataque no sistema, incluindo teclas digitadas, motivação dos atacantes e ferramentas utilizadas.

A implementação de *honeypots* de alta interatividade requer cuidados específicos. Afinal, por disponibilizar um sistema não emulado, o mesmo pode ser comprometido, permitindo aos atacantes terem acesso completo ao sistema e utilizá-lo para lançar ataques a terceiros. Tipicamente, sistemas *honeypots* de alta interatividade são implementados utilizando modificações no núcleo do sistema operacional (*Kernel Modules*). Por outro lado, os *honeypots* de baixa interatividade utilizam ferramentas específicas para emular um conjunto de funcionalidades que serão acessíveis às sondagens. A ferramenta honeyd (PROVOS, 2003), por exemplo, é uma solução bastante utilizada para implementar *honeypots* de baixa interatividade na qual diferentes serviços de rede podem ser emulados e disponibilizados para coletar detalhes de ataques.

Os benefícios de utilizar *honeypots* em uma estrutura de rede já foi discutida em diferentes trabalhos (VIRTI et al., 2006; HOLZ; RAYNAL, 2005; PROVOS; HOLZ, 2008). No contexto de análise de *malware*, por outro lado, os *honeypots* podem ser utilizados para compor o ambiente de análise coletando as ações de rede efetuadas pelo *malware* analisado. Sendo assim, os *honeypots* podem atuar como iscas numa infraestrutura provendo serviços para serem sondados. Quando um *malware* propaga-se através de serviços vulneráveis, por exemplo, essa propagação tende a chegar no *honeypot* onde pode ser estudada e, em última análise, revelar características do *malware*.

Essa característica dos *honeypots* de prover serviços vulneráveis para atrair atacantes e estudar suas técnicas é um conceito que tangencia a solução de análise de *malware* proposta. De certa maneira, a arquitetura MARS molda-se ao ambiente onde o *malware* é executado. MARS busca prover subsídios com os quais o *malware* sinta-se estimulado a realizar ações maliciosas. Como consequência, assim como os *honeypots*, a arquitetura MARS tem por objetivos estudar as ações desempenhadas por um *malware* durante a sua execução.

2.4 Análise de Malware

A análise de códigos maliciosos, propriamente dita, é composta por um conjunto de técnicas e procedimentos que visam investigar as características de arquivos suspeitos. Essa investigação busca determinar o impacto do *malware* quando executado em um sistema, bem como mapear características que podem ser utilizadas para a sua identificação. Sabe-se que boa parte das soluções de segurança, tais como anti-vírus e sistemas de detecção de intrusão, empregam técnicas de análise de *malware* para desenvolver assinaturas e aprimorar a detecção de ameaças nos sistemas. Portanto, é de fundamental importância compreender as funcionalidades dos *malwares* para aprimorar a segurança dos mecanismos de defesa.

O processo de investigar as funcionalidades de um programa malicioso é dispendioso. Seria muito mais fácil utilizar o código fonte dos *malwares* para inferir o seu comportamento. No entanto, isso nem sempre é possível. O volume de *malwares* cujo o código fonte está disponível é ínfimo. Para citar algumas exceções, têm-se os *malwares* Agobot (MCLAUGHLIN, 2004), Bashlite (BASHLITE, 2017) e Mirai (MI-

RAI, 2017) que, de alguma forma, tiveram o seu código fonte publicamente disponível. Mesmo assim, observam-se diferentes versões desses *malwares* que não refletem o seu código fonte publicamente disponível. Na prática, os *malwares* são disponibilizados em formato executável e incluem diferentes artifícios para evitar a sua análise incluindo técnicas de auto modificação, compressão e empacotamento (CHEN et al., 2008).

Diante disso, cabe aos analistas desenvolver técnicas eficientes que permitam analisar arquivos maliciosos de maneira segura evitando que sistemas legítimos sejam comprometidos. A literatura classifica as técnicas de análise de *malwares* em duas categorias:

Análise Estática: nesta abordagem, busca-se avaliar o funcionamento de um programa sem a execução do mesmo, baseando-se apenas no seu código executável. As técnicas mais tradicionais consistem em extrair instruções do código ASSEMBLY do programa e inferir conclusões com base na sequência lógica das instruções a serem executadas (BALL; RAJAMANI, 2002).

Análise Dinâmica: nesta abordagem o arquivo suspeito é executado em um ambiente controlado onde suas ações são monitoradas. Usualmente, as implementações valem-se de *sandboxes* para investigar as ações do *malware* em um sistema operacional. Alternativamente, um código malicioso pode ser executado de maneira progressiva utilizando *debuggers* para identificar as instruções desempenhadas pelo código (MATLOFF; SALZMAN, 2008).

Conforme descrito no Capítulo 1, este trabalho propõe uma arquitetura de análise baseada na metodologia dinâmica de análise de *malware*. Na sequência, a análise dinâmica é discutida com mais detalhes.

2.4.1 Análise Dinâmica

A análise dinâmica pode ser implementada utilizando diferentes técnicas e metodologias. Como descrito, o objetivo das ferramentas que implementam a análise dinâmica de *malware* é prover informações comportamentais através da execução do *malware* em um ambiente controlado. O processamento dessas análises permite identificar modificações no sistema e detectar comportamentos incomuns nos dispositivos infectados.

Neste contexto, o uso de *sandboxes* tem se demonstrado uma abordagem eficaz (EGELE et al., 2012). Afinal, tais ferramentas permitem executar um arquivo malicioso de maneira automatizada provendo escalabilidade ao processo de investigação. Boa parte dos *sandboxes* utilizam máquinas virtuais para executar um arquivo malicioso e, após a sua execução, restaurar o sistema para o seu estado inicial. Como resultado, fornecem ao analista um conjunto de relatórios no qual as ações do *malware* no sistema são descritas, incluindo:

- arquivos criados ou modificados;
- acessos ou modificações da chave do registro do sistema;
- bibliotecas dinâmicas instanciadas;
- áreas da memória virtual utilizada;
- processos criados;
- conexões de rede instanciadas;
- dados transmitidos pela rede.

Esses relatórios estão condicionados às particularidades de implementação de cada ferramenta de análise. Alguns *sandboxes* conseguem mapear ações do *malware* com maior nível de detalhamento porque atuam monitorando chamadas de sistema. A ferramenta Cuckoo Sandbox (OKTAVIANTO; MUHARDIANTO, 2013), por exemplo, fornece relatórios comportamentais bastante completos descrevendo ações no sistema e assinaturas associadas ao tráfego de rede. A Figura 3 ilustra um fragmento do relatório no qual é possível identificar assinaturas comportamentais de um determinado *malware* classificado como `Trojan.VBS.Downloader.ADR`.

No relatório apresentado na Figura 3 observam-se três assinaturas comportamentais identificadas pelo *sandbox*: *"File has identified by at least one AntiVirus on VirusTotal as malicious"*, *"Steals private information from local Internet browsers"*; e *"Installs itself for autorun at Windows startup"*. Essas assinaturas descrevem ações nas quais o *malware* rouba informações sensíveis do navegador Web e instala-se na inicialização do sistema Windows para autoexecução. Adicionalmente, é apontado que esse *malware* já possui assinatura maliciosa identificada por um antivírus.

Figura 3: Relatório das ações de um *malware* disponibilizado pelo Cuckoo Sandbox.

Signatures

File has been indentified by at least one AntiVirus on VirusTotal as malicious

Steals private information from local Internet browsers

```

process: {u'process_id': 200, u'process_name': u'RunDll32.exe'}
signs: [{u'type': u'api', u'value': {u'category': u'filesystem', u'status': True, u'return': u'0x00000000', u'timestamp': u'2017-01-04 17:16:16,255',
u'thread_id': u'2008', u'repeated': 0, u'api': u'NtCreateFile', u'arguments': [{u'name': u'ShareAccess', u'value': u'3'}, {u'name': u'FileName', u'value':
u'C:\\Documents and Settings\\User\\Local Settings\\History\\History.IE5\\index.dat'}, {u'name': u'DesiredAccess', u'value': u'0xc0100080'}, {u'name':
u'CreateDisposition', u'value': u'3'}, {u'name': u'FileHandle', u'value': u'0x000002a0'}], u'id': 1054}]]
process: {u'process_id': 200, u'process_name': u'RunDll32.exe'}
signs: [{u'type': u'api', u'value': {u'category': u'filesystem', u'status': True, u'return': u'0x00000000', u'timestamp': u'2017-01-04 17:16:16,255',
u'thread_id': u'2008', u'repeated': 0, u'api': u'NtCreateFile', u'arguments': [{u'name': u'ShareAccess', u'value': u'3'}, {u'name': u'FileName', u'value':
u'C:\\Documents and Settings\\User\\Local Settings\\History\\History.IE5\\index.dat'}, {u'name': u'DesiredAccess', u'value': u'0xc0100080'}, {u'name':
u'CreateDisposition', u'value': u'3'}, {u'name': u'FileHandle', u'value': u'0x000002a0'}], u'id': 1057}]]
process: None
signs: [{u'type': u'registry', u'value': u'HKEY_USERS\\S-1-5-21-1547161642-507921405-839522115-1004\\Software\\Microsoft\\Windows
NT\\CurrentVersion\\Winlogon}]
process: None
signs: [{u'type': u'registry', u'value': u'HKEY_LOCAL_MACHINE\\System\\CurrentControlSet\\Services\\Tcpip\\Parameters}]
process: None
signs: [{u'type': u'file', u'value': u'C:\\WINDOWS\\win.ini}]

```

Installs itself for autorun at Windows startup

Fonte: Autor.

Os aspectos comportamentais presentes nos relatórios são essenciais para entender os vetores de ataque e motivações dos atacantes. Com isso, novas metodologias e mecanismos de defesas podem ser desenvolvidos para aprimorar a segurança dos sistemas computacionais.

2.5 Requisitos Operacionais da Análise Dinâmica de Malware

Determinar os principais requisitos de uma arquitetura de análise de *malware* é fundamental para identificar limitações e propor aprimoramentos. Muitas instituições e pesquisadores utilizam uma estrutura própria para analisar *malwares* de forma dinâmica com objetivo de extrair características funcionais do arquivo suspeito através da sua execução (WILLEMS; HOLZ; FREILING, 2007). Revisando a literatura, nota-se que não existe uma arquitetura padronizada, mas sim elementos que estão presentes na maioria das soluções utilizadas para análise de *malware*.

A Figura 4 ilustra um ambiente controlado – uma sub-rede da própria instituição – onde um analista executa a sua investigação de *malwares*. Essa sub-rede de análise tipicamente é composta por um *sandbox* e demais servidores úteis para a composição do ambiente de avaliação.

Figura 4: Arquitetura simplificada para análise de *malware*.

Fonte: Autor.

É importante observar que essa sub-rede está isolada da instituição através de um *firewall*. No entanto, isso não significa efetivamente um isolamento completo. Afinal, se não corretamente isoladas, as conexões de rede podem acessar outras sub-redes da própria instituição. De forma complementar, ao observar apenas a sub-rede de análise, pode-se notar que não existe nenhuma restrição de acesso entre o *sandbox* e os demais elementos da arquitetura. Com isso, a análise de um arquivo malicioso pode ocasionar o comprometimento dos elementos presentes na sub-rede de análise.

Confinar os acessos da sub-rede impedindo que o *malware* acesse a Internet é uma alternativa viável, porém pouco efetiva. Boa parte dos *malwares* implementam mecanismos que ativam a sua execução apenas quando realizam acessos à Internet (SANTOS et al., 2010). Sendo assim, a contenção dos acessos realizada pelo *firewall* é crítica. Deve-se assegurar que os acessos da sub-rede de análise não afetem demais máquinas da instituição, mas também que os mesmos acessos consigam comunicar-se com a Internet de modo a não causar danos a terceiros. Muitas vezes é necessário colaboração de esforços, haja vista que a rede da instituição pode ter outros mecanismos intermediários de segurança, como por exemplo, um *firewall* de borda. Logo, o *firewall* da sub-rede deve estar alinhado com a política dos demais dispositivos intermediários da rede.

No processo de investigação de *malware* pode-se definir que as principais responsabilidades do analista compreende:

- configurar o sistema onde os *malwares* serão executados;
- atualizar os sistemas de contenção de acesso de rede (interno e externo);
- analisar as características do *malware*.

Neste processo investigativo, para cada *malware* a ser analisado, o analista deve definir a configuração do ambiente de análise e instanciá-lo previamente. Essa configuração compreende características do *sandbox*, tal como o sistema operacional onde o *malware* efetivamente será executado, e demais serviços da sub-rede de análise. Sabe-se, no entanto, que essa configuração do ambiente pode influenciar na execução do *malware*. Afinal, a presença de um determinado serviço, ou até mesmo a existência de um software vulnerável, pode desencadear novas ações do *malware* analisado. Por exemplo, o *malware* de espionagem russo, conhecido como Uroburos, ativa suas funcionalidades apenas quando um navegador Web é executado no sistema afetado (SECURITYLABS, 2014). Diante disso, muitas vezes o analista necessita executar um *malware* em diferentes cenários a fim de mapear essa variação comportamental.

Figura 5: Arquitetura tradicional de análise de *malwares* utilizando paralelização.

Fonte: Autor.

Uma variação da arquitetura tradicional de análise de *malware* é apresentada na Figura 5. Nesta figura é ilustrado o conceito de uma arquitetura de análise de *malware* paralelizada. Além da escalabilidade, uma arquitetura paralela permite que um arquivo malicioso seja analisado em diferentes cenários. Cenários que fazem o uso de paralelização requerem cuidados extras, demandando uma sintonia entre isolamento dos elementos e respectivas políticas de acesso.

Por outro lado, as limitações das arquiteturas tradicionais ficam mais evidentes quando deseja-se implementar funcionalidades mais complexas as quais permitam, por exemplo, reconfigurar dinamicamente o ambiente de análise.

Nesse contexto, identificou-se um conjunto de funcionalidades que são críticas em uma arquitetura de análise de *malware*. A listagem abaixo, composta com base nos estudos de Ligh et al. (2010), enumera os principais requisitos operacionais de uma arquitetura de análise de *malware*:

- **isolamento** - é muito comum que diferentes elementos estejam executando de forma paralela. Neste contexto, é fundamental que haja a possibilidade de restringir as comunicações de cada elemento da arquitetura. Desse modo, permite-se que análises paralelas sejam executadas sem interferência;
- **contenção** - além da possibilidade de isolar os elementos, é importante aplicar políticas de acesso e contenção. Os *malwares* analisados tipicamente acessam redes externas, logo, restringir determinadas comunicações evitam que sistemas de terceiros sejam atacados;
- **reprodutibilidade** - é importante que o analista consiga reproduzir cenários a fim de analisar diferentes *malwares*. A reprodutibilidade do cenário não deve ficar apenas no nível do sistema operacional – no qual o *malware* é executado – mas também nos demais elementos de rede, tais como serviços e dispositivos disponíveis;
- **alocação dinâmica de serviços** - uma interface que permita a alocação dinâmica de serviços na rede, respeitando o isolamento de cada sistema de análise, é valiosa para um analista de *malware*. Com isso, cenários mais complexos podem ser instanciados incorporando novos recursos na rede de análise;
- **gerenciamento** - o controle e o gerenciamento dos elementos da arquitetura e recursos disponíveis é um mecanismo essencial. Dessa maneira, pode-se obter informações associadas às atividades de cada cenário e detectar possíveis exceções.

As arquiteturas tradicionais apresentam limitações em dispor as funcionalidades descritas acima. Boa parte das soluções existentes são pouco flexíveis e requerem

elementos não intrínsecos à arquitetura para que as funcionalidades sejam disponibilizadas. Por consequência, o controle dos diferentes elementos da rede aumenta o nível de complexidade da gerência, dada a natureza descentralizada das soluções. Sendo assim, acredita-se que exista uma oportunidade para desenvolver soluções integradas nas quais funcionalidades descritas por este capítulo possam ser compostas no núcleo da rede.

2.6 Sumário do Capítulo

Este capítulo apresenta a fundamentação teórica necessária para o entendimento da arquitetura proposta. Inicialmente foram apresentadas as características da tecnologia SDN e, posteriormente, a ferramenta *honeypot* e os principais tipos de códigos maliciosos. Na sequência foi conceitualizado o processo de análise de *malware* discutindo as diferentes abordagens disponíveis. Por fim, este capítulo finaliza com um levantamento das requisitos funcionais para a concepção de uma arquitetura de análise de *malware*.

3 TRABALHOS RELACIONADOS

Analisar o comportamento de um arquivo malicioso é de fundamental importância na proteção de usuários legítimos contra ameaças de segurança. Diferentes abordagens podem ser utilizadas para identificar detalhes de funcionamento de arquivos maliciosos e aprimorar os atuais mecanismos de defesa. Em especial, técnicas baseadas em análise dinâmica têm demonstrado bons resultados no processo de análise de *malware*, sendo amplamente utilizadas pela comunidade de segurança para estudar as funcionalidades de um arquivo malicioso (PAITHANE et al., 2017). Como comentado, as implementações tradicionais de análise dinâmica consistem em executar um arquivo malicioso em um *sandbox* e monitorar ações do arquivo suspeito (CHEN et al., 2008; OKTAVIANTO; MUHARDIANTO, 2013; KRUEGEL; KIRDA; BAYER, 2006; LAURIANNE, 2004). Os *sandboxes* empregam diversas tecnologias para monitorar as ações de um *malware* e, como resultado, produzem relatórios que descrevem as atividades do *malware* no sistema. Esses relatórios são essenciais para investigar o comportamento de *malwares* e aprimorar ferramentas de segurança. De fato, existem muitas soluções que focam em aprimorar a qualidade de informações coletadas no *sandbox* e, até mesmo, em estudar técnicas de evasão de *sandboxes* (KIRAT; VIGNA; KRUEGEL, 2014; FLECK et al., 2013; NORMAN, 2003; MOSER; KRUEGEL; KIRDA, 2007; LIANG; YIN; SONG, 2008; SHI; ALWABEL; MIRKOVIC, 2014).

As ações desempenhadas por um *malware* são altamente dependentes de sua interação com outros sistemas. Sendo assim, o *sandbox* deve prover um certo nível de conectividade com a Internet de modo a permitir que a comunicação com outros sistemas seja exposta. Por outro lado, também é necessário evitar que os *malwares* ataquem sistemas legítimos de terceiros. Quando um *worm* é analisado, por exemplo, suas conexões podem acessar múltiplos sistemas na Internet e, sem a devida política de contenção, pode danificar sistemas de terceiros. Diante disso, já foram propostas diferentes abordagens para contenção de tráfego de rede aplicado ao contexto de análise de *malwares* (GRAZIANO; LEITA; BALZAROTTI, 2012; KREIBICH et al., 2011).

Algumas dessas abordagens, no entanto, desassociam a contenção de tráfego do próprio processo de análise de *malware*. Para isso, essas soluções utilizam dispositivos externos (*middleboxes*), nos quais são definidas regras de contenção de tráfego e políticas de acesso apropriadas (LINDORFER; KOLBITSCH; COMPARETTI, 2011). Na solução descrita por Graziano, Leita e Balzarotti (2012), por exemplo, a contenção de tráfego é implementada através da manipulação das conexões realizadas pelo *malware*. Com isso, as requisições destinadas a determinados serviços são encaminhadas para serviços emulados, aptos a interagir com protocolos determinísticos. Esse redirecionamento de tráfego permite obter detalhes sobre as reais intenções dos *malwares*, fazendo-os interagirem com serviços controlados. Apesar dessa interação ser limitada, por tratar-se de um serviço emulado, é possível identificar informações sensíveis incluindo as credenciais utilizadas por um *malware* para acessar um controlador de *botnet*.

Semelhante à proposta de Graziano, Leita e Balzarotti (2012) - a ferramenta Anubis (KRUEGEL; KIRDA; BAYER, 2006) implementa a contenção de tráfego redirecionando conexões associadas ao envio de e-mails para servidores de e-mails locais. Desse modo, os *malwares* que se caracterizam pelo envio de mensagens indesejáveis podem ser estudados sem efetivamente prejudicar serviços de terceiros.

A ferramenta Norman Sandbox (NORMAN, 2003) também disponibiliza serviços de rede emulados. No entanto, tal ferramenta possui um escopo mais específico do que na proposta de Graziano, Leita e Balzarotti (2012). O Norman Sandbox busca estudar *worms* que se propagam através do envio de e-mails e compartilhamento de arquivos. Para isso, disponibiliza serviços em sua estrutura que correspondem a um servidor de e-mail e um servidor de arquivos. Dessa forma, as conexões destinadas a esses serviços emulados são investigadas de modo a identificar as características de propagação dos *malwares*. Adicionalmente, a ferramenta Anubis viabiliza de maneira limitada que certas conexões acessem a Internet utilizando um serviço de *proxy*.

De forma similar, a solução denominada IoT POT (PA et al., 2015) propõe mecanismos específicos para análise de *malwares* com foco em dispositivos IoT. Para isso, os autores desenvolveram um *sandbox* que permite analisar arquivos suspeitos em diferentes arquiteturas de CPU. O *sandbox* emula o sistema operacional OpenWrt (FAINELLI, 2008) – uma distribuição Linux para dispositivos embarcados – e im-

plementa um conjunto de regras para o controle de acesso de rede. Esse controle é realizado por uma entidade externa (*firewall*) que é responsável por limitar o tráfego de saída do *malware* permitindo apenas conexões destinadas a serviços de DNS e HTTP. Adicionalmente, todas as conexões destinadas à porta 23/TCP (TELNET) são redirecionadas para um servidor local para que características do ataque sejam avaliadas.

Uma abordagem diferente é apresentada por GQ (KREIBICH et al., 2011), a qual propõe uma infraestrutura de análise de *malware* com suporte a políticas de contenção de tráfego. Para isso, o GQ implementa uma estrutura de controle externa que gerencia as conexões de rede originadas pelo *malware* utilizando protocolos customizados. Por ser uma infraestrutura centralizada, o GQ pode atuar diretamente nos diferentes fluxos de rede, podendo: bloquear conexões, encaminhar tráfego de modo transparente, restringir acesso a determinados destinos e limitar taxa de pacotes.

Com exceção da solução GQ, é possível concluir que as soluções não implementam mecanismos intrínsecos ao processo de análise de *malware* para o emprego de políticas de contenção de tráfego de rede. A falta desses mecanismos de contenção intrínsecos ao processo de análise adiciona uma camada extra de complexidade aos atuais sistemas de análise, demandando que elementos externos de contenção sejam monitorados e configurados de acordo com a política de acesso definida.

Destaca-se outra característica fundamental para estudar os *malwares* modernos: trata-se da flexibilidade de configuração do ambiente de análise. As atuais soluções de análise de *malwares* devem identificar e prover os diferentes recursos demandados pelos *malwares* analisados. Afinal, existe um conjunto de *malwares* que apenas revelam suas atividades maliciosas quando certos serviços estão disponíveis. Outros *malwares*, porém, modificam seu comportamento diante de possíveis restrições de acesso impostas ao ambiente de análise. Mais recentemente, *malwares* modernos também observam o ambiente de rede que circunda o *sandbox* de modo a selecionar suas ações maliciosas. Em especial, ataques direcionados a alvos específicos requerem condições próprias para que ações maliciosas sejam plenamente desempenhadas pelo *malware*. A literatura descreve *malwares* que atacam exclusivamente certos serviços de rede, bem como outros que observam o espaço de endereçamento IP em que estão sendo executados a fim de determinar as ações a serem desempenha-

das (VIRVILIS; GRITZALIS, 2013). Em particular, *malwares* utilizados em espionagem apenas ativam suas funcionalidades quando são executados em determinadas redes, tal como redes governamentais e militares, baseado no endereçamento IP. Portanto, a camada de rede tem se tornado cada vez mais relevante no processo de análise de *malware*. Afinal, a configuração e disponibilidade dos recursos da rede interferem diretamente nas ações desempenhadas pelos *malwares* (CERON; MARGI; GRANVILLE, 2016). Diante disso, elencou-se funcionalidades de rede úteis no processo de investigação de *malwares* e, por meio da Tabela 2, traçou-se um comparativo entre as principais soluções existentes.

Tabela 2: Comparativo das funcionalidades de redes disponíveis nas diferentes soluções de análise de *malware*.

	MARS	GQ	Cuckoo	Norman	Graziano	Anubis	IoT POT
Funcionalidades de rede							
Acesso à Internet (filtrado/não filtrado)	✓	✓	✓	✓	—	✓	✓
Serviços de rede emulados	✓	✓	—	✓	✓	✓	-
Mecanismos para contenção de acessos (<i>traffic shaping, rate limit</i>)	✓	✓	—	—	—	✓	✓
Redirecionamento de tráfego (NAT/PROXY)	✓	✓	✓	✓	✓	✓	✓
Reescrita de pacote (<i>payload</i>)	✓	—	—	—	—	—	-
Filtro de conteúdo (<i>payload</i>)	✓	—	—	—	—	—	-
Reendereçoamento dos elementos/dispositivos	✓	—	—	—	—	—	-
Alteração dinâmica de topologia	✓	—	—	—	—	—	-
Adição de novos recursos baseado em características de tráfego	✓	—	—	—	—	—	-
Suporte a IPv6	✓	—	—	—	—	—	-

Fonte: Autor.

De maneira geral, boa parte das soluções utilizam funcionalidades associadas com a contenção de acessos à rede, incluindo redirecionando de tráfego e filtro de pacotes. Já as demais funcionalidades de rede são negligenciadas pelas atuais ferramentas de análise de *malware*, em particular mecanismos relacionados à configuração dos elementos que compõe a arquitetura, filtro dinâmico de conteúdo e alterações de topologia. No decorrer deste trabalho mostra-se como essas características implementadas por MARS, e não consideradas por outras soluções, podem ser efetivas no processo de análise de *malwares* modernos.

A arquitetura MARS não apenas introduz novas características no processo de análise de *malware*, mas também busca aprimorar as funcionalidades existentes nas soluções tradicionais. Em especial, por utilizar uma infraestrutura na qual a camada de rede faz parte do processo de análise de *malwares*, torna-se possível desenvol-

ver mecanismos de contenção de tráfego mais precisos considerando características de fluxos e do próprio conteúdo dos pacotes (protocolos não cifrados). Aliado a isso, a orquestração das funcionalidades de maneira centralizada traz flexibilidade ao processo de análise de *malware* permitindo que diferentes perfis de configuração sejam aplicados a um mesmo ambiente, sem a necessidade de intervenção ou configuração manual requeridas pelas soluções tradicionais. Como resultado, um mesmo *malware* pode ser investigado em diferentes cenários para que variações comportamentais decorrentes da configuração do cenário de avaliação sejam convenientemente mapeadas.

As funcionalidades introduzidas por MARS valem-se de características disponibilizadas pelo paradigma SDN. A literatura já tem demonstrado a versatilidade de SDN no contexto de segurança de sistemas. Apesar de não existirem trabalhos específicos no contexto de análise de *malware* dinâmica, é possível encontrar investigações que tangenciam o problema abordado por MARS. Por exemplo, existem estudos que utilizam SDN para lidar com questões associadas a varreduras (HU et al., 2014), detecção de ataques que buscam exaurir recursos de rede (ROWE; WILKINSON, 2014), e avaliação de vulnerabilidades (KREUTZ; RAMOS; VERISSIMO, 2013). Já outras investigações buscam portar mecanismos de segurança atuais para o contexto SDN: Jin e Wang (2014) propõem uma solução análoga a um IDS, na qual assinaturas de tráfego malicioso são incorporadas no controlador da rede de modo a bloquear acessos suspeitos.

3.1 Sumário do Capítulo

O estudo da literatura permitiu identificar que o processo de análise de *malware* pode ser aprimorado para contemplar a investigação de *malwares* modernos. Conclui-se que boa parte dos trabalhos focam em aprimorar funcionalidades do *sandbox* menosprezando os elementos que o circundam, conforme ilustrado na Tabela 2. Nos trabalhos em que recursos externos ao *sandbox* são considerados, o suporte à rede é implementado de forma superficial. Essa tese busca abordar as limitações existentes e, através de uma arquitetura de análise de *malwares* integrada, introduzir novas funcionalidades ao processo de análise de *malware*.

4 ARQUITETURA PROPOSTA

Este capítulo descreve a arquitetura MARS, a solução para análise dinâmica de *malwares* proposta por este trabalho. Inicialmente, apresenta-se uma visão geral da arquitetura e como seus componentes se relacionam. Na sequência, descrevem-se as funcionalidades da solução proposta apresentando possíveis cenários de análise de *malware*.

4.1 Visão Geral

MARS define uma arquitetura para análise de códigos maliciosos que utiliza SDN para controlar a execução do *malware* e gerenciar a configuração do ambiente de análise. Graças a estrutura centralizada, provida pelo paradigma SDN, é possível implementar funcionalidades que permitem modificar a composição do ambiente de definir políticas de acesso à rede e interagir com a execução do *malware*.

A solução proposta especifica uma arquitetura modular baseada em eventos que integra a camada de rede ao processo de análise de *malware*. Cada evento é associado a uma determinada ação que irá interagir com o ambiente configurando e/ou coordenando a execução do *malware* no *sandbox*. Dessa forma, aspectos comportamentais resultantes da execução do *malware* geram eventos de configuração que atuarão no ambiente de investigação.

A Figura 6 ilustra a estrutura da arquitetura MARS e a relação entre os elementos que a compõe. Os principais elementos funcionais da arquitetura são destacados, sendo: *sandbox* (❶), controlador SDN (❷), recursos de rede (❸). Esses elementos permitem a execução de um *malware* de maneira controlada na qual a composição da infraestrutura se dá por meio de interfaces de configuração pré-definidas. Por exemplo, o controle dos acessos de rede realizados pelo *malware* é gerenciado pelo controlador SDN, o qual intervém no tráfego através das seguintes ações de rede: encaminhamento de pacotes; redirecionamento de tráfego; reescrita de pacotes; restrição do número de conexões; e descarte de pacotes.

Figura 6: Arquitetura MARS.

Fonte: Autor.

Os elementos que compõe a arquitetura MARS possuem funcionalidades específicas. Na sequência, os mesmos são descritos com maiores detalhes.

Sandbox - é o elemento da arquitetura responsável por executar um arquivo executável e mapear suas funcionalidades no sistema. Na arquitetura proposta, não é especificada a tecnologia do *sandbox*, tampouco sistemas operacionais em que o *malware* deve ser executado. Independentemente da solução utilizada, é importante que o *sandbox* permita o controle do processo de análise de *malware* via interface remota. Uma vez que o *sandbox* tenha essa interface para controle de análise, o *malware* pode ser executando em sistemas Windows, Linux, Android ou qualquer outro suportado pelo *sandbox*.

Recursos de rede - é um repositório de recursos utilizado para compor o ambiente em que o *malware* será executado. Tais recursos compreendem diversos serviços de rede, incluindo servidores e dispositivos que podem ser acessados diretamente pelo *sandbox* da arquitetura. Esses recursos podem ser dinamicamente incluídos no ambiente de análise de modo a alterar a topologia e disponibilizar serviços a serem explorados pelo *malware* em execução. Sabe-se que alguns *malwares* apenas executam funcionalidades maliciosas assim que determinados serviços estejam disponíveis na rede. Existem *malwares* que sondam o ambiente buscando atacar servidores Web (NARANG; REDDY; HOTA, 2013). Outros, porém, atacam exclusivamente

sistemas industriais (LANGNER, 2011). Sendo assim, disponibilizar um conjunto de recursos permite que um analista tenha flexibilidade de especificar diferentes cenários de análise e, com isso, mapear características de ataques destinados a tais serviços.

Controlador SDN - mais do que prover as funcionalidades básica de comunicação de rede, o controlador da arquitetura MARS foi estendido com mecanismos que permitem controlar o processo de análise de *malware*. Essa extensão foi realizada através de módulos implementados no próprio controlador SDN. Um diagrama dos módulos implementados no controlador da rede são exibido na Figura 7.

Figura 7: Diagrama dos módulos funcionais implementados no controlador SDN.

Fonte: Autor.

Os módulos de software implementados pelo controlador SDN são estruturados em dois grupos funcionais: no primeiro grupo, os módulos descrevem funcionalidades generalistas que incluem o encaminhamento de pacotes, endereçamento da rede e filtragem de fluxos de rede. O segundo grupo – detalhado na Tabela 3 – congrega módulos SDN com funcionalidades específicas para o processo de análise de *malware*.

Tabela 3: Módulos SDN responsáveis por gerir o processo de análise de *malware*.

Módulo SDN	Funcionalidade
inspeção	responsável por gerar eventos baseados na análise de tráfego de rede.
gerente de configuração	trata eventos relacionados com a reconfiguração dos elementos da rede.
contenção	trata eventos relacionados ao bloqueio e contenção do tráfego da rede.
controle de análise	controla a análise interagindo com o <i>sandbox</i> e demais elementos do ambiente de análise.

Fonte: Autor.

Esse módulos são elementos críticos na arquitetura MARS, uma vez que os mesmos permitem que o controlador SDN interaja com o *sandbox* e, conseqüentemente,

as ações desempenhadas pelo *malware* analisado. Na sequência, as características dos módulos SDN são detalhadas.

4.2 Módulo de Inspeção

Este módulo é responsável por analisar o tráfego de rede e gerar eventos que serão tratados pelos demais módulos. Os eventos são gerados em resposta a características de tráfego de rede, e podem considerar: número de pacotes, número de sessões TCP/UDP, conteúdo do pacote, vazão, endereço de origem/destino e portas TCP/UDP. Diante desses eventos, a arquitetura instancia ações que podem reconfigurar a infraestrutura de análise e interagir com as conexões do *malware*. A Listagem 4.1 ilustra um fragmento do arquivo utilizado pelo *Módulo de Inspeção*.

Listagem 4.1: Conjunto de regras utilizado para caracterizar tráfego de rede implementado pelo *Módulo de Inspeção*.

```
1  # redirecionamento de tráfego e reconfiguração de topologia
2  if (packet.dst_port == 80 and packet.proto == TCP):
3      server = add_webserver()
4      redirect_traffic(server)

6  # reescrita de respostas NXDOMAIN
7  if (packet.dns.response.error_code == NXDOMAIN):
8      packet.dns.response.answer.set.new.ip(192.0.2.10)

10 # regras de contenção
11 if (packet.ip.dst == "www.example.org"):
12     drop(packet)

14 # limite de sessões TCP
15 if (tcp.sessions > 10) and (packet.dst_port == 22)
16     limit(tcp.sessions,dst_port=22)
```

A Listagem 4.1 apresenta quatro regras utilizadas pelo controlador para caracterizar o tráfego de rede e definir eventos de configuração. A primeira regra – linhas 2, 3 e 4 – analisa fluxos destinados à porta 80/TCP e os redireciona para um servidor Web localizado na própria rede de análise. Esse redirecionamento permite coletar detalhes de ataques e sondagens endereçados a serviços Web. A segunda regra – linhas 7 e 8 – ilustra um exemplo de manipulação de pacotes, mais precisamente, a reescrita de respostas do protocolo DNS. A terceira regra – linhas 11 e 12 – exemplifica uma política de acesso que restringe conexões destinadas a um determinado destino.

No exemplo, as conexões destinadas à `www.example.org` são bloqueadas pelo próprio controlador da rede. A última regra – linhas 15 e 16 – demonstram como limitar o número de conexões TCP simultâneas relacionadas ao serviço SSH. Essa regra, por exemplo, pode ser utilizada para evitar que o *malware* analisado no *sandbox* faça ataques exaustivos de força bruta.

Por fim, os eventos são distribuídos para os demais módulos da arquitetura tendo em vista a natureza das ações demandadas: *Módulo de Contenção* - para ações relacionadas com política de acesso; *Módulo Gerente de Configuração* - para configuração do ambiente de análise; *Módulo de Controle da Análise* - para ações nas quais a interação com o *sandbox* é necessária.

4.3 Módulo de Contenção

O *Módulo de Contenção* é responsável por tratar eventos relacionados com políticas de acesso à rede aplicadas ao ambiente de análise. Essas políticas determinam ações destinadas aos pacotes de dados bem como às conexões de rede, podendo incluir: bloqueio, reescrita, limitação e redirecionamento de fluxos.

A contenção de tráfego, como já descrito, é um recurso crítico em sistemas de análise de *malware*. Afinal, implementar mecanismos para a contenção do tráfego de rede é fundamental para evitar que o *malware* analisado ataque sistemas de terceiros. Apesar dos riscos, é fundamental permitir que os *malwares* tenha um certo nível de comunicação com entidades externas, pois muitos *malware* necessitam baixar componentes adicionais e obter instruções de ataque. Por outro lado, existem *malwares* com comportamento agressivo que buscam comprometer sistemas tão logo são executados. Por exemplo, uma variante do *malware* Bashlite – *hash* SHA-1 `d483984a44d0a7d3a9216065d3273a8bdef18f1f` – assim que é executada busca infectar dispositivos IoT que estão utilizando determinadas senhas para o serviço TELNET. Logo que o acesso é concedido, esse *malware* executa série de comandos, exemplificados na Listagem 4.2.

Listagem 4.2: Comandos executados por um *malware* tentando abusar de sistemas externos.

```
1 cd /tmp | cd /var/run;wget http://XXX.XX.1.38/gtop.sh;\n2 sh gtop.sh;rm -rf gtop.sh;tftp -r tftp1.sh -g ip;\n3 sh tftp1.sh; tftp ip -c get tftp2.sh; sh tftp2.sh;\n4 rm -rf tftp.sh tftp2.sh gtop.sh
```

Na Listagem 4.2 o endereço IP foi sanitizado; no entanto, é possível observar um conjunto de comandos sequenciais que são executados pelo *malware* no sistema infectado. Esses comandos, se executados com sucesso, comprometem o sistema atacado. Para evitar esse tipo de ataque um analista pode escrever regras de modo a bloquear pacotes contendo instruções específicas, como o comando `wget`.

A utilização de SDN no contexto de análise de *malware* possibilita uma série de vantagens significativas que passam pelo controle centralizado da rede. Com isso, é possível implementar regras de contenção diretamente no núcleo da rede sem a necessidade de dispositivos externos, assim como *firewall* e IDS. Essas regras de contenção podem considerar diferentes aspectos da rede, incluindo:

- **conteúdo do pacote** - identificar características de pacotes, sobretudo na área de dados, é um recurso poderoso para evitar ataques, como ilustrado na Listagem 4.2. Regras de contenção podem identificar características de pacotes e demandar ações, não apenas filtrando, mas também redirecionando para outros elementos da arquitetura. A utilização de muitas regras que demandam a inspeção completa dos pacotes pode afetar o desempenho do controlador. No entanto, isso não é um problema crítico; afinal, o ambiente de análise é utilizado exclusivamente para o processo de análise de *malware*, cujo possível impacto de desempenho pode ser detectado facilmente, não afetando a efetividade da análise;
- **vazão** - determinar a taxa de transferência máxima é uma maneira de identificar as características do sistema infectado. Um sistema com acesso rápido à Internet é mais valioso para os atacantes sendo utilizados em atividades maliciosas, tal como propagação de SPAM. Sendo assim, limitar a vazão do sistema é um fator que pode influenciar o resultado da análise de um *malware*, além de proteger sistemas externos de ataques;

- **taxa de pacotes** - de forma similar à vazão, é possível configurar a taxa de pacotes associada a cada elemento da arquitetura. Essa configuração é utilizada para evitar certos tipos de ataques, tal como DoS, que utilizam uma alta taxa de pacotes para tornar um sistema inacessível;
- **número de conexões** - outros ataques podem ser facilmente inibidos através da limitação do número de conexões realizadas pelo *malware*. Varreduras de vulnerabilidades e ataques de força bruta, por exemplo, podem ser mitigados através da limitação do número de conexões;
- **listas de bloqueio de IP** - existem cenários nos quais é importante impedir que determinados endereços sejam acessíveis a partir da arquitetura de análise. Diante disso, os *malwares* com essa configuração ficam impossibilitados de acessar e, conseqüentemente, atacar os endereços previamente definidos.

Múltiplas regras de contenção podem ser aplicadas ao ambiente de análise. De maneira mais ampla, as políticas de acesso podem ser especificadas para toda a arquitetura ou restringidas para um subconjunto de serviços disponíveis. A Listagem 4.3 ilustra a configuração de regras de contenção para um determinado ambiente de análise de *malware*.

Listagem 4.3: Configuração de regras de contenção de um determinado ambiente de análise.

```
1  # perfil de contenção de tráfego
2  containment_name="001"

4  # limites gerais do ambiente (global)
5  set limit packet_rate={8000}
6  set limit throughput={1M}

8  drop proto tcp from any to 200.160.0.0/20 port = ssh
9  drop proto tcp from any to INTERNET port = smtp
10 drop proto udp from SANDBOX to any limit session 100
11 drop proto tcp from SANDBOX to any limit session 100
```

Na primeira parte da Listagem 4.3 – linhas 5 e 6 – são estabelecidas regras globais associadas ao limite da taxa de pacotes e vazão definidos pelo ambiente. Nas linhas 8, 9, 10 e 11, são descritas regras associadas a serviços específicos (SSH e SMTP) e também regras que limitam o número de sessões UDP e TCP realizadas pelo *sandbox*.

Essa configuração é aplicada ao ambiente de análise antes de iniciar a execução

do *malware*. Contudo, essas regras não são estáticas, e podem ser modificadas dinamicamente durante a execução da análise, conforme descrito no *Módulo de Inspeção*.

4.4 Módulo Gerente de Configuração

Este módulo é responsável por tratar eventos relacionados com a configuração do ambiente de análise. Tais eventos visam definir características da topologia da rede e modificar a sua configuração em consequência de padrões de tráfego pré-definidos. Entre as possíveis ações utilizadas para configurar o ambiente de análise têm-se:

- **adicionar novos elementos na rede** - em resposta a um determinado ataque, o controlador SDN pode modificar a topologia da rede inserindo um novo serviço na rede local. Dessa maneira, o ataque pode ser redirecionado para o novo serviço para que as sondagens sejam detalhadas;
- **reendereçoamento** - é possível reendereçar de maneira automatizada todos os elementos do cenário de avaliação. O endereçamento IP dos elementos pode influenciar no resultado da análise, uma vez que existem *malwares* que só executam em determinados contextos, tais como: domínios governamentais ou endereçamento de certas organizações;
- **segmentação** - o compartimento do cenário realizado no nível de enlace – semelhante ao uso de redes virtuais – é mais um artifício que pode ser utilizado por um analista para especificar topologias complexas de rede;
- **isolamento** - isolar determinado dispositivo no cenário de análise visa trazer maior flexibilidade à arquitetura. Por exemplo, caso o analista suspeite que algum elemento do cenário de avaliação esteja comprometido, o mesmo pode isolar o dispositivo e deixá-lo em quarentena para que o seu tráfego seja monitorado.

A configuração do ambiente de análise é especificada utilizando um arquivo no qual todos os recursos do ambiente são definidos. Dessa maneira, a configuração dos recursos pode ser previamente definida e instanciada para a execução do *malware*. O uso de arquivos de configuração permite utilizar diferentes configurações para um mesmo ambiente de análise e, até mesmo, definir modelos (*templates*), tais como: cenários com suporte ao protocolo IPv6; cenários com dispositivos móveis; cenário que

simula uma rede acadêmica; cenário representando uma instituição governamental. A Listagem 4.4 descreve um exemplo de um modelo de configuração.

Listagem 4.4: Configuração inicial de um determinado ambiente de análise de *malware*.

```
1  # identificação do modelo de configuração
2  configuration_name = "001"

4  # recursos disponíveis no ambiente de análise
5  web-server = {ip=192.0.2.1,ip=192.0.2.2,ip=192.0.2.3}
6  smtp-sever = {ip=192.0.2.4,status=waiting}
7  sandbox = {ip=192.0.2.10}
```

Como observado, a Listagem 4.4 descreve os elementos disponíveis no ambiente de análise e as suas respectivas configurações. A configuração inclui três servidores Web, um (1) servidor SMTP e o *sandbox*. Cada elemento possui os seus respectivos atributos, no exemplo, pode-se observar a configuração de endereçamento IP utilizado pelos servidores. O servidor SMTP encontra-se em estado de espera (*waiting*) e pode ser incorporado ao ambiente dependendo das ações do *malware* previamente descritas pelo analista.

4.5 Módulo Controle da Análise

Esse módulo trata eventos associados com o controle da análise de *malwares* nos quais a comunicação com o *sandbox* é necessária. Por meio de um conjunto de mensagens, o controlador SDN pode demandar ações ao *sandbox* controlando remotamente a execução do *malware*. Na arquitetura MARS é definido um conjunto de ações extensíveis direcionadas a esse controle, sendo:

- **reiniciar a execução do *malware*** - o ambiente de análise, como previamente descrito, pode ser modificado durante a processo de execução do *malware* no qual novos serviços e configurações podem ser incorporados ao ambiente. Nessas situações, um analista pode reiniciar o processo de análise de *malware* para investigar novas ações comportamentais ocasionadas pela modificação do ambiente de análise;
- **finalizar análise** - a execução da análise pode ser interrompida diante de um evento pré-definido, tal como tempo de execução e determinadas características de tráfego de rede;

- **reverter ao estado inicial** - essa ação envolve restaurar o estado inicial da máquina em que o *malware* é executado, e também reverter a configuração do ambiente de análise para o estado inicial.

A comunicação entre o controlador e o *sandbox* é realizada através de um protocolo de mensagens. Para isso, foi especificado um protocolo – cuja estrutura de mensagem é ilustrada na Figura 8 – de modo a implementar e controlar o processo de análise. As diretivas do protocolo especificado são descritas na sequência:

Figura 8: Protocolo especificado para a comunicação entre o *sandbox* e o controlador SDN.

Sandbox ID			
Instruction Code			
Malware Hash			
Src. IP	Src. Port	Dst. IP	Dst. Port

Fonte: Autor.

- **sandbox ID** - código alfanumérico que identifica o *sandbox* em execução. Isso é útil para distinguir quando mais de um *sandbox* está em execução em cenários logicamente separados;
- **instruction code** - define o código das ações – descritos na Tabela 4 – que podem ser executadas pelo controlador no ambiente de análise;
- **malware hash** - SHA-1 - este campo identifica o *malware* que está sendo analisado. Dessa maneira, o controlador pode identificar os fluxos de rede relacionados com o *malware* analisado;
- **src. IP e dst. IP** - define, respectivamente, o endereço de origem e o endereço de destino do pacote de instrução;
- **src. port e dst. port** - define, respectivamente, a porta de origem e a porta de destino do pacote de instrução.

As diferentes mensagens são utilizadas para interagir no processo de investigação determinando ações de configuração que devem ser aplicadas no ambiente de análise. A Tabela 4 descreve as instruções disponibilizadas pelo protocolo definido, ilustrando respectivamente, o código da mensagem, a descrição da ação e a entidade responsável por processar a instrução de comando.

Tabela 4: Código das instruções utilizados nas mensagens do protocolo de comunicação definido.

Código	Descrição	Destino
000	Sandbox iniciou a análise	controlador
001	Sandbox finalizou a análise	controlador
010	Sandbox deve iniciar	sandbox
011	Sandbox deve parar	sandbox
100	Sandbox deve reiniciar	sandbox

Fonte: Autor.

Por exemplo, o código de instrução 000 identifica uma mensagem de notificação utilizada para informar o controlador SDN que uma análise de *malware* iniciou. Deste modo, o controlador sabe que um *malware* está executando e que deve analisar o respectivo tráfego de rede. Já, o código de instrução 011 solicita ao *sandbox* para que a execução do *malware* seja interrompida.

4.6 Funcionalidades Específicas

Em complementação às ações de configuração implementadas pelos módulos SDN, a arquitetura MARS também descreve um conjunto de funcionalidades específicas ao processo de análise que visam facilitar a investigação de *malware*. Essa seção, inicialmente detalha o processo de manipulação de pacote intrínseco à arquitetura e, posteriormente, como a configuração do ambiente de análise favorece a automatização da investigação de *malware*.

4.6.1 Manipulação de Pacotes

Como previamente descrito, as características dos pacotes podem ser utilizadas para desencadear ações de configuração no ambiente de análise. No entanto, devido às características da própria tecnologia SDN, torna-se possível também manipular características de pacotes de forma transparente. Com isso, determinados pacotes de dados podem ser interceptados e manipulados pelo controlador SDN antes de encaminhá-los ao seu destino final. Um analista pode, por exemplo, definir as seguintes ações de manipulação:

- **requisições DNS** - manipular respostas do serviço de nomes é uma técnica bastante conhecida no contexto de análise de *malware*. Utilizando o controlador

SDN é possível fazer a manipulação de respostas diretamente no centro da rede sem utilizar serviços auxiliares. É possível, por exemplo, alterar o endereço IP de determinado nome, associar endereços a outros nomes, e ainda, reescrever todas as requisições DNS sem resposta (NXDOMAIN / NODATA) para um endereço IP previamente configurado;

- **diretivas de protocolo** - a manipulação de diretivas de certos protocolos pode influenciar na resposta das conexões. Requisições HTTP com diretivas que indicam o uso de *proxy*, ou ainda conexões que revelam características do sistema (*User-Agent*) podem ser alteradas para desencadear variações comportamentais;
- **injeção de pacotes** - em certos contextos, pode ser interessante criar um pacote e injetá-lo na rede de modo a observar o comportamento do *malware* analisado. Por exemplo, ao analisar uma *botnet*, o controlador SDN pode construir uma mensagem com determinadas ações para um *bot* de maneira a observar o respectivo comportamento desencadeado.

É importante salientar que a manipulação de pacotes só pode ser realizada em conexões nas quais os dados não são cifrados. Do contrário, as funcionalidades de manipulação de pacotes tornam-se inaptas. No entanto, essa limitação não é uma exclusividade da ferramenta MARS. Demais soluções que necessitam avaliar características de pacotes de rede apresentam limitações ao analisar fluxos cifrados (WILLEMS; HOLZ; FREILING, 2007).

4.6.2 Automação

Analisar arquivos maliciosos de forma automatizada é uma funcionalidade essencial provida pela técnica de análise dinâmica de *malwares*. No entanto, nas soluções tradicionais, a automatização não é extensiva para o ambiente de rede que circunda o *sandbox*. Na solução proposta, a automatização é estendida para a camada de rede através do uso de arquivos de configuração. Logo, um *malware* pode ser analisado em diferentes cenários sem depender da intervenção humana para configurar o ambiente de análise. Para isso, o controlador é responsável por aplicar as configurações definidas no ambiente de análise e instanciar a execução do *malware* no *sandbox*.

Figura 9: Processo de análise de *malware* composto por múltiplas execuções.

Fonte: Autor.

A investigação de um *malware* pode demandar que o mesmo exemplar seja executado múltiplas vezes, sendo que em cada execução uma configuração diferente pode ser aplicada no ambiente de análise. A Figura 9 ilustra um fluxo de execução, no qual o controlador modifica a configuração do ambiente com base nas ações do *malware*. A partir disso, a análise é reiniciada para que o mesmo *malware* seja executado em uma nova configuração.

Figura 10: Especificação de um processo de análise de *malware* composto por múltiplas execuções.

sample=a1ada2cac088c5a5881ade3bfb7aa928d2a3d160,cycle=3,sequence=cycle1,cycle2,cycle3		
containment-profile="001" environment-configuration-profile="002" inspection-profile="001" platform = windows label = cycle1 runtime=5	containment-profile="002" environment-configuration-profile="002" inspection-profile="001" platform = windows label = cycle2 runtime=5	containment-profile="003" environment-configuration-profile="001" inspection-profile="001" platform = windows label = cycle3 runtime=5

A Figura 10 ilustra a configuração de um processo de investigação de análise composto por três ciclos com suas respectivas configurações. Na parte superior são descritas características gerais do processo, tal como o *malware* a ser executado e o número de ciclos (execuções no *sandbox*). Na parte inferior, é apresentada a configuração de cada um dos três ciclos de análise definidos. Na sequência, os atributos de configuração utilizados para especificar cada um dos ciclos são discutidos:

- **containment-profile** - arquivo de configuração que descreve a política de acesso do ambiente de análise;
- **environment-configuration-profile** - arquivo de configuração que descreve a topologia e configuração dos elementos do ambiente de análise;

- ***inspection-profile*** - arquivo que descreve os eventos de rede que devem ser observados pelo controlador de modo a gerar ações de reconfiguração do ambiente;
- ***runtime*** - a quantidade de minutos que o *malware* deve ser executado no *sandbox*;
- ***platform*** - descreve a plataforma na qual o *malware* a ser analisado será executado (plataforma do *sandbox*);
- ***label*** - parâmetro que identifica o arquivo de configuração.

Como resultado, o processo de análise descrito na Figura 10 disponibiliza três relatórios que relatam o comportamento do *malware* em cada ambiente de configuração. Esses relatórios fornecem dados para que um analista detecte variações comportamentais nos diferentes ambientes de análise e, baseado nisso, aprimore a sua investigação.

4.7 Sumário do Capítulo

Este capítulo apresenta os componentes que compõe a arquitetura MARS e como os mesmos se relacionam no processo de análise de *malwares*. Na sequência, são descritos os módulos SDN especificados e implementados no controlador da rede. Por fim, demonstra-se como esses módulos podem ser configurados para instanciar um processo de investigação considerando as ações desempenhadas pela *malware* para modificar o ambiente de análise.

5 ANÁLISE COMPORTAMENTAL UTILIZANDO MARS

Este capítulo apresenta uma avaliação experimental na qual um conjunto de *malwares* heterogêneos foi analisado utilizando a arquitetura proposta. Deseja-se identificar aspectos comportamentais dos *malwares* exibidos em decorrência da configuração utilizada no ambiente de análise. Para tal, utilizou a flexibilidade provida por MARS para especificar diferentes configurações para a mesma infraestrutura de análise. Através da execução de um mesmo *malware* nas múltiplas configurações definidas, torna-se possível mapear variações comportamentais ocasionadas pela alteração das características do ambiente de análise.

Como resultado, a arquitetura pode identificar quais configurações podem ser mais adequadas para investigar um determinado *malware* e, em última análise, identificar os recursos demandados pelo *malware* para desencadear funcionalidades não observadas nas soluções tradicionais.

O capítulo teve como base a estrutura utilizada no seguinte artigo científico, aprovado para publicação no periódico científico *Computer Networks*:

- CERON, J. M.; MARGI, C. B.; GRANVILLE, L. Z. MARS: From Traffic Containment to Network Reconfiguration in Malware-Analysis Systems. In *Computer Networks*. ISSN: 1389-1286, 2017. DOI: 10.1016/j.comnet.2017.10.003.

De modo a evitar redundância de conteúdo, a discussão associada aos trabalhos relacionados do artigo original não é apresentada neste capítulo, já que a mesma foi discutida no Capítulo 3 desta tese. O restante do capítulo é estruturado da seguinte maneira: a Seção 5.1 apresenta uma contextualização do problema; a Seção 5.2 discute a metodologia utilizada para avaliar variações comportamentais. A Seção 5.3 descreve como a arquitetura MARS foi implementada e configurada para investigar os *malwares*. A Seção 5.4 apresenta o estudo de caso proposto; e, a Seção 5.5 apresenta uma discussão a respeito dos resultados obtidos.

5.1 Contextualização

Analisar códigos maliciosos é um processo fundamental para compreender as suas funcionalidades e desenvolver mecanismos de defesa apropriados. Uma abordagem muito utilizada para identificar as ações de um *malware* consiste em executar um arquivo suspeito em um ambiente controlado. Entretanto, com a evolução dos ataques, nota-se que os *malwares* estão implementando mecanismos para dificultar a sua investigação em ferramentas de análise. Além de técnicas que dificultam a sua execução em um *sandbox*, observam-se *malwares* que verificam o contexto que estão executando. Nota-se que boa parte dos *malwares* adaptam seu comportamento com base nos recursos disponíveis no ambiente de análise.

Neste contexto, este capítulo descreve uma pesquisa onde a arquitetura MARS foi utilizada para identificar variações comportamentais dos *malwares*. Para isso, foi implementada uma avaliação experimental na qual múltiplas famílias de *malwares* foram analisadas de maneira automatizada buscando identificar os aspectos funcionais dos códigos maliciosos.

De modo a estimular que novos comportamentos sejam revelados, a arquitetura MARS foi configurada com funcionalidades que permitem interagir na execução do *malware*. Mais especificamente, as funcionalidades possibilitam que o ambiente de análise seja reconfigurado dinamicamente em resposta a comportamentos de rede exibidos pelo *malware*. Através da inspeção de tráfego realizada por MARS, é possível redirecionar fluxos de rede, reescrever pacotes DNS, modificar a topologia e introduzir serviços vulneráveis no ambiente para coletar detalhes de ataques. Como resultado, deseja-se investigar como essas funcionalidades podem ser efetivas para revelar novos comportamentos dos *malwares* que, de outra maneira, ficariam dormentes em soluções de análise tradicionais.

5.2 Metodologia

Esta seção descreve a metodologia utilizada para detectar variações comportamentais apresentadas pelos *malwares* analisados. Como previamente discutido, o processo de análise dinâmica de *malwares* caracteriza-se por disponibilizar um con-

junto de relatórios nos quais os aspectos comportamentais mapeados são descritos. Com base nas informações presentes nesses relatórios é possível investigar as ações dos *malwares* e detectar variações comportamentais.

Para isso, torna-se crítico determinar que parâmetros comportamentais são relevantes para caracterizar uma variação comportamental exibida pelo *malware* analisado. No entanto, fez-se um estudo bibliográfico e observou-se que não existe uma padronização nos parâmetros utilizados para comparar soluções de análise de *malware*.

As soluções que traçam comparativos utilizam métricas arbitrárias. Por exemplo, na solução Pytrigger (FLECK et al., 2013), os autores avaliam a eficácia de sua solução observando parâmetros associados às atividades do *malware* no *sandbox*, incluindo acessos ao sistema de arquivos e modificações na configuração do sistema (chaves de registro do sistema Windows).

Lindorfer, Kolbitsch e Comparetti (2011) descrevem uma métrica que caracteriza ações do *malware* no *sandbox* através da investigação de chamadas de sistemas. Apesar de efetiva em contextos restritos, essa técnica apresenta limitações para observar certos comportamentos: chamadas de sistemas relacionadas com comunicações inter-processo e chamadas assíncronas são ignoradas pela técnica. Por outro lado, existem metodologias que consideram as conexões estabelecidas pelo *malware* de modo a determinar variações comportamentais. Por exemplo, na arquitetura GQ (KREIBICH et al., 2011), as diferenças comportamentais são determinadas pelo número de conexões estabelecidas pelo *malware*.

Sabe-se, no entanto, que o uso de uma metodologia baseada em parâmetros individualizados pode introduzir imprecisões na avaliação. Sobretudo em *malwares* modernos, é importante considerar múltiplos parâmetros da análise para determinar uma variação comportamental. Comparar o número de conexões estabelecidas pelo *malware* em diferentes análises pode não representar uma variação comportamental; afinal alguns *malwares* conectam a serviços com endereçamento dinâmico – *botnet* P2P, domínios *fast-flux* e CDNs. Como consequência, cada ciclo de análise pode revelar diferentes conexões que, no fundo, estão associadas a uma mesma ação comportamental. Portanto, constata-se que uma métrica de avaliação deve examinar múltiplos parâmetros para identificar assertivamente as ações realizadas pelo *malware*.

Diante da ausência de métricas comuns para comparar soluções de análise de *malware*, este capítulo discute um conjunto de parâmetros que podem ser utilizados para refletir a variação comportamental de um *malware*. São propostos os seguintes parâmetros obtidos da execução do *malware*:

- **conversação IP** - uma conversação corresponde ao tráfego IP realizado entre dois pares observado no ambiente de análise. Essa informação é fornecida pelo próprio controlador SDN, o qual possui uma visão centralizada da rede, possibilitando coletar informações de todos os elementos do ambiente;
- **endereço IP** - todos os endereços IP observados no ambiente de análise, incluindo os endereços demandados pelo *malware* analisado;
- **porta TCP/UDP** - fornece informações relativas a todas as portas TCP/UDP requisitadas pelo *malware*;
- **assinatura comportamental** - são assinaturas fornecidas pelo *sandbox* e descrevem aspectos comportamentais do *malware* analisado. Essas assinaturas representam ações genéricas, tais como: “Foi detectado tráfego HTTP”, “*Malware* conectou a um servidor IRC”, “*Malware* conecta-se à rede Tor”, “*Malware* que sequestra arquivos do sistema encriptando-os (*ransomware*)”, “*Malware* tenta desativar *firewall* e antivírus do sistema”, e demais padrões previamente definidos;
- **dados transmitidos** - quantifica o número de conversações nas quais foram transmitidos mais de 1000 *bytes* (entrada e saída). O volume de dados transmitidos em uma conversação pode revelar características comportamentais do *malware* bem como identificar ataques. O limiar (1000 bytes) foi definido com base nos resultados analisados: conversações com menos de 1000 bytes tendem a ser menos significativas em relação a revelar um comportamento malicioso;
- **duração** - quantifica o número de conversações que possuem duração superior a 2 segundos. Assim como *dados transmitidos*, a duração das conversações

aliados com os demais parâmetros pode revelar aspectos funcionais dos *malwares*. Um ataque de varredura, por exemplo, pode ser caracterizado por conexões destinadas a diferentes endereços IP com curto tempo de duração e com poucos dados transmitidos. Novamente, uma investigação manual permitiu definir esse limiar (2 segundos), uma vez que representam conexões mais expressivas em termos de revelar comportamento malicioso.

Utilizando os diferentes parâmetros descritos é possível identificar mais consistentemente a natureza das análises de *malwares* e, conseqüentemente, evitar inconsistências no processo de comparação dos resultados obtidos das ferramentas de análise. Em consequência, um analista pode comparar soluções de análise e determinar quais foram mais eficientes para mapear as ações desempenhadas por um determinado *malware*. No entanto, as soluções relacionadas com a arquitetura MARS, assim como descrito no capítulo 3 desta tese, não estão publicamente disponíveis e não podem ser utilizadas como fonte de comparação. Portanto, este trabalho utiliza uma abordagem semelhante à proposta por Balzarotti (BALZAROTTI et al., 2010) na qual um sistema de referência é implementado para efetuar comparações entre os resultados de análise. Logo, a metodologia proposta compara a execução do *malware* no sistema de referência e na solução proposta. Cada *malware* é analisado nos dois sistemas (MARS e referência) fornecendo um conjunto de relatórios comportamentais de onde os parâmetros descritos pela metodologia são extraídos.

Tabela 5: Comparação dos parâmetros do *malware* TR/Dropper.Gen quando analisado na arquitetura de referência e no MARS.

Arquitetura	Conversação	IP	Assinatura	Duração	Dados Transmitidos	TCP	UDP
Referência	11	6	2	1	1	0	3
MARS	12	8	5	4	2	0	3

Fonte: Autor.

A Tabela 5 ilustra um exemplo onde a metodologia proposta foi utilizada para investigar os aspectos comportamentais do *malware* classificado como TR/Dropper.Gen. Na comparação, nota-se que a arquitetura MARS revelou mais eventos relacionados aos seguintes parâmetros: *conversação*, *IP*, *assinatura*, *duração*, *dados transmitidos*. Em números, a arquitetura MARS identificou:

- 12 conversações de rede;
- 8 endereços IP distintos;
- 5 assinaturas do *sandbox*;
- 4 conversações com duração superior a 2 segundos;
- 2 conversações nas quais foram transmitidos mais de 1000 bytes;
- 3 portas UDP distintas.

Neste exemplo pode se concluir que a arquitetura MARS foi mais efetiva para revelar novos eventos que, em última análise, reflete uma variação comportamental do *malware* analisado.

Por fim, na metodologia proposta, considera-se que um *malware* apresentou variação comportamental quando ao menos quatro dos sete parâmetros definidos pela metodologia apresentam um incremento no número de eventos. Na avaliação do *malware* descrito na Tabela 5, no qual cinco parâmetros apresentam um incremento no número de eventos em relação a arquitetura de referência, conclui-se que o *malware* apresentou variação comportamental na arquitetura MARS. A quantidade de parâmetros foi definida com base na análise manual dos relatórios de análise. Um número inferior a quatro parâmetros tende a revelar variações comportamentais associadas a funcionalidades homólogas.

5.2.1 Arquitetura de Referência

A *Arquitetura de Referência* foi desenvolvida para fins de comparação, sendo uma versão simplificada da solução MARS. O objetivo é mapear os parâmetros de avaliação em um ambiente similar ao MARS, mas sem as funcionalidades providas pelo MARS, tal como a reconfiguração do ambiente de análise e as políticas de acesso de contenção. Como resultado, é possível traçar um paralelo comparativo em relação aos eventos observados nas duas soluções (MARS e Referência).

A Figura 11 ilustra os elementos da *Arquitetura de Referência* implementada. É importante notar que essa arquitetura foi constituída utilizando uma rede convencional, sem uso da tecnologia SDN.

Figura 11: Arquitetura de Referência implementada para analisar *malwares* de modo a estabelecer um paralelo comparativo.

Portanto, as ações do *malware* não sofrem intervenções. Assim como as soluções tradicionais, as conexões realizadas pelo *malware* na arquitetura de referência possuem acesso irrestrito à Internet. Essa infraestrutura é composta pelos seguintes elementos: *repositório de Malware*, *sandbox*, e *gateway*.

Para manter consistência nas comparações, a configuração dos elementos foi mantida igual na *Arquitetura de Referência* e no protótipo da arquitetura MARS implementado. De modo a evitar redundância de informações, a configuração dos elementos da arquitetura são descritos na Seção 5.3, na qual o protótipo da arquitetura MARS é detalhado.

5.3 Implementação da Arquitetura MARS

Tendo em vista as especificações da arquitetura MARS, descritas no Capítulo 4, foi implementado um protótipo com funcionalidades específicas com intuito de observar variações comportamentais de *malwares*. Na sequência, é descrito o ambiente experimental e, posteriormente, os detalhes de configuração dos elementos.

5.3.1 Ambiente Experimental

O ambiente experimental especifica a infraestrutura utilizada para executar os arquivos maliciosos e identificar o seu comportamento. Nessa infraestrutura, ilustrada na Figura 12, é definida uma sub-rede na qual todos os elementos estão no mesmo domínio de *broadcast*. Essa rede de análise possui uma estrutura centralizada na qual a comunicação com os elementos da sub-rede, bem como os acessos à Internet,

é gerenciada pelo controlador SDN. Na sequência, os elementos que compõem a infraestrutura de análise são detalhados.

Figura 12: Ambiente experimental do MARS implementado para analisar *malwares*.

Sandbox - utilizou-se a ferramenta Cuckoo Sandbox (OKTAVIANTO; MUHARDIANTO, 2013) a qual foi estendida para implementar as interfaces necessárias para o controle das análises. Como característica, o *sandbox* instancia uma máquina virtual de modo a traçar as ações do *malware* no sistema. Para isso, foi provida uma máquina virtual *Windows XP Home Edition Service Pack 3* disponibilizando o navegador *Internet Explorer 6.0*. Adicionalmente, foram utilizadas as boas práticas de configuração para evitar técnicas *anti-sandbox* e dificultar a identificação de ferramenta de análise (SHI; ALWABEL; MIRKOVIC, 2014).

Repositório de Malware - compreende um conjunto de arquivos maliciosos que serão submetidos para análise na infraestrutura. Para este estudo foram utilizados 100 *malwares* coletados de fontes públicas, como VirusShare¹. Os *malwares* foram selecionados observando um conjunto de requisitos, incluindo: i) diferentes famílias de *malware* segundo assinatura do antivírus ClamAV (ESLER; NOGEL; JUDGE, 2017); ii) *hash* criptográfico; e iii) remoção de *malwares* análogos conforme algoritmo *fuzzy hash* (STEIN, 2011). Adicionalmente, para aumentar a diversidade dos exemplares, o repositório foi composto por 50% dos *malwares* classificados como “APT” segundo o VirusShare, e 50% dos exemplares classificados como “não-APT”. A lista completa de *malwares* é disponibilizada no Apêndice A desta tese.

¹<http://www.virusshare.com/>

Servidor Web - são servidores independentes cada qual com o seu próprio endereço físico e configuração. São utilizados para compor o ambiente de análise atuando para coletar ataques destinados ao serviço Web. Foram disponibilizados três servidores, conforme abaixo:

- servidor Web utilizando a configuração padrão do software Nginx versão 1.6.2, executando em um sistema Linux kernel 3.14;
- servidor Web utilizando a configuração padrão do software Apache versão 2.4.23, executando em um sistema Linux kernel 3.14;
- servidor Web utilizando o software Wordpress em conjunto com o servidor Apache versão 2.4.23, executando em um sistema Linux kernel 3.14.

Servidor SMTP - implementa o serviço SMTP executando em um servidor Linux kernel 3.14. Para tal, foi utilizado o software Postfix (versão 3.0.2) customizado para aceitar mensagens de e-mail, mas sem repassá-las.

Honeypot - sistema implementado pelo software honeyd (PROVOS, 2003) que emula serviços arbitrários para coletar informações de ataques. No ambiente, o *honeypot* emula os seguintes serviços: TELNET (23/TCP), FTP (21/TCP), VPN (5900/TCP), IRC (6667/TCP).

Gateway - permite que máquinas da rede interna acessem à Internet. Por questões de flexibilidade, a mesma máquina faz o papel de servidor DHCP e também efetua a tradução dos endereços que acessam a rede externa (NAT). Trata-se de uma máquina virtual Linux kernel 3.14 valendo-se de um conjunto de regras de encaminhamento de pacotes incorporadas no software *Iptables*.

OpenVSwitch - comutador de pacotes responsável por prover a comunicação entre todos os elementos da rede seguindo as políticas definidas pelo controlador SDN.

Controlador POX - foi implementado utilizando um servidor Linux Kernel 3.14 e o software POX (GUDE et al., 2008). Conforme previamente descrito, através da configuração dos módulos de software implementados no controlador SDN é possível customizar e imprimir diferentes características na infraestrutura que o *malware* será analisado.

5.3.2 Características de Configuração

Assim como discutido, arquitetura MARS pode ser personalizada para compor diferentes ambientes de investigação. Sendo assim, para a análise experimental proposta por esta investigação, foram definidas duas funcionalidades para interação com o ambiente de análise: a primeira está associada à política de contenção de tráfego disponibilizada ao *sandbox*; e, a segunda, com a configuração dinâmica dos elementos presentes no ambiente de análise. Na sequência, as mesmas são descritas com maior nível de detalhamento.

5.3.2.1 Políticas de Contenção

As ações desempenhadas por um *malware* estão diretamente associadas com a sua interação com outros sistemas. Logo, implementar diferentes políticas de acesso no ambiente que circunda o *sandbox* pode fazer que novos comportamentos sejam revelados, já que muitos *malwares* modificam suas ações tendo em vista a disponibilidade de recursos. Por exemplo, quando um *malware* tenta acessar um serviço que está indisponível, o *malware* pode adaptar suas ações buscando um serviço alternativo, revelando assim novas informações comportamentais.

No protótipo implementado, foi utilizado um conjunto de regras pré-definidas que podem ser aplicadas nos elementos que compõem o ambiente de análise. Essas regras corresponde a três perfis de configuração, sendo:

- **open** - sem restrições, *i.e.*, todos os acessos de rede originados por elementos do ambiente de análise são permitidos, incluindo acessos à Internet;
- **partial** - sem restrições para comunicação local. No entanto, o acesso à Internet é limitado. Apenas as conexões destinadas aos seguintes serviços são permitidas: 80/TCP, 6667/TCP, 53/TCP, 53/UDP, 67/UDP, 68/UDP;
- **close** - apenas acesso local é permitido. As conexões destinadas aos elementos externos ao ambiente de análise não são permitidas.

Os perfis são utilizados para configurar o ambiente de análise antes do processo a ser instanciado. Com isso, pode-se automatizar a análise de *malware* permitindo que o mesmo seja executado em cada um dos perfis acima descritos.

5.3.2.2 Regras de Reconfiguração

Uma outra maneira de estimular com que o *malware* analisado exiba novos comportamentos se dá através da reconfiguração do ambiente de análise. Sabe-se que a presença de determinados serviços na rede de análise pode desencadear novas ações do *malware*. Mais do que modificar a topologia de rede, as regras de reconfiguração definidas buscam interagir com o ambiente durante a execução do *malware*. No protótipo implementado, foi especificado um conjunto de ações pré-definidas que buscam interagir com o tráfego de rede efetuado pelo *malware*:

- **adicionar serviço de rede** - quando o controlador detecta um pacote destinado à porta 80/TCP, o mesmo adiciona os servidores Web no ambiente de análise ativando as respectivas portas do comutador OpenVSwitch. A ideia é observar ataques destinados às aplicações Web. Em casos específicos, a inclusão de um servidor Web no ambiente de análise pode revelar novos comportamentos maliciosos e expor detalhes dos ataques que não são demonstrados em outros cenários de avaliação;
- **redirecionamento de tráfego** - todas as conexões do ambiente de análise são monitoradas. As conexões destinadas a um conjunto de serviços – TELNET (23/TCP), FTP (21/TCP), VPN (5900/TCP) – são transparentemente redirecionadas para o respectivo serviço emulado no *honeypot*, permitindo assim coletar maiores detalhes sobre o ataque;
- **manipulação de pacotes DNS** - o controlador intervém nas consultas DNS realizadas pelo *malware*. Consultas que têm como resposta uma mensagem de domínio não existente (NXDOMAIN) são reescritas apontando para o IP do *honeypot* disposto no ambiente de análise. Como consequência, as ações do *malware* associadas ao endereço indisponível podem ser examinadas no *honeypot*. As conexões destinadas para os serviços emulados no *honeypot* são devidamente tratadas. Já as conexões destinadas aos serviços não emulados são salvas pelo próprio *honeypot* para futura análise.

Esse conjunto de funcionalidades é implementado no controlador SDN de forma paralela. Sendo assim, todas as conexões desempenhadas pelo *malware* são avaliadas por cada uma das três regras de configuração descritas.

5.4 Avaliação Experimental

Esta seção apresenta os experimentos definidos para avaliar a arquitetura MARS com relação ao processo de identificação de variações comportamentais dos *malwares*. Foram definidos dois experimentos: no primeiro experimento, a arquitetura MARS é parametrizada para incorporar características de reconfiguração do ambiente de análise, conforme previamente detalhado; no segundo experimento, além das regras de reconfiguração, buscou-se executar os *malwares* nos diferentes perfis de contenção de acesso de rede especificados. Os experimentos e suas respectivas características de configuração são sumarizados na Tabela 6.

Tabela 6: Aspectos de configuração dos experimentos definidos.

	Políticas de contenção (Seção 5.3.2.1)	Regras de reconfiguração (Seção 5.3.2.2)
Experimento I	-	✓
Experimento II	✓	✓

Fonte: Autor.

Os experimentos foram conduzidos nas duas arquiteturas de análise, conforme especificados pela metodologia proposta (Seção 5.2). Com isso, foi possível avaliar os aspectos comportamentais dos *malwares* quando executados nos diferentes cenários de avaliação. Na sequência, os experimentos são detalhados.

5.4.1 Experimento I: Reconfiguração

Neste experimento são avaliadas como as regras de reconfiguração do ambiente de análise, detalhadas na Seção 5.3.2.2, podem aprimorar o processo de análise revelando novas ações comportamentais.

Cada *malware* foi executado no *sandbox* por um período de três minutos produzindo 200 relatórios de análise (100 análises utilizando MARS e 100 análises utilizando a *Arquitetura de Referência*). Na sequência, foi implementado um processo automatizado que extrai os parâmetros de avaliação e fornece as informações para que os diferentes aspectos da análise sejam confrontados.

Como resultado, observou-se que 38% dos exemplares analisados apresentaram variações comportamentais. Isso significa que as funcionalidades implementadas

neste experimento permitiram à arquitetura MARS revelar mais eventos comportamentais do que a Arquitetura de Referência. Do contrário, esses comportamentos permaneceriam dormentes. Por exemplo, na investigação associada ao *malware* com *hash* SHA-1 `b5c75c0be343f5acdf05b239331097f456ed928b`, a arquitetura MARS identificou mais eventos de rede do que a *Arquitetura de Referência*. No geral, como ilustrado na Tabela 7, todos os parâmetros de avaliação apresentaram um incremento sensível, sendo: 28 novas conversações IP; 15 novos endereços IP; 1 nova assinatura comportamental; 15 novas conversações com mais de que 2 segundos de duração; 11 novas conexões com mais de 1000 bytes transmitidos; e 1 nova porta UDP.

Tabela 7: Parâmetros de avaliação do *malware* *hash* SHA-1 `b5c75c0be343f5acdf05b239331097f456ed928b` obtidos no Experimento I.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmitidos	TCP	UDP
MARS	42	31	11	21	21	3	6
Arquitetura Referência	14	16	10	6	10	3	5

Fonte: Autor.

Uma análise mais detalhada deste *malware* demonstrou que seu comportamento é condicional a um servidor FTP específico. O *malware* revela suas ações quando esse servidor FTP específico está acessível. No entanto, quando a análise foi executada, o servidor FTP estava indisponível (NXDOMAIN). Como consequência, na *Arquitetura de Referência* o *malware* interrompe suas atividades. Por outro lado, na arquitetura MARS, as ações do *malware* foram mapeadas graças às funcionalidades associadas à manipulação de DNS e inclusão dos serviços de rede. Logo, quando esse tráfego destinado ao servidor FTP é observado, o mesmo é direcionado para o *honeypot* presente na rede local.

Apesar dessas funcionalidades serem efetivas na maioria dos casos, a manipulação de pacotes implementada neste experimento pode apresentar efeitos adversos em casos pontuais, como por exemplo, no comportamento revelado pelo *malware* com *hash* SHA-1 `39fdf1bfd3b7f1c11ef67be8d07309446fc41aae`. Esse *malware* tenta acessar um endereço IP específico antes de iniciar uma varredura na rede. No entanto, como o endereço IP não está disponível (NXDOMAIN), a arquitetura MARS re-escreve a conexão e a encaminha para o servidor Web local. O servidor Web local, entretanto, não possui o que o *malware* espera e o exemplar não inicializa a varredura na rede, tal como faria em uma análise regular. Contudo, ainda assim, a arquitetura

MARS provê recursos para identificar esses comportamentos, permitindo a um analista reconfigurar o ambiente para abordar essa situação. A Tabela 8 apresenta os parâmetros de avaliação do referido *malware*.

Tabela 8: Parâmetros de avaliação do *malware* hash SHA-1 39fdf1bfd3b7f1c11ef67be8d07309446fc41aae obtidos no Experimento I.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmitidos	TCP	UDP
MARS	14	16	5	5	8	2	6
Arquitetura Referência	18	19	5	6	10	2	6

Fonte: Autor.

As funcionalidades disponibilizadas pela arquitetura MARS também demonstraram ser úteis para identificar particularidades dos ataques. Em especial, observam-se *malwares* que buscam enviar massivamente mensagens de e-mails para diversos servidores de e-mail. Como configurado nesse experimento, todas as comunicações associadas ao serviço SMTP (25/TCP) foram redirecionadas para o servidor local. Com isso, torna-se possível estudar as campanhas de SPAM e coletar informações dos ataques que não seriam possíveis sem a manipulação de tráfego provida por MARS.

Figura 13: Parâmetros de avaliação obtidos na análise experimental do Experimento I.

Fonte: Autor.

A Figura 13 apresenta um comparativo dos parâmetros de avaliação obtidos nas análises realizadas na arquitetura MARS e Arquitetura de Referência. Para cada um dos parâmetros de avaliação é apresentado o somatório de eventos em cada uma das respectivas arquiteturas de análise. Com exceção dos parâmetros *IP*, todos os outros apresentaram um incremento em relação à Arquitetura de Referência.

Em particular, essa discrepância no número de endereços IP contatados está associada com um único *malware* que possui funcionalidades análogas a um *worm* fazendo varreduras pela rede. Tal *malware*, quando foi executado na arquitetura MARS, contatou menos endereços IP que na *Arquitetura de Referência* devido as regras de configuração implementadas, assim como observado na análise do *malware hash* SHA-1 b5c75c0be343f5acdf05b239331097f456ed928b previamente comentada. É importante correlacionar esses eventos com o parâmetro *Conversação*, que quantifica o número de conexões estabelecidas pelo *malware*. Como ilustrado, a quantidade de conversações não cresce em taxa semelhante ao número de endereços IP. Logo, um analista pode concluir que houve muitos acessos a endereços externos, mas essas conexões não foram estabelecidas, o que caracteriza um comportamento de varredura ou negação de serviço.

5.4.2 Experimento II: Reconfiguração e Contenção

Neste experimento buscou-se expandir as funcionalidades descritas pelo primeiro experimento incorporando políticas de acesso à rede (Seção 5.3.2.1). Dessa maneira, cada *malware* foi executado em cada uma das três políticas de acesso definidas. Na primeira execução, o *malware* é analisado no perfil “open”, equivalente a configuração de acesso implementado no Experimento I. Na segunda execução, utilizou-se o perfil “partial”, e na terceira análise o perfil “close”. Desta maneira, os parâmetros de avaliação de cada análise são agrupados e comparados com os parâmetros da sua respectiva execução na *Arquitetura de Referência*.

Para caracterizar esse processo, foram criadas tabelas de comparação nas quais os aspectos de avaliação de cada *malware* são sumarizados. A Tabela 9 exemplifica os aspectos de avaliação do exemplar com *hash* SHA-1

f5f82285141fe18b8b6f592510d85d2d0aa9707f classificado pelo Antivírus ClamAV como Win.Trojan.Gloox-1. Os demais aspectos de avaliação relacionadas às outras famílias de *malwares* podem ser encontrados no Anexo B.

Tabela 9: Eventos identificados pelo *malware* Win.Trojan.Gloox-1 executado em diferentes configurações do ambiente de análise.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmitidos	TCP	UDP
Open	21	22	5	6	11	3	6
Partial	27	25	5	18	10	8	5
Close	19	18	4	12	4	8	6
MARS Eventos únicos	67	38	5	36	25	10	6
Arquitetura Referência	18	19	5	5	12	3	6

As funcionalidades incorporadas na arquitetura MARS neste segundo experimento revelaram ações inéditas dos *malwares* analisados. Como resultado, foram detectadas variações comportamentais em 100% dos exemplares analisados. No geral, os experimentos demonstraram uma maior quantidade de eventos quando os *malwares* foram executados no perfil de contenção “open” e “partial”. No perfil “close”, observou-se um decremento no número geral de eventos. Porém, esses acessos bloqueados pelo perfil “close” podem revelar informações valiosas dos ataques, como por exemplo, endereços associados aos repositórios de artefatos.

Os distintos perfis de acesso permitiram identificar o cenário de adaptabilidade dos *malwares*: alguns *malwares* testam a política de contenção utilizando diferentes portas de saída. O exemplar com *hash* SHA-1 b3423fef635638bb078b01c34166d1e6eb638d36, por exemplo, tentou conectar em diversos serviços na Internet. No entanto, como os serviços não estavam acessíveis devido à política de contenção de rede implementada, o *malware* modificou suas ações acessando portas alternativas. De forma complementar, nota-se que a maioria dos *malwares* tentam acessar serviços externos utilizando portas TCP associadas com o tráfego Web e portas associadas com serviços de *proxy*. As portas TCP mais observadas foram: 80/TCP em 99% das análises, 443/TCP em 83% das análises, e 1053/TCP em 20% das análises.

A Figura 14 ilustra um comparativo dos parâmetros de avaliação obtidos na análise do Experimento II. Observa-se, que este experimento foi mais efetivo para desencadear novos eventos para todos os parâmetros definidos.

Figura 14: Parâmetros de avaliação obtidos na análise experimental do Experimento II.

Fonte: Autor.

Novamente o parâmetro IP apresenta uma maior variação. No entanto, desta vez, esse comportamento é observado na arquitetura MARS. Observa-se que a utilização das diferentes políticas de acesso no ambiente de análise, aliadas às regras de reconfiguração de rede, possibilitaram que o *malware* contactasse mais endereços que anteriormente não eram acessados. Esse comportamento é oriundo de *malwares* que fazem varredura na rede. Como neste experimento o *malware* é executado por três vezes, em cada execução diferentes endereços aleatórios são contactados gerando esse valor acumulativo.

Tabela 10: Comparação dos parâmetros de avaliação nos dois experimentos realizados.

	Experimento I	Experimento II	Incremento (%)
IP	5176	29277	465%
Duração	1043	3268	213%
Conversação	2783	7617	173%
Dados Transmitidos	958	2264	136%
TCP	661	796	16%
UDP	453	572	26%
Assinaturas	613	630	2%

Fonte: Autor.

Para concluir esse experimento, compararam-se os parâmetros de análise observados nos dois experimentos. A Tabela 10 ilustra os valores identificados, sendo que as primeiras colunas descrevem o somatório dos eventos obtidos em todas as análises do respectivo experimento. A última coluna apresenta o percentual de variação que o Experimento II obteve com relação ao Experimento I. Nota-se que o parâmetro de avaliação *IP* é o que mais apresentou incremento em relação ao Experimento I. Como discutido, o grande número de endereços IP está associado aos *malwares* que fazem varredura na rede, acessando diferentes endereços em cada execução no *sandbox*. Por outro lado, os parâmetros *Duração*, *Conversação* e *Dados Transmítidos* apresentam um incremento menor, no entanto quando combinados tendem a refletir aspectos comportamentais mais representativos.

5.5 Discussão

Analisar códigos maliciosos modernos demandam características até então negligenciadas pelas ferramentas de análise. Além de disponibilizar um *sandbox* no qual as ações do *malware* são mapeadas, as ferramentas devem prover mecanismos para que os demais elementos que circundam o *sandbox* também sejam controlados.

Este capítulo descreve como a arquitetura MARS foi instanciada para analisar *malwares* modernos, permitindo que os diferentes aspectos do ambiente de análise sejam controlados. Para isso, os módulos da arquitetura SDN foram parametrizados de modo a prover funcionalidades relacionadas com a manipulação do tráfego de rede, alteração de topologia e política de acesso a rede. Em consequência, foi analisado um conjunto de *malwares* heterogêneos para os quais foram especificadas diferentes configurações para o ambiente de análise. Como resultado, pode-se avaliar a efetividade da arquitetura para revelar as ações dos *malwares*: no primeiro experimento foi possível observar variação comportamental em 38% das análises. Já no segundo experimento, no qual foram especificadas diferentes políticas de acesso, notou-se variação comportamental em 100% das análises. De forma resumida, o estudo de caso descrito neste capítulo possibilitou as seguintes conclusões:

- a metodologia de detecção de variações comportamentais deve considerar diversos aspectos das análises a fim de evitar imprecisões;

- no conjunto de *malwares* analisados, identificaram-se exemplares que adaptaram o seu comportamento com base na política de acesso disponível. Muitos *malwares* tentam acessar diferentes portas quando uma porta demandada não está acessível. Em particular, esse comportamento foi mais observado com relação a portas TCP do que portas UDP;
- o encaminhamento de tráfego e a manipulação de pacotes DNS são mecanismos efetivos para estimular novos comportamentos em *malwares*;
- a maioria dos *malwares* requer recursos de rede externos, assim como contatar um serviço para obter um componente adicional. Na ausência desses recursos, os *malwares* tendem a ficar dormentes durante a análise;
- simples modificações na configuração do ambiente de análise podem estimular grandes mudanças no comportamento do *malware*.

5.6 Sumário do Capítulo

Este capítulo busca responder a pergunta de pesquisa **P1** descrita na Seção 1.3. Para isso, o capítulo apresenta dois experimentos de análise. No primeiro experimento as conexões do *malware* são manipuladas de modo a fazer com que o *malware* analisado exiba comportamentos que não seriam observados diante da ausência dos recursos demandados. No segundo experimento, utilizaram-se diferentes políticas de acesso à rede para observar as ações dos *malwares*. Como resultado, foram observados novos aspectos comportamentais em 100% dos *malwares* analisados em decorrência das funcionalidades implementadas pela arquitetura MARS. A pergunta de pesquisa **P2** também é tangenciada neste capítulo. Para isso, foram desenvolvidos diferentes domínios de análise e cenários de avaliação nos quais a arquitetura MARS foi efetiva para desencadear novos eventos comportamentais nos *malwares* analisados.

6 ANÁLISE DE MALWARES PARA IoT

Este capítulo apresenta uma avaliação experimental na qual *malwares* projetados para IoT foram analisados utilizando a arquitetura proposta. Como objetivo, deseja-se demonstrar que as funcionalidades da arquitetura MARS facilitam a investigação de *malwares* mitigando o tráfego de ataque destinado a outros sistemas sem afetar a qualidade dos resultados. Para isso, a arquitetura foi customizada tendo em vista as características comportamentais das duas famílias de *malware* IoT mais populares, *i.e.*, Mirai e Bashlite. Como resultado, o ambiente de análise pode se adaptar durante a execução do *malware* para prover os recursos demandados pelo *malware* de maneira adequada.

O capítulo é desenvolvido observando a estrutura utilizada no seguinte artigo científico, que se encontra em fase de avaliação:

- CERON, J. M.; STEDING-JESSEN, K.; HOEPERS, C.; MARGI, C. B.; GRANVILLE, L. Z.; C. A *Responsive Analysis Approach for IoT Malware*. *IEEE Internet of Things Journal*, 2017. [Em avaliação.]

O capítulo está estruturado da seguinte maneira: a Seção 6.1 apresenta uma contextualização do problema. A Seção 6.2 discute os trabalhos relacionados. A Seção 6.3 apresenta as características dos *malwares* projetados para IoT. A Seção 6.4 apresenta as assinaturas comportamentais desenvolvidas. A Seção 6.5 descreve como a arquitetura MARS foi implementada e configurada para investigar os *malwares* IoT. A Seção 6.6 apresenta a metodologia; e, a Seção 6.7 discute os resultados obtidos na avaliação experimental.

6.1 Contextualização

A segurança dos dispositivos IoT representa uma ameaça significativa para a Internet, pois a maioria desses dispositivos não implementa as melhores práticas de segurança. Esses dispositivos possuem um conjunto limitado de recursos e, normalmente, não recebem atualizações de software regulares para corrigir suas deficiências de segurança. Diante disso, nota-se a proliferação de *malwares* que exploram as fragilidades desses dispositivos para realizar ataques.

A maioria dos *malwares* projetados para IoT, ou simplesmente *malwares* IoT, tiram proveito das vulnerabilidades dos sistemas para comprometer dispositivos e transformá-los em um *bot*, *i.e.*, um dispositivo controlado remotamente conectado a uma infraestrutura de controle comum (LAND, 2017). Um dispositivo conectado a uma *botnet* pode ser utilizado para realizar diferentes tipos de ataques, incluindo ataques de negação de serviço distribuído (DDoS). De fato, ataques DDoS originados por *botnets* constituídas por dispositivos IoT, podem ser poderosos, capazes de afetar sistemas de grande escala na Internet (DynDNS, 2017; KREBS, 2017).

Observam-se diferentes famílias de *malwares* produzidas exclusivamente para dispositivos IoT, como por exemplo: Mirai, Bashlite, Tsunami, RemainTen, Carna, LizardStresser, LightAidra e Zollard. Além disso, existem muitas variantes que foram desenvolvidas com base nos *malwares* que tiveram o seu código fonte exposto publicamente (CONSTANTINOS et al., 2017). Uma maneira escalável de analisar essa quantidade de variantes é utilizar a análise dinâmica. Como já descrito, na análise dinâmica os arquivos suspeitos são executados em um ambiente controlado para que o seu comportamento seja estudado. No entanto, as soluções atuais apresentam limitações para atuar no contexto de *malwares* IoT. Em particular, os *malwares* IoT são desenvolvidos para múltiplas arquiteturas de CPU, incluindo MIPS, ARM e PowerPC. Logo, uma solução de análise dinâmica deve fornecer suporte para que o *malware* seja executado em diferentes plataformas e assim, suas ações sejam mapeadas adequadamente.

Outro desafio é definir políticas de contenção de acesso para evitar que o *malware* em execução ataque outros sistemas. Da mesma forma que os *worms* da Internet, os *malwares* IoT são muito ativos: logo que infectam um sistema, esses *malwares* efetuam varreduras pela Internet buscando identificar sistemas vulneráveis e comprometê-los. No entanto, implementar o bloqueio de todas as conexões destinadas à Internet impossibilitará que um *bot* também acesse o seu respectivo servidor de comando e controle (C&C). Como consequência, o *bot* não ingressa na *botnet* nem recebe instruções de ataque, tornando o sistema de análise incapaz de investigar as funcionalidades da rede. A próxima seção descreve as principais soluções utilizadas para analisar *malwares* projetados para dispositivos IoT.

6.2 Trabalhos Relacionados

O aumento no número de ataques que utilizam dispositivos IoT como uma plataforma tem motivado a comunidade de segurança a estudar os códigos maliciosos projetados para IoT. Boa parte dos estudos disponíveis correspondem a análises técnicas que descrevem funcionalidades dos *malwares* (ANTONAKAKIS et al., 2017). Essas análises, no entanto, são realizadas utilizando uma abordagem manual onde o código malicioso é inspecionado sem a sua execução. Além de não ser escalável, essas investigações que não executam o *malware* podem não refletir o real comportamento do *malware*. Em particular, não é possível investigar as instruções recebidas pelo *bot* porque o fluxo de comunicação com o controlador não é estabelecido. Com base nisso, os pesquisadores estão investigando abordagens escaláveis que permitam a investigação dos diferentes *malwares* IoT.

O projeto intitulado IoTPot (PA et al., 2015) aborda esse problema implantando uma infraestrutura para analisar famílias de *malwares* IoT que se propagam via ataques de força bruta destinados ao serviço de TELNET. Para tal, o IoTPot utiliza um *sandbox* capaz de mapear funcionalidades de *malwares* em diferentes arquiteturas de CPU. Esse *sandbox* caracteriza-se por emular o sistema OpenWrt (FAINELLI, 2008) - uma distribuição Linux popular para dispositivos embarcados - e implementa um simples controle de acesso. Dessa maneira, os *malwares* executados no *sandbox* tem o seu tráfego de rede destinado à porta 23/TCP redirecionado para um servidor local com o objetivo de estudar as características de propagação de *malware*.

Uma solução semelhante é apresentada por Detux (2017), a qual disponibiliza uma solução multiplataforma para executar *malwares* com objetivo de caracterizar o tráfego de rede exibido durante a análise. Como resultado, fornece relatórios contendo todas as tentativas de conexão e suas respectivas respostas. Detux é uma ferramenta de código aberto e não fornece mecanismos intrínsecos para o controle do tráfego de rede gerado pelo *malware*. Assim sendo, executar um arquivo malicioso na ferramenta Detux sem mecanismos externos de contenção pode afetar sistemas de terceiros.

No trabalho apresentado por Elzen e Heugten (2017) busca-se detectar dispositivos IoT comprometidos utilizando assinaturas da rede. Ao invés de fornecer um ambiente controlado para executar *malwares*, os autores modelam as características

de tráfego de rede dos dispositivos IoT comprometidos e os comparam ao tráfego legítimo para detectar anomalias.

Diferentemente de outras soluções, a arquitetura MARS possibilita analisar *malwares* IoT de forma automatizada provendo flexibilidade e segurança. A política de acesso é definida com base em assinaturas de rede que podem incluir os atributos completos dos pacotes (cabeçalho e carga útil), não apenas a porta de destino e protocolo, como em outras soluções. De forma complementar, o sistema pode modificar sua configuração de modo a adaptar o ambiente de análise conforme os recursos demandados pelo *malware*.

6.3 Malwares IoT

Os *malwares* projetados para IoT são muito heterogêneos, sendo desenvolvidos para múltiplas arquiteturas de CPU e diferentes sistemas operacionais. Em particular, as funcionalidades desses *malwares* são adaptadas para contemplar as limitações dos dispositivos IoT, onde memória, processamento, e espaço em disco são recursos escassos.

Apesar das limitações, os *malwares* projetados para IoT podem ser sofisticados. Observam-se famílias de *malwares* que incorporam funcionalidades avançadas, dificultando a sua análise em ferramentas automatizadas. A família Mirai, por exemplo, analisa aspectos do sistema infectado antes de acessar a respectiva entidade de administração (C&C da *botnet*). Se os requisitos não são satisfeitos, o *malware* simula conectar em um C&C falso, evitando que o C&C legítimo seja revelado. Outras variantes monitoram os processos sendo executados no dispositivo infectado buscando identificar processos associados a outros *malwares*. Diante disso, o *malware* compreende que o dispositivo já foi comprometido e, através do término do processo associado ao outro *malware*, pode interromper a execução de um rival otimizando os recursos do dispositivo comprometido (MIRAI, 2017).

A maioria dos *malwares* é desenvolvido utilizando a linguagem de programação C e posteriormente compilados para as diversas arquiteturas de CPU. Os códigos maliciosos incorporam todos os recursos necessários para a sua execução diminuindo a

dependência de bibliotecas externas o que, de fato, tende a potencializar as chances de sua execução nos sistemas heterogêneos. *Malwares* IoT implementam funcionalidades associadas a ataques de negação de serviço, ataques de força bruta e instalação de *backdoor*. No entanto, ainda se observa que o maior propósito dos *malwares* IoT é efetuar ataques de negação de serviço distribuído (ANTONAKAKIS et al., 2017). As famílias Bashlite e Mirai foram especialmente desenvolvidas para constituir *botnets* de grande escala cuja função principal é efetuar ataques de negação de serviço.

Malwares projetados para IoT propagam-se através de varreduras automatizadas cujo objetivo é comprometer dispositivos vulneráveis. Esse processo de varredura inclui enumerar serviços de rede disponíveis e efetuar tentativas de acesso usando diferentes credenciais (ataque de força bruta). Um erro comum é assumir que apenas dispositivos diretamente conectados à Internet estão expostos a esse tipo de ataque. Muitos dispositivos IoT implementam a tecnologia *Universal Plug and Play* (UPnP) que automaticamente negocia o encaminhamento de portas para a rede interna do dispositivo. Isso significa que um dispositivo IoT atrás de uma rede com *Network Address Translation* (NAT) pode ser explorado por varreduras, se o *gateway* da rede possuir o serviço UPnP habilitado (US-CERT, 2016).

Na sequência, são descritas as funcionalidades das duas famílias de *malware* mais populares e utilizadas na avaliação experimental executada por esta pesquisa.

6.3.1 Bashlite

Bashlite é uma família de *malware* caracterizada como *bot* que infecta dispositivos baseados em Linux para desempenhar ataques de negação de serviço. Essa família de *malware* possui diversas variantes sendo também conhecidas pelos nomes de Gafgyt, Lizkebab, Qbot, Torlus e LizardStresser (CONSTANTINOS et al., 2017).

O Bashlite propaga-se utilizando ataques de força bruta destinado ao serviço TELNET (23/TCP). Esses ataques são destinados a endereços IP aleatórios e caracterizando-se por utilizar um conjunto de credenciais embutidas no próprio *malware*. Ademais, o *malware* conecta-se a uma infraestrutura de C&C utilizando um protocolo personalizado baseado no *Internet Relay Chat* (IRC) no qual mensagens são trocadas em texto claro através de sessões TCP. O protocolo é descrito

em maiores detalhes na Seção 6.4. Com algumas diferenças entre as variantes, essa família de *malware* disponibiliza um conjunto de instruções que incluem mensagens específicas para ataques de negação de serviço, varreduras, e até um mecanismo que permite executar comandos arbitrários no sistema infectado.

6.3.2 Mirai

A família Mirai apresenta características muito similares ao Bashlite. No entanto, o Mirai possui aspectos mais sofisticados para controlar o dispositivo infectado e efetuar ataques de negação de serviço distribuído. Em particular, o Mirai implementa diferentes ataques volumétricos baseados no protocolos TCP, UDP e *Generic Routing Encapsulation* (GRE). Essas funcionalidades permitem ao Mirai realizar potentes ataques coordenados de negação de serviço distribuídos (ANTONAKAKIS et al., 2017).

A propagação do Mirai é muito similar à implementada pelo Bashlite na qual credenciais pré-definidas são utilizadas em ataques de força bruta. As primeiras versões do Mirai concentravam seus ataques as portas 23/TCP e 2323/TCP. Posteriormente, outras portas foram incorporadas no processo de varredura ampliando a profusão dos ataques (US-CERT, 2016).

A infraestrutura de controle e comando utilizada pelo Mirai possui diferenças consideráveis em relação a outras famílias de *malwares*. No lugar de uma, o Mirai possui duas entidades que gerenciam a infraestrutura de comando e controle (vide detalhes na Seção 6.4). Aliado a isso, o Mirai implementa funcionalidades que dificultam a sua análise em ferramentas automatizadas, como por exemplo: detectar ferramentas de depuração (*debuggers*), avaliar processo sendo executados e inspecionar parâmetros de execução. Como particularidade, o Mirai deve ser instanciado utilizando o nome `dvrHelper` e respectivo parâmetro baseado na sua arquitetura, assim como `./drvHelper telnet.mips`. Do contrário, o *malware* apresenta um comportamento dissuasivo e não contata o C&C legítimo.

6.4 Análise de Malware Adaptativa

Conforme descrito anteriormente, a solução proposta implementa um ambiente de execução que pode se adaptar de acordo com o tráfego de rede observado. O comportamento do *malware* pode ser mapeado em assinaturas e assim utilizado para ativar ações de configuração no ambiente de análise. Portanto, a definição de assinaturas de rede é um processo crítico pois, somente dessa maneira, o controlador SDN pode identificar os padrões de rede e aplicar ações de configuração.

Como prova de conceito, foi desenvolvido um conjunto de assinaturas para avaliar as famílias dos *malwares* Bashlite e Mirai. Ambas as famílias possuem variantes; no entanto, para esse estudo foram utilizados os respectivos códigos publicamente disponíveis: Bashlite (BASHLITE, 2017) e Mirai (MIRAI, 2017). De fato, não é necessário estudar o código fonte para desenvolver uma assinatura. Outras técnicas podem ser utilizadas para detectar padrões de rede, como descritos por Elzen e Heugten (2017). Uma vez que o escopo desse trabalho é fornecer um ambiente para analisar *malwares*, optou-se por criar assinaturas de rede utilizando o código fonte, o que torna o processo mais fácil.

Assinatura Bashlite C&C: o protocolo de comunicação utilizado por esta família de *malware* possui uma assinatura particular. Todas as mensagens são em texto claro onde as instruções – contidas no conteúdo dos pacotes – começam com o símbolo “!”. Dessa maneira, uma abordagem factível para detectar um C&C da *botnet* é inspecionar o conteúdo dos pacotes utilizando tal padrão. Assim que o C&C é identificado, torna-se possível bloquear demais conexões e, conseqüentemente, ataques destinados a entidades externas. Uma visão geral do protocolo de comunicação implementado pela família Bashlite e suas variantes é exibida na Figura 15.

O servidor mantém a conexão ativa com o *bot* trocando mensagens PING e PONG a cada 60 segundos. As demais mensagens do protocolo descrevem instruções para o dispositivo infectado: SCANNER para efetuar varreduras destinadas à porta 23/TCP; UDP para efetuar ataques DoS utilizando o protocolo UDP; TCP para efetuar ataques DoS utilizando o protocolo TCP; HOLD para manter um determinado número de sessões TCP conectadas à máquina alvo; e, GETLOCALIP para identificar o endereço IP local.

Figura 15: Protocolo de troca de mensagens implementado pelo C&C do Bashlite.

Fonte: Autor.

Assinatura Mirai C&C: a infraestrutura de comando e controle implementada pelo Mirai, como previamente mencionado, além do dispositivo infectado (*bot*) é composta por duas entidades: o C&C da *botnet* e o *reporter*. Essas entidades trocam informações utilizando um protocolo baseado em mensagens em formato hexadecimal. O processo de comunicação implementado pelo Mirai é ilustrado na Figura 16.

Figura 16: Protocolo de troca de mensagens implementado pelo C&C do Mirai.

Fonte: Autor.

O processo de ingresso na *botnet* Mirai é muito peculiar. Para isso, o dispositivo infectado deve enviar um pacote para o C&C da *botnet* utilizando uma mensagem específica, composta pelo código 0x00,0x00,0x00,0x00,0x01. Depois de 10 segun-

dos, mais uma mensagem deve ser encaminhada, dessa vez contendo um literal que representa a arquitetura do dispositivo (*arch*). Após essas duas mensagens, o C&C da *botnet* responde com uma mensagem de reconhecimento iniciando o processo de *keep-alive*, ou seja, troca de mensagens PING e PONG.

Assinatura Mirai Scan: O mecanismo de propagação implementado pelo Mirai consiste em varrer um conjunto de portas predefinidas, tal como às portas 23/TCP e 2323/TCP. As varreduras são iniciadas antes mesmo de estabelecer uma conexão com o respectivo C&C, sendo bastante agressivas. O simples bloqueio dos acessos destinados a tais portas tende a afetar a análise. Afinal, muitos C&C utilizam as mesmas portas para encaminhar instruções aos *bots*. Portanto, bloquear conexões para à porta 23/TCP e 2323/TCP pode inibir que o *bot* contate o respectivo controlador. Sendo assim, faz-se necessário desenvolver uma assinatura de rede que permita distinguir o tráfego de propagação da comunicação regular da *botnet*. Apesar da implementação da funcionalidade de propagação utilizar parâmetros aleatórios que impedem a sua identificação, foi possível utilizar uma característica presente no código fonte do Mirai – salientada na Figura 17 – que foi utilizada para desenvolver a assinatura de propagação.

Figura 17: Os pacotes de varredura do Mirai são construídos utilizando o mesmo valor para o campo “endereço IP” e “TCP Sequence number”.

```

1  for (i = 0; i < SCANNER_RAW_PPS; i++)
    {
        struct sockaddr_in paddr = {0};
        struct iphdr *iph = (struct iphdr *)scanner_rawpkt;
5   struct tcphdr *tcph = (struct tcphdr *) (iph + 1);

        iph->id = rand_next();
        iph->saddr = LOCAL_ADDR;
        iph->daddr = get_random_ip();
        iph->check = 0;
10  iph->check = checksum_generic((uint16_t *)iph, sizeof(struct iphdr));
        if (i % 10 == 0)
        {
            tcph->dest = htons(2323);
        }
15  else
        {
            tcph->dest = htons(23);
        }
        tcph->seq = iph->daddr;
    }

```

mirai/bot/scanner.c

Fonte: Autor.

A Figura 17 ilustra um fragmento do código fonte associado com o mecanismo de propagação do *malware*. É possível observar (linha 11) que 10% das varreduras são destinadas a porta 2323/TCP e 90% à porta 23/TCP. Em particular, os pacotes de varreduras são criados com uma duplicação de valores: o campo *sequence number* recebe o mesmo valor do endereço IP destino (linhas 8 e 19). Sendo assim, pode-se desenvolver uma assinatura que distingue ataques de propagação do tráfego regular da *botnet*.

6.5 Implementação da Arquitetura MARS

Tendo em vista as especificações da arquitetura MARS, descritas no Capítulo 4, foi implementado um protótipo com funcionalidades customizadas para analisar *malwares* da família Bashlite e Mirai. Na sequência são descritos detalhes de configuração e os demais elementos utilizados para compor a infraestrutura de análise.

6.5.1 Ambiente Experimental

O ambiente experimental compreende a infraestrutura utilizada para executar um arquivo suspeito e monitorar o seu comportamento. Na infraestrutura especificada (ilustrada na Figura 18) definiu-se uma sub-rede dedicada com acesso à Internet sendo gerenciada pelo controlador SDN. Além do controlador SDN, a rede de análise contém o *sandbox*, o repositório de *malwares* e um servidor que emula um C&C de *botnet*. Na sequência, os elementos utilizados para compor a infraestrutura de análise são detalhados.

Figura 18: Ambiente experimental implementado para analisar *malwares* IoT.

Fonte: Autor.

Sandbox - utilizou-se o *sandbox* Detux, o qual foi estendido para prover as interfaces necessárias para controle e parametrização das análises. Como característica, o Detux utiliza máquinas virtuais para instanciar as análises, sendo que neste experimento, foram implementadas quatro máquinas virtuais com diferentes arquiteturas de CPU (x86, ARM, MIPS, MIPSSEL) executando Linux Kernel 2.6. Cada *malware* foi executado no *sandbox* por um período de 1440 minutos (24 horas). Esse parâmetro foi definido com base nos experimentos realizados nos quais esse período demonstrou-se representativo para coletar instruções do C&C e determinar características de *bot-nets*.

Repositório de *Malware* - é responsável por disponibilizar os arquivos maliciosos que serão submetidos para análise no *sandbox*. Nesta investigação utilizou-se apenas exemplares da família Bashlite e Mirai que foram coletados e fornecidos pelo projeto “Honeypots Distribuídos” mantido pelo CERT.br (HoneyTARG, 2017). A listagem dos *malwares* analisados é exibida no Anexo C.

Controlador de *Botnet* - é um servidor Linux responsável por disponibilizar um serviço de rede que emula um C&C de *botnet*. Esse software foi desenvolvido utilizando a linguagem Perl e suporta o protocolo de comunicação descrito pelas duas famílias (Bashlite e Mirai). Para tal, foi implementado um conjunto de mensagens do protocolo que possibilita a associação do *bot* no C&C e registrar toda a comunicação gerada. O software que emula o C&C adapta-se automaticamente à família de *malware* tendo em vista as características da conexão, mas especificamente, através do conteúdo do pacote de inicialização. Sendo assim, caso um *malware* Bashlite tente conectar no serviço emulado, o mesmo responderá como um C&C de Bashlite. O mesmo comportamento se aplica para a família Mirai.

Controlador de SDN - foi implementado por um servidor Linux utilizando o software POX (GUDE et al., 2008) e OpenVSwitch (PFAFF et al., 2015) – comutador de pacotes que executa ações em nível de rede. Através da parametrização dos módulos SDN implementados pelo controlador foi possível customizar a configuração da análise para cada uma das famílias dos *malwares*. A próxima seção aborda com mais detalhes as características da configuração do controlador SDN implementado.

6.5.2 Características de Configuração

Como descrito, um analista define a configuração que será aplicada no ambiente que o *malware* é executado. Essa configuração é descrita utilizando um arquivo texto que, posteriormente é traduzido para regras OpenFlow pelo controlador SDN. A Listagem 6.1, por exemplo, ilustra um fragmento da configuração onde características do tráfego do Mirai são definidas para que o controlador SDN as considere no processo de análise.

Listagem 6.1: Arquivo de configuração que descreve a assinatura de propagação do Mirai.

```
1 # Assinatura de propagação da família de malware Mirai
2 sandbox-ip = 192.0.2.30
3 subnet = 192.0.2.0/24
4 bpf-expression = "tcp[4:4] == ip[16:4]"
5 action = "drop"
6 proto = "tcp"
7 detux-args = "dvrHelper telnet.mips"
8 arch = "mips"
9 time = "1440 minutes"
```

O arquivo de configuração apresenta informações a respeito dos parâmetros de análise, incluindo assinatura de propagação do *malware* e atributos dos elementos que compõem a infraestrutura. Essa configuração é especificada em formato texto valendo-se de um conjunto de diretivas de configuração, sendo:

- **sandbox-ip:** descreve o endereço IP do *sandbox*;
- **bpf-expression:** define uma expressão no formato *Berkeley Packet Filter* que será monitorada na rede;
- **proto:** descreve protocolo utilizado pela assinatura;
- **action:** define a ação que deve ser aplicada nos pacotes identificados pela assinatura;
- **arch:** define a arquitetura do *sandbox* que deve ser instanciado;
- **time:** representa a quantidade de minutos que o *sandbox* deve executar o *malware*.

Por fim, as assinaturas podem ser combinadas de modo a especificar um processo estruturado de análise. Na sequência, é descrito como as assinaturas foram sistematizadas no processo de investigação definido.

6.5.3 Processo de Análise

Foram desenvolvidos dois processos de análises baseado nas assinaturas de rede do Bashlite e Mirai. Esses processos foram incorporados no controlador SDN permitindo identificar automaticamente a família do *malware* que está sendo analisada e, como consequência, customizar o ambiente para a operação apropriada. Na sequência, são descritos os dois processos implementados para a avaliação.

6.5.3.1 Bashlite

O processo de análise relacionado à família Bashlite envolveu o uso de uma única assinatura de rede (*Bashlite C&C*). O Bashlite caracteriza-se por inicialmente estabelecer uma sessão TCP com o C&C e aguardar por instruções. Assim que o *malware* conecta no C&C, ele recebe uma mensagem *SCANNER ON* instruindo para que o dispositivo faça varreduras buscando propagar-se. Tendo em vista esse comportamento, foi desenvolvido um processo que busca evitar que essa propagação seja efetivada. O fluxo de execução do processo de avaliação é ilustrado na Figura 19.

Figura 19: Fluxo de execução da análise do *malware* Bashlite: o *malware* estabelece uma conexão com seu C&C e posteriormente realiza varreduras de propagação.

Fonte: Autor.

O controlador SDN avalia cada pacote gerado pelo *malware* e toma ações de repasse ou descarte. Assim que o endereço IP do C&C da *botnet* é detectado (*Assinatura Bashlite C&C*), o sistema automaticamente atualiza as regras de contenção permitindo apenas conexões para o endereço IP do C&C e um conjunto de serviços de infraestrutura, tais como DNS e NTP. Consequentemente, os ataques destinados aos sistemas externos – incluindo ataques de propagação – são sumariamente descartados.

6.5.3.2 Mirai

A investigação dos *malwares* da família Mirai envolveu o uso de duas assinaturas de rede: *Mirai C&C* e *Mirai SCAN*. O processo de inicialização do Mirai é diferente do comportamento apresentado pelo Bashlite. Assim que o *malware* é executado, o processo de autopropagação é iniciado instantaneamente. O *malware* contata o seu respectivo C&C somente após ter iniciado o processo de propagação do *malware*. Diante desse comportamento, o processo de análise deve ser capaz de distinguir o tráfego de propagação do tráfego do C&C. Como consequência, o sistema pode bloquear ataques mas permitir a comunicação com o C&C da *botnet*. A Figura 20 ilustra o fluxo de execução implementado pelo sistema proposto.

Figura 20: Fluxo de execução da análise do Mirai: assim que executado, o *malware* faz varreduras de propagação e concomitantemente conecta o C&C da *botnet*.

Fonte: Autor.

Os pacotes gerados pelo Mirai são inspecionados tendo em vista os padrões de tráfego definidos. Quando um ataque de propagação (*Mirai Scan*) é identificado, o próprio controlador SDN descarta os pacotes evitando que afetem sistemas de terceiros. Em conjunto, caso o controlador SDN identifique um fluxo de rede que possui a assinatura *Mirai C&C* – o endereço IP do controlador – a política de acesso do ambiente de análise é modificada. Dessa maneira, apenas são permitidos fluxos destinados ao IP do C&C e para um conjunto de serviços básicos de infraestrutura, como implementado no processo de execução do Bashlite.

6.6 Avaliação Experimental

Esta seção apresenta os experimentos definidos para analisar as duas famílias de *malwares* descritas. O objetivo é observar como as funcionalidades da arquitetura MARS podem aprimorar o processo de análise de *malware* em termos da qualidade dos resultados obtidos. A Tabela 11 sumariza os três experimentos definidos.

Tabela 11: Aspectos de configuração dos experimentos definidos.

	bloqueio de tráfego dinâmico	manipulação de pacotes (cabeçalho)	manipulação de pacotes (carga útil)
Experimento I	✓	-	-
Experimento II	-	✓	-
Experimento III	-	-	✓

Fonte: Autor.

No primeiro experimento investiga-se funcionalidades relacionadas com a contenção de tráfego. Mais especificamente, com a habilidade da arquitetura de identificar e bloquear o tráfego de rede malicioso gerado pelo *malware*. Nos últimos dois experimentos são investigados mecanismos de manipulação de tráfego de rede que permitem interagir com as conexões desempenhadas pelo *malware*.

6.6.1 Experimento I: Contenção de Acessos de Rede

Esse experimento investiga como a arquitetura pode adaptar a política de contenção dos acessos à rede dinamicamente, bloqueando ataques realizados pelo *malware*, mas permitindo a comunicação com o C&C da *botnet*. Para isso, foram analisados

exemplares das duas famílias de *malwares* – Bashlite e Mirai – os quais foram executados por um período de 24 horas. Apesar de serem distintas, as famílias exibem um comportamento de rede muito peculiar. Por exemplo, a Figura 21 sumariza a taxa de pacotes por hora originadas na execução do *malware* Bashlite.

Figura 21: Bashlite: número de pacotes por hora gerados pelo *malware* analisado.

Fonte: Autor.

Como ilustrado, a maioria dos pacotes são associados aos ataques de propagação, *i.e.*, pacotes destinados à porta 23/TCP (TELNET). O ataque de varredura inicia assim que o *bot* contata o C&C e persiste de forma praticamente constante durante todo o período da análise (24 horas). Observa-se uma pequena quantidade de pacotes associados à comunicação com o C&C da *botnet* que representam basicamente mensagens de *keep-alive* e comandos de ataques. Em particular, durante a análise foram detectadas cinco instruções de ataques encaminhadas para a máquina infectada. Esses comandos representam instruções de ataques de negação de serviços almejando o serviço HTTP (GET e POST) destinadas para cinco endereços IP distintos. Devido às características do ataque, o número de pacotes não é representativo no gráfico. Para ser efetivo, ataques DoS destinados aos serviços TCP requerem uma sessão estabelecida. No entanto, no sistema implementado, esses pacotes de ataques são subitamente bloqueados impedindo que a sessão TCP seja estabelecida. Apesar de bloquear esses ataques, a arquitetura ainda assim pode mapear as características dos mesmos, tal como, alvos e aspectos dos pacotes.

O Mirai possui comportamento de rede muito similar. No entanto, o volume de tráfego de saída é significativamente diferente do Bashlite. A Figura 22 realça o comportamento de rede no sistema de análise. É importante notar o grande volume de pacotes associados à assinatura de *Mirai SCAN*. Esses pacotes correspondem a ataques destinados às portas 23/TCP e 2323/TCP buscando realizar ataques de força bruta. Devido ao volume de propagação da varredura, os pacotes destinados ao C&C são indistinguíveis na linha do gráfico. Além de mensagens *keep-alive*, foi detectado um único comando de ataque. Esse comando instruiu a máquina infectada a atacar um (1) IP alocado na Rússia utilizando ataque DoS do tipo *syn flood*. Esse ataque foi realizado por 10 segundos e gerou 92419 pacotes.

Figura 22: Mirai: número de pacotes por hora gerados pelo *malware* analisado.

Fonte: Autor.

Neste experimento, a taxa de pacotes de saída manteve-se constante, correspondendo à taxa de transmissão de 120 pacotes por segundos. No pico do gráfico (ataque), o sistema gerou 9361 pacotes por segundo, chegando à capacidade máxima de vazão do dispositivo de análise.

Para concluir esse experimento, comparou-se o tráfego gerado pelos dois exemplares. Conforme ilustrado na Figura 23, o Mirai realizou uma propagação muito mais agressiva que o Bashlite. Em geral, o número de pacotes realizados pelo Mirai é 13

vezes superior que o Bashlite. Adicionalmente, foi observado um comportamento similar nas duas análises: mesmo durante as instruções de ataques, os pacotes de propagação são contínuos e mantiveram-se praticamente estáveis.

Figura 23: Bashlite *versus* Mirai: comparação do número de pacotes gerados pelos exemplares analisados.

Fonte: Autor.

Os *malwares* das famílias Bashlite e Mirai são muito ativos e, sem as devidas políticas de contenção, podem alcançar um grande número de sistemas de terceiros. Na investigação apresentada, o controlador SDN identificou e bloqueou comandos de ataques das duas famílias de *malware*. O experimento também foi bem-sucedido ao monitorar a comunicação com o C&C da *botnet*, provendo informações valiosas relacionadas com os ataques.

6.6.2 Experimento II: Manipulação de Pacotes

Neste experimento foi estudada a capacidade do sistema de reescrever características de pacotes durante a análise de *malware*. Mais do que reescrever o cabeçalho do pacote, foi modificado o conteúdo dos pacotes (carga útil) de modo a manipular comandos de ataques. Para ilustrar esse processo, examinou-se a comunicação entre o *bot* Bashlite e seu respectivo C&C. Em especial, foi implementada uma assinatura que identifica mensagens contendo instruções de varredura, assim como a mensagem

SCANNER ON (veja Figura 15). Quando o controlador SDN detecta tal instrução, o pacote é interceptado e seu conteúdo é reescrito para SCANNER OFF e assim repassado para o *sandbox*. Esse processo de reescrita evita que o dispositivo infectado realize ataques de varreduras, o qual deveria ser efetuado em resposta à mensagem SCANNER ON. A Listagem 6.2 ilustra um fragmento da conversação onde a instrução é reescrita pelo controlador SDN.

Listagem 6.2: Processo de reescrita de instruções do Bashlite.

```

1  T 192.0.2.30:41828 -> 46.XX.XX.XX:666 [AP]
2  .[32mDEVICE CONNECTED .[31m-> .[37m192.0.2.30|. [32mTYPE .[31m-> .[37mSERVER.
3  T 46.XX.XX.XX:666 -> 192.0.2.30:41828 [AP]
4      !* SCANNER ON.
5  T 46.XX.XX.XX:666 -> 192.0.2.30:41828 [AP]
6      !* SCANNER OFF.
7  T 46.XX.XX.XX:666 -> 192.0.2.30:41828 [AP]
8      PING
9  T 192.0.2.30:41828 -> 46.XX.XX.XX:666 [AP]
10     PONG.

```

Nessa conversação, destaca-se a comunicação do *malware* Bashlite sendo executado no *sandbox* (192.0.2.30) com o seu respectivo C&C (46.XX.XX.XX). Inicialmente, o *malware* estabelece uma sessão TCP com seu C&C encaminhando um *banner* de identificação. Assim que essa comunicação é estabelecida, o C&C da *botnet* envia uma mensagem SCANNER ON que é interceptada pelo controlador SDN. Essa mensagem é reescrita e encaminhada para o *sandbox*, como consequência, o *malware* não executa uma varredura, mas inicializa o processo de troca de mensagens de PING e PONG.

A capacidade de reescrita do conteúdo de pacotes é um recurso flexível e pode ser expandido para outras mensagens do protocolo, não apenas para instruções da *botnet*. Com exceções de comunicações criptografadas, outras conversações podem ser manipuladas, incluindo mensagens do protocolo HTTP e SMTP.

6.6.3 Experimento III: Encaminhamento de *Bot*

Neste experimento, buscou-se redirecionar um *bot* para um C&C ilegítimo, localizado na rede de análise. Como esperado, os *malwares* possuem informações embutidas a respeito do seu C&C da *botnet*. No instante em que um determinado *malware* é analisado o seu C&C pode não estar ativo; afinal, o C&C pode estar temporariamente desativado ou ter sido desabilitado. Neste contexto, analisar um *malware* pode não revelar informações detalhadas sobre o processo de ingresso na *botnet*, tal como parâmetros de comunicação e credenciais utilizadas. Para lidar com esse problema, a arquitetura MARS permite redirecionar o C&C requisitado para outro com funcionalidades homólogas.

Para explorar o processo de encaminhamento de *bots*, foram realizados dois testes experimentais: redirecionamento local e redirecionamento externo. No primeiro teste, exemplares do Bashlite foram redirecionado para um servidor local que impersona um controlador C&C (descrito na Seção 6.5). Como resultado, pode-se coletar as mensagens iniciais do processo de registro no C&C da *botnet*, revelando *banners* de identificação que expõem aspectos de configuração do *sandbox*, os quais podem ser utilizados para observar as características de variantes de *malwares*. A Listagem 6.3 ilustra o conjunto de *banners* obtidos no C&C emulado.

Listagem 6.3: *Banners* coletados de diferentes *bots* da família Bashlite analisados.

```
1 [Bash-Lite] [Connected [SERVER] [192.0.2.30]
2 SERVER Connected [192.0.2.30]
3 BUILD SERVER:[192.0.2.30]
4 [DEVICE CONNECTED] -> [192.0.2.30|TYPE -> SERVER]
```

Na segunda investigação, o *bot* é redirecionado para uma *botnet* ilegítima. Ou seja, faz com que o *bot* contate um IP externo diferente daquele para o qual ele foi designado. No experimento realizado foi possível confirmar a viabilidade dessa redireção, pois a maioria das IoT não incorpora mecanismos de autenticação. Alguns C&Cs de *botnet*, no entanto, para evitar reinfecção, restringem o número de conexões baseadas no endereço IP de origem. É comum receber mensagens de desconexão, tal como DUP e KILLMYSELF quando múltiplos *bots* do mesmo endereço IP tentam registrar-se

no C&C da *botnet*. Muito embora, se necessário, um analista pode manipular tal comportamento e reescrever essas mensagens de maneira a evitar a desconexão do *bot*.

De forma complementar, uma implementação mais elaborada do software que emula o C&C da *botnet* pode incorporar outras funcionalidades, incluindo injeção de pacotes. Com isso, pode-se forjar uma mensagem de ataque e estudar as ações que essa mensagem desencadeará no dispositivo infectado.

6.7 Discussão

A segurança dos dispositivos IoT representa uma ameaça significativa para a Internet, pois a maioria desses dispositivos não implementa as melhores práticas de segurança. Esses dispositivos possuem falhas de segurança e, normalmente, não recebem atualizações de software regulares para corrigir suas deficiências. Diante disso, nota-se a proliferação de *malwares* que exploram as fraquezas desses dispositivos para realizar ataques poderosos de negação de serviço distribuído. Uma forma de proteger os sistemas desses ataques é investigar tais ameaças com o objetivo de aprimorar os mecanismos de defesa.

Neste capítulo, foi descrita uma abordagem de análise de *malware* que se adapta segundo ações realizadas pelo *malware*. Para isso, foram utilizadas assinaturas para minimizar o tráfego de rede para outros dispositivos sem afetar a qualidade dos resultados. Para avaliar a solução proposta, foram analisados exemplares de *malwares* das famílias Bashlite e Mirai utilizando três experimentos. O primeiro experimento foca na contenção de tráfego de rede, no qual a política de acesso é adaptada conforme as conexões realizadas pelo *malware*. No segundo experimento é explorada a capacidade de reescrita de pacotes e, como resultado, foi possível manipular mensagens do protocolo fazendo com que o *malware* exibisse novos comportamentos de rede. Por fim, no terceiro experimento, foram exploradas as características de manipulação dos fluxos de rede fazendo que o *malware* fosse redirecionado para um C&C ilegítimo.

6.8 Sumário do Capítulo

Este capítulo busca responder as perguntas de pesquisas **P2** e **P3** descritas na Seção 1.3. A pergunta **P2** é respondida com a análise dos resultados do Experimento I e Experimento II. Essa investigação demonstrou que a arquitetura foi efetiva para desencadear novos aspectos comportamentais dos *malwares* projetados para IoT. A pergunta **P3** é respondida com a análise dos resultados do Experimento II no qual os comandos de ataque da *botnet* são manipulados de modo a modificar seus aspectos comportamentais. Por fim, nota-se que a flexibilidade provida pela arquitetura MARS permitiu identificar novos comportamentos de rede, e também aprimorar o processo de análise de *malware* mitigando ataques a sistemas de terceiros.

7 CONSIDERAÇÕES FINAIS

Este capítulo discute as principais contribuições desta tese e apresenta a possível continuidade da pesquisa. Para tal, este capítulo está estruturado da seguinte maneira: a Seção 7.1 apresenta as conclusões deste trabalho. A Seção 7.2 recapitula as respostas para as perguntas de pesquisas propostas. A Seção 7.3 apresenta as limitações desta arquitetura proposta. Por fim, a Seção 7.4 discute os próximos passos para a continuidade da pesquisa.

7.1 Conclusões

Arquivos maliciosos modernos podem adaptar o seu comportamento diante do contexto em que são executados. As soluções atuais apresentam limitações para detectar essas variações, sobretudo quando essas adaptações são ocasionadas pela configuração do ambiente que circunda o *sandbox*. Diante disso, esta tese buscou aprimorar o processo de investigação de *malware* propondo uma arquitetura flexível para analisar códigos maliciosos na qual podem-se detectar variações comportamentais de maneira automática.

Esta tese apresentou uma arquitetura para análise de *malware* denominada MARS *MAIwaRe Analysis Architecture based on SDN* – que emprega a tecnologia SDN para suprir as limitações das soluções atuais. MARS utiliza a camada de rede para integrar a operação do *sandbox* com os demais componentes do ambiente de análise. Com isso, o ambiente de análise pode assumir múltiplas configurações em termos de disponibilidade de serviços, topologia e política de acesso. Adicionalmente, o ambiente pode modificar suas características dinamicamente baseado nas ações do *malware* e, deste modo, moldar o ambiente conforme recursos demandados.

Para avaliar a solução proposta foram definidas duas avaliações experimentais: na primeira, foi analisado um conjunto heterogêneo de *malwares* compostos por códigos maliciosos modernos; na segunda, foram analisados *malwares* projetados para dispositivos IoT. A análise dos resultados experimentais possibilitou obter as seguintes conclusões:

- a arquitetura proposta permitiu mapear alterações comportamentais dos *malwares* de maneira automatizada;
- a integração da camada de rede junto ao processo de análise demonstrou-se eficiente para controlar o tráfego gerado pelo *malware* e também para controlar os demais dispositivos presentes no ambiente de análise;
- o processo de detecção de variações comportamentais deve considerar múltiplos parâmetros a fim de evitar imprecisões;
- os recursos demandados pelos *malwares* fazem que novas ações sejam mapeadas e, em última análise, que variações comportamentais sejam identificadas;
- o controle unificado do ambiente de análise permite um analista desenvolver ambiente complexos fazendo o uso de arquivos de configuração. Com isso, um *malware* pode ser analisado em diferentes cenários sem depender da intervenção humana para configurar manualmente os elementos, bem como definir a política de acesso em cada execução do *malware*;
- a avaliação experimental demonstrou que simples modificações no ambiente de análise podem desencadear novas ações comportamentais dos *malwares*;
- a política de acesso disponibilizada para o *malware* tende a afetar as suas ações comportamentais. Alguns *malwares* adaptam suas ações de modo a superar as limitações impostas pela política de acesso presente no ambiente;
- o encaminhamento de tráfego e manipulação de pacotes DNS foram funcionalidades efetivas para desencadear comportamentos inéditos na maioria dos *malwares* analisados.

Mais do que revelar novos comportamentos maliciosos, os experimentos revelaram que MARS fornece funcionalidades que não estão presentes nas soluções existentes. A flexibilidade provida pela arquitetura MARS permite determinar diferentes configurações para a infraestrutura de análise, bem como interagir com as conexões geradas pelo *malware*. Com isso, um analista pode especificar um processo de investigação automatizado no qual um determinado *malware* pode ser analisado nos diferentes cenários identificando suas variações comportamentais.

7.2 Respostas às Perguntas de Pesquisa

Assim como proposto na Seção 1.3, para investigar a efetividade da solução proposta, buscou-se responder um conjunto de perguntas de pesquisa. Nesta seção essas perguntas são lembradas e respondidas.

P1: *A arquitetura proposta é capaz de detectar malwares que apresentam variações comportamentais tendo em vista a configuração do ambiente que circunda o sandbox?*

Tendo como base os resultados apresentados na avaliação experimental do Capítulo 5, foi possível afirmar que as funcionalidades incorporadas na arquitetura proposta foram efetivas para desencadear comportamentos não observados por soluções tradicionais. Em particular, foi possível observar a adaptação dos *malwares* frente a política de acesso e também detectar ações comportamentais decorrentes da reconfiguração do ambiente de análise.

P2: *É possível especificar uma arquitetura flexível de tal forma que possa ser utilizadas em diferentes domínios de análise?*

Essa pergunta **P2** foi respondida com base nos resultados de duas avaliações experimentais apresentados no Capítulo 5 e Capítulo 6. A análise experimental demonstrou que a MARS é flexível para atuar em diferentes contextos sendo efetiva para mapear características inéditas de *malwares* projetados para IoT, assim como *malwares* avançados projetados para sistemas tradicionais. O processo de investigação de *malwares* em diferentes domínios envolve pequenos ajustes na infraestrutura da arquitetura MARS, como por exemplo, definição do *sandbox* e dos elementos que irão compor o ambiente de análise. No entanto, as funcionalidades de reconfiguração e controle do ambiente são transparentes, uma vez que todo o processo é gerido pela camada de rede.

P3: *A arquitetura MARS permite que um analista interaja com os fluxos de rede de modo a estimular malwares a revelar detalhes de suas funcionalidades?*

Essa pergunta foi abordada no Capítulo 6, no qual um conjunto de *malwares* projetados para IoT é analisado. Os resultados da avaliação experimental demonstraram a adaptação do ambiente de análise permitindo bloquear ataques de *malwares* e, ao mesmo tempo, manipular comandos de ataques. A manipulação dos fluxos de rede permitiram redirecionar ataques para um servidor local e, através da reescrita de pacotes, modificar os aspectos das requisições do *malware* intervindo nos ataques.

7.3 Limitações

A arquitetura proposta é dependente das atividades realizadas pelo *malware* executado no *sandbox*. Isso torna o *sandbox* um elemento crítico na infraestrutura de análise. Os resultados obtidos na arquitetura MARS são prejudicados quando o *malware* não é executado apropriadamente no *sandbox*, seja pelo uso de técnicas *anti-sandbox* ou pela ausência de recursos. Essa limitação afeta a arquitetura MARS, mas também todas as soluções que fazem uso de *sandbox* para investigar códigos maliciosos. Portanto, para aprimorar os sistemas frente essa limitação, é importante investigar contramedidas para prevenir a evasão dos *malwares* das ferramentas de análise, bem como utilizar configurações customizadas para evitar a sua identificação. Em particular, os recursos providos pela arquitetura MARS, como manipulação de pacotes e política de contenção, também podem ser identificados por *malwares* que inspecionam características dos sistemas. Consequentemente, deve-se estudar técnicas que dificultam a identificação de tais recursos e implementar mecanismos que garantam a integridade dos componentes que compõe o ambiente de análise.

A arquitetura MARS também está vulnerável a ataques destinados aos modelos SDN. Caso o controlador SDN seja comprometido, o sistema torna-se impreciso, afetando o processo de investigação de *malwares*. Mesmo não sendo o escopo deste trabalho, é essencial considerar esses aspectos para mapear vulnerabilidades do sistema e aprimorar sua segurança.

7.4 Trabalhos Futuros

Como trabalho futuro, deseja-se desenvolver uma biblioteca com diferentes configurações para a arquitetura. Dessa maneira, as configurações podem ser utilizadas por analistas para compor processos customizados de análise de *malware* de maneira simplificada. Nesta biblioteca de configurações, por exemplo, seriam disponibilizadas:

- **assinatura de rede** - características comportamentais de rede que podem ser utilizadas para gerar eventos de configuração na arquitetura. Por exemplo: assinatura de varredura do *malware* Mirai;
- **política de contenção** - política de acesso disponibilizada para o ambiente de análise, como por exemplo: acesso à Internet exclusivamente utilizando porta 80/TCP;
- **manipulação de tráfego** - regras associadas à manipulação dos pacotes gerados pelo *malware*, tais como: regras que reescrevem respostas de requisições DNS.

Adicionalmente, pretende-se implementar um conjunto de procedimentos autônomos que visam facilitar a análise dos *malwares*. Por exemplo, um método de investigação automático no qual um mesmo *malware* é executado em diferentes configurações do ambiente de análise e, como resultado, apresente um relatório com as ações comportamentais do *malware* nos respectivos cenários de avaliação. Essa investigação automatizada permite observar aspectos comportamentais do *malware* sem a necessidade de customização prévia da arquitetura. A partir disso, um analista tem uma visão geral das ações do *malware* e pode começar a adaptar a arquitetura conforme necessário.

REFERÊNCIAS

- ANTONAKAKIS, M. et al. Understanding the Mirai Botnet. In: **26th USENIX Security Symposium (USENIX Security 17)**. Vancouver, BC: USENIX Association, 2017. p. 1093–1110. ISBN 978-1-931971-40-9. Disponível em: <https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis>.
- BALL, T.; RAJAMANI, S. K. The S LAM project: debugging system software via static analysis. In: ACM. **ACM SIGPLAN Notices**. ACM, 2002. v. 37, n. 1, p. 1–3. ISBN 978-1-4503-1083-3. Disponível em: <http://dl.acm.org/citation.cfm?id=2103656>.
- BALZAROTTI, D. et al. Efficient detection of split personalities in malware. In: **NDSS**. The Internet Society, 2010. Disponível em: <http://www.isoc.org/isoc/conferences/ndss/10/>.
- BASHLITE. **The malware Bashlite source code**. 2017. Acessado em Novembro de 2017. Disponível em: <https://github.com/anthonygtellez/BASHLITE/>.
- CERON, J. M.; MARGI, C. B.; GRANVILLE, L. Z. MARS: an SDN-based malware analysis solution. In: IEEE. **Computers and Communication (ISCC), 2016 IEEE Symposium on**. IEEE Computer Society, 2016. p. 525–530. ISBN 978-1-5090-0679-3. Disponível em: <http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=7536670>.
- CHEN, X. et al. Towards an understanding of anti-virtualization and anti-debugging behavior in modern malware. In: IEEE. **Dependable Systems and Networks With FTCS and DCC, 2008. DSN 2008. IEEE International Conference on**. IEEE Computer Society, 2008. p. 177–186. ISBN 978-1-4244-2397-2. Disponível em: <http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=4610267>.
- CONSTANTINOS, K. et al. DDoS in the IoT: Mirai and Other Botnets. **Computer**, IEEE Computer Society, Los Alamitos, CA, USA, v. 50, n. 7, p. 80–84, 2017. ISSN 0018-9162.
- DETUX. Detux - The Linux Sandbox. outubro 2017. Acessado em Novembro de 2017. Disponível em: <https://github.com/detuxsandbox/detux>.
- DINABURG, A. et al. Ether: malware analysis via hardware virtualization extensions. In: ACM. **Proceedings of the 15th ACM conference on Computer and communications security**. 2008. p. 51–62. ISBN 978-1-4503-4946-8. Disponível em: <http://doi.acm.org/10.1145/3133956>.
- DynDNS. Dyn Statement on 10/21/2016 DDoS Attack. outubro 2017. Acessado em Novembro de 2017. Disponível em: <http://dyn.com/blog/dyn-statement-on-10212016-ddos-attack/>.
- EGELE, M. et al. A survey on automated dynamic malware-analysis techniques and tools. **ACM Computing Surveys (CSUR)**, ACM, v. 44, n. 2, p. 6, 2012.

ELZEN, I. van der; HEUGTEN, J. van. **Techniques for detecting compromised IoT devices**. Amsterdam, The Netherlands, The Netherlands, 2017. v. 1, n. 1, 10 p.

ESLER, J.; NOGEL, H.; JUDGE, T. **ClamAV**. 2017. Acessado em Novembro de 2017. Disponível em: <<http://www.clamav.net/>>.

FAINELLI, F. The OpenWrt embedded development framework. In: **Proceedings of the Free and Open Source Software Developers European Meeting**. Atlanta, Georgia, USA: ACM, 2008. ISBN 978-1-59593-995-1.

FLECK, D. et al. Pytrigger: A system to trigger & extract user-activated malware behavior. In: **2013 International Conference on Availability, Reliability and Security, ARES 2013, Regensburg, Germany, September 2-6, 2013**. Regensburg, Germany, September: IEEE Computer Society, 2013. p. 92–101. ISBN 978-0-7695-5008-4. Disponível em: <<http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6656131>>.

GRAZIANO, M.; LEITA, C.; BALZAROTTI, D. Towards network containment in malware analysis systems. In: ACM. **28th Annual Computer Security Applications Conference, ACSAC 2012, Orlando, FL, USA, 3-7 December 2012**. 2012. p. 339–348. Disponível em: <<http://doi.acm.org/10.1145/2420950.2421000>>.

GUDE, N. et al. Nox: towards an operating system for networks. **ACM SIGCOMM Computer Communication Review**, ACM, v. 38, n. 3, p. 105–110, 2008.

HALEPLIDIS, E. et al. **Software-Defined Networking (SDN): Layers and Architecture Terminology**. Fremont, USA, 2015. <<http://www.rfc-editor.org/rfc/rfc7426.txt>>. Disponível em: <<http://www.rfc-editor.org/rfc/rfc7426.txt>>.

HOLZ, T.; RAYNAL, F. Detecting honeypots and other suspicious environments. In: IEEE. **Information Assurance Workshop, 2005. IAW'05. Proceedings from the Sixth Annual IEEE SMC**. West Point, New York, USA, 2005. p. 29–36.

HoneyTARG. **Distributed Honeypots Project**. 2017. Acessado em Novembro de 2017. Disponível em: <<http://honeytarg.cert.br/honeypots/>>.

HU, H. et al. Flowguard: building robust firewalls for software-defined networks. In: ACM. **Proceedings of the third workshop on Hot topics in software defined networking**. New York, USA, 2014. p. 97–102.

JIN, R.; WANG, B. Malware detection for mobile devices using software-defined networking. In: **Proceedings of The 2014 IAJC/ISAM Joint International Conference**. Orlando, FL, USA: IEEE Computer Society, 2014. (IAJC/ISAM '14). ISBN 978-1-60643-379-9.

KIRAT, D.; VIGNA, G.; KRUEGEL, C. Barecloud: bare-metal analysis-based evasive malware detection. In: **Proceedings of the 23rd USENIX conference on Security Symposium (SEC'14)**. USENIX Association, Berkeley, CA, USA. USENIX, 2014. p. 287–301. Disponível em: <<https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/kirat>>.

KREBS, B. KrebsOnSecurity Hit With Record DDoS. outubro 2017. Acessado em Novembro de 2017. Disponível em: <http://dyn.com/blog/dyn-statement-on-10212016-ddos-attack/>.

KREIBICH, C. et al. Gq: Practical containment for measuring modern malware systems. In: ACM. **Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference**. ACM, 2011. p. 397–412. ISBN 978-1-4503-0797-0. Disponível em: <http://doi.acm.org/10.1145/2018436>.

KREUTZ, D.; RAMOS, F.; VERISSIMO, P. Towards secure and dependable software-defined networks. In: **Proceedings of ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN'13)**. New York, USA: ACM, 2013. ISBN 978-1-4503-2056-6. Disponível em: <http://dl.acm.org/citation.cfm?id=2486001>.

KRUEGEL, C.; KIRDA, E.; BAYER, U. TAnalyze: A tool for analyzing malware. In: **Proceedings of the 15th European Institute for Computer Antivirus Research (EICAR 2006) Annual Conference**. Vienna, Austria: EICAR, 2006. v. 4.

LAND, J. Systemic Vulnerabilities in Customer-Premises Equipment (CPE) Routers. Software Engineering Institute, 2017. CMU/SEI-2017-SR-019, July 2017.

LANGNER, R. Stuxnet: Dissecting a cyberwarfare weapon. **Security & Privacy, IEEE**, IEEE, v. 9, n. 3, p. 49–51, 2011.

LAURIANNE, M. **Bot Software Spreads, Causes New Worries**. 2004. Disponível em: <https://doi.org/10.1109/MDSO.2004.7>.

LIANG, Z.; YIN, H.; SONG, D. Hookfinder: Identifying and understanding malware hooking behaviors. **Department of Electrical and Computing Engineering**, p. 41, 2008.

LIGH, M. et al. **Malware Analyst's Cookbook and DVD: Tools and Techniques for Fighting Malicious Code**. 1. ed. Wiley, 2010. Paperback. ISBN 0470613033. Disponível em: <http://www.worldcat.org/isbn/0470613033>.

LINDORFER, M.; KOLBITSCH, C.; COMPARETTI, P. M. Detecting environment-sensitive malware. In: SPRINGER. **Recent Advances in Intrusion Detection**. 2011. p. 338–357. Disponível em: https://doi.org/10.1007/978-3-642-23644-0_18.

MATLOFF, N.; SALZMAN, P. J. **The Art of Debugging with GDB, DDD, and Eclipse**. San Francisco, CA, USA: No Starch Press, 2008. ISBN 1593271743, 9781593271749.

MCKEOWN, N. et al. Openflow: enabling innovation in campus networks. **SIGCOMM Comput. Commun. Rev.**, ACM, New York, NY, USA, v. 38, n. 2, p. 69–74, mar. 2008. ISSN 0146-4833.

MCLAUGHLIN, L. Bot software spreads, causes new worries. **Distributed Systems Online, IEEE**, IEEE, v. 5, n. 6, p. 1, 2004.

MIRAI. **The malware Mirai source code**. 2017. Acessado em Novembro de 2017. Disponível em: <https://github.com/jgamblin/Mirai-Source-Code/>.

MOSER, A.; KRUEGEL, C.; KIRDA, E. Exploring multiple execution paths for malware analysis. In: IEEE. **Security and Privacy, 2007. SP'07. IEEE Symposium on**. IEEE Computer Society, 2007. p. 231–245. ISBN 0-7695-2848-1. Disponível em: <http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=4223200>.

NARANG, P.; REDDY, J. M.; HOTA, C. Feature selection for detection of peer-to-peer botnet traffic. In: ACM. **Proceedings of the 6th ACM India Computing Convention**. Bangalore, India, 2013. p. 16.

NORMAN. **Norman sandbox technical repor**. New York, USA, 2003.

NUNES, B. et al. A survey of software-defined networking: Past, present, and future of programmable networks. **Communications Surveys Tutorials, IEEE**, v. 16, n. 3, p. 1617–1634, Third 2014. ISSN 1553-877X.

OKTAVIANTO, D.; MUHARDIANTO, I. **Cuckoo Malware Analysis**. New York, USA: Packt Publishing Ltd, 2013.

ONF. **Open Networking Foundation**. 2017. Acessado em Novembro de 2017. Disponível em: <https://www.opennetworking.org/>.

PA, Y. M. P. et al. Iotpot: Analysing the rise of iot compromises. **EMU**, v. 9, p. 1, 2015. Disponível em: <https://www.usenix.org/conference/woot15/workshop-program/presentation/pa>.

PAITHANE, S. et al. **System and method for detecting time-bomb malware**. San Francisco, CA, USA: Google Patents, 2017. US Patent 9,536,091.

PFAFF, B. et al. The design and implementation of open vswitch. In: **NSDI**. Oakland, CA, USA: NSDI, 2015. p. 117–130.

PROVOS, N. Honeyd-a virtual honeypot daemon. In: **10th DFN-CERT Workshop, Hamburg, Germany**. Germany: DFN-CERT, 2003. v. 2, p. 4.

PROVOS, N.; HOLZ, T. **Virtual Honeypots - From Botnet Tracking to Intrusion Detection**. New York, NY: Addison-Wesley, 2008. I-XXIII, 1-440 p. ISBN 978-0-321-33632-3.

ROWE, D. C.; WILKINSON, L. Leveraging next-generation virtualization technologies for advanced malware analysis in the classroom. 2014.

SANTOS, C. dos et al. Botnet master detection using a mashup-based approach. In: **Network and Service Management (CNSM), International Conference on**. IEEE, 2010. p. 390–393. ISBN 978-1-4244-8910-7. Disponível em: <http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=5668576>.

SECURITYLABS, G. D. Uroburos highly complex espionage software with russian roots. **G Data Software AG**, 2014.

SHI, H.; ALWABEL, A.; MIRKOVIC, J. Cardinal pill testing of system virtual machines. In: . Berkeley, CA, USA: USENIX Association, 2014. (SEC'14), p. 271–285. ISBN 978-1-931971-15-7.

STEIN, B. Fuzzy-fingerprints for text-based information retrieval. In: **Proceedings of the 5th international conference on knowledge management (I-KNOW 05), Graz, Journal of Universal Computer Science**. Graz, Austria: ACM, 2011. p. 572–579.

US-CERT. Heightened DDoS Threat Posed by Mirai and Other Botnets. In: UNITED STATES COMPUTER EMERGENCY READINESS TEAM. Washington, USA: Alert TA16-288A, 2016.

VIRTI, É. S. et al. Honeypots as a security mechanism. **MonAm (2006 set.: Tübingen, Germany). Proceedings of the IEEE/IST. Tübingen: IEEE, 2006.**, 2006.

VIRVILIS, N.; GRITZALIS, D. The big four-what we did wrong in advanced persistent threat detection? In: IEEE. **Availability, Reliability and Security (ARES), 2013 Eighth International Conference on**. Regensburg, Germany, September: IEEE Computer Society, 2013. p. 248–254. ISBN 978-0-7695-5008-4. Disponível em: <http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6656131>.

WILLEMS, C.; HOLZ, T.; FREILING, F. Toward automated dynamic malware analysis using cwsandbox. **IEEE Security & Privacy**, IEEE, n. 2, p. 32–39, 2007.

GLOSSÁRIO

Ambiente de Análise	<i>estrutura de rede dedicada à análise de malwares na qual um sandbox é instanciado para mapear as ações de um malware.</i>
Análise Dinâmica	<i>metodologia de análise de malware que consiste em executar um arquivo suspeito e mapear o seu comportamento.</i>
Análise Estática	<i>metodologia de análise de malware que busca avaliar o funcionamento de um programa sem a execução do mesmo.</i>
Bot	<i>tipo de código malicioso que permite com que o sistema comprometido seja controlado remotamente.</i>
Botnet	<i>uma rede de máquinas comprometidas aptas a serem controladas remotamente.</i>
C&C	<i>estrutura de comando e controle utilizada para gerenciar uma botnet.</i>
Fast-flux	<i>é uma técnica na qual múltiplos endereços IP são associados a um mesmo nome de domínio. Através de uma configuração, esses IP são alterados rapidamente fazendo com que o nome do domínio corresponda a um IP diferente.</i>
Firewall	<i>dispositivo de rede que busca aplicar uma política de segurança de modo a controlar o tráfego de entrada e saída de rede.</i>
Keylogger	<i>código malicioso que se caracteriza por coletar e armazenar as teclas digitadas por usuários do sistema infectado.</i>
Malware	<i>códigos maliciosos especificamente desenvolvidos para</i>

causar danos a sistemas computacionais sem o consentimento do usuários.

Proxy *sistema dedicado que atua como um intermediário para comunicações de rede.*

Sandbox *sistema controlado preparado para executar arquivos suspeitos e mapear as suas ações no sistema.*

Screenlogger *código malicioso que se caracteriza por coletar imagens da tela do sistema infectado.*

APÊNDICE A – MALWARES ANALISADOS

A listagem abaixo ilustra detalhes dos 100 malwares que tiveram o seu comportamento analisado pelas arquiteturas definidas. Nessa listagem é apresentado o *hash* criptográfico de cada *malware* e sua respectiva assinatura obtida pelo antivírus ClamAV em outubro de 2017.

Malware (hash SHA-1)	Assinatura
00590628752de21c18d444f625261692c12a287e	Win.Trojan.Agent-30575
0320ee1aca89fb00ade98a4d546c8e07dade01ec	Win.Trojan.Sdbot-28
0931655cd5eac3abb6d3bace2b8c7e408c65db48	Win.Trojan.Downbot-3
0d3c49bf681e6128a7c3c663d6eb5701af70759e	Sem assinatura
1164fa9eb0ca28d23b7cd48f731b89f422c2d230	Sem assinatura
11cd20c734c90d09c8faa5b6e0964d27a666938e	Sem assinatura
1571105b3fcabc117dd7c9308286af30d34787a92	Sem assinatura
19aee19f5ba66743da4f59858df1cb162841b572	Sem assinatura
1ae0c8b0190b05c72b902779f1e471a3bb4feedf	Win.Trojan.Perl-35
1dda56a95ff272d627c89c6a7b37975790c8fcff	Sem assinatura
2872de82a851dbd4fe27b51459c5fb3ccfcdcbcf	Heuristics.W32.Polipos.A
2aa58c26deac88c63b544318da7ab08e2a5ffff97	Sem assinatura
2ca57931c0e0347200f3fed1d188fb4152fb67a5	Win.Trojan.Agent-30684
2ec7139d5fdb02fb1e7557a7339af5c91fb21ed7	Sem assinatura
30f6a8dba8d73b867c05e16a486f2ca6cdb6e104	Win.Worm.Deadhat-1
325acdf79a66cb744c48363a7d97f19fbaf79c20	Sem assinatura
32d3604c26a4bb339136119b26ec69c32c156bff	Win.Trojan.Agent-400667
3794d50471d7a937241efa60e0b847c65e54c4c8	Win.Trojan.Agent-442576
39fd1b1fd3b7f1c11ef67be8d07309446fc41aae	Sem assinatura
3a93cddb50d96d1472ddedcd8fc2d146ea489ce9	Win.Trojan.Downloader-67059
3e5879ba2b137f949732c918661e73f18baf0352	Win.Trojan.Autoit-2022
3f752c5c97e80d41f5498d8a90f5ee8afd1c4372	Win.Trojan.Agent-521064
4072c65ef571735157453e4c7f68a210df62948f	Win.Trojan.Cossta-71
42081abf164c948d19075b90e169747f390924f8	Sem assinatura
44627df5595278068872eb1277d2c188907baf94	Win.Trojan.Dalbot-1
44fbd1e9d7d0c0a7f3eb5e9de2e640bf5d64e2e2	Win.Trojan.Ircbot-1256
48eb49bcd11b7d27e66def91b5a08b7ecbf1ac2	Win.Downloader.83571-1
4d053b26ad52a8248c80ddde63d4577f27a12dd1	Win.Trojan.Agent-30657

Continua na próxima página

Malware (hash SHA1)	Assinatura
4d57817bfbf70e06fb7dd5c6a8d340443fd6cfc4	Sem assinatura
4d5d2bc7a7008ef5f6920495e2ba96b14d2d96d7	Win.Trojan.Tomvode-1
4f9ac6ed2eb74a8bb8eb5d23d834c2be6f1ddaf7	Win.Trojan.Agent-30696
4faf2bc5ae8ec46c89cb305e8dce25a50c8b697c	Sem assinatura
556aa22e34c05711cece7ca1fc53bbbf2ad977cf	Win.Trojan.Scar-989
58f2e511843d7346f1baba3a77a84a24205696fd	Sem assinatura
5b56ec6fbe4a9a948dcfd5ba3d549b09bc4ad08a	Win.Trojan.Mybot-11624
61796e8d48d366269430054c4e3aeb2bbbec9d29	Win.Downloader.133181-1
69a0616c005a49fdfd1a0f18cfe87eb5bacae623	Win.Trojan.Agent-30718
6a95860594cd8b7e3636bafa8f812e05359a64ca	Win.Trojan.Agent-30723
6c37e484b00f9fd80a92559d16e3de7b68e7b167	Win.Trojan.Agent-1316411
71afc0989a17954d0eb6c480257c378f5d81847a	Sem assinatura
724e06f187271e8cba649d8ce6a72f4036b9b7ab	Sem assinatura
745b7a2ad45a1e46daceb20bb182a07522772605	Win.Trojan.Agent-474351
7487377374a97a444bdfbeef6177894bb982e034	Sem assinatura
7534ff6ed4afa03465f41a25e98e6212298c74a6	Win.Trojan.Zusy-765
76e6b13837a6712e3f3f768e3ad2e144fbc28076	Win.Trojan.Zloyfly-1
7a2b17e11d6b8f31bb261195e2a419bc43ee9e71	Win.Trojan.Ircworm-5
7a4186dfbb206fc05f4c222a71e09fc56ce3a49e	Win.Trojan.Agent-30705
7f4d6745b9053583b55b87bb16a88840e56e5621	Win.Trojan.Agent-474236
80260c1b637366f301d25cdcd9a7c82e9f09ea5	Win.Trojan.Merong-1
83511103946b0664ace5384b09a978fdf49c85b90	Win.Downloader.74679-1
85555450fb89aca482f74c924504788ff4159323	Win.Trojan.Rbot-2443
86ec7431cb4a933b0711be80143c59045d07897e	Sem assinatura
86f654a312f3247a915f3fa4eea4567513b25628	Sem assinatura
88fc0b30bdaafc1093cf873f5a805423d6c9e7225	Win.Trojan.Agent-30589
8d8cc34190831c0b86acc85be5f516261ef60bab	Sem assinatura
91a7bb3dbca171f9dd7a793317e06b1b15f8e1a7	Win.Trojan.Buzus-4415
957c669dda88b8e960c1b1c8f50d8624bccc3f4b	Win.Trojan.Agent-30683
9a6c465a4ccd2a8b6fbab75657f532fdcca27ed0	Win.Trojan.Virtob-2
9f3d6d3fd87ebb83098e5615e98c6c8e929eab84	Sem assinatura
a417fd8b203083e5a3ebcb212f661c4aeebc6015	Sem assinatura
a915c8f23cc2082656522ad06e5c00bbaa0d168f	Win.Trojan.Zapchast-2506
a9e2d5f84e9cf5fc4ab585415f3669a474f1c426	Sem assinatura
aa38bef9bb26dcc0b0ebdede1537ddd76665f3f9	Win.Trojan.Delf-2270
aad4b86ad3545c508d5e02a6f78a8f363f9de6dc	Win.Downloader.16819-1
ac7a3bbd2f7da330ff30b7bdd0c60de2b2d873ed	Win.Trojan.IRCBot-3485
acf9d93e6ebb915d435529102607649a19414bf9	Sem assinatura
adcead6817f5d0575c09b03eb94a9772584eafae	Win.Trojan.Agent-30677
b3423fef635638bb078b01c34166d1e6eb638d36	Win.Trojan.VanBot-1
b5c75c0be343f5acdf05b239331097f456ed928b	Sem assinatura
b7e9c0cef3b1b62f4435850d2c0db0bd6b68b10a	Sem assinatura

Continua na próxima página

Malware (hash SHA1)	Assinatura
b9de3adc708bc59d806d1f706897b2fa582045a7	Sem assinatura
be08b1025dd9d46c7012aa1ae752d4e0af91fc71	Win.Trojan.Agent-30710
c176b323f94759fbcda20a4ccca17993339727c6	Win.Trojan.Agent-30623
c33377212dc40200933e346786f2e30ae9665406	Sem assinatura
c35d12f564201f6eeeb1b76b86610e6e8f64a113b	Sem assinatura
c4343f6b7d899ceae825af31209985824e97cd23	Win.Worm.Sasser-2
cc91bc68522fa8912825e71abeb2ddf9ffce1220	Win.Trojan.Cossta-649
cccbb4b74fdf7a78b663ec1488f8bc804100dcd4	Sem assinatura
ce80c629fda3722449405b7c51857efc7bed0bd7	Win.Trojan.Agent-30691
d10f26181ac7491329df4772372a9e8a2d2b1faf	Win.Trojan.Refroso-5119
d20581741e64c8306fa94c7c8605e768a83168d4	Win.Downloader.81796-1
d2fbdd167613e1c851cb6b362d56737bb6f578b8	Win.Trojan.Swisyn-1821
d70af9deeab81d6cb9263583703049311e4459bd	Sem assinatura
d78ae37341a26378a86b460dfaddf054105c25dd	Sem assinatura
d7a6662a9be5364b6e5f8fc96e3e0314ef3105e2	Win.Trojan.Lithium-3
d9a2cf4db9b8cda4190e3eb79f37cf5c8c2dcb9f	Sem assinatura
ddb518c695c27f4dbdf77ca969643843cdc8acbe	Win.Trojan.Bot-13
e0872ab939c03ccb740fc2aecf97f2bd9c618795	Win.Trojan.Agent-385437
e3c48e3d1bb0afd39adf5575f8c6d554c8ab93cf	Sem assinatura
e508966f45b73fe1f9b0ebef3b2501e7845da5ea	Win.Trojan.Small-13891
e90b4c961f4ec67229ebfca4ba72d1eefaedc9cd	Win.Trojan.Agent-440760
ec13ddb6c448a8fa7619da79beced8d9b19ae416	Sem assinatura
ec56d119dc543082860a6ac1fb8a1374d57e37d8	Win.Trojan.IRCBot-3800
ed47563dd5cc300716a9ba7946424d538f095ce6	Win.Downloader.134278-1
ee1183fcf26263d45ec1c26ddeccf36ce64b5a25	Win.Trojan.Downloader-2215
eed4c14d576c9d0a16aabb6516761f8d1eb9160b	Win.Trojan.Agent-30583
ef40b9699a3b9156219b1b1ae1bb920f61cda37d	Win.Worm.Allapple-307
ef6b4afea678eaa80b8ea3ba1fea485c9e678a6e	Sem assinatura
f5f82285141fe18b8b6f592510d85d2d0aa9707f	Win.Trojan.Gloox-1
fcbb6d23d43236efe38afbe05c9c803091b8941e	Win.Trojan.Poebot-29
fd3c14eb2a27adb8f8085dfed51b98579d8df592	Win.Trojan.Agent-30715

APÊNDICE B – PARÂMETROS DE AVALIAÇÃO

A listagem abaixo apresenta os parâmetros de avaliação dos *malwares* utilizados na avaliação experimental descrita no Capítulo 5. Essas avaliações são resultantes da execução dos *malwares* na respectiva arquitetura realizada no período de 09 a 11 de junho de 2016.

Tabela 1: Parâmetros de avaliação do *malware* hash SHA-1 00590628752de21c18d444f625261692c12a287e.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	13	14	4	6	8	2	5
Partial	21	21	4	14	8	2	5
Close	15	15	3	7	6	2	6
MARS Eventos únicos	49	29	4	27	22	2	6
Arquitetura Referência	14	16	4	6	8	2	6

Tabela 2: Parâmetros de avaliação do *malware* hash SHA-1 0320ee1aca89fb00ade98a4d546c8e07dade01ec.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	5	9	3	0	1	0	4
Partial	2	4	3	0	0	0	2
Close	8	12	3	0	1	1	4
MARS Eventos únicos	15	12	3	0	2	1	4
Arquitetura Referência	3	5	3	0	1	0	3

Tabela 3: Parâmetros de avaliação do *malware* hash SHA-1 0931655cd5eac3abb6d3bace2b8c7e408c65db48.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	19	20	4	7	11	2	6
Partial	34	23	4	16	19	2	5
Close	25	23	3	15	7	2	6
MARS Eventos únicos	78	38	4	38	37	2	6
Arquitetura Referência	18	20	4	6	11	2	6

Tabela 4: Parâmetros de avaliação do *malware hash* SHA-1 *0d3c49bf681e6128a7c3c663d6eb5701af70759e*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	29	24	4	17	8	3	6
Partial	28	29	4	17	9	3	6
Close	21	21	3	14	5	3	6
MARS Eventos únicos	78	37	4	48	22	4	6
Arquitetura Referência	20	20	4	6	10	2	6

Tabela 5: Parâmetros de avaliação do *malware hash* SHA-1 *1164fa9eb0ca28d23b7cd48f731b89f422c2d230*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	39	35	10	16	12	15	6
Partial	25	23	8	14	11	10	5
Close	32	32	6	16	7	11	6
MARS Eventos únicos	96	47	10	46	30	16	6
Arquitetura Referência	31	29	8	14	13	14	6

Tabela 6: Parâmetros de avaliação do *malware hash* SHA-1 *11cd20c734c90d09c8faa5b6e0964d27a666938e*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	8	9	4	2	3	2	4
Partial	7	8	4	1	1	1	5
Close	8	8	3	1	1	2	4
MARS Eventos únicos	23	13	4	4	5	2	5
Arquitetura Referência	6	8	3	1	1	1	4

Tabela 7: Parâmetros de avaliação do *malware hash* SHA-1 *1571105b3fcbc117dd7c9308286af30d34787a92*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	4	6	3	0	1	0	4
Partial	7	9	3	0	0	1	4
Close	5	7	3	0	1	1	4
MARS Eventos únicos	16	10	3	0	2	2	4
Arquitetura Referência	4	6	3	0	1	0	4

Tabela 8: Parâmetros de avaliação do *malware hash* SHA-1 *19aee19f5ba66743da4f59858df1cb162841b572*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	4	8	4	0	1	0	4
Partial	4	6	4	1	0	1	3
Close	6	8	4	0	1	2	3
MARS Eventos únicos	14	11	4	1	2	2	4
Arquitetura Referência	3	6	4	0	1	0	3

Tabela 9: Parâmetros de avaliação do *malware hash* SHA-1 *1ae0c8b0190b05c72b902779f1e471a3bb4feedf*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	29	22	4	21	11	3	6
Partial	21	21	4	14	8	3	6
Close	28	23	3	21	5	3	6
MARS Eventos únicos	78	31	4	56	24	5	6
Arquitetura Referência	17	16	4	4	10	2	6

Tabela 10: Parâmetros de avaliação do *malware hash* SHA-1 *2872de82a851dbd4fe27b51459c5fb3ccfcdbcfc*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	85	23	7	23	24	57	6
Partial	69	13	7	10	9	57	5
Close	77	23	6	15	7	57	6
MARS Eventos únicos	231	32	7	48	40	64	6
Arquitetura Referência	27	20	7	12	11	10	6

Tabela 11: Parâmetros de avaliação do *malware hash* SHA-1 *2aa58c26deac88c63b544318da7ab08e2a5fff97*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	9	11	6	4	7	1	5
Partial	17	17	6	5	9	1	6
Close	19	22	5	8	4	2	5
MARS Eventos únicos	45	29	6	17	20	2	6
Arquitetura Referência	12	14	6	4	8	1	6

Tabela 12: Parâmetros de avaliação do *malware hash* SHA-1 *2ca57931c0e0347200f3fed1d188fb4152fb67a5*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	17	18	5	6	11	2	6
Partial	34	24	5	15	10	2	6
Close	22	21	4	11	9	2	6
MARS Eventos únicos	73	36	5	32	30	2	6
Arquitetura Referência	18	17	5	8	11	2	6

Tabela 13: Parâmetros de avaliação do *malware hash* SHA-1 *2ec7139d5fdb02fb1e7557a7339af5c91fb21ed7*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	81	23	7	13	10	57	6
Partial	76	23	7	12	11	56	6
Close	74	21	6	10	6	57	6
MARS Eventos únicos	231	32	7	35	27	59	6
Arquitetura Referência	28	21	7	12	11	10	6

Tabela 14: Parâmetros de avaliação do *malware hash* SHA-1 *30f6a8dba8d73b867c05e16a486f2ca6cdb6e104*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	22	21	7	7	7	3	6
Partial	30	22	7	13	8	3	6
Close	25	22	6	14	5	3	6
MARS Eventos únicos	77	31	7	34	20	3	6
Arquitetura Referência	16	16	7	5	10	2	6

Tabela 15: Parâmetros de avaliação do *malware hash* SHA-1 *325acdf79a66cb744c48363a7d97f19fbaf79c20*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	9	13	4	3	3	2	5
Partial	7	9	4	3	2	2	4
Close	7	8	3	1	1	2	4
MARS Eventos únicos	23	16	4	7	6	2	5
Arquitetura Referência	6	8	4	3	3	1	4

Tabela 16: Parâmetros de avaliação do *malware hash* SHA-1 *32d3604c26a4bb339136119b26ec69c32c156bff*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	6	8	3	2	1	2	4
Partial	8	10	3	1	0	1	4
Close	8	11	3	1	1	2	5
MARS Eventos únicos	22	15	3	4	2	2	5
Arquitetura Referência	5	7	3	1	1	1	4

Tabela 17: Parâmetros de avaliação do *malware hash* SHA-1 *3794d50471d7a937241efa60e0b847c65e54c4c8*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	17	18	5	5	12	2	6
Partial	16	15	5	6	11	2	5
Close	16	17	4	7	9	2	6
MARS Eventos únicos	49	29	5	18	32	2	6
Arquitetura Referência	19	18	5	7	14	2	6

Tabela 18: Parâmetros de avaliação do *malware hash* SHA-1 *39fdf1bfd3b7f1c11ef67be8d07309446fc41aae*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	14	16	5	5	8	2	6
Partial	23	21	5	16	9	3	5
Close	27	25	4	14	7	3	6
MARS Eventos únicos	64	39	5	35	24	4	6
Arquitetura Referência	18	19	5	6	10	2	6

Tabela 19: Parâmetros de avaliação do *malware hash* SHA-1 *3a93cddb50d96d1472ddedcd8fc2d146ea489ce9*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	18	19	5	6	11	2	6
Partial	30	29	5	16	9	2	6
Close	14	14	4	8	7	2	5
MARS Eventos únicos	62	35	5	30	27	2	6
Arquitetura Referência	16	18	5	7	11	2	6

Tabela 20: Parâmetros de avaliação do *malware hash* SHA-1 *3e5879ba2b137f949732c918661e73f18baf0352*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	31	25	6	5	10	2	6
Partial	26	23	6	12	10	2	6
Close	11	13	4	8	3	2	5
MARS Eventos únicos	68	34	6	25	23	2	7
Arquitetura Referência	25	22	6	8	13	2	6

Tabela 21: Parâmetros de avaliação do *malware hash* SHA-1 *3f752c5c97e80d41f5498d8a90f5ee8afd1c4372*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	12	12	4	6	9	2	5
Partial	22	19	4	15	9	2	5
Close	23	23	3	16	8	2	6
MARS Eventos únicos	57	33	4	37	26	2	6
Arquitetura Referência	17	17	4	6	11	2	6

Tabela 22: Parâmetros de avaliação do *malware hash* SHA-1 *4072c65ef571735157453e4c7f68a210df62948f*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	18	17	4	6	9	2	6
Partial	24	22	4	13	10	2	5
Close	21	18	3	9	8	2	6
MARS Eventos únicos	63	34	4	28	27	2	6
Arquitetura Referência	14	13	4	5	9	2	5

Tabela 23: Parâmetros de avaliação do *malware hash* SHA-1 *42081abf164c948d19075b90e169747f390924f8*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	16	17	7	1	4	3	5
Partial	8	10	7	1	3	2	5
Close	10	12	6	3	2	2	6
MARS Eventos únicos	34	22	7	5	9	5	6
Arquitetura Referência	9	10	7	2	5	2	5

Tabela 24: Parâmetros de avaliação do *malware hash* SHA-1 *44627df5595278068872eb1277d2c188907baf94*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	25	21	4	10	9	2	6
Partial	27	21	4	19	7	2	6
Close	22	19	3	12	7	2	6
MARS Eventos únicos	74	36	4	41	23	2	6
Arquitetura Referência	22	19	4	11	9	2	6

Tabela 25: Parâmetros de avaliação do *malware hash* SHA-1 *44fbd1e9d7d0c0a7f3eb5e9de2e640bf5d64e2e2*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	52	24	14	25	19	24	6
Partial	40	20	14	11	13	23	6
Close	48	24	12	24	7	21	6
MARS Eventos únicos	140	35	14	60	39	34	6
Arquitetura Referência	25	18	14	12	12	9	6

Tabela 26: Parâmetros de avaliação do *malware hash* SHA-1 *48eb49bcdb11b7d27e66def91b5a08b7ecbf1ac2*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	16	17	4	5	9	2	6
Partial	20	21	4	14	8	2	5
Close	18	17	3	6	6	2	6
MARS Eventos únicos	54	37	4	25	23	2	6
Arquitetura Referência	17	18	4	5	10	2	6

Tabela 27: Parâmetros de avaliação do *malware hash* SHA-1 *4d053b26ad52a8248c80ddde63d4577f27a12dd1*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	6	8	4	0	2	1	4
Partial	9	11	3	0	1	2	5
Close	6	7	3	1	1	1	4
MARS Eventos únicos	21	13	4	1	4	2	5
Arquitetura Referência	5	8	3	0	1	0	5

Tabela 28: Parâmetros de avaliação do *malware hash* SHA-1 *4d57817bfbf70e06fb7dd5c6a8d340443fd6cfc4*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	6	10	3	2	3	1	4
Partial	7	8	3	3	2	1	3
Close	6	8	2	1	1	1	4
MARS Eventos únicos	19	14	3	6	6	1	5
Arquitetura Referência	8	8	3	3	3	1	4

Tabela 29: Parâmetros de avaliação do *malware hash* SHA-1 *4d5d2bc7a7008ef5f6920495e2ba96b14d2d96d7*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	28	21	5	11	6	2	6
Partial	35	29	5	28	7	2	6
Close	23	20	4	11	7	2	6
MARS Eventos únicos	86	35	5	50	20	2	6
Arquitetura Referência	18	14	5	12	9	2	5

Tabela 30: Parâmetros de avaliação do *malware hash* SHA-1 *4f9ac6ed2eb74a8bb8eb5d23d834c2be6f1ddaf7*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	18	19	4	6	10	2	6
Partial	31	29	4	15	11	2	6
Close	21	20	3	12	8	2	6
MARS Eventos únicos	70	38	4	33	29	2	6
Arquitetura Referência	15	15	4	5	12	2	5

Tabela 31: Parâmetros de avaliação do *malware hash* SHA-1 *4faf2bc5ae8ec46c89cb305e8dce25a50c8b697c*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	29	24	6	12	9	3	6
Partial	22	20	6	15	7	3	5
Close	19	21	5	14	4	3	6
MARS Eventos únicos	70	38	6	41	20	4	6
Arquitetura Referência	17	18	6	5	10	2	6

Tabela 32: Parâmetros de avaliação do *malware hash* SHA-1 *556aa22e34c05711cece7ca1fc53bbbf2ad977cf*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	15	16	4	5	9	2	6
Partial	27	26	4	14	8	2	6
Close	14	14	3	6	6	2	6
MARS Eventos únicos	56	33	4	25	23	2	6
Arquitetura Referência	17	17	4	5	10	2	6

Tabela 33: Parâmetros de avaliação do *malware hash* SHA-1 *58f2e511843d7346f1baba3a77a84a24205696fd*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	29	22	5	9	8	2	6
Partial	20	19	5	13	7	3	5
Close	19	20	4	8	3	2	6
MARS Eventos únicos	68	36	5	30	18	3	6
Arquitetura Referência	17	18	5	6	9	2	6

Tabela 34: Parâmetros de avaliação do *malware hash* SHA-1 *5b56ec6fbe4a9a948dcfd5ba3d549b09bc4ad08a*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	82	21	8	8	9	58	6
Partial	69	12	8	10	8	59	5
Close	77	20	7	10	6	60	6
MARS Eventos únicos	228	29	8	28	23	64	6
Arquitetura Referência	19	12	8	13	9	10	5

Tabela 35: Parâmetros de avaliação do *malware hash* SHA-1 *61796e8d48d366269430054c4e3aeb2bbbec9d29*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	21	21	6	5	11	2	6
Partial	28	26	8	16	8	2	6
Close	12	11	3	6	5	2	5
MARS Eventos únicos	61	33	9	27	24	2	6
Arquitetura Referência	17	17	6	6	11	2	6

Tabela 36: Parâmetros de avaliação do *malware hash* SHA-1 *69a0616c005a49fd1a0f18cfe87eb5bacae623*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	23	21	6	10	11	2	6
Partial	34	29	6	25	9	2	6
Close	18	13	5	14	6	2	5
MARS Eventos únicos	75	37	6	49	26	2	6
Arquitetura Referência	24	20	6	10	12	2	6

Tabela 37: Parâmetros de avaliação do *malware hash* SHA-1 *6c37e484b00f9fd80a92559d16e3de7b68e7b167*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	6	10	3	1	1	1	4
Partial	9	11	4	4	2	1	4
Close	6	7	3	1	1	1	4
MARS Eventos únicos	21	16	4	6	4	1	5
Arquitetura Referência	7	8	4	4	3	1	4

Tabela 38: Parâmetros de avaliação do *malware hash* SHA-1 *71afc0989a17954d0eb6c480257c378f5d81847a*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	23	22	5	6	8	2	6
Partial	22	20	5	15	7	2	5
Close	15	15	4	7	6	2	6
MARS Eventos únicos	60	35	5	28	21	2	6
Arquitetura Referência	19	18	5	5	10	2	6

Tabela 39: Parâmetros de avaliação do *malware hash* SHA-1 724e06f187271e8cba649d8ce6a72f4036b9b7ab.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	5	7	3	0	2	0	5
Partial	7	9	3	0	0	1	4
Close	5	7	3	0	1	1	4
MARS Eventos únicos	17	11	3	0	3	2	5
Arquitetura Referência	4	6	3	0	1	0	4

Tabela 40: Parâmetros de avaliação do *malware hash* SHA-1 745b7a2ad45a1e46daceb20bb182a07522772605.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	8	10	4	1	4	1	5
Partial	11	13	4	3	3	2	5
Close	11	13	3	0	2	1	6
MARS Eventos únicos	30	21	4	4	9	2	6
Arquitetura Referência	6	8	4	1	3	1	4

Tabela 41: Parâmetros de avaliação do *malware hash* SHA-1 7487377374a97a444bdfbeef6177894bb982e034.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	40	34	10	21	10	14	6
Partial	31	32	8	16	9	10	6
Close	28	30	7	15	6	10	6
MARS Eventos únicos	99	45	10	52	25	16	6
Arquitetura Referência	30	30	9	11	13	14	6

Tabela 42: Parâmetros de avaliação do *malware hash* SHA-1 7534ff6ed4afa03465f41a25e98e6212298c74a6.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	52	22	19	13	33	2	6
Partial	47	23	19	15	34	3	6
Close	28	23	17	18	6	3	6
MARS Eventos únicos	127	30	19	46	73	4	6
Arquitetura Referência	41	17	19	7	35	2	6

Tabela 43: Parâmetros de avaliação do *malware hash* SHA-1 76e6b13837a6712e3f3f768e3ad2e144fbc28076.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	8	13	5	1	4	1	5
Partial	9	12	5	1	2	2	6
Close	7	8	4	1	1	2	4
MARS Eventos únicos	24	19	5	3	7	3	6
Arquitetura Referência	6	8	5	1	3	1	4

Tabela 44: Parâmetros de avaliação do *malware hash* SHA-1 *7a2b17e11d6b8f31bb261195e2a419bc43ee9e71*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	84	22	8	20	9	59	6
Partial	77	21	8	12	9	59	6
Close	78	23	7	14	7	59	6
MARS Eventos únicos	239	31	8	46	25	64	6
Arquitetura Referência	15	15	8	5	11	2	6

Tabela 45: Parâmetros de avaliação do *malware hash* SHA-1 *7a4186dfbb206fc05f4c222a71e09fc56ce3a49e*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	27	20	4	14	10	10	6
Partial	30	22	4	20	9	9	5
Close	30	24	3	20	6	9	6
MARS Eventos únicos	87	38	4	54	25	14	6
Arquitetura Referência	24	18	4	13	11	10	6

Tabela 46: Parâmetros de avaliação do *malware hash* SHA-1 *7f4d6745b9053583b55b87bb16a88840e56e5621*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	68	17	4	58	59	3	6
Partial	74	20	4	62	61	3	5
Close	70	17	4	56	60	2	6
MARS Eventos únicos	212	32	4	176	180	4	6
Arquitetura Referência	22	16	4	13	13	2	5

Tabela 47: Parâmetros de avaliação do *malware hash* SHA-1 *80260c1b637366f301d25cdcda9a7c82e9f09ea5*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	18	18	6	9	13	2	6
Partial	14	14	6	8	10	2	5
Close	26	23	5	12	9	2	6
MARS Eventos únicos	58	32	6	29	32	2	6
Arquitetura Referência	14	16	6	9	9	2	6

Tabela 48: Parâmetros de avaliação do *malware hash* SHA-1 *80648f7606b36d12f67914907eb86c2fe2c7b26a*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	23	15	5	5	15	2	5
Partial	30	28	4	23	9	2	6
Close	23	20	3	18	5	2	6
MARS Eventos únicos	76	30	5	46	29	2	6
Arquitetura Referência	32	24	5	13	24	2	6

Tabela 49: Parâmetros de avaliação do *malware hash* SHA-1 *8351103946b0664ace5384b09a978df49c85b90*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	27	24	5	7	11	2	6
Partial	17	17	5	7	12	2	5
Close	25	22	4	13	6	2	6
MARS Eventos únicos	69	33	5	27	29	2	6
Arquitetura Referência	19	19	5	6	13	2	6

Tabela 50: Parâmetros de avaliação do *malware hash* SHA-1 *85555450fb89aca482f74c924504788ff4159323*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	53	25	13	10	12	25	6
Partial	38	16	13	11	11	25	5
Close	39	22	11	11	6	22	6
MARS Eventos únicos	130	34	13	32	29	31	6
Arquitetura Referência	27	22	13	13	12	9	6

Tabela 51: Parâmetros de avaliação do *malware hash* SHA-1 *86ec7431cb4a933b0711be80143c59045d07897e*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	31	25	13	13	11	2	6
Partial	35	27	13	17	10	2	6
Close	22	22	4	15	6	2	6
MARS Eventos únicos	88	39	13	45	27	2	6
Arquitetura Referência	20	21	13	7	12	2	6

Tabela 52: Parâmetros de avaliação do *malware hash* SHA-1 *86f654a312f3247a915f3fa4eea4567513b25628*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	2	4	4	0	1	0	2
Partial	3	5	4	0	1	0	3
Close	3	5	4	0	1	1	2
MARS Eventos únicos	8	6	4	0	3	1	4
Arquitetura Referência	3	6	4	0	1	0	3

Tabela 53: Parâmetros de avaliação do *malware hash* SHA-1 *88fc0b30bda4c1093cf873f5a805423d6c9e7225*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	10	12	4	5	5	2	5
Partial	10	13	4	6	2	2	3
Close	7	8	3	3	1	2	4
MARS Eventos únicos	27	17	4	14	8	2	5
Arquitetura Referência	10	11	4	5	5	2	5

Tabela 54: Parâmetros de avaliação do *malware hash* SHA-1 *8d8cc34190831c0b86acc85be5f516261ef60bab*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	12	15	7	5	4	1	5
Partial	13	14	7	5	3	1	6
Close	11	12	6	5	3	1	6
MARS Eventos únicos	36	22	7	15	10	1	6
Arquitetura Referência	11	13	7	5	5	1	6

Tabela 55: Parâmetros de avaliação do *malware hash* SHA-1 *91a7bb3dbca171f9dd7a793317e06b1b15f8e1a7*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	16	17	11	7	8	4	6
Partial	22	21	11	14	9	4	6
Close	13	13	10	7	6	4	5
MARS Eventos únicos	51	30	11	28	23	6	6
Arquitetura Referência	16	14	11	9	9	6	6

Tabela 56: Parâmetros de avaliação do *malware hash* SHA-1 *957c669dda88b8e960c1b1c8f50d8624bccc3f4b*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	18	19	4	5	7	2	6
Partial	21	20	4	13	9	2	5
Close	15	16	3	6	4	2	6
MARS Eventos únicos	54	32	4	24	20	2	6
Arquitetura Referência	16	18	4	5	9	2	6

Tabela 57: Parâmetros de avaliação do *malware hash* SHA-1 *9a6c465a4ccd2a8b6fbab75657f532fdcca27ed0*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	32	25	14	23	21	4	6
Partial	21	16	17	10	8	5	5
Close	32	22	12	27	5	5	6
MARS Eventos únicos	85	32	17	60	34	9	6
Arquitetura Referência	20	18	14	12	10	5	6

Tabela 58: Parâmetros de avaliação do *malware hash* SHA-1 *9f3d6d3fd87ebb83098e5615e98c6c8e929eab84*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	14	15	9	7	8	2	6
Partial	26	21	9	16	10	2	6
Close	7	9	6	3	1	2	4
MARS Eventos únicos	47	28	9	26	19	2	7
Arquitetura Referência	10	11	9	4	5	2	5

Tabela 59: Parâmetros de avaliação do *malware hash* SHA-1 *a417fd8b203083e5a3ebcb212f661c4aeebc6015*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	4	6	3	0	1	0	4
Partial	7	9	3	0	0	1	4
Close	6	9	3	0	1	1	5
MARS Eventos únicos	17	12	3	0	2	2	5
Arquitetura Referência	6	8	4	1	3	1	4

Tabela 60: Parâmetros de avaliação do *malware hash* SHA-1 *a915c8f23cc2082656522ad06e5c00bbaa0d168f*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	29	23	12	16	10	2	6
Partial	17	17	12	8	10	2	5
Close	29	23	10	17	6	2	6
MARS Eventos únicos	75	34	12	41	26	2	6
Arquitetura Referência	15	15	12	6	12	2	5

Tabela 61: Parâmetros de avaliação do *malware hash* SHA-1 *a9e2d5f84e9cf5fc4ab585415f3669a474f1c426*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	29	23	10	12	10	3	6
Partial	24	21	10	14	11	3	5
Close	26	22	5	14	5	2	7
MARS Eventos únicos	79	36	11	40	26	4	7
Arquitetura Referência	15	15	10	6	12	2	6

Tabela 62: Parâmetros de avaliação do *malware hash* SHA-1 *aa38bef9bb26dcc0b0ebdede1537ddd76665f3f9*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	86	23	7	10	10	60	6
Partial	74	15	7	8	11	61	5
Close	79	22	6	11	6	61	6
MARS Eventos únicos	239	32	7	29	27	66	6
Arquitetura Referência	23	15	7	14	11	10	6

Tabela 63: Parâmetros de avaliação do *malware hash* SHA-1 *aad4b86ad3545c508d5e02a6f78a8f363f9de6dc*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	15	16	4	7	12	2	6
Partial	22	20	4	15	9	2	5
Close	16	16	3	8	6	2	6
MARS Eventos únicos	53	31	4	30	27	2	6
Arquitetura Referência	20	21	4	6	11	2	6

Tabela 64: Parâmetros de avaliação do *malware hash* SHA-1 *ac7a3bbd2f7da330ff30b7bdd0c60de2b2d873ed*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	54	25	17	26	27	25	6
Partial	39	14	17	11	9	24	5
Close	38	21	15	16	5	21	6
MARS Eventos únicos	131	33	17	53	41	32	6
Arquitetura Referência	26	19	16	12	11	8	6

Tabela 65: Parâmetros de avaliação do *malware hash* SHA-1 *acf9d93e6ebb915d435529102607649a19414bf9*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	39	19	13	12	10	24	6
Partial	49	25	13	13	13	25	6
Close	33	15	11	10	5	22	6
MARS Eventos únicos	121	32	13	35	28	34	6
Arquitetura Referência	25	18	13	12	13	9	6

Tabela 66: Parâmetros de avaliação do *malware hash* SHA-1 *adcead6817f5d0575c09b03eb94a9772584eafae*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	5	7	4	0	1	1	4
Partial	11	13	4	0	1	2	6
Close	7	9	4	0	1	1	4
MARS Eventos únicos	23	15	4	0	3	2	6
Arquitetura Referência	5	7	4	0	2	0	5

Tabela 67: Parâmetros de avaliação do *malware hash* SHA-1 *b3423fef635638bb078b01c34166d1e6eb638d36*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	20	20	12	9	8	3	6
Partial	22	17	14	9	9	5	5
Close	21	19	12	11	7	4	6
MARS Eventos únicos	63	33	17	29	24	7	6
Arquitetura Referência	23	22	14	10	10	4	6

Tabela 68: Parâmetros de avaliação do *malware hash* SHA-1 *b5c75c0be343f5acdf05b239331097f456ed928b*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	42	31	11	21	21	3	6
Partial	27	22	10	21	9	3	5
Close	34	27	9	19	9	3	6
MARS Eventos únicos	103	50	11	61	39	3	6
Arquitetura Referência	14	16	10	6	10	3	5

Tabela 69: Parâmetros de avaliação do *malware hash* SHA-1 *b7e9c0cef3b1b62f4435850d2c0db0bd6b68b10a*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	3	6	2	0	1	0	3
Partial	4	6	2	0	0	1	2
Close	2	4	2	0	1	0	2
MARS Eventos únicos	9	8	2	0	2	1	3
Arquitetura Referência	3	6	2	0	1	0	3

Tabela 70: Parâmetros de avaliação do *malware hash* SHA-1 *b9de3adc708bc59d806d1f706897b2fa582045a7*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	7	10	3	3	3	1	5
Partial	6	7	2	1	1	1	5
Close	6	7	2	1	1	1	4
MARS Eventos únicos	19	13	3	5	5	1	6
Arquitetura Referência	6	8	2	2	2	1	5

Tabela 71: Parâmetros de avaliação do *malware hash* SHA-1 *be08b1025dd9d46c7012aa1ae752d4e0af91fc71*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	24	21	11	14	17	2	6
Partial	25	25	10	10	11	2	6
Close	24	22	9	14	9	2	6
MARS Eventos únicos	73	40	11	38	37	2	6
Arquitetura Referência	25	24	11	11	14	2	6

Tabela 72: Parâmetros de avaliação do *malware hash* SHA-1 *c176b323f94759fbcda20a4ccca17993339727c6*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	18	17	5	7	11	2	6
Partial	23	21	5	14	9	2	5
Close	17	15	4	9	6	2	6
MARS Eventos únicos	58	31	5	30	26	2	6
Arquitetura Referência	18	19	5	7	8	2	6

Tabela 73: Parâmetros de avaliação do *malware hash* SHA-1 *c33377212dc40200933e346786f2e30ae9665406*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	8	10	3	3	4	2	4
Partial	6	7	3	3	2	1	4
Close	4	7	2	1	1	1	3
MARS Eventos únicos	18	13	3	7	7	2	5
Arquitetura Referência	8	9	3	3	4	1	5

Tabela 74: Parâmetros de avaliação do *malware hash* SHA-1 *c35d12f564201f6eee1b76b86610e6e8f64a113b*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	16	17	4	9	8	2	6
Partial	22	21	4	17	7	2	5
Close	18	15	3	10	6	2	6
MARS Eventos únicos	56	32	4	36	21	2	6
Arquitetura Referência	22	21	4	9	12	2	6

Tabela 75: Parâmetros de avaliação do *malware hash* SHA-1 *c4343f6b7d899ceae825af31209985824e97cd23*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	127	1408	8	106	15	4	5
Partial	102	1294	9	95	8	4	5
Close	92	9903	7	83	6	4	6
MARS Eventos únicos	321	12539	9	284	29	6	6
Arquitetura Referência	113	1219	7	93	8	3	6

Tabela 76: Parâmetros de avaliação do *malware hash* SHA-1 *cc91bc68522fa8912825e71abeb2ddf9ffce1220*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	18	19	4	6	7	2	6
Partial	27	28	4	19	8	2	6
Close	17	17	3	5	6	2	6
MARS Eventos únicos	62	36	4	30	21	2	6
Arquitetura Referência	13	14	4	5	10	2	5

Tabela 77: Parâmetros de avaliação do *malware hash* SHA-1 *cc-cbb4b74fdf7a78b663ec1488f8bc804100dcd4*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	74	26	15	12	12	50	6
Partial	24	18	15	16	11	10	5
Close	37	25	15	22	9	10	6
MARS Eventos únicos	135	37	16	50	32	53	6
Arquitetura Referência	24	18	15	13	12	9	6

Tabela 78: Parâmetros de avaliação do *malware hash* SHA-1 *ce80c629fda3722449405b7c51857efc7bed0bd7*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	5	7	3	0	1	1	4
Partial	9	11	4	2	0	2	4
Close	7	9	3	0	1	1	5
MARS Eventos únicos	21	14	4	2	2	2	5
Arquitetura Referência	4	6	3	1	1	0	4

Tabela 79: Parâmetros de avaliação do *malware hash* SHA-1 *d10f26181ac7491329df4772372a9e8a2d2b1faf*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	32	17	8	10	8	19	6
Partial	31	15	8	7	11	19	5
Close	28	14	7	8	5	18	6
MARS Eventos únicos	91	26	8	25	24	23	6
Arquitetura Referência	22	17	8	12	10	8	6

Tabela 80: Parâmetros de avaliação do *malware hash* SHA-1 *d20581741e64c8306fa94c7c8605e768a83168d4*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	43	19	5	28	34	3	6
Partial	45	21	6	34	32	2	5
Close	47	24	5	32	33	2	6
MARS Eventos únicos	76	37	6	52	50	3	6
Arquitetura Referência	25	20	5	14	10	2	5

Tabela 81: Parâmetros de avaliação do *malware hash* SHA-1 *d2fbdd167613e1c851cb6b362d56737bb6f578b8*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	9	12	5	1	1	2	3
Partial	7	10	5	0	2	2	4
Close	7	11	5	0	2	2	5
MARS Eventos únicos	23	18	5	1	5	4	5
Arquitetura Referência	5	9	5	1	1	1	4

Tabela 82: Parâmetros de avaliação do *malware hash* SHA-1 *d70af9deeeab81d6cb9263583703049311e4459bd*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	27	22	4	11	8	2	6
Partial	26	20	4	18	7	2	5
Close	20	15	3	11	3	2	6
MARS Eventos únicos	73	34	4	40	18	2	6
Arquitetura Referência	23	20	4	11	10	2	6

Tabela 83: Parâmetros de avaliação do *malware hash* SHA-1 *d78ae37341a26378a86b460dfaddf054105c25dd*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	28	24	6	6	9	3	6
Partial	20	20	6	12	8	3	5
Close	28	21	5	14	5	3	6
MARS Eventos únicos	76	36	6	32	22	3	6
Arquitetura Referência	18	18	6	5	11	2	6

Tabela 84: Parâmetros de avaliação do *malware hash* SHA-1 *d7a6662a9be5364b6e5f8fc96e3e0314ef3105e2*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	26	23	11	13	21	2	6
Partial	27	22	10	17	12	2	5
Close	25	21	9	17	10	2	6
MARS Eventos únicos	78	40	11	47	43	2	6
Arquitetura Referência	23	21	11	10	19	2	6

Tabela 85: Parâmetros de avaliação do *malware hash* SHA-1 *d9a2cf4db9b8cda4190e3eb79f37cf5c8c2dcb9f*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	3	5	3	0	1	1	2
Partial	4	7	3	1	0	1	3
Close	4	6	3	0	1	2	2
MARS Eventos únicos	11	10	3	1	2	2	3
Arquitetura Referência	3	5	3	1	1	1	2

Tabela 86: Parâmetros de avaliação do *malware hash* SHA-1 *ddb518c695c27f4dbdf77ca969643843cdc8acbe*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	5	7	5	1	1	1	4
Partial	6	8	5	2	0	3	3
Close	10	11	5	2	1	3	5
MARS Eventos únicos	21	12	5	5	2	4	5
Arquitetura Referência	6	8	5	1	2	2	4

Tabela 87: Parâmetros de avaliação do *malware hash* SHA-1 *e0872ab939c03ccb740fc2aecf97f2bd9c618795*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	21	20	5	8	11	2	6
Partial	23	21	5	14	9	2	6
Close	18	18	4	9	8	2	6
MARS Eventos únicos	62	37	5	31	28	2	6
Arquitetura Referência	19	18	5	8	10	2	6

Tabela 88: Parâmetros de avaliação do *malware hash* SHA-1 *e3c48e3d1bb0afd39adf5575f8c6d554c8ab93cf*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	10	14	4	4	3	1	4
Partial	8	9	4	5	2	2	3
Close	7	8	3	3	1	2	4
MARS Eventos únicos	25	18	4	12	6	3	5
Arquitetura Referência	9	10	4	4	3	1	5

Tabela 89: Parâmetros de avaliação do *malware hash* SHA-1 *e508966f45b73fe1f9b0ebef3b2501e7845da5ea*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	15	17	4	6	9	2	6
Partial	21	20	4	13	10	2	5
Close	15	16	3	8	4	2	6
MARS Eventos únicos	51	31	4	27	23	2	6
Arquitetura Referência	19	20	4	6	11	2	6

Tabela 90: Parâmetros de avaliação do *malware hash* SHA-1 *e90b4c961f4ec67229ebfca4ba72d1eefaedc9cd*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	43	21	5	30	11	2	6
Partial	38	29	5	29	9	2	6
Close	23	14	4	15	6	2	5
MARS Eventos únicos	104	39	5	74	26	2	6
Arquitetura Referência	40	18	5	29	10	2	6

Tabela 91: Parâmetros de avaliação do *malware hash* SHA-1 *ec13ddb6c448a8fa7619da79beced8d9b19ae416*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	26	20	4	5	11	2	6
Partial	22	21	4	14	10	2	5
Close	25	21	3	14	6	2	6
MARS Eventos únicos	73	37	4	33	27	2	6
Arquitetura Referência	19	12	4	5	8	2	5

Tabela 92: Parâmetros de avaliação do *malware hash* SHA-1 *ec56d119dc543082860a6ac1fb8a1374d57e37d8*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	4	8	3	0	1	0	4
Partial	3	6	3	0	1	0	3
Close	4	7	3	0	1	1	3
MARS Eventos únicos	11	9	3	0	3	1	4
Arquitetura Referência	3	6	3	0	1	0	3

Tabela 93: Parâmetros de avaliação do *malware hash* SHA-1 *ed47563dd5cc300716a9ba7946424d538f095ce6*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	21	22	6	6	11	2	6
Partial	20	20	6	12	10	2	5
Close	26	22	5	10	7	2	6
MARS Eventos únicos	67	35	6	28	28	2	6
Arquitetura Referência	20	20	6	6	12	2	6

Tabela 94: Parâmetros de avaliação do *malware hash* SHA-1 *ee1183fcf26263d45ec1c26ddeccf36ce64b5a25*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	18	20	5	6	10	2	6
Partial	26	25	5	14	9	2	6
Close	15	12	3	9	5	2	5
MARS Eventos únicos	59	36	5	29	24	2	6
Arquitetura Referência	16	16	5	7	10	2	6

Tabela 95: Parâmetros de avaliação do *malware hash* SHA-1 *eed4c14d576c9d0a16aabb6516761f8d1eb9160b*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	6	9	3	0	2	0	6
Partial	7	9	3	1	0	1	4
Close	6	8	3	0	2	1	5
MARS Eventos únicos	19	14	3	1	4	2	6
Arquitetura Referência	5	8	3	0	1	0	5

Tabela 96: Parâmetros de avaliação do *malware hash* SHA-1 *ef40b9699a3b9156219b1b1ae1bb920f61cda37d*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	301	2054	7	111	24	4	6
Partial	156	3505	7	33	8	4	5
Close	25	8387	6	20	3	2	6
MARS Eventos únicos	482	13920	7	164	35	4	6
Arquitetura Referência	167	4422	7	27	10	3	6

Tabela 97: Parâmetros de avaliação do *malware hash* SHA-1 *ef6b4afea678eaa80b8ea3ba1fea485c9e678a6e*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	18	19	5	5	10	2	6
Partial	20	19	5	12	9	2	5
Close	25	23	4	14	7	2	6
MARS Eventos únicos	63	36	5	31	26	2	6
Arquitetura Referência	19	20	5	5	13	2	6

Tabela 98: Parâmetros de avaliação do *malware hash* SHA-1 *f5f82285141fe18b8b6f592510d85d2d0aa9707f*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmítidos	TCP	UDP
Open	21	22	5	6	11	3	6
Partial	27	25	5	18	10	8	5
Close	19	18	4	12	4	8	6
MARS Eventos únicos	67	38	5	36	25	10	6
Arquitetura Referência	18	19	5	5	12	3	6

Tabela 99: Parâmetros de avaliação do *malware hash* SHA-1 *fcbb6d23d43236efe38afbe05c9c803091b8941e*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmitidos	TCP	UDP
Open	33	24	5	7	10	5	6
Partial	47	14	5	5	8	2	6
Close	54	22	4	5	6	3	6
MARS Eventos únicos	134	32	5	17	24	5	6
Arquitetura Referência	21	20	6	7	11	3	6

Tabela 100: Parâmetros de avaliação do *malware hash* SHA-1 *fd3c14eb2a27adb8f8085dfed51b98579d8df592*.

Perfil de contenção	Conversação	IP	Assinatura	Duração	Dados Transmitidos	TCP	UDP
Open	18	19	6	6	9	2	6
Partial	28	28	6	18	8	2	6
Close	16	17	5	6	5	2	6
MARS Eventos únicos	62	39	6	30	22	2	6
Arquitetura Referência	16	17	6	5	9	2	6

APÊNDICE C – MALWARES IoT ANALISADOS

A listagem abaixo ilustra o conjunto de *malwares* projetados para IoT analisado na avaliação experimental descrita no Capítulo 6. Nessa listagem é apresentado o *hash* criptográfico de cada *malware* e sua respectiva assinatura obtida pelo antivírus ClamAV em outubro de 2017.

Malware (hash SHA-1)	Assinatura
15333a1880f9f3f72a5428420d68fb5343adfc20	Unix.Trojan.Mirai-1
20778f15154c7070f9766c1c98c22978ddc61d54	Unix.Trojan.Mirai-1
210e8097567ded3216eb8e428c18b49fc1fa0957	Unix.Trojan.Mirai-1
463850f960d01f6a69ac1e1823ff529bac3ffcff	Unix.Trojan.Mirai-1
71d21123e876c8e01f2da7cfc338f298be732c1b	Unix.Trojan.Mirai-1
850c302503e8359fa83e57d0bbc9c9f0988e6e45	Unix.Trojan.Mirai-1
a55d400207855483897ca98307ca14a8208cb366	Unix.Trojan.Mirai-1
a5ee2f8f37ee9458f59227e231b67270a63693b2	Unix.Trojan.Mirai-1
b178b6cb8f28763f984d8b12bc08ce1c69c29545	Unix.Trojan.Mirai-1
fa82440b2216361496f9730f4b700bd61d1511e8	Unix.Trojan.Mirai-1
376d2d876b1331c3b3442cec52de4da1c817b214	Unix.Malware.Agent-6333943-0
399e04a43787cb7a39304349ad78c1a7f7653712	Sem assinatura
3ebafce776c45cd37c2b94f97487401e0e5300cb	Sem assinatura
4d4bac2c05480785ae164e1171473d426c5b8c97	Sem assinatura
5b39134f715f793d5cb1e03b8b858c6e5eb35cb4	Legacy.Trojan.Agent-37003
5c85beb347c4a5dd149c9d8e67756aa76f987907	Unix.Trojan.Mirai-5607483-0
7c922e80369f2e6b8a4724d42d59ef9775d671f3	Sem assinatura
b866ae4ad377d9274dbd2b3be0fc86a4b7bf8f02	Unix.Trojan.Mirai-5607483-0
cd2301e25f2838684bc2b1e26c7aee51140abba1	Sem assinatura
fb15723cbfd497b8b327992cf7bf928276c9503b	Sem assinatura