CEFET/RJ
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO

UM ARCABOUÇO DE CAIXA-PRETA PARA A DETECÇÃO DE TRÁFEGO MALICIOSO EM AMBIENTES DE TIC

Carlos Alberto Martins de Sousa Teles

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação, do Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, CEFET/RJ, como parte dos requisitos necessários à obtenção do título de mestre.

Orientador(a): Felipe da Rocha Henriques
Coorientador(a): Raphael Carlos Santos Machado

Rio de Janeiro,
Dezembro 2019

UM ARCABOUÇO DE CAIXA-PRETA PARA A DETECÇÃO DE TRÁFEGO MALICIOSO EM AMBIENTES DE TIC

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação, do Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, CEFET/RJ, como parte dos requisitos necessários à obtenção do título de mestre.

Carlos Alberto Martins de Sousa Teles

Banca Examinadora:

Presidente, Professor D.Sc. Felipe da Rocha Henriques (Orientador(a))

Professor D.Sc. Raphael Carlos Santos Machado (UFF) (Coorientador(a))

Professor D.Sc. Pedro Henrique González Silva (CEFET/RJ)

Professor D.Sc. Michel Pompeu Tcheou (UERJ)

Rio de Janeiro,

Dezembro 2019

Ficha catalográfica elaborada pela Biblioteca Central do CEFET/RJ

T269 Teles, Carlos Alberto Martins de Sousa.
Um arcabouço de caixa-preta para a detecção de tráfego malicioso em ambientes de TIC / Carlos Alberto Martins de Sousa Teles – 2019.
78f. [+apêndice] : il.color. , grafs. ; enc.

Dissertação (Mestrado). Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, 2019.
Bibliografia : f. 73-77.
Orientador : Felipe da Rocha Henriques.
Co-orientador: Raphael Carlos Santos Machado.

1. Redes de computadores – Medidas de segurança. 2. Proteção de dados. 3. Segurança da informação. I. Henriques, Felipe da Rocha (Orient.). II. Machado, Raphael Carlos Santos (Co-orient). III. Título.

CDD 005.8

Elaborada pelo bibliotecário Leandro Mota de Menezes CRB-7/5281

DEDICATÓRIA

Dedico este trabalho a minha esposa Angelica Baptista Ramos, aos meus filhos Antônio Carlos e Maria Antônia, que não mediram esforços para me ajudar nessa etapa e por compreender as ausências no período.

AGRADECIMENTOS

O presente trabalho foi desenvolvido com o apoio da Coordenação de Aperfeicionamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

A todos os meus amigos do PPCIC, com os quais pude compartilhar as experiências.

À Carlos Viana, por muito mais que a amizade. Carrego um irmão para vida, meu muito obrigado!

Ao Gabriel Nascimento e Alexandre Martins que nos momentos mais dificeis me fizeram ter confiança em não desistir no decorrer do processo, meu muito obrigado!

À Marcia Lemos, Anselmo Ramalho e Robson da Cruz, companheiros da Embratel;

Aos professores Eduardo Ogasawara e Joel dos Santos, pela paciência, objetividade, presteza e gentileza à frente do PPCIC;

Ao professor Raphael Carlos Santos Machado que, muito gentilmente, me recebeu como seu orientando quando estava no início de minha jornada, mas, por motivos de força maior, não foi possível prosseguir;

Ao professor Felipe da Rocha Henriques, pela viabilidade dessa dissertação, por seu profissionalismo, sua orientação, atenção, controle da minha ansiedade e paciência; muito obrigado.

À banca examinadora, pelo convite aceito para a avaliação deste trabalho.

RESUMO

Um Arcabouço de Caixa-Preta para a Detecção de Tráfego Malicioso em Ambientes de TIC

A segurança da informação está se tornando gradualmente uma área que desempenha um papel importante em nossas vidas cotidianas, em função do crescimento dos ativos de Tecnologia da Informação e de Comunicações (TIC) e os ambientes cada vez mais conectados. Como resultado, os problemas de segurança da informação agora devem ser considerados relevantes a nossa sociedade. Cada vez mais temos informações de Governos, empresas e pessoas tendo seus dados vazados em função de falhas de segurança da informação, tanto em Hardware e Software dos ativos de TIC. Para a inspeção dos ativos de TIC existe o processo de avaliação de segurança, que são procedimentos para verificar o atendimento a requisitos por meio da execução de testes sistemáticos denominados ensaios, os chamados de Programas de Avaliação da Conformidade (Programas de Avaliação da Conformidade (PACs)). No intuito de identificar as falhas dos ativos de TIC, atráves da detecção de tráfego malicioso, este trabalho propõe uma estrutura baseada em caixa preta em ambientes de TIC. O método de caixa preta permite monitorar a rede sem acessar o código do software, que pode ser inviável nas empresas, para seus funcionários. Em nossa proposta, unimos a segurança da informação e do uso de recursos de rede para executar uma detecção confiável de tráfego malicioso. Primeiramente, coletamos informações de tráfego de rede, gerando um conjunto de dados a partir de ferramentas comerciais de rede. Nosso esquema de detecção proposto foi validado verificando a correlação entre alertas de segurança de rede e uso de recursos de rede, como taxa de transferência e número de conexões TCP. Os resultados mostram que o esquema proposto pode detectar com segurança o tráfego malicioso em um ambiente real de rede de TIC.

Palavras-chave: Segurança da informação, TIC, Black–box, Tráfego Malicioso

ABSTRACT

A Black-Box Framework for Malicious TrafficDetection in ICT Environments

Information security is gradually becoming an area that plays an important role in our daily lives as information and communications technology (ICT) assets grow and increasingly connected environments. As a result, information security issues should now be considered relevant to our society. Increasingly we have information from governments, companies and people having their data leaked due to information security flaws in both hardware and software of ICT assets. For the inspection of ICT assets there is a security assessment process, which are procedures for verifying compliance with requirements by performing systematic tests called tests, called Conformity Assessment Programs (Conformity Assessment Programs (PACs)). In order to identify failures of ICT assets, through the detection of malicious traffic, this work proposes a Black–Box–based framework that aims to detect malicious traffic in ICT environments. The black–box method allows to monitor the network without accessing the software code, which may be infeasible in companies, for their employees. In our proposed framework, we ally information of security and network resource usage to perform a reliable malicious traffic detection. Firstly, we collected network traffic information, gererating a dataset from comercial networking tools. Our proposed detection scheme was validated verifying the correlation between network security alerts and network resource usage such as throughput and number of TCP connections. Results show that the proposed scheme can reliably detect malicious traffic in a real ICT network environment.

Keywords: Information Security, ICT, Black–box, Malicious Traffic

LISTA DE ILUSTRAÇÕES

Figura 1 – Arquiteturas de Virtualização Completa - figura adaptada de [Scarfone, 2011] 30

Figura 2 – Arquitetura conceitual 41

Figura 3 – Arquitetura com elementos instanciados 53

Figura 4 – Análise de Segurança da Informação - cenário i - guardiao - evento DNS em IPv4 55

Figura 5 – Análise de Segurança da Informação - cenário i - guardiao - evento DNS em IPv6 55

Figura 6 – Análise de Segurança da Informação - cenário i - guardiao - evento TLS 56

Figura 7 – Análise de desempenho - cenário i - guardião - Pilha TCP 56

Figura 8 – Análise de desempenho - cenário i - guardião - datagrama UDP 57

Figura 9 – Análise de desempenho - cenário i - guardião - tráfego de rede 57

Figura 10 – Análise de Segurança da Informação - cenário i - decodificador de TV - evento HTTP e DNS - 1 58

Figura 11 – Análise de Segurança da Informação - cenário i - decodificador de TV - evento HTTP e DNS - 2 58

Figura 12 – Análise de desempenho - cenário i - decodificador de TV - Pilha TCP 59

Figura 13 – Análise de desempenho - cenário i - decodificador de TV - datagrama UDP 59

Figura 14 – Análise de desempenho - cenário i - decodificador de TV - tráfego de rede e pacotes 59

Figura 15 – Quantidade de pacotes de ENTRADA - cenário i - média por dia 60

Figura 16 – Quantidade de pacotes de ENTRADA - cenário i - média por dia 61

Figura 17 – Quantidade de *Alerts* - Cenário i - total por dia 61

Figura 18 – Quantidade de eventos de DNS - Cenário i - total por dia 62
Figura 19 – Quantidade de eventos de HTTP - Cenário i - total por dia 62
Figura 20 – Análise de Segurança da Informação - cenário ii - Câmera IP - evento DNS em IPv4 - 01 63
Figura 21 – Análise de Segurança da Informação - cenário ii - Câmera IP - evento DNS em IPv4 - 02 64
Figura 22 – Análise de desempenho - cenário ii - Câmera IP - Pilha TCP 65
Figura 23 – Análise de desempenho - cenário ii - Câmera IP - datagrama UDP 65
Figura 24 – Análise de desempenho - cenário ii - Câmera IP - tráfego de rede 66
Figura 25 – Análise de desempenho - cenário ii - Câmera IP - tráfego de pacotes 66
Figura 26 – Vazão de entrada em bits por segundo - Cenário ii - média por dia 67
Figura 27 – Quantidade de pacotes de ENTRADA - Cenário ii - média por dia 68
Figura 28 – Quantidade de *Alerts* - Cenário ii - total por dia 68
Figura 29 – Quantidade de eventos de DNS - Cenário ii - total por dia 69

Figura 30 – Tela execução do Ettercap 81
Figura 31 – Tela inicial do Grafana 87
Figura 32 – Troca de senha inicial do Grafana 88
Figura 33 – Tela inicial do Grafana após login 89
Figura 34 – Tela *Add data source* do Grafana 89
Figura 35 – Configurar *data source* do Grafana 90
Figura 36 – Salvar configuração do *data source* do Grafana 90
Figura 37 – Inflxudb *data source* do Grafana 91
Figura 38 – Como importar o dashboard 91
Figura 39 – Importar *dashboard* 91
Figura 40 – Dados do *dashboard* 928 - 1 92
Figura 41 – Dados do *dashboard* 928 - 2 92
Figura 42 – *Dashboard* 928 - 01 93
Figura 43 – *Dashboard* 928 - 02 93
Figura 44 – *Dashboard* 928 - 03 94
Figura 45 – *Dashboard* 928 - 04 94
Figura 46 – EveBox em execução 96
Figura 47 – Dados do EveBox - 01 97

Figura 48 – Dados do EveBox - 02 97

LISTA DE TABELAS

Tabela 1 – Comparação entre as técnicas de teste *White Box* e *Black Box* 25

Tabela 2 – Comparação entre as técnicas Virtualização 28

Tabela 3 – Resultados consolidados de teste de escalabilidade da solução 70

Tabela 4 – Exemplo para documentação de Rede 79

LISTA DE ALGORITMOS

Algoritmo 1 – Sistema de detecção de tráfego malicioso baseado em informações de segurança e desempenho da rede. 42

LISTA DE ABREVIATURAS E SIGLAS

CPU	Central Process Unit - Unidade Central De Processamento
DHCP	Dynamic Host Configuration Protocol
DHS	Department Of Homeland Security - Departamento De Segurança Interna (EUA)
EPA	U.S. Environmental Protection Agency - Agência De Proteção Ambiental (EUA)
HTTP	Hypertext Transfer Protocol
IDMEF	Intrusion Detection Message Exchange Format
IDS	Sistemas De Detecção De Intrusão
IEC	International Electrotechnical Commission - Comissão Internacional Eletrotécnica
IOT	Internet Of Things - Internet Das Coisas
IP	Internet Protocol
IPS	Inline Intrusion Prevention - Prevenção Contra Intrusões
IPV4	Internet Protocol Version 4
IPV6	Internet Protocol Version 6
ISO	International Organization For Standardization - Organização Internacional Para Padronização
IT	Information Technology - Tecnologia Da Informação
KVM	Kernel-based Virtual Machine - Máquina Virtual Baseada Em Kernel
MAC	Media Access Control
MQTT	Message Queuing Telemetry Transport
NE	Network Element - Elemento De Rede
NSA	National Security Agency - Agência De Segurança Nacional (EUA)
NSM	Monitoramento Seguro De Rede
PACS	Programas De Avaliação Da Conformidade
SIMOC	Simulador De Operações De Guerra Cibernética
TCP	Transmission Control Protocol
TDI	Turbocharged Direct Injection
TIC	Tecnologia Da Informação E De Comunicações
UAT	User Acceptance Testing - Teste De Aceitação Do Usuário
UDP	User Datagram Protocol

VEE Virtual Execution Environment - Ambiente De Execução Virtual

VMM Virtual Machine Monitor - Monitor De Máquinas Virtuais

SUMÁRIO

1	**Introdução**	**17**
1.1	Motivação	21
1.2	Objetivos e contribuições	22
1.3	Estrutura	23
2	**Referencial Teórico: Conceitos Fundamentais**	**24**
2.1	Testes de Software	24
2.2	Virtualização	26
2.2.1	Tipos de Virtualização	27
2.3	Segurança da Informação e Desempenho	31
2.3.1	Monitoração de eventos de segurança	31
2.3.2	Ferramentas de detecção de intrusão	31
2.3.3	Segurança na Internet das Coisas	33
2.3.4	Análise de comportamento e Avaliação da Conformidade	34
2.4	Ambientes para monitoração e simulação de ativos de TIC	36
3	**Proposta do Arcabouço de Caixa-Preta para o Monitoramento de Tráfego Malicioso**	**38**
3.1	Premissas	38
3.2	Arquitetura	39
3.3	Algoritmo de detecção proposto	41
4	**Metodologia Experimental**	**43**
4.1	Rede	43
4.1.1	Servidor DHCP	44
4.1.2	Espelhamento de portas e *Sniffer*	44
4.2	Monitoração	45
4.2.1	Desempenho	46

4.2.2 Segurança da Informação 47
4.3 Virtualização 49
4.3.1 Oracle VirtualBox 50
4.3.2 KVM e QEMU 51

5 Resultados Experimentais 53
5.1 Cenário i - Software de comunicação com banco 54
5.2 Cenário i - Decodificador de TV a cabo 57
5.3 Cenário i - Análise consolidada 60
5.4 Cenário ii - Câmera IP 63
5.5 Cenário ii - Análise consolidada 67
5.6 Análise de escalabilidade da solução 69

6 Conclusões 71

Referências 72

A Orientações para instalação e configuração de ambiente de testes 78
A.1 Introdução 78
A.1.1 Rede 78
A.1.2 Monitoração 82
A.1.3 Virtualização 97

1- Introdução

A segurança dos atuais ambientes tecnológicos é altamente dependente do adequado comportamento dos diversos ativos de tecnologia da informação e de comunicações que os compõem – onde ativos de tecnologia da informação e de comunicações (ativos de Tecnologia da Informação e de Comunicações (TIC)) são utilizados para contemplar todo o tipo de hardware e software capaz de executar processamento computacional e que esteja envolvido em atividades de tecnologia da informação e de comunicações. O termo TIC refere-se à conjugação da tecnologia computacional com a tecnologia das telecomunicações e tem nas redes de computador, e principalmente na Internet, a sua maior expressão [Miranda et al., 2007]. De fato, o funcionamento não adequado de um simples elemento de um sistema computacional, sendo hardware ou software, pode comprometer toda a segurança daquele sistema. É fundamental, portanto, que se desenvolvam mecanismos que permitam averiguar e atestar que ativos de TIC atendem a requisitos desejáveis de segurança. Tal objetivo, no entanto, é um enorme desafio, na medida em que os ativos de TIC usados nos atuais sistemas computacionais tornam-se cada vez mais complexos e estão inseridos em atividades cada vez mais críticas e relevantes para o bem-estar da sociedade [Carlson, 2001].

Uma abordagem que tem ganhado cada vez mais força na conformidade de produtos, serviços, processos, sistemas e pessoas a requisitos normativos é a implantação de Programas de Avaliação da Conformidade (PACs) [Fernandes, 2011]. Estes programas tratam sistemicamente, utilizando-se de tradicionais técnicas de gestão da qualidade, como inspeções, ensaios, amostragem e auditorias, para analisar o atendimento a requisitos preestabelecidos em normas e regulamentos, para cada produto, processo, serviço ou profissional em avaliação. Programas de Avaliação da Conformidade já são usados nas mais diversas áreas para averiguar o atendimento aos mais diversos tipos de requisitos, incluindo a calibração de instrumentos de medição, eficiência energética, compatibilidade eletromagnética, detecção de radiação, isolamento térmico, ergonomia e construção civil [Inmetro, 2018].

Na área de segurança de ativos de TIC, o uso de PACs para verificar o atendimento a requisitos de segurança cibernética ganhou um forte impulso, na última década,

com a publicação dos padrões internacionais: International Organization for Standardization - Organização Internacional para Padronização (ISO)/International Electrotechnical Commission - Comissão Internacional Eletrotécnica (IEC) 15408-1:2009 [ISO, 2015] (*Information technology – Security techniques – Evaluation criteria for Information Technology - Tecnologia da Informação (IT) security*) e ISO/IEC 19790:2012 (*Information technology – Security techniques – Security requirements for cryptographic modules*)[ISO, 2018]. No entanto, ainda que a área de segurança da informação caminhe no sentido da padronização internacional e da busca por procedimentos de ensaio sistemáticos para a avaliação da conformidade, a avaliação de ativos de TIC contempla avançados desafios técnicos decorrentes da complexidade do comportamento de tais ativos, o que demanda a realização de pesquisas para o desenvolvimento de ensaios sistemáticos e conclusivos.

De fato, a história recente do mundo da avaliação da conformidade está repleta de exemplos nos quais ativos de TIC eram capazes de “reconhecer”, ou melhor, de identificar que estavam sob avaliação e, passavam a apresentar o comportamento esperado, sem falhas ou atividades suspeitas, distinto do comportamento sob operação normal.

Nos últimos anos ocorreram diversos casos de ativos de TIC comprometidos, conforme relatado por [Kirtley and Memmel, 2018], em que, por exemplo, vários modelos de babá eletrônica sofreram incidentes de segurança, onde em alguns casos ocorria o envio de áudios assustadores para as crianças, além do roubo de credenciais de acesso a lojas da Apple e Google, além da obtenção indevida da autorização de outros usuários para visualizar e controlar remotamente o monitor. Em outro relato no artigo de [Cziesla et al., 2015], diversas empresas tiveram seus dados violados pela National Security Agency - Agência de Segurança Nacional (EUA) (NSA) através de brechas de segurança em ativos de TIC. E por fim o ”Escândalo Volkswagen” [Blackwelder et al., 2016], onde a mesma declarou ao mundo em 18 de setembro de 2015 com sua resposta ao aviso da Agência de Proteção Ambiental dos EUA (U.S. Environmental Protection Agency - Agência de Proteção Ambiental (EUA) (EPA)) de que os veículos “limpos a diesel” da empresa estavam violando a Lei do Ar Limpo dos EUA. Em sua resposta, a montadora alemã admitiu ter equipado sua linha de motores a diesel com injeção direta turbinada (Turbocharged Direct Injection (TDI)) com um “dispositivo de derrota” que se destinava a “ignorar” os elementos inoperantes do sistema de controle de emissão de um veículo durante o teste de emissões. Os relatórios iniciais indicaram que aproximadamente 500.000 carros a diesel Volkswagen, modelo de 2009 a 2011, foram afetados pelo

dispositivo. Como o escândalo continuou a se desdobrar, pois, o sistema foi vendido a outras montadoras, o número estimado de veículos afetados cresceu para 11 milhões de veículos, a partir de um cruzamento de várias marcas em todo o mundo.

A este cenário de ativos de TIC, junta-se a Internet of Things - Internet das Coisas (IoT) [Alaba et al., 2017]. Este é um novo paradigma tecnológico concebido como uma rede global de máquinas e dispositivos capazes de interagir uns com os outros. A IoT é reconhecida como uma das áreas mais importantes da tecnologia futura e está ganhando grande atenção de uma ampla gama de indústrias. O verdadeiro valor da IoT para as empresas pode ser totalmente percebido quando os dispositivos conectados conseguem se comunicar uns com os outros e se integram a sistemas de inventário gerenciados por fornecedores, sistemas de suporte ao cliente, e análise de negócios [Lee and Lee, 2015].

A empresa de tecnologia Cisco Systems prevê que até 2020, haverá mais de 50 bilhões de dispositivos IoT conectados pela Internet, incluindo carros, utensílios de cozinha, televisões, câmeras Internet Protocol (IP), smartphones, medidores de serviços públicos, sensores intra-corpo, termostatos e quase tudo que pudermos imaginar. Consequentemente, previu-se que as receitas anuais poderiam exceder US$ 470 bilhões para os fornecedores de IoT que vendem hardware, software e serviços. A aplicação de produtos e serviços da IoT permeia todos os setores e indústrias, desde casas a cidades inteligentes, educação, saúde, manufatura, mineração, serviços públicos, comércio, transporte, vigilância, gerenciamento de infraestrutura, cadeia de suprimentos e logística [Cisco, 2017] [Columbus, 2016].

Embora os benefícios da IoT sejam inegáveis, a realidade é que a segurança não está acompanhando o ritmo da inovação. À medida que a IoT se expande, de forma generalizada, espera-se que sua heterogeneidade e escala aumentem as ameaças existentes à Segurança da Informação. Uma vez que humanos, sensores, carros, robôs e drones sejam capazes de interagir perfeitamente entre si de qualquer parte do mundo através da IoT, uma série de ameaças que não podemos sequer imaginar hoje serão reveladas. Se as devidas precauções necessárias não forem tomadas, os indivíduos mal-intencionados aproveitarão a conectividade da IoT para interromper as comunicações, obter vantagens monetárias significativas ou até prejudicar as pessoas fisicamente.

Em outra pesquisa, segundo Restuccia et al. [2018], provou-se que os carros conectados à Internet, através de dispositivos IoT, podem ser controlados remotamente, incluindo operações como destrancar as portas ou até mesmo desligar o carro em

movimento. Os casos mais preocupantes, no entanto, envolvem dispositivos médicos e podem ter consequências fatais na saúde dos pacientes . Conforme já citado o caso de vulnerabilidades em babá eletrônica, os mesmos também são exemplos de ativos de TIC em IoT.

Assim, em alguns anos teremos a rede da IoT como parte de nosso dia a dia. Mais do que nunca, abordar as questões de Segurança da Informação e Privacidade da IoT se tornou mais importante. Percebendo a importância de resolver este problema, o Senado dos EUA recentemente propôs a Lei bipartidária de Melhorias na Segurança Cibernética da Internet das Coisas de 2017, que busca impulsionar a Segurança da Informação em dispositivos conectados à Internet [Restuccia et al., 2018]. A legislação proposta exige que os fornecedores garantam que os dispositivos não contenham vulnerabilidades de segurança conhecidas quando vendidos. Além disso, cumpre garantir a divulgação adequada de novas vulnerabilidades de segurança e para preparar planos de correção para qualquer dispositivo IoT onde vulnerabilidades conhecidas tenham sido descobertas. Isso implica que os fabricantes de IoT precisarão ser proativos e reativos no que se refere à segurança. Em outro esforço para garantir a Segurança da Informação na IoT, foi apresentado pelo Departamento de Segurança Interna (Department of Homeland Security - Departamento de Segurança Interna (EUA) (DHS)) dos EUA alguns princípios estratégicos que são os seguintes: Incorporar a segurança da informação na fase de projeto; Gerenciamento avançado de atualizações de segurança e vulnerabilidades; Desenvolva práticas de segurança comprovadas; Priorizar as medidas de segurança de acordo com o impacto potencial o produto ou serviço; Promover a transparência na IoT e; Conectividade continuada e sem interrupções. O documento do DHS especifica diretamente que a segurança da informação deve ser avaliada como um componente integral de qualquer dispositivo conectado à rede, e que a segurança da IoT deve ser projetada, tendo em mente, interrupções do sistema e de sua operação [Restuccia et al., 2018] [DHS, 2016].

1.1- Motivação

Um dos grandes desafios na avaliação de segurança de ativos de TIC é a realização de ensaios de avaliação da conformidade quando não se dispõe de todos os documentos de engenharia, tais como diagramas elétricos, documentos de engenharia de software e código fonte. Neste caso, é necessário fazer ensaios do tipo *Black Box*, ou em tradução livre, “caixa preta” [Kumar et al., 2015]. Os testes *Black Box* são realizados sem o conhecimento do funcionamento interno do software e hardware dos ativos de TIC. Este tipo de teste também é conhecido como teste funcional ou teste orientado por saída. Em um exemplo de um teste de caixa preta em ativos de TIC, o testador só conhece as entradas e quais devem ser os resultados esperados e como o software trata e gera essas saídas. O testador não examina o código-fonte do software e não precisa de nenhum conhecimento adicional do programa além de suas especificações, se as tiver. Por este motivo, o testador e o programador podem ser independentes um do outro, evitando a tendência do programador em relação ao seu próprio trabalho [Kumar et al., 2015].

De hackers a eventos de segurança, pessoas descuidadas a dispositivos e sistemas operacionais desatualizados ou vulneráveis, a computação em nuvem pública e móvel a provedores de serviços terceirizados, estamos rotineiramente expostos a ameaças de segurança da informação. Dada a natureza onipresente e inevitável dos riscos de segurança, um tempo de resposta rápido é essencial para manter a segurança do sistema, e o monitoramento de segurança contínuo e automatizado é a chave para a rápida detecção e resposta a ameaças.

Mas o que o monitoramento realmente faz? Com a existência de uma grande variedade de Hardware e Software, como testar todos estes ativos de TIC? Uma maneira de responder a essa pergunta é em termos de monitoramento via ensaios *Black Box* e *White Box*. O último termo refere-se ao monitoramento de aplicativo e o primeiro ao monitoramento de servidor ou rede.

Assim, para garantir a conformidade do comportamento do ativo de TIC, mesmo não tendo acesso aos detalhes de sua especificação de software e hardware é uma tarefa árdua. Então, temos nosso principal questionamento: como garantir que, em condições específicas – e não conhecidas pelo avaliador e/ou testador – o ativo de TIC não poderá apresentar comportamento indesejado?

Diante de todos estes potenciais incidentes de segurança podemos concluir que é fundamental desenvolver metodologias que permitam avaliar ativos de TIC, sendo os mesmos IoT ou não, mesmo sem acesso aos seus detalhes de concepção, projeto e implementação.

1.2- Objetivos e contribuições

Propomos uma avaliação sistemática da conformidade de ativos de TIC a requisitos de segurança da informação quando não se tem disponíveis todos os detalhes de implementação do ativo, ou seja, avaliação por meio de testes do tipo caixa preta. Mostramos como uma avaliação baseada no monitoramento do comportamento da rede pode revelar características anômalas e sinalizar não-conformidades. Defendemos a ideia de que por meio da reprodução de cenários de uso diversos, e por meio do monitoramento continuado dos ativos de TIC sob avaliação e do ambiente onde está instalado, é que seja possível identificar tais não-conformidades. Neste sentido, é importante possuir um ambiente de ensaio dotado de recursos que permitam reproduzir cenários de uso distintos e monitorar o comportamento do ativo sob avaliação, identificando e caracterizando não-conformidades.

Neste trabalho, propomos um arcabouço de caixa preta (*Black–box*) baseado em ferramentas de código aberto e que permitam a coleta de dados de segurança de rede, além de possibilitar a detecção de tráfego malicioso em tempo real. Através do ambiente proposto, dados reais de uma rede corporativa foram coletados e uma base de dados foi gerada. Através da correlação entre informações de segurança e do uso de recursos da rede, pode-se realizar a detecção de tráfego malicioso de maneira confiável.

Desse modo, podemos resumir as contribuições deste trabalho através dos itens a seguir:

1. Desenvolvimento de um ambiente de testes de ativos de TIC baseado em ferramentas de virtualização. O ambiente proposto se diferencia das propostas observadas na literatura, na medida em que ele possibilita a instanciação de sofisticados ambientes de teste, inclusive com o uso de ativos de TIC virtualizados. O ambiente foi disponibilizado publicamente na forma de um produto de software em

`https://github.com/carlos-teles/etsg/` para que os experimentos possam ser reproduzidos pela comunidade acadêmica.

2. Desenvolvimento de uma metodologia para a realização de ensaios sistemáticos com foco em segurança. A metodologia proposta se diferencia das observadas na literatura, uma vez que os testes de segurança e desempenho permitem avaliar tanto as falhas de segurança que permitem que os ativos sofram ataques quanto permitem identificar eventuais comportamentos maliciosos por parte do próprio dispositivo sob avaliação.

3. Construção de uma base de dados de referência para o estudo de eventos de segurança e ataques a ativos de TIC a partir do tráfego capturado pela monitoração durante os experimentos.

1.3- Estrutura

Além da presente Introdução, este trabalho está organizado em outros 5 capítulos. O Capítulo 2 introduz o referencial teórico, descrevendo e apresentando uma taxonomia dos principais conceitos e das publicações relacionadas a área de segurança, monitoração e virtualização. No Capítulo 3 apresentaremos as premissas, arquitetura para o ambiente proposto, além do algoritmo para detecção. O Capítulo 4 destaca a metodologia de implementação, ferramentas utilizadas e sua organização para realização dos testes e monitoração do Desempenho e Segurança. No Capítulo 5, explicaremos como o ambiente foi concebido e seus resultados obtidos. E finalmente, concluímos esta dissertação no Capítulo 6.

2- Referencial Teórico: Conceitos Fundamentais

No Capítulo 1, apresentamos um breve histórico, motivação e objetivo do trabalho. Neste capítulo abordaremos o referencial teórico que nos levaram à organização da nossa proposta e implementação do ambiente de testes. Assim, este capítulo foi dividido em quatro seções, de acordo com os conceitos levados em consideração para a proposta: trabalhos sobre ensaios Testes de Software em 2.1, sobre Virtualização em 2.2, sobre Segurança da Informação e Desempenho em 2.3 e a busca por ambientes existentes para monitoração e simulação de ativos de TIC em 2.4.

Nesta última seção, uma busca na literatura indicou se há ambientes em funcionamento com as características elencadas por nós neste trabalho como essenciais para o devido monitoramento de ativos de TIC em rede, quais sejam: monitoração de todo o tráfego da rede (com visão de segurança da informação e desempenho) onde se encontra o ativo de TIC; possibilidade de virtualização de ativos de TIC em múltiplas arquiteturas; e exportação de dados para correlação.

2.1- Testes de Software

Na área de software, o principal objetivo dos testes é identificar falhas para que os mesmos possam ser corrigidos. O teste de software também é utilizado em relação a outros fatores de qualidade, como confiabilidade, usabilidade, integridade, segurança, capacidade, eficiência, portabilidade, capacidade de manutenção, compatibilidade, etc. Esta é uma área muito ampla, que envolve aspectos técnicos e áreas não-técnicas, como especificação, design e implementação, manutenção, processos e gerenciamento de problemas em Engenharia de Software. As técnicas mais importantes usadas para encontrar erros são os testes de *Black Box* e os de *White Box* [Kumar et al., 2015].

Além dos testes de caixa preta e caixa branca, existem os testes de caixa cinza (*grey box test*), relatada por [Khan et al., 2012], [Acharya and Pandya, 2012] e [Sjöberg et al., 1995]. Contudo, esse tipo de teste não será considerado neste trabalho, pois, é

uma técnica com conhecimento limitado para o detalhamento de ativos de TIC, tanto de software e ou hardware. Em princípio, isto poderia significar que o testador conhece algumas partes do código fonte e não outras, mas, na prática, isto geralmente significa que o testador tem acesso a artefatos mais detalhados do que especificações ou requisitos [Michael et al., 2013].

A técnica para testes *Black Box* é uma atividade de teste sem que haja qualquer conhecimento, via código-fonte, ou acesso ao funcionamento da lógica interna e estrutura do software. Esta técnica examina os aspectos fundamentais de um sistema, via software e hardware, de entradas e saídas aguardadas do ativo de TIC a ser testado. Uma das características, segundo Sjöberg et al. [1995], é que nenhum resultado deve ser excluído dos testes, pois, há um espectro muito rico das descrições de modelos possíveis que devem ser tratados.

A técnica para testes *White Box* é uma atividade investigativa detalhada da lógica interna e estrutura do software, via código-fonte, e hardware, via esquemas e diagramas elétricos. Para tal é preciso que o testador tenha grande conhecimento do ativo de TIC a ser testado. A Tabela 1 apresenta uma comparação entre as técnicas de teste de caixa branca e de caixa preta.

Tabela 1 – Comparação entre as técnicas de teste *White Box* e *Black Box*

	Teste Black Box	*Teste White Box*
1	Não é adequado para teste de algoritmo	É adequado para testes de algoritmo (adequado para todos)
2	Pode testar apenas pelo método de tentativa e erro	Teste melhor: domínios de dados e limites internos
3	É menos exaustivo e demorado	Potencialmente mais exaustivo e demorado
4	O teste é baseado em exceções externas - o comportamento interno do programa é ignorado	Interno são totalmente conhecidos
5	Executado por usuários finais e também por testador e desenvolvedores (User Acceptance Testing - Teste de aceitação do usuário (UAT))	É realizado por desenvolvedores e testadores
6	A granularidade do teste é baixa	A granularidade é alta
7	Aspectos fundamentais da análise apenas, ou seja, sem margem comprovada de trabalho interno	Pleno conhecimento do Software e Hardware

Tabela adaptada de [Khan et al., 2012]

2.2- Virtualização

A virtualização tornou-se parte integrante da maioria das organizações e está cada vez mais difundida em vários setores. Isto reduziu os custos de TIC e aumentou consideravelmente sua receita [Ameen and Hamo, 2013]. A vantagem da virtualização depende de sua capacidade de reduzir custos e fornecer um meio eficaz de gerenciar os ativos de TIC. Sua definição é: uma técnica capaz de ocultar as características físicas dos recursos de computação da maneira como outros sistemas, aplicativos ou usuários finais interagem com esses recursos.

Para [Francia III et al., 2012], é introduzido uma camada de abstração de software entre o hardware e o sistema operacional e os aplicativos executados sobre ela. Esta camada de abstração (Virtual Machine Monitor - Monitor de Máquinas Virtuais (VMM)), basicamente, oculta os recursos físicos do sistema de computação do sistema operacional. Como os recursos de hardware são controlados diretamente pelo VMM e não pelo sistema operacional, é possível executar vários sistemas operacionais (possivelmente diferentes) em paralelo no mesmo hardware. Como resultado, a plataforma de hardware é particionada em uma ou mais unidades lógicas chamadas de máquinas virtuais.

Segundo Francia III et al. [2012] e Ameen and Hamo [2013] a virtualização foi desenvolvida primeiramente pela IBM Corporation em 1960, originalmente para particionar grandes computadores, *mainframe*, em várias instâncias lógicas e para rodar em um único hardware físico como o *host*. Esse recurso foi inventado porque a manutenção dos computadores (*mainframe*) tornou-se incômoda. Em Ameen and Hamo [2013], os cientistas da IBM perceberam que esta capacidade de particionamento permite que vários processos e aplicativos sejam executados ao mesmo tempo, aumentando assim a eficiência do ambiente e diminuindo a sobrecarga de manutenção.

A partir dos 1980 e início dos anos 1990 ocorreu uma mudança que trouxe a computação distribuída para os *data centers*. A computação centralizada e o interesse da virtualização diminuíram, sendo substituídos por servidores individuais e com funções dedicadas: e-mail, banco de dados, web e aplicativos. Após investimentos significativos em arquiteturas e computação distribuída, houve uma renovação no foco em torno das máquinas virtuais, como solução complementar para projetos de servidores virtualizados, além do gerenciamento de *data center* [Ameen and Hamo, 2013].

Assim, a virtualização foi criada no *mainframe* e após 30 anos de sua criação, no início dos anos de 1990, foi introduzida na plataforma x86. Avanços tecnológicos em hardware e software tornaram as máquinas virtuais estáveis e com menor custo [Ameen and Hamo, 2013].

Segundo Zhang et al. [2010], a virtualização é uma tecnologia que abstrai os detalhes do *hardware* e fornece recursos virtualizados para aplicativos de alto nível. Um servidor virtualizado é comumente chamado de máquina virtual (VM). A virtualização forma a base da computação em nuvem, pois fornece a capacidade de agrupar recursos de computação de *clusters* de servidores e atribuir dinamicamente, acrescentando ou removendo, recursos virtuais a aplicativos sob demanda.

2.2.1 Tipos de Virtualização

Conforme explicado por [Singh and Singh, 2018], existem diversos tipos de virtualização, em função de suas arquiteturas, e alguns dos mais importantes quais são:

1. Virtualização completa;
2. Virtualização assistida por hardware;
3. Paravirtualização;
4. Virtualização parcial;
5. Virtualização híbrida e;
6. Virtualização em nível de sistema operacional.

Neste momento daremos foco, com uma breve definição, em três tipos de Virtualização importantes para este trabalho: Virtualização completa, Virtualização assistida por hardware e Paravirtualização.

A Virtualização completa realiza a abstração total do sistema físico e cria um sistema virtual completo. Não é necessário fazer qualquer modificação no Sistema Operacional ou na aplicação que está rodando nesta modalidade. Este tipo de Virtualização

facilita a migração de máquinas virtuais entre servidores físicos, pois existe total independência das aplicações e dos recursos físicos do servidor. Também, a segurança é facilitada pelo isolamento entre as máquinas virtuais. O *hypervisor*, que é parte desta solução de virtualização, são responsáveis por controlar o hardware e criar um ambiente virtualizado seguro para os usuários trabalharem. Entretanto, o desempenho neste caso pode ser prejudicado, pois o *hypervisor* controla todo processo e toda a chamada ao hardware é feita sob a sua supervisão. Também a implementação de uma máquina virtual que emule cada dispositivo de hardware é uma tarefa complexa, pois isto é feito baseado em hardwares genéricos, o que influi no desempenho [VERAS, 2011].

A Paravirtualização surgiu como uma forma de contornar as desvantagens de uso da Virtualização completa, no que diz respeito ao processamento. A máquina virtual enxerga uma abstração do hardware que não é idêntico ao hardware físico. Os dispositivos de hardware são acessados por drivers de dispositivo do próprio *hypervisor*, o que é interessante, pois otimiza o desempenho. Uma desvantagem é que a Paravirtualização requer modificação do sistema operacional convidado [VERAS, 2011].

Virtualização assistida por hardware é uma abordagem de virtualização de plataforma que permite a virtualização completa eficiente usando a ajuda de recursos de hardware, principalmente dos processadores *host*. A virtualização assistida por hardware foi adicionada aos processadores x86 (Intel VT-x ou AMD-V) em 2005 e 2006 (respectivamente) [VERAS, 2011]. A Tabela 2 apresenta a comparação entre as três técnicas de virtualização supracitadas.

Tabela 2 – Comparação entre as técnicas Virtualização

	Virtualização completa	Paravirtualização	Virtualização assistida por hardware
Técnica	Translação Binária e Execução Diretas	Saída para modo *root* nas instruções privilegiadas	*Hypercalls*
Compatibilidade	Alta	Alta	Baixa
Desempenho	Bom	Médio	Bom (sob certos aspectos)
Independência entre SO convidado e VMM	Sim	Sim	Não

Tabela adaptada de [VERAS, 2011]

Na virtualização completa, o *hypervisor* fornece a maioria das mesmas interfaces de hardware fornecidas pela plataforma física do hardware. Isso significa que os sistemas operacionais e os aplicativos em execução na virtualização completa não precisam ser modificados para que a virtualização funcione se os sistemas operacionais e os aplicativos forem compatíveis com o hardware subjacente.

O recente aumento no uso de produtos e serviços de virtualização completa foi impulsionado por muitos benefícios. Um dos motivos mais comuns para a adoção da virtualização completa é a eficiência operacional: as organizações podem usar o hardware existente (e novas aquisições de hardware) com mais eficiência, colocando mais carga em cada computador. Em geral, os servidores que usam a virtualização completa podem usar mais recursos de processamento e memória do computador do que os servidores que executam uma única instância do sistema operacional e um único conjunto de serviços. Avanços recentes nas arquiteturas de Central Process Unit - Unidade Central de Processamento (CPU) tornaram a virtualização completa mais rápida do que há alguns anos, e espera-se que avanços semelhantes continuem a ser feitos tanto pelos fornecedores de CPU quanto pelos fornecedores de software de virtualização. Além disso, as alterações na arquitetura da CPU tornaram a virtualização completa mais segura reforçando as restrições de *hypervisor* nos recursos.

A virtualização completa tem algumas implicações negativas de segurança. Ela adiciona camadas de tecnologia, o que pode aumentar a sobrecarga de gerenciamento de segurança, exigindo controles de segurança adicionais. Além disso, combinar muitos sistemas em um único computador físico pode causar um impacto maior se ocorrer um comprometimento de segurança. Além disso, alguns sistemas de virtualização facilitam o compartilhamento de informações entre os sistemas; essa conveniência pode se tornar um vetor de ataque se não for cuidadosamente controlada. Em alguns casos, os ambientes virtualizados são bastante dinâmicos, o que torna a criação e manutenção dos limites de segurança necessários mais complexos.

Existem duas formas de virtualização completa, conforme descrevemos na figura 1. Na virtualização completa nativa, o *hypervisor* é executado diretamente no hardware subjacente, sem um sistema operacional hospedado; o *hypervisor* pode até ser incorporado no firmware do computador. Na outra forma de virtualização completa, conhecida como virtualização completa hospedada, o *hypervisor* é executado na parte superior do sistema operacional hospedado; o sistema operacional hospedado pode ser Windows,

Linux ou MacOS. As arquiteturas de virtualização hospedadas geralmente também têm uma camada adicional de software (o aplicativo de virtualização) em execução no sistema operacional convidado que fornece utilitários para controlar a virtualização enquanto no sistema operacional convidado, como a capacidade de compartilhar arquivos com o sistema operacional hospedado. As arquiteturas de virtualização hospedadas também permitem que os usuários executem aplicativos como navegadores da Web e clientes de e-mail juntamente com o aplicativo de virtualização hospedado, ao contrário das arquiteturas nativas, que só podem executar aplicativos em sistemas virtualizados [Scarfone, 2011]. Em nosso trabalho utilizaremos a Virtualização Completa Hospedada, em função das características acima definidas.

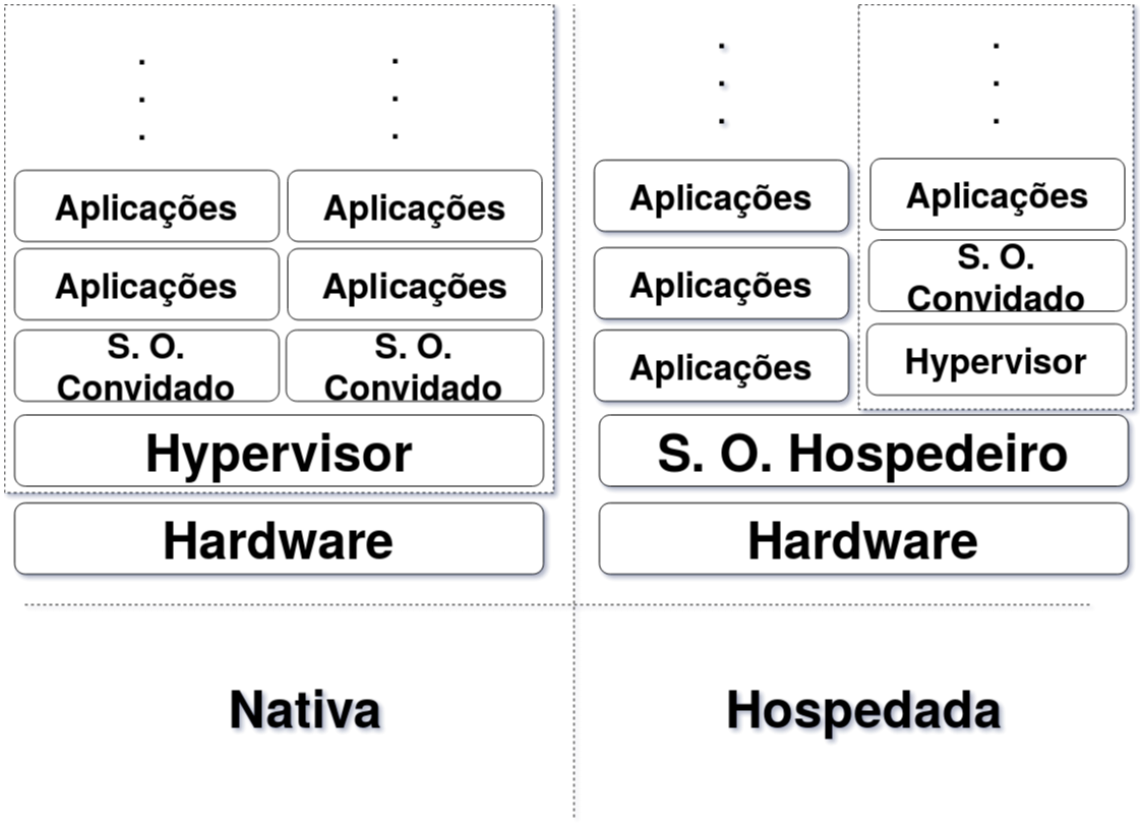

Figura 1 – Arquiteturas de Virtualização Completa - figura adaptada de [Scarfone, 2011]

2.3- Segurança da Informação e Desempenho

Conforme identificado no Capítulo 1, a Segurança da Informação tornou-se relevante na maioria das organizações. Seu objetivo é assegurar a continuidade dos negócios e minimizar os danos causados ao prevenir e minimizar o impacto de incidentes de segurança sobre os ativos de TIC. A mesma é influenciada por três propriedades principais: Confidencialidade para limitar o acesso à informação somente aos agentes autorizados; Integridade para preservar que as informações tenham suas características fiéis à sua origem e que qualquer alteração durante o processo tenha sido realizada com autorização e controle e; Disponibilidade para garantir que a informação esteja acessível aos autorizados a todo tempo que precise ser resgatada [von Solms, 1998] [Sêmola, 2003].

Incidentes de Segurança ou ataques segurança da informação são classificados como ataques passivos, que incluem leitura não autorizada de uma mensagem de arquivo e análise de tráfego ou ataques ativos, como modificação de mensagens ou arquivos e negação de serviço [Stallings, 2017].

2.3.1 Monitoração de eventos de segurança

A monitoração de segurança, envolve a coleta e análise de informações para detectar comportamento suspeito ou alterações não autorizadas do sistema na rede, definindo quais tipos de comportamento devem acionar alertas e tomar medidas em alertas conforme necessário [Mendonça, 2015].

2.3.2 Ferramentas de detecção de intrusão

Os *Intrusion Detection Systems* (Sistemas de Detecção de Intrusão (IDS)) são ferramentas de segurança que, como outras medidas, como software antivírus, *firewalls* e

esquemas de controle de acesso, têm por objetivo fortalecer a segurança dos sistemas de informação e comunicação. Uma de suas características é que este reconhece uma intrusão em um determinado dispositivo e salva como mensagem Intrusion Detection Message Exchange Format (IDMEF) o endereço Internet Protocol version 4 (IPv4) ou Internet Protocol version 6 (IPv6) e o endereço Media Access Control (MAC) do dispositivo [Wood and Erlinger, 2007].

Segundo Hallstensen [2017], um *Network Security Monitoring* (Monitoramento Seguro de Rede (NSM)) pode ser visto como um laço de repetição, consistindo de coleta, detecção e análise de dados de segurança de rede para abordar os quatro principais elementos, quais sejam:

1. Proteger a rede, concentrando-se em proteger sistemas e impedir a exploração e a invasão de sistemas endógenos de rede e computador, a varredura de vulnerabilidades e o gerenciamento de vulnerabilidades, avaliações de riscos e gerenciamento de riscos;

2. Detectar ameaças para a rede, concentrando-se na detecção de intrusões atualmente ativas ou intrusões que foram bem-sucedidas no passado, monitorando sistemas, detectando ataques e emitindo alarmes e avisos;

3. Responder ameaças na rede, concentrando-se em responder a intrusões, isolando ativos comprometidos, realizando análises forenses e de rede, análise de *malware* e relatórios;

4. Sustentar as capacidades operacionais do CND, concentrando-se no gerenciamento de pessoas, processos e tecnologias nas formas de desenvolvimento de capacidades, implementação de sistemas, contratação de pessoal, desenvolvimento de políticas e redação de rotinas.

Uma adição à gama de produtos de segurança é o sistema de prevenção de intrusões (Inline Intrusion Prevention - Prevenção contra intrusões (IPS)). Este é uma extensão de um IDS que inclui a capacidade de tentar bloquear ou impedir atividades maliciosas detectadas. Como um IDS, um IPS pode ser baseado em *host*, baseado em rede ou distribuído. Da mesma forma, ele pode usar a detecção de anomalia para identificar comportamentos que não são de usuários legítimos ou assinatura para identificar um comportamento malicioso conhecido. Quando um IDS detecta atividade maliciosa, ele

pode responder modificando ou bloqueando pacotes de rede em um perímetro ou em um *host*, ou modificando, ou bloqueando chamadas do sistema por programas executados em um *host*. Assim, um IPS de rede pode bloquear o tráfego, como faz um *firewall*, mas faz uso dos tipos de algoritmos desenvolvidos para IDSs para determinar quando fazer isso. É uma questão de terminologia se um IPS de rede é considerado um novo tipo de produto, ou simplesmente outra forma de *firewall* [Stallings, 2017].

2.3.3 Segurança na Internet das Coisas

Com o avanço e a popularização da IoT, novos projetos, soluções e pesquisas surgem em diferentes áreas do conhecimento, tais como Saúde e áreas de cuidados médicos, Industria, Logística, Eletricidade entre outras [Al-Fuqaha et al., 2015], [Atzori et al., 2010] [Whitmore et al., 2015]. A IoT possui diversos desafios a serem desenvolvidos e um deles está relacionado a área de Segurança da Informação e medições de Desempenho.

Segundo Shinde [2016], com o crescimento da utilização de dispositivos IoT, há um impulso nas aplicações de missão crítica, existindo, assim, a necessidade crucial de se monitorar e gerenciar proativamente seus níveis de escalabilidade, de disponibilidade, desempenho e segurança. Uma falha nas medições pode levar a um impacto direto nos negócios e, agora, atingindo a vida humana, à medida que a IoT é cada vez mais usada na área da saúde. Em sua pesquisa, o monitoramento está relacionado a utilização de protocolos de mensagens como Message Queuing Telemetry Transport (MQTT), RESTful[1] Hypertext Transfer Protocol (HTTP), Transmission Control Protocol (TCP) e etc.

Para Gupta [2015], no advento da IoT, supõe-se que muitos outros dispositivos serão conectados à infraestrutura de rede existente. Como resultado, espera-se que o monitoramento se torne mais complexo para os administradores, pois as redes tendem a se tornar mais heterogêneas. Além disso, o endereçamento para IoTs será mais complexo, dada a escala em que os dispositivos serão adicionados à rede e, portanto, o monitoramento está fadado a se tornar uma tarefa difícil devido ao gerenciamento de

[1] É um conjunto de princípios que definem como os padrões da Web, como HTTP e URIs, devem ser usados [Wilde and Pautasso, 2011].

uma gama maior de endereços. Em sua pesquisa, demonstra-se o uso dos sistemas *Big Brother* e o *Zenoss*. Ambos podem monitorar uma grande variedade de aplicativos, incluindo processos, eventos e até registros. Ambos podem funcionar em ambientes heterogêneos, onde, além de máquinas virtuais ou físicas, outros dispositivos de rede, como roteadores e switches, também podem ser monitorados.

Além disso, conforme Restuccia et al. [2018], uma das características mais inovadoras da IoT é a permissão para que os objetos sejam detectados ou controlados remotamente em toda a infraestrutura de rede existente ou *ad hoc*, criando uma infinidade de oportunidades para uma integração mais direta com o mundo físico. Isso resulta em maior eficiência, precisão e benefício econômico, além da redução da intervenção humana. Outra característica marcante da IoT é sua Heterogeneidade. Devido à enorme possibilidade de dispositivos conectados, uma infinidade de diferentes protocolos, algoritmos e padrões de IoT coexistirão necessariamente, especialmente no domínio de rede, enquanto alguns fabricantes estão adotando padrões de IoT mais abertos, como MQTT e os protocolos da IETF (Internet Engineering Task Force) para dispositivos IoT restritos. Além disso, a maioria das pesquisas existentes supõe a existência de uma associação estática entre os recursos da IoT e as entidades do mundo real vizinhas. Pelo contrário, o ambiente de IoT é extremamente heterogêneo, tanto em arquitetura de *Hardware* e *Software*, além de dinâmico, pois, os dispositivos de IoT podem experimentar mobilidade imprevisível, o que resulta em variações súbitas de capacidade de comunicação e posicionamento ao longo do tempo. Esse ambiente torna a resolução de dispositivos IoT disponíveis uma tarefa desafiadora.

2.3.4 Análise de comportamento e Avaliação da Conformidade

A análise de comportamento de um ativo de TIC é uma tarefa dividida em dois principais pontos: a conectividade com a rede e o monitoramento.

A conectividade fornecida com a rede a um ativo de TIC deve ser transparente e de forma que o mesmo saiba que está em uso, ou melhor, em operação. Alguns dispositivos apresentam um modo de ”teste”, ”demo” ou ”loja”. Este tipo de operação demonstra algumas funcionalidades do dispositivo, mas, principalmente, como modo de exibição de

suas funcionalidades principais. Entretanto podem também esconder funcionalidades como captura de áudio, de localização, imagens ou vídeo. Outro modo existente em diversos ativo de TIC é o "*standby*". Geralmente, quando um dispositivo se encontra em algum dos modos acima citados, sua conectividade não é transparente, ou seja, pode estar transmitindo e/ou operando de forma a esconder funcionalidades.

Pode-se fazer um paralelo com o caso da Volkswagen [Blackwelder et al., 2016], onde a mesma admitiu que instalou um dispositivo nos carros, e os resultados sobre emissões de poluentes foram alterados. O software foi instalado em mais de 11 milhões de veículos a diesel em todo o mundo, em modelos de várias marcas pertencentes ao grupo. A desconfiança partiu da diferença entre níveis de emissão encontrados em testes de rodagem e os oficiais. Após investigar, a Agência de Proteção Ambiental (EPA) concluiu que um software instalado pela montadora detecta quando o carro está sendo inspecionado para verificar o nível de emissão de poluentes e só então passa a controlar os gases que o veículo solta na atmosfera. Este controle fica desligado em situações normais de rodagem, fazendo com que os carros poluam muito além do nível exigido no país [Presse, 2015].

Assim, o monitoramento de um ativo de TIC deve estar atento a todas as formas de operação. Segundo a Organização Mundial do Comércio (OMC), a avaliação da conformidade engloba qualquer procedimento utilizado, direta ou indiretamente, para determinar que as prescrições pertinentes a regulamentos técnicos ou normas são cumpridas [Inmetro, 2015].

Além desta definição, o Instituto Nacional de Metrologia, Qualidade e Tecnologia (Inmetro) entende que: "A Avaliação da Conformidade é um processo sistematizado, com regras preestabelecidas, devidamente acompanhado e avaliado, de forma a propiciar adequado grau de confiança de que um produto, processo ou serviço, ou ainda uma pessoa, atende a requisitos preestabelecidos em normas ou regulamentos, com o melhor custo benefício possível para a sociedade" [Inmetro, 2012].

Podemos definir uma não conformidade como o não atendimento a um requisito, ou seja, quando uma empresa, instituição ou pessoa não opera de acordo com os requisitos. Este requisito não necessariamente precisa estar definido apenas na norma ISO, podendo ser também um requisito do cliente, um procedimento interno ou ainda a falha em atender a um requisito legal, que quando não cumprido, se torna uma não conformidade. Podemos ainda afirmar que um produto se encontra em não conformidade

quando apresenta um comportamento fora de suas especificações ou que apresenta uma funcionalidade contra suas especificações encontra-se em não conformidade.

Além disso, podemos ter um comportamento suspeito, ou seja, ainda se falando em produto, este, passa por um comportamento ou funcionalidade apresentada pelo produto que não vai explicitamente contra as especificações, mas que pode representar potencial vulnerabilidade de segurança ou ação maliciosa.

2.4- Ambientes para monitoração e simulação de ativos de TIC

Em busca de tornar o ambiente em que o ativo deTIC estará em operação de forma a não ter visibilidade que está sendo testado ou inspecionado, foi feita uma pesquisa *ad hoc* visando encontrar às iniciativas que estivessem relacionados a monitoração de Segurança da Informação e Desempenho, associadas ao uso de Virtualização e Teste *Black Box*. Encontramos ao menos quatro ambientes que possuem algumas características, mas, não todas.

O GNS3 [Welsh, 2013] é um emulador de software de rede. Nos permite a combinação de dispositivos virtuais e reais, assim como nosso trabalho, para simular redes complexas. Possui suporte a utilização do *Dynamips*, que permite emular roteadores Cisco e fornece uma coleção de dispositivos e interfaces genéricos; Kernel-based Virtual Machine - Máquina Virtual baseada em Kernel (KVM) e QEMU para máquinas virtuais; emulação de dispositivos Cisco ASA e roteadores Juniper; Também o Pemu[2] é uma variação do QEMU utilizada para *firewalls* Cisco PIX e o; Oracle VirtualBox que citaremos abaixo. Além disso, permite a utilização em conjunto do Wireshark, que é aplicativo para captura de pacotes de código aberto e do Virtual PC Simulator (VPCS) que permite simular até nove máquinas virtuais, além de executar ping e traceroute. Entretanto, é necessário bastante de CPU e memória RAM para alocar todos estes recursos.

O Simulador de Operações de Guerra Cibernética (SIMOC) [Machado, 2018] é um sistema utilizado para treinar os militares a combates no espaço digital. Este é empregado pela Seção de Guerra Cibernética, do Centro de Instrução de Guerra Eletrônica (CIGE), na formação dos alunos do Curso de Guerra Cibernética do Exército Brasileiro. Pode

[2]https://github.com/haishanh/pemu - (Acesso em: 30 set. 2019)

ser utilizado também para treinamento de civis. Possui como vantagens a possibilidade de escalabilidade, custo baixo de implementação e redução de tempo para treinamento. Como desvantagens foram identificadas a limitação do uso de um simulador baseado em uma solução de virtualização de rede com os principais ativos de TIC. Assim, os cenários criados podem se tornar muito acadêmicos e teóricos, divergindo do objetivo do treinamento, que é fazer uso de redes corporativas com ativos de TIC. O projeto utiliza uma interface Web em Java e como plataforma de virtualização privada selecionou o VMware.

O Docker Security Playground Perrone and Romano [2017] é a implementação de uma arquitetura que utiliza uma abordagem baseada em microsserviços para construir infraestruturas de rede complexas especificamente adaptadas ao estudo da segurança de rede. Este foi concebido como uma ferramenta para ensinar segurança da informação em redes de computadores com uma abordagem prática. Vários laboratórios podem ser instanciados. Utiliza-se de virtualização em nível de sistema operacional via Docker. O projeto utiliza de interface Web em NodeJS.

O CoreLab [Nakao et al., 2008] propõe um novo *testbed* de rede e é baseado em um VMM como um Virtual Execution Environment - Ambiente de Execução Virtual (VEE) para que os serviços de rede sejam executados. Utiliza o desenvolvimento de um protótipo de plataforma de rede usando o KVM e QEMU como um VMM hospedado.

Uma das considerações que fazemos em não selecionar os sistemas acima citados foi que estes, após configurados, não permitem a conexão de ativos de TIC aos ambientes. Existe apenas a possibilidade de expandir os ambientes através de ferramentas existentes nos mesmos.

Comparativamente, o GNS3 e o DSP poderiam ser adaptados ao nosso trabalho; entretanto, necessitariam de alteração em código-fonte, mais memória, CPU e armazenamento. O DSP e o Corelab não possuem uma plataforma de monitoração associada. O SIMOC e o DSP utilizam apenas arquitetura x86 e x86-64, não instanciando outras (arm, mips e etc.).

Este trabalho difere principalmente dos trabalhos apresentados nesta seção por possibilitar: (i) inspeção de pacotes de todos os ativos de TIC do ambiente com detecção de eventos/incidentes de segurança; (ii) análise de desempenho de todos os ativos de TIC do ambiente; (iii) extração de dados de desempenho e segurança para análise e histórico.

3- Proposta do Arcabouço de Caixa-Preta para o Monitoramento de Tráfego Malicioso

A partir das definições apresentadas no Capítulo 2, discutiremos nossa proposta, levando em consideração os ensaios *Black Box* e *White Box* em 2.1, sobre Virtualização em 2.2 e ainda sobre Segurança da Informação e Desempenho em 2.3, fazendo sua ligação e relação para avaliação de ativos de TIC através de monitorações, conforme descreveremos a seguir.

3.1- Premissas

Nossa proposta tem em vista a utilização de ferramentas comerciais que possam ser integradas, e que nos permitam realizar uma monitoração com foco em segurança da informação e desempenho. Além disso, pretende-se que a proposta seja genérica, podendo ser basicamente composta por:

1. Uma ou mais redes com ativos de TIC virtualizados;
2. Uma ou mais redes rede com ativos de TIC físicos;
3. Uma ou mais redes com ativos de TIC virtualizados e físicos.

Conforme apresentamos na seção 2.1, os testes de *Black Box* e *White Box* possuem diferentes abordagens e características. É importante perceber a importância dos dois tipos de monitoramento. Historicamente, havia uma lacuna no monitoramento de aplicativos, e isso apresentava muitos problemas porque o monitoramento de *Black box* captava problemas com sistemas, como alta carga de CPU ou alto tráfego de rede, mas não haveria informações no lado do aplicativo para mostrar por que isso estava acontecendo - e, na maioria das vezes, esses problemas são causados por problemas de aplicativos, não por problemas de ativos de TIC.

Um bom exemplo de monitoramento de *White box* e sua importância pode ser demonstrado com um alerta de verificação de monitoramento de *Black box*, mostrando como os dois trabalham juntos. Tomemos, por exemplo, um alerta de monitoramento de caixa preta que informa que o uso da CPU do nosso servidor está em 100%. Vamos investigar esse problema e ver que os processos do MySQL são a causa desse alerta. Se tivermos monitoramento de caixa branca para verificar também as consultas em execução no MySQL, a quantidade de conexões no MySQL e a quantidade de tempo que elas levam para que as consultas sejam executadas, então temos muito mais informações para ajudar a diagnosticar o problema. Isso pode nos permitir demonstrar que um aplicativo está executando uma consulta que é muito intensiva em recursos ou foi mal projetada e fornece retorno às equipes de aplicativos com evidências concretas.

Assim, em nosso trabalho, utilizaremos a abordagem *Black Box* associada com a Monitoração, não nos furtando se necessário do uso de monitoramento de *White box*. Tal abordagem se faz necessária nos ativos de TIC em função de:

1. Ativos de TIC são multi arquitetura (x86, x86_64, arm, aarch64, mips e etc). Uma análise de projeto de uma especificação de *Hardware* demanda muito tempo, o que fará com que projetos possam ser inviabilizados.
2. Uma revisão de código-fonte também demanda muito tempo, fazendo que determinados projetos seriam inviabilizados;
3. Nem sempre todos os detalhes da implementação de *Hardware* e/ou *Software* estão disponíveis;
4. Existirem funcionalidades de *Hardware* e/ou *Software* serem deliberadamente ocultadas.

3.2- Arquitetura

A partir da apresentação dos conceitos, na seção anterior, descreveremos nossa proposta de testes de caixa preta em ativos de TIC. Dividiremos este ambiente em duas partes: um ambiente de monitoração e um ambiente dos ativos de TIC, conforme podemos observar na Figura 2.

No ambiente de ativos de TIC, estarão elementos virtualizados, elementos físicos, elementos que devem simular acesso à Internet, elementos que devem simular acesso à rede local, elementos que devem simular acesso ao ativo de TIC que passará pelos testes, ou seja, todos os elementos conectados, por cabo ou sem fio, com acesso à Internet, serão monitorados. Sua conexão com rede se dará por um ativo de TIC preparado para capturar e copiar Network Element - Elemento de rede (NE) todas as informações que passam na rede, ou seja, todos os dados dos ativos de TIC a serem monitorados serão copiados para outro ambiente que processará esta informação de forma a identificar se há comportamento suspeito, malicioso ou em não conformidade.

Esta outra parte da proposta de teste de caixa preta em ativos de TIC é composta pelo ambiente de monitoração. Este ambiente receberá uma cópia de todo o tráfego, entrada e saída para Internet ou rede local, do ambiente de ativos de TIC. Estas informações serão processadas pelos componentes de Segurança e de Desempenho. A partir da investigação de tais informações, o sistema proposto visa identificar não-conformidades, ou seja, comportamentos, requisitos não especificados e funcionalidades incompatíveis com as especificações do ativo de TIC em teste. Estas informações serão processadas pelo ambiente de monitoração, sem o auxílio do código-fonte do ativo de TIC em teste ou da documentação de engenharia do mesmo. Assim, eventuais desvios serão identificados pelo ambiente de monitoração, somente por meio da observação do comportamento do ativo de TIC.

Cabe ressaltar que tal ambiente, para testes de caixa preta ao qual o ativo de TIC estará conectado, deve simular um ambiente real, de forma que não seja possível pelo ativo de TIC em teste identificar que está sendo testado ou inspecionado, visto que há ativos de TIC com software maliciosos e que tendem a buscar tal contexto. Por isso, o ambiente ao qual o ativo de TIC estará conectado deve ter uma conexão com a Internet, visto que outra possibilidade é a busca por uma informação externa, pelo ativo de TIC em teste, para averiguar o ambiente ao qual estará conectado. A busca por uma heterogeneidade de arquiteturas e sistemas operacionais para as máquinas virtuais existentes no ambiente de ativos de TIC foi uma das soluções encontradas em nosso trabalho para que não seja possível, pelo ativo de TIC em teste, descobrir um padrão de atividades ao qual está submetido no teste, de maneira que não se identifique também em teste. Todos os ativos de TIC, mesmo máquinas virtuais que fazem parte do ambiente, também serão monitorados, para que se veja a interação do ativo de TIC em teste.

Portanto, para tal ambiente de testes de caixa preta, faz-se necessário o uso de virtualização com heterogeneidade de arquiteturas, para simular ativos de TIC, de forma que um ativo de TIC em teste, não descubra um contexto de teste no qual está inserido ou que estejam sob avaliação. Além disso, ainda há um ambiente de monitoramento, ao qual deverá também estar conectado, para podermos identificar não-conformidades ou comportamentos suspeitos existente em ativos de TIC diante da grande variedade e complexidade existentes no mercado.

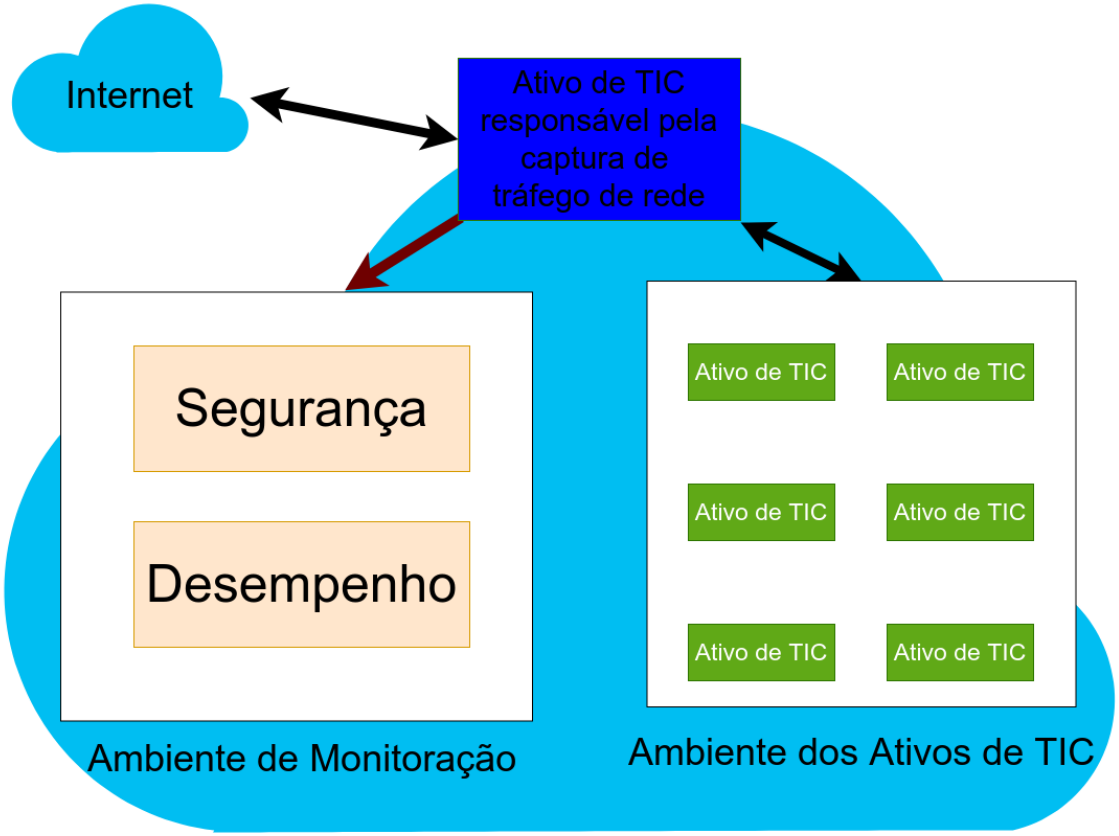

Figura 2 – Arquitetura conceitual

3.3- Algoritmo de detecção proposto

O **Algoritmo 1** apresenta os passos de detecção de tráfego malicioso utilizados pelo sistema proposto neste trabalho. A variável n representa o instante de medição dos dados; d é o número de dispositivos da rede; a variável booleana A_r recebe valor 0

quando não há alerta de segurança, e recebe 1 quando há alerta de segurança. Nesse caso, quando $A_r \leftarrow 1$, o sistema verifica o desempenho (ou o uso de recursos) da rede; a variável $L_r(d)$ define um limite inferior para que se identifique que há ataques, baseado no número de dispositivos conectados à rede, ou seja, quando há aumento no tráfego decorrente de tentativas de invasão, com consequente aumento da vazão da rede ou do número de conexões TCP. $L_r(d)$ é proporcional ao número de dispositivos conectados, pois se há um aumento do número de conexões na rede, aumenta-se o tráfego, o que faz aumentar o limite de detecção. Repare que a detecção se dá em duas etapas, uma referente à identificação de um alerta (com $A_r = 1$) e outra referente à verificação de um aumento do uso de recursos da rede, através da análise de D_r. Essas duas etapas servem para que o sistema de detecção seja robusto, de modo que não haja algum falso–positivo. Além disso, quando há a detecção, uma contramedida é realizada pelo sistema, que pode representar o acionamento de um *firewall* ou até mesmo o desligamento de um ativo de TIC.

Algoritmo 1 – Sistema de detecção de tráfego malicioso baseado em informações de segurança e desempenho da rede.

```
   input : n, d, A_r, L_r
 1 n ← 1;
 2 A_r ← 0;
 3 d ← número de dispositivos conectados na rede;
 4 SI_r[n] ← Dados de segurança da rede;
 5 D_r[n] ← Dados de desempenho da rede;
 6 while A_r = 0 do
 7     Leia SI_r[n];
 8     if SI_r[n] == 'alerta' then
 9         Leia D_r[n];
10         if D_r > L_r[d] then
11             A_r ← 1;
12             Acionamento de contra–medida;
13     n ← n + 1;
```

4- Metodologia Experimental

O presente capítulo apresentará as ferramentas integradas que utilizamos em nosso trabalho. Neste contexto descrevemos um ambiente para Teste de Caixa Preta em ativos de TIC. Apresentaremos os componentes e ferramentas usados para construir tal ambiente. Os detalhes de instalação e configuração são apresentados no Anexo A do presente documento. Esta apresentação foi dividida de acordo com seu uso e macro função que são: (i) Rede; (ii) Monitoração e; (iii) Virtualização.

4.1- Rede

Abaixo, descreveremos componentes e soluções para a infraestrutura de redes, relativos a conectividade e captura de tráfego de dados, que podem ser utilizados.

Para configuração do ambiente, algumas premissas básicas:

1. Conectividade com a Internet;
2. Plano de endereçamento IP interno;
3. Equipamento para prover conectividade com a Internet e o plano de endereçamento IP;
4. Servidor para instalação do Ambiente de Monitoração e;
5. Servidor para instalação do Ambiente de Virtualização.

A definição de plano de endereçamento IP ajudará na identificação dos ativos de TIC que estarão sujeitos a teste e dos que não estarão.

Ainda sobre endereçamento e identificação dos ativos de TIC, sugerimos, se for possível, que ocorra a identificação dos endereços de MAC de todos os ativos de TIC, pois, como poderemos utilizar uma faixa Dynamic Host Configuration Protocol (DHCP) para os ativos em teste, os mesmos poderão mudar de IP dentro desta faixa. Via de regra, os equipamentos que proveem conectividade têm esta informação.

4.1.1 Servidor DHCP

Para endereçamento automático dos ativos de TIC e elementos virtualizados, em IPv4 e IPv6, pode-se utilizar um servidor DHCP. Este pode associado a algum dos ativos de TIC que poderá prover a conectividade ou instalado em um servidor a parte. Tal utilização, sugerida, visa que os ativos de TIC interpretem que estão em operação e não em modo de teste [Droms, 1997].

4.1.2 Espelhamento de portas e *Sniffer*

O espelhamento de portas (*Port Mirroring* ou *Switched Port Analyzer* (SPAN)) é uma abordagem para monitoramento de tráfego de rede que envolve o envio de uma cópia de cada pacote de uma porta de origem do *switch* (ou VLAN) para uma porta de destino, fazendo o espelhamento do tráfego para análise [Orebaugh et al., 2006].

Este permite que ao verificar do desempenho do *switch* e tráfego de rede, coloque um analisador de rede, ou analisador de protocolo, na porta que está recebendo os dados espelhados. O analisador captura, examina e avalia os dados sem afetar a comutação de tráfego do cliente na porta de origem. Em nosso caso, quem receberá a analisará este tráfego é o IDS [Ornaghi and Valleri, 2005].

Um *sniffer* é um tipo de software utilizado para monitorar e analisar o tráfego de rede para detectar problemas e manter um fluxo eficiente. Eles capturam todo tráfego de rede que passa por eles, inclusive senhas e nomes de usuários não criptografados. Um *sniffer* pode ser instalado em qualquer computador conectado a uma rede local. Ele não precisa ser instalado no próprio aparelho que se deseja monitorar. Em outras palavras, ele pode permanecer oculto durante a conexão [Orebaugh et al., 2006].

Em nosso trabalho selecionamos Ettercap para função de *sniffer*. Sua seleção foi baseada nas suas características, onde verificamos a sua capacidade de interceptar o tráfego de rede, além possuir um conjunto de ferramentas para ataques do tipo ”*man-in-the-middle*”e ”*denial of service*”, também pode observar conexões ativas e filtrar conteúdo de conexões ativas. Possui suporte a diversos protocolos de rede para análise ativa e

passiva. Em uma das formas de operação, o Ettercap funciona colocando a interface de rede em modo promíscuo e por envenenamento ARP nas máquinas alvo. Assim, ele pode atuar como um ”*man-in-the-middle*”.

O Espelhamento de portas e o *sniffer* são duas formas que podemos utilizar para a captura de tráfego da rede para análise. Ressaltamos que a forma mais ’elegante’ para captura de tráfego é o espelhamento de portas, entretanto, é necessário que o ativo de TIC, para conectividade de rede, possua tal capacidade. Assim, caso não haja disponibilidade de um equipamento com tal característica, nossa sugestão passa a ser o uso de *sniffer*.

Ao se utilizar o espelhamento de portas devemos ter duas interfaces físicas de rede. Uma por onde passará todo o tráfego de rede (entrada e saída) e outra que receberá uma cópia do tráfego.

Ettercap

O pacote Ettercap é instalado a partir de um *script* (Seção A.1.2), mas, também não é habilitado para iniciar automaticamente. Este pacote ajudará a capturar o tráfego de rede caso seja utilizada a técnica *sniffing*. Além disso, este pacote deverá ser executado no mesmo servidor que estará o Suricata IDS e o EveBox.

4.2- Monitoração

O Ambiente de Monitoração deve ser instalado em um Servidor com sistema Operacional Linux. Utilizamos como base a versão de 16 do Ubuntu e também a versão de CentOS 7. Lembramos que uma vez que todos os pacotes estejam instalados é necessário que sejam verificados os estados dos processos que devem estar em execução.

4.2.1 Desempenho

Para a medição de desempenho existem diversas ferramentas. Algumas, por exemplo, utilizando-se SNMP, como CACTI[1], NAGIOS[2] e LibreNMS[3]. Entretanto, tais ferramentas tradicionalmente, utilizam medições de 5 em 5 minutos. Nossas indicações são por ferramentas que possibilitam manipular tal medição, ou melhor, por ferramentas em que a granularidade dos dados seja menor. Conforme Glavic and Dittrich [2007], a granularidade dos dados e o seu nível de detalhe são conceitos inversamente proporcionais. Assim, quanto maior for a granularidade, menor o nível de detalhe e quanto menor a granularidade, maior o nível de detalhe. Assim, as ferramentas propostas seguem abaixo.

Ferramentas de coleta e armazenamento

O Telegraf e o Influxdb são desenvolvidos pela empresa InfluxDATA. Sua plataforma é composta por outros dois projetos que são o Chronograf e o Kapacitor. Estes são chamados coletivamente de ecossistema *TICK*. São projetos de código aberto. Em nosso trabalho utilizaremos apenas os componentes Telegraf e o Influxdb.

O Telegraf é um agente baseado em servidores para coletar métricas e dados. Possui integrações para obter uma variedade de métricas, eventos e registros diretamente dos contêineres e sistemas ao qual está em execução. Ele também possui plug-ins de saída para enviar métricas para uma variedade serviços e filas de mensagens, incluindo InfluxDB, Graphite, OpenTSDB, Datadog, Librato, Kafka, MQTT e muitos outros.

O InfluxDB é um banco de dados de séries temporais de código aberto e sem esquema. Ele está escrito na linguagem de programação Go e é otimizado para manipular dados de séries temporais. Possui uma linguagem de consulta semelhante a SQL [Naqvi et al., 2017].

Os pacotes Influxdb e Telegraf são instalados a partir do *script* descrito na seção A.1.2, mas não são habilitados para iniciar automaticamente, pois, os mesmos devem

[1]Sistema de monitoração de desempenho de rede. - https://www.cacti.net/
[2]Sistema de monitoração de falhas de rede https://www.nagios.org/
[3]Sistema de monitoração de falhas e desempenho de rede - https://www.librenms.org/

ser previamente configurados. Estes pacotes ajudarão a apresentar as informações de Desempenho.

Ferramentas de visualização

O componente Grafana, que mostrará os dados coletados via Telegraf e Influxdb, é uma plataforma aberta para análises e monitoramento de dados. Com o Grafana é possível consultar, visualizar, alertar e entender suas métricas, independentemente de onde elas estejam armazenadas. Possui uma vasta documentação e disponibilidade de painéis de controle prontos, para visualização dos dados coletados via Telegraf e Influxdb e podem ser consultados em https://grafana.com/dashboards/ [Grafana, 2018].

Durante a instalação do Grafana, o mesmo é habilitado automaticamente. Também é instalado a partir do *script* na seção A.1.2. Este pacote ajudará a apresentar as informações de Desempenho.

4.2.2 Segurança da Informação

Em nosso trabalho, utilizaremos Suricata [(OISF), 2018], pois, o mesmo acumula as funções de IDS, IPS e NSM. É um sistema de intrusão de rede, que pode ser usado para inspecionar o tráfego da rede usando uma linguagem de regras e assinaturas. O Suricata é utilizado para detecção de invasão de rede, prevenção de invasão de rede e prevenção de monitoramento de segurança. O sistema é capaz de lidar com tráfego alto, exibi-lo na tela e também enviar alertas por e-mail; suporta *Multi-Threading*, para que se possa usar mais de uma CPU por vez; fornece aceleração de hardware incorporada, para que se possa usar placas gráficas para inspecionar o tráfego da rede; suporte a scripts Lua que podem ser usados para detectar ameaças complexas; suporte vários sistemas operacionais, como Unix, Linux, FreeBSD e Windows.

O pacote Suricata IDS é instalado a partir do *script* descrito na seção A.1.2, mas, não é habilitado para iniciar automaticamente. Este pacote ajudará a coletar informações

de Segurança.

Ferramentas de visualização

As informações coletadas pela detecção de intrusão, deverão ser indexadas. Apresentaremos o EveBox, que permite a visualização destes dados.

O EveBox é uma ferramenta de gerenciamento de eventos e alertas baseada na web para eventos gerados pelo mecanismo de detecção de ameaças da rede Suricata. É possível configurar a exportação de dados para buscas simples com o banco de dados SQLite, PostgreSQL ou ainda indexação via Elasticsearch. As duas abordagens são válidas, pois, não exigem muito conhecimento de configuração de ambas. Sem o Elasticsearch, a visualização de dados se torna mais rápida, e sem alguns relatórios, assim como a possibilidade de exportação dos mesmos. Nesta característica, utilizando-se os bancos de dados SQLite ou PostgreSQL, a função relatórios está desabilitada [Ish, 2017].

Ao executar o EveBox, este lê o arquivo eve.json, que é gerado pelo Suricata, e grava estes dados em um banco de dados para depois os apresentar. Além disso, esta ferramenta, pode ser executada em dois modos, onde o processo *Evebox_import* tem a função de importar alertas de ameaças e indexá-los no banco de dados do Elasticsearch. Estas ameaças podem ser filtradas, exibidas em diferentes gráficos, arquivadas, excluídas, identificadas como específicas ou analisadas para obter informações detalhadas [Uramová et al., 2018].

Dependendo do volume de dados gerados a partir do Suricata, as bases de dados do EveBox e do Elasticsearch podem ficar muito grandes, além do alto consumo de CPU e memória. Assim utilizaremos apenas o EveBox, com banco de dados PostgreSQL, sem a integração com o Elasticsearch.

O pacote EveBox é instalado a partir do *script* descrito na seção A.1.2, mas, também não é habilitado para iniciar automaticamente. Este pacote ajudará a mostrar as informações de Segurança coletadas.

4.3- Virtualização

Todas as máquinas virtuais são gerenciadas e controladas por um gerenciador de máquinas virtuais. Estes são geralmente denominados como VMM ou *hypervisor*, e seu foco principal é fornecer abstração ao hardware subjacente. O sistema no qual o VMM é instalado ou executado é denominado como máquina *host* e todas as outras máquinas virtuais em execução dentro da máquina *host* são denominadas *guest*. O *host* e o *guest* usam quase a mesma interface para usar os diferentes aplicativos. A máquina *host* e todas as máquinas *guest* disponíveis em execução na máquina *host* são independentes umas das outras [Yadav et al., 2019].

Existem vários *hypervisors* no mercado atualmente e também podem ser divididos em duas categorias: “Tipo 1” e “Tipo 2”.

Os *hypervisors* “Tipo 1” são executados na parte superior do hardware do sistema e, por isso, são denominados como máquina virtual nativa. De outra forma, podemos dizer que esse *hypervisor* “Tipo 1” toma o lugar do sistema operacional e pode acessar diretamente o hardware disponível para seu uso. Os *hypervisors* “Tipo 1” têm um favor que se qualquer máquina virtual falhar ou não responder por qualquer motivo, outro sistema operacional convidado não será afetado. O “Tipo 1” é executado no modo kernel e, por causa disso, possui CPUs físicas exclusivas. Exemplos de *hypervisors* “Tipo 1” são o Microsoft Hyper-V, o VMware ESXi Server, o Citrix / Xen Server, KVM, Proxmox etc. Por outro lado, os *hypervisors* “Tipo 2” são executados no sistema operacional instalado no *hypervisor* ou, em outras palavras, podemos dizer que são qualquer outro software aplicativo que seja executado no sistema operacional. Os *hypervisors* de “Tipo 2” também são conhecidos como máquinas virtuais hospedadas. Exemplos de *hypervisors* “Tipo 2” são o Microsoft Virtual PC, o VMware Workstation e o Oracle VirtualBox, QEMU etc.

Para configuração do ambiente de virtualização seguem algumas premissas básicas para o ambiente:

1. Conectividade com a Internet já implementada
2. Servidor instalado do Ambiente de Monitoração
3. Servidor para instalação do Ambiente de Virtualização

O Ambiente de Virtualização deve ser instalado em um Servidor com sistema operacional Linux ou Windows. Cada um destes ambientes pode ser utilizado com uma finalidade de virtualização. Para que o ativo de TIC em teste não detecte que está em inspeção ou averiguação, em nosso caso, utilizamos cada um destes ambientes com os seguintes propósitos:

1. Virtualização de dispositivos para simulação de uma rede real, ou seja, o ativo de TIC pode verificar sua vizinhança e dado que há uma quantidade de outros ativos de TIC, o mesmo não se considera em testes.

2. Virtualização dispositivos em volta para simular acesso ao ativo de TIC e para ambiente de monitoração verificar o tráfego de rede;

3. Virtualização do ativo de TIC em teste, se for possível.

Abaixo seguem algumas sugestões de *hypervisors*.

4.3.1 Oracle VirtualBox

O Oracle VirtualBox é um software de virtualização desenvolvido pela empresa Innotek depois comprado pela Sun Microsystems que posteriormente foi comprada pela Oracle Corporation. Este visa criar ambientes virtualizados para instalação de sistemas distintos.

O Oracle VirtualBox possui uma arquitetura modular com interfaces de programação interna bem definidas e um desenho cliente/servidor. Isso torna fácil o controle de diversas interfaces de uma só vez. Uma de suas características limitantes é o suporte apenas as arquiteturas x86 e x86-64 tanto para sistemas convidados quanto para sistemas hospedados [Oracle, 2018].

O Oracle VirtualBox é voltado para virtualização de plataforma x86 e AMD64/Intel64 e pode ser instalado em diversos sistemas operacionais. A lista dos sistemas operacionais suportados e o download pode ser feito em *https://www.virtualbox.org/wiki/Downloads*.

Para instalar o Oracle VirtualBox, no Linux, deve-se saber, previamente, se sua distribuição é suportada. Seguem alguns exemplos para sistemas baseados em paco-

tes DEB e RPM. Sugerimos distribuição Linux baseada em RPM o CentOS 7 e como distribuição Linux baseada em DEB o Ubuntu 16.

4.3.2 KVM e QEMU

O KVM e o QEMU funcionam juntos para virtualização em sistema Linux. O KVM é uma tecnologia de virtualização de código aberto baseada no Linux. Especificamente, com o KVM, há a possibilidade de se transformar o Linux em um *hypervisor*, permitindo que uma máquina *host* execute vários ambientes virtuais isolados, chamados máquinas *guest* ou máquinas virtuais.

O KVM é parte do Linux desde kernel 2.6.20 e nas versões mais recentes. Esta foi anunciada pela primeira vez em 2006 e, um ano depois, inserida à versão de kernel do Linux da linha principal. Como a KVM é parte do código do Linux atual, esta aproveita imediatamente todos os recursos, correções e avanços novos do Linux sem engenharia adicional [Goto, 2011].

O QEMU é um software emulador de máquina: ele pode executar um sistema operacional de destino não modificado (como Windows ou Linux) e seus aplicativos em uma máquina virtual. O próprio QEMU é executado em vários sistemas operacionais hospedados, como Linux, Windows e Mac OS X. As CPUs hospedadas e as convidadas podem ser diferentes [team, 2018]. Seu principal uso é hospedar um sistema operacional convidado, como o Windows no Linux ou o Linux no Windows. Outro uso é a depuração, porque a máquina virtual pode ser facilmente interrompida e seu estado pode ser inspecionado, salvo e restaurado. Além disso, dispositivos embarcados específicos podem ser simulados adicionando novas descrições de máquina e novos dispositivos emulados.

Conforme Bellard [2005], o mesmo é integrado a um emulador de modo de usuário específico do Linux. É um subconjunto do emulador de máquina que executa processos do Linux para uma CPU de destino em outra CPU. Ele é usado principalmente para testar o resultado de compiladores cruzados ou para testar o emulador de CPU sem precisar iniciar uma máquina virtual completa. Algumas de suas características são: (i) Emulador de CPU; (ii) Emulador de dispositivos; (iii) Dispositivos genéricos; (iv) *Debugger*; (v) Interface gráfica; (vi) Diversas arquiteturas como x86, x86-64, mips, arm, entre outras.

Para o KVM e QEMU existem diversas formas de se criar máquinas virtuais. Em nosso exemplo, uma das necessidades era iniciar várias máquinas virtuais dentro da rede de testes dos ativos de TIC. Para esta solução devemos utilizar o conceito de *bridge* (ponte).

5- Resultados Experimentais

Este capítulo apresenta os resultados, extraídos de ambientes previamente configurados com as ferramentas anteriormente apresentadas, como forma de validar nosso trabalho de testes de caixa preta em ativos de TIC. Nem todos os resultados de nosso trabalho serão mostrados, pois, devido ao volume em relação quantidade de eventos detectados nas coletas de segurança da informação, apenas os resultados mais relevantes serão expostos.

Os ambientes seguiram a arquitetura definida na figura 2. A figura 3 apresenta um exemplo de implementação de um dos ambientes que extraímos informações. Destes ambientes, extraímos análises de ativos de TIC. Estes apresentaram comportamentos suspeitos, ou seja, alinhados com a não conformidade.

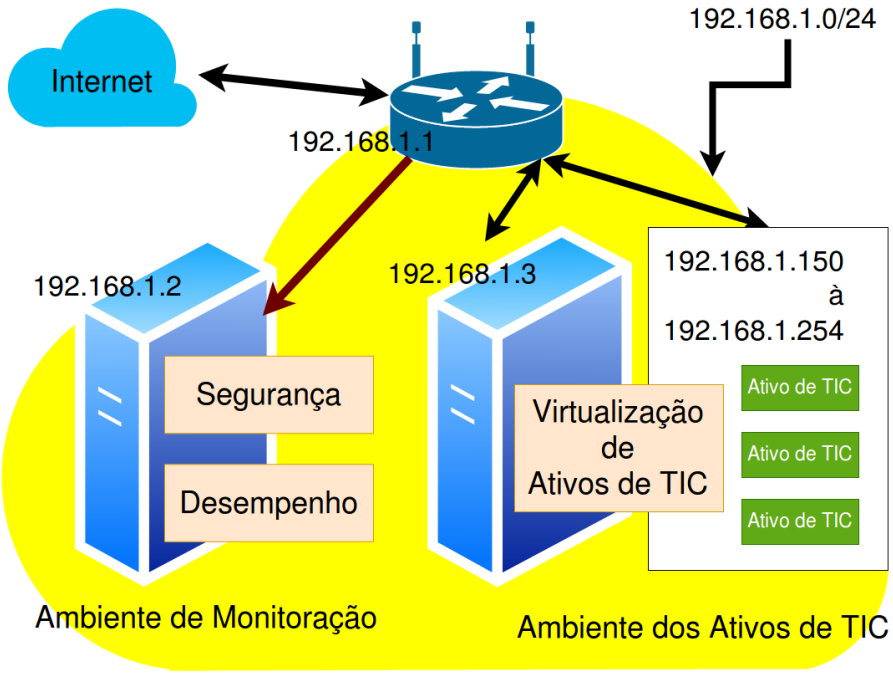

Figura 3 – Arquitetura com elementos instanciados

Para validar o funcionamento da proposta deste trabalho, dois cenários foram construídos: (i) Escritório em casa; e (ii) Internet. Cada um caracteriza-se por uma topologia e ativos de TIC distintos. Para o cenário (i) temos: um Roteador Modem Wifi, três *Smartphones*, um servidor, um desktop, uma Impressora Multifuncional, um extensor

de rede, uma *SmartTV*, um decodificador de TV a Cabo e um *firewall*. Este ambiente é uma rede local. O servidor possui o Ambiente de Monitoração e uma placa de rede, onde a esta provê conectividade e recebe todo o tráfego de rede para análise.

No cenário (ii) temos: um *switch*, uma câmera IP, um servidor para monitoração, um servidor para *honeypot* e não há um *firewall*. Este ambiente está conectado diretamente na Internet. Há um servidor com o Ambiente de Monitoração instalado e possui duas placas de rede. Uma provê conectividade ao ambiente e a outra placa de rede recebe todo o tráfego de rede para análise. Isto é feito via *Port Mirroring* no *switch* para o servidor do Ambiente de Monitoração.

Os resultados foram obtidos a partir dos experimentos relativos aos cenários descritos. Todas as evidências são efetuadas por consultas via EveBox, Grafana ou gráficos com as informações extraídas dos bancos de dados InfluxDB (Grafana) e PostgreSQL (EveBox). Estão disponíveis em *https://github.com/carlos-teles/etsg-results*, em uma base de dados construída a partir do monitoramento da rede. Entretanto, não é possível expor todas as evidências, pois, seu volume supera 500GB e o site Github.com não permite tal volume. Assim, a partir de então, descrevemos as mais significativas.

5.1- Cenário i - Software de comunicação com banco

A partir de consultas aos eventos gerados pelo Suricata e publicados pelo EveBox, identificamos acessos indevidos de uma aplicação, que apesar de instalada, não era mais utilizada há pelo menos 01 ano. Entretanto identificamos a aplicação através de um evento do tipo DNS para IPv4 e IPv6. Esta é a chamada 'guardiao.itau.com.br', conforme as figuras 4 e 5.

Continuando as consultas aos eventos gerados pelo Suricata e publicados pelo EveBox, identificamos mais acessos indevidos de uma aplicação. Identificamos a aplicação através de um evento do tipo TLS também para o domínio guardiao.itau.com.br, conforme a figura 6.

Confirmamos tais eventos acima, também, a partir do gráfico das conexões da pilha TCP na figura 7, pela aplicação DNS pelo gráfico de datagramas do protocolo UDP na figura 8 e também do tráfego de rede a partir das informações na figura 9.

Figura 4 – Análise de Segurança da Informação - cenário i - guardiao - evento DNS em IPv4

Figura 5 – Análise de Segurança da Informação - cenário i - guardiao - evento DNS em IPv6

Figura 6 – Análise de Segurança da Informação - cenário i - guardiao - evento TLS

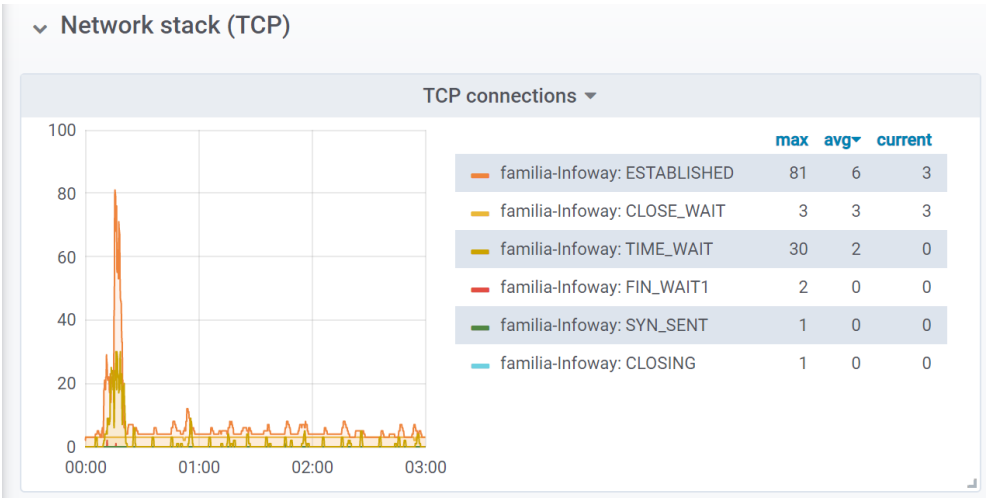

Figura 7 – Análise de desempenho - cenário i - guardião - Pilha TCP

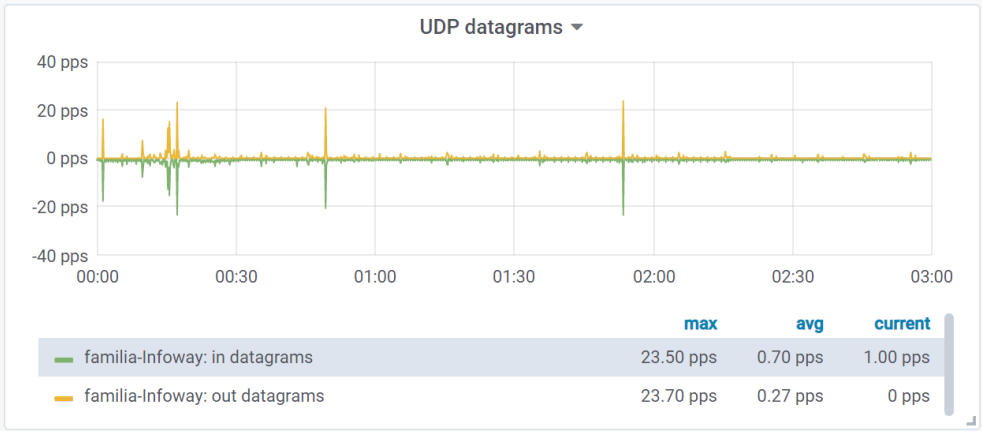

Figura 8 – Análise de desempenho - cenário i - guardião - datagrama UDP

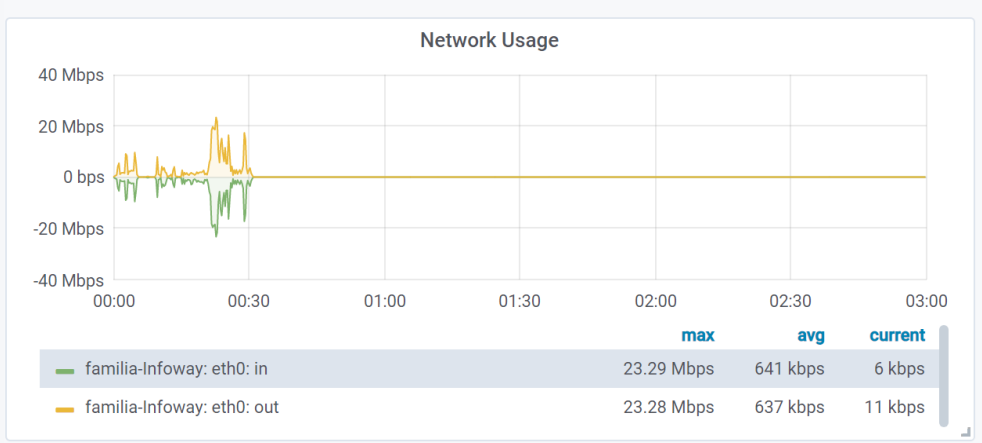

Figura 9 – Análise de desempenho - cenário i - guardião - tráfego de rede

5.2- Cenário i - Decodificador de TV a cabo

No cenário i, identificamos mais um ativo de TIC com comportamento suspeito. Este foi um decodificador de TV a cabo. Nas consultas aos eventos de segurança da informação gerados pelo Suricata e publicados pelo EveBox obtivemos eventos pela aplicação DNS e HTTP, conforme as figuras 10 e 11. Neste momento, o decodificador de TV a cabo estava em modo *stand by*, ou seja, não estava em operação. Assim, no período identificado, houve acesso à Internet sem que o decodificador de TV a cabo estivesse efetivamente ligado.

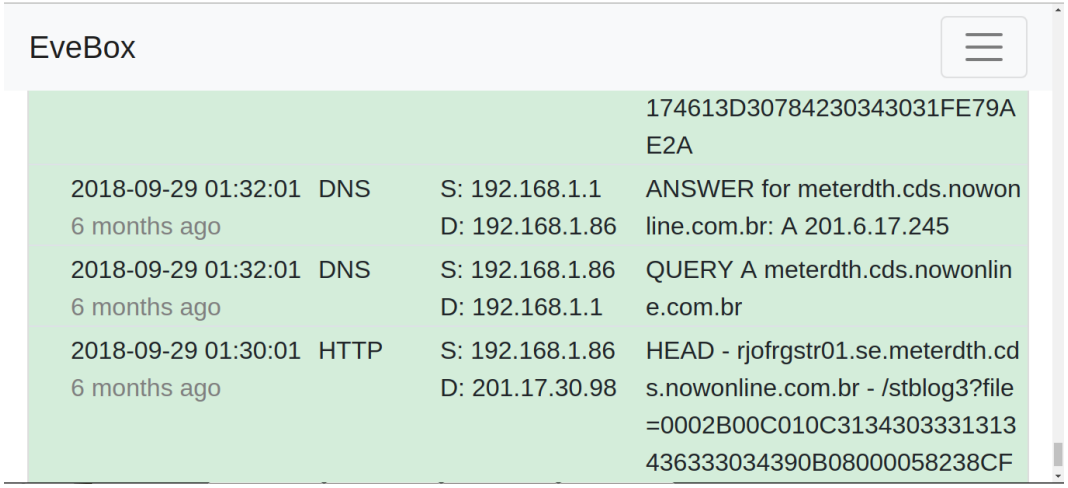

Figura 10 – Análise de Segurança da Informação - cenário i - decodificador de TV - evento HTTP e DNS - 1

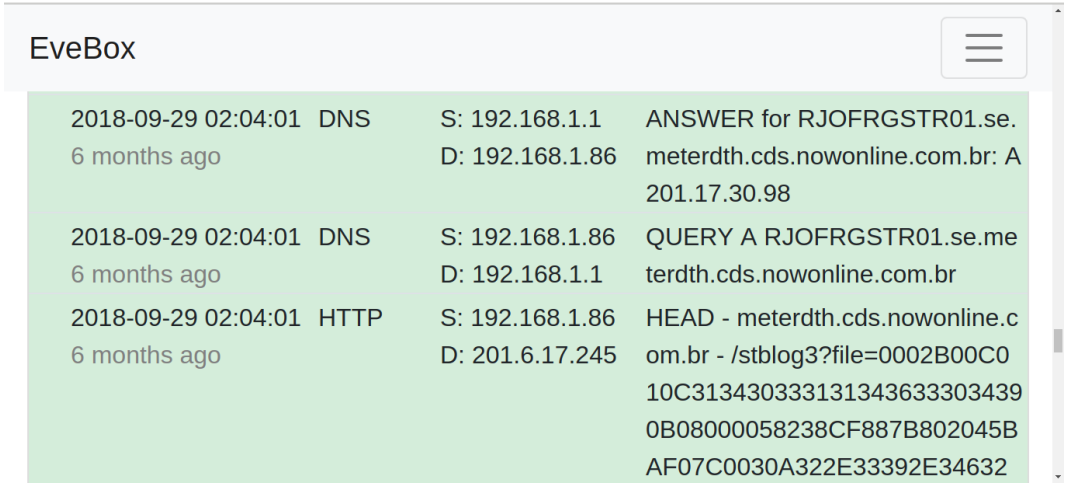

Figura 11 – Análise de Segurança da Informação - cenário i - decodificador de TV - evento HTTP e DNS - 2

Confirmamos tal tráfego de rede a partir das informações nas figuras 12 e 13, onde observamos, tanto tráfego UDP e TCP, respectivamente, além da utilização da rede na figura 14.

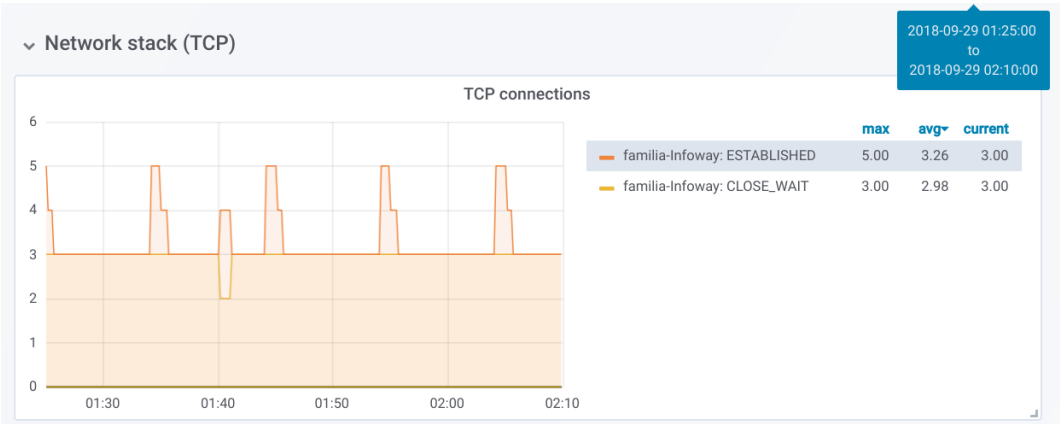

Figura 12 – Análise de desempenho - cenário i - decodificador de TV - Pilha TCP

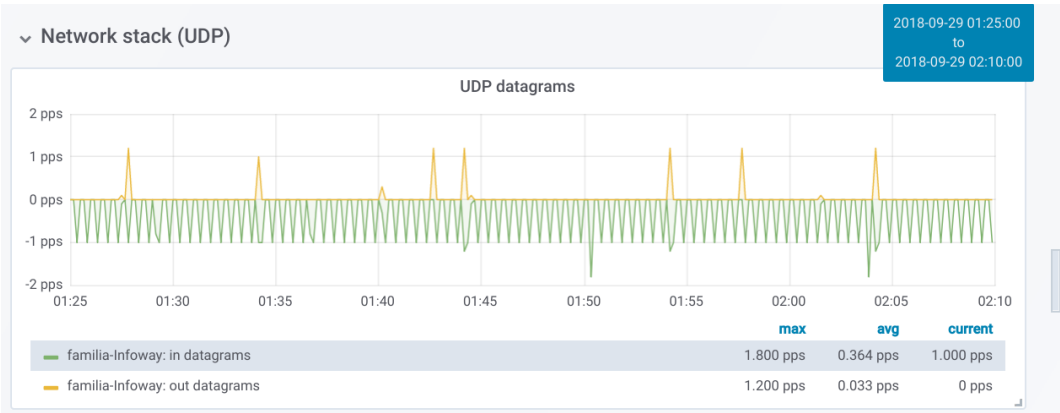

Figura 13 – Análise de desempenho - cenário i - decodificador de TV - datagrama UDP

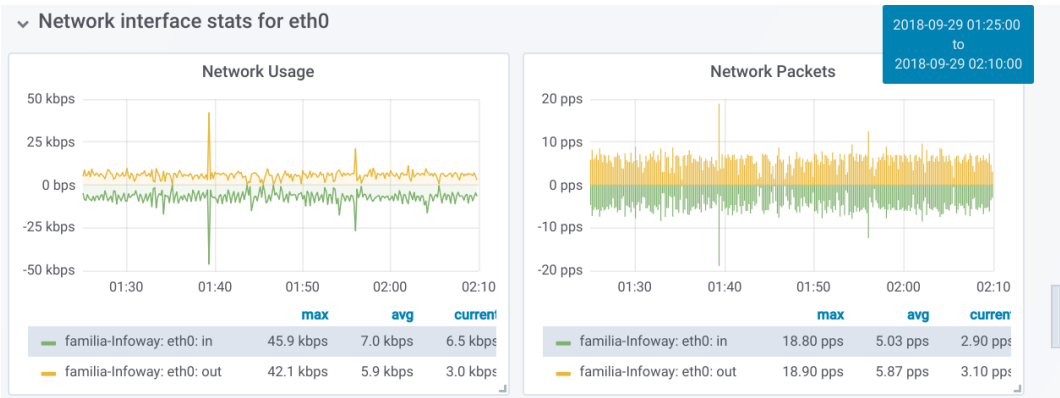

Figura 14 – Análise de desempenho - cenário i - decodificador de TV - tráfego de rede e pacotes

5.3- Cenário i - Análise consolidada

A partir das informações coletadas no cenário i extraímos, da monitoração de desempenho, as médias diárias do pacotes na rede, entrada de pacotes na Figura 15 e saída de pacotes Figura 16. Extraímos, da monitoração de segurança da informação, o total por dia, de alertas gerados (Figura 17) e de eventos DNS (Figura 18). Verificamos que os picos dos dias 16/09/2018 apresentados na seção 5.1 e do dia 29/09/2018 e na seção 5.2 coincidem com os das figuras 15, 16, 17, 18, 19 e colaboram com a validação da solução. Em tempo, lembramos que na seção 5.2 tivemos eventos do tipo HTTP.

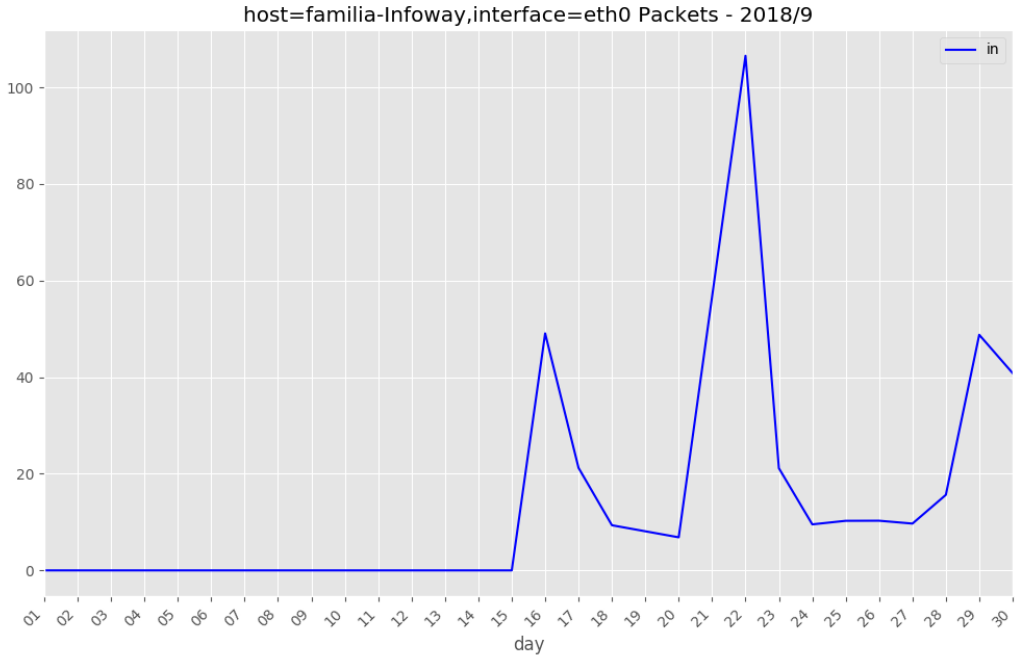

Figura 15 – Quantidade de pacotes de ENTRADA - cenário i - média por dia

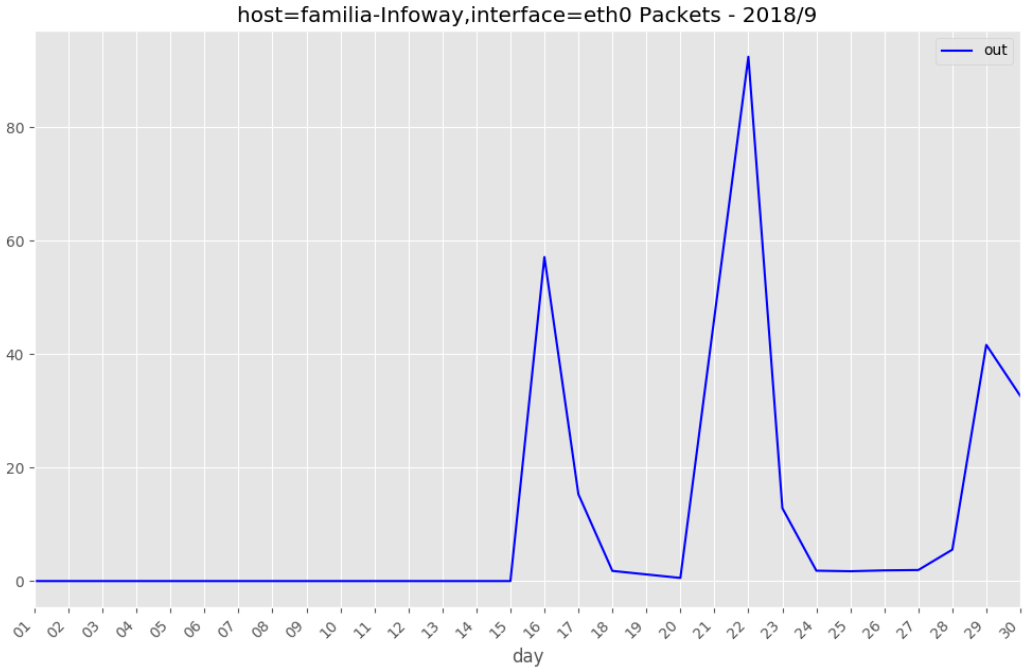

Figura 16 – Quantidade de pacotes de ENTRADA - cenário i - média por dia

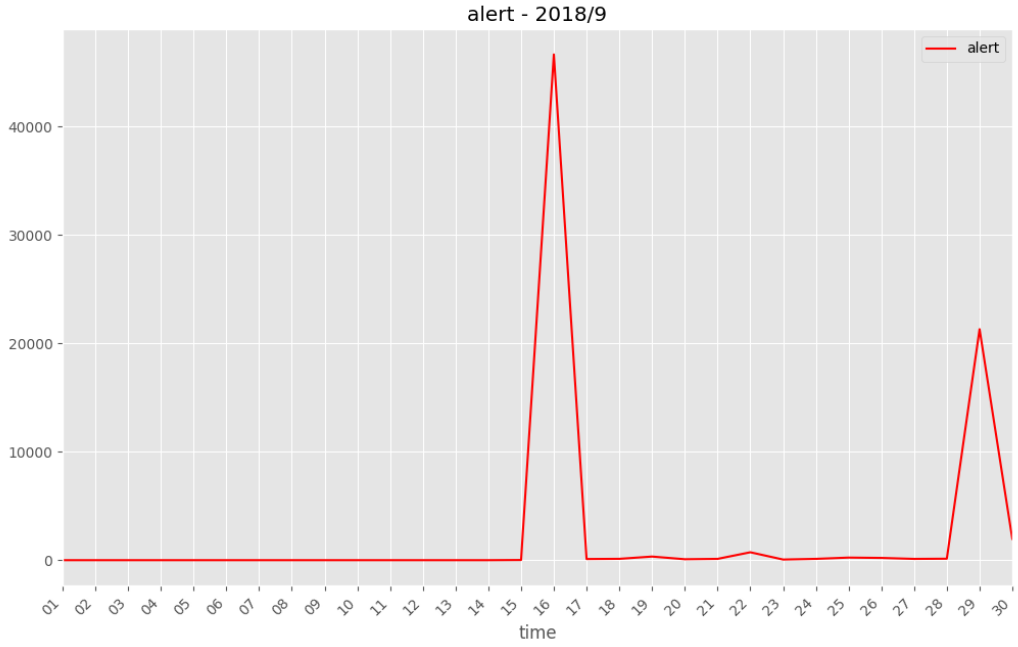

Figura 17 – Quantidade de *Alerts* - Cenário i - total por dia

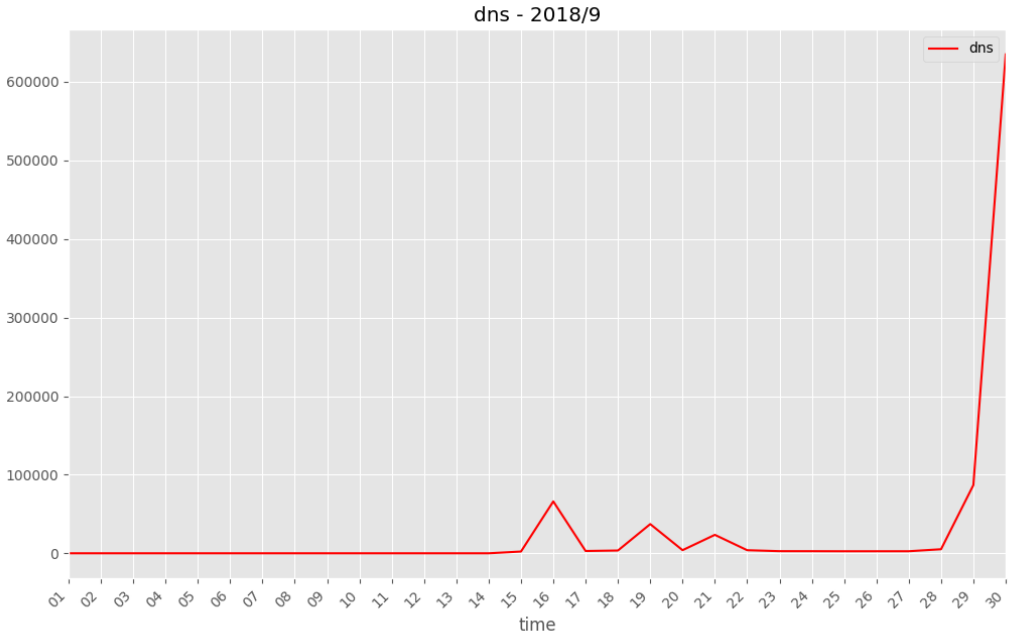

Figura 18 – Quantidade de eventos de DNS - Cenário i - total por dia

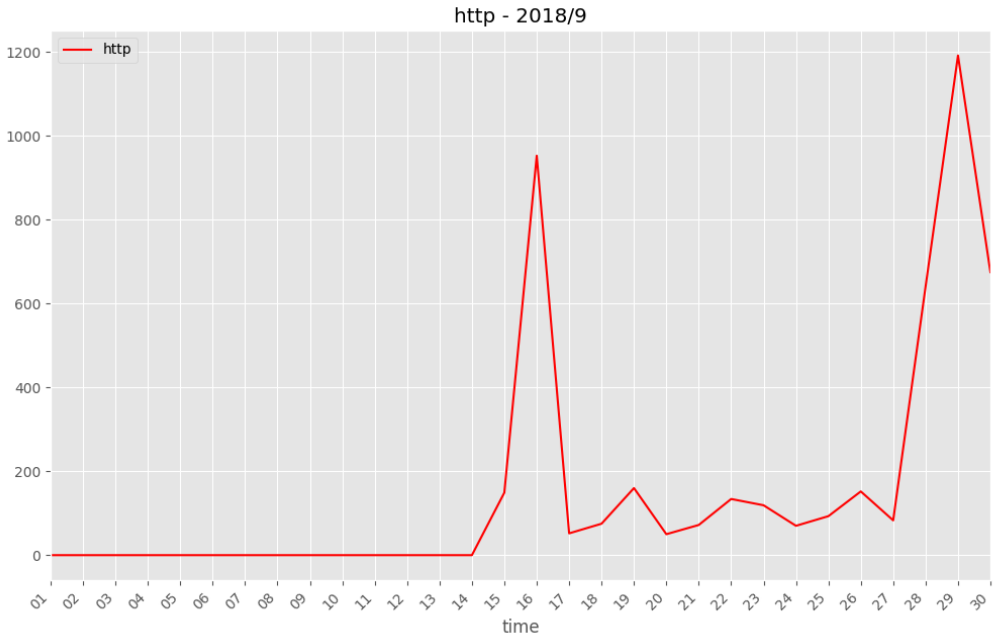

Figura 19 – Quantidade de eventos de HTTP - Cenário i - total por dia

5.4- Cenário ii - Câmera IP

Em um dos ambientes instalados tínhamos uma câmera IP sendo monitorada. Esta possuía o IPv4 configurado ’201.57.200.161’, em um ambiente com conectividade a Internet. Nossas buscas se iniciam no ambiente EveBox, onde foram publicados os dados do Suricata. Estas se basearam no filtro do disponibilizado pelo ambiente EveBox, onde é possível informar o IP de destino ou IP de origem, ou seja, quem estaria recebendo ou enviando a informação pela Internet.

De acordo com as figuras 20 e 21 em consultas no EveBox a eventos com IP de origem ’201.57.200.161’, a câmera IP efetuou consultas de DNS ao IP ’8.8.8.8’, conhecido como DNS público do Google [Huang et al., 2011], e obteve retorno. Isto é o indício de uma troca de dados pela Internet que não foi informada em momento algum pelo fabricante.

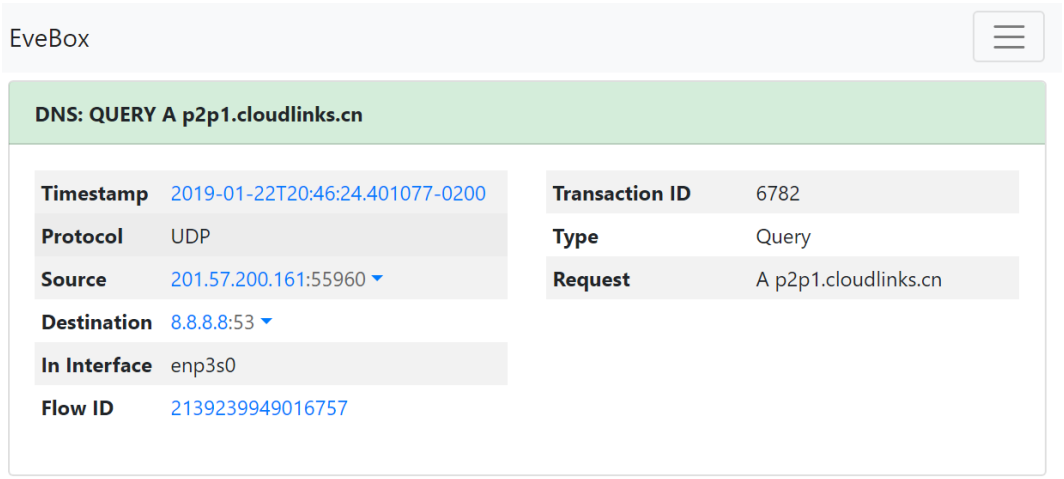

Figura 20 – Análise de Segurança da Informação - cenário ii - Câmera IP - evento DNS em IPv4 - 01

As figuras 20 e 21 também apresentam o detalhamento pelo EveBox, onde se apresenta o *Source* (IP de Origem) da câmera IP ’201.57.200.161’ e o *Destination* (Ip de Destino) o DNS público do Google ’8.8.8.8’. Pela consulta DNS apresentada, aparentemente, há uma sequência de consultas aos servidores do fabricante. Esta sequência foi confirmada com uma consulta no banco de dados onde extraímos os seguintes nomes sendo consultados por DNS: ”p2p1.cloudlinks.cn”, ”p2p2.cloudlinks.cn”, ”p2p3.cloudlinks.cn”, ”p2p4.cloudlinks.cn”, ”p2p5.cloudlinks.cn”, ”p2p6.cloudlinks.cn”, ”p2p7.cloudlinks.cn”,

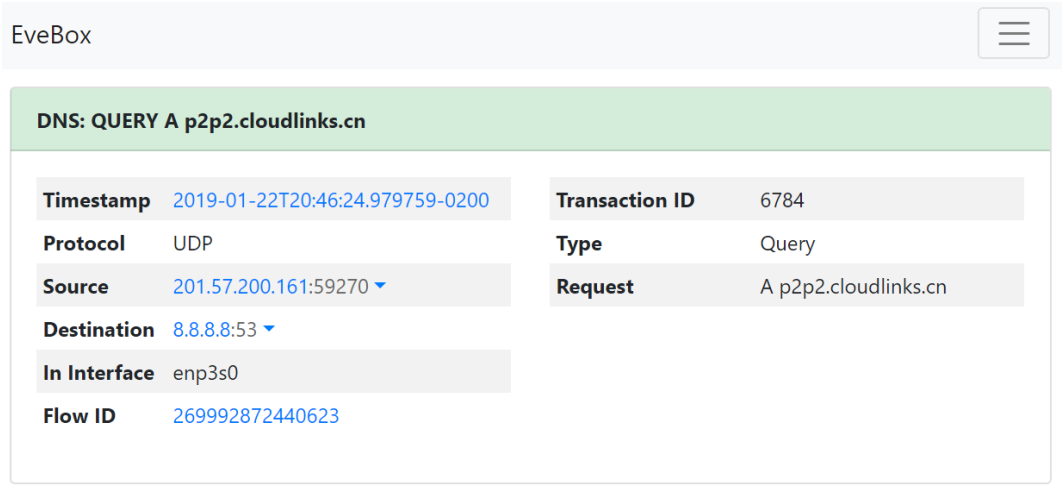

Figura 21 – Análise de Segurança da Informação - cenário ii - Câmera IP - evento DNS em IPv4 - 02

”p2p8.cloudlinks.cn”, ”p2p9.cloudlinks.cn”e ”p2p10.cloudlinks.cn”

Sobre a análise de desempenho, a mesma vem a comprovar que ocorreram trocas de informações da aplicação DNS, que utiliza protocolo User Datagram Protocol (UDP) da pilha TCP/IP para o transporte da informação. Assim, as figuras 22 e 23 apresentam, respectivamente, o transporte de pacotes da TCP e UDP. A figura 24 apresenta o tráfego de rede (*Network Usage*) e a figura 25 o tráfego de pacotes (*Network Packets*). Em todos os gráficos as informações são globais, ou seja, tínhamos outros ativos de TIC sob monitoração, mas é importante confirmar tal informação para a câmera, mesmo com as informações de todos.

Em tempo, as figuras 24 e 25, respectivamente, são exibidas apenas os dados de entrada, tráfego de rede e pacotes, pois, neste ambiente, a interface de rede enp3s0 recebe todo o tráfego via espelhamento de portas (SPAN). Assim, o tráfego que aparece é todo como de entrada.

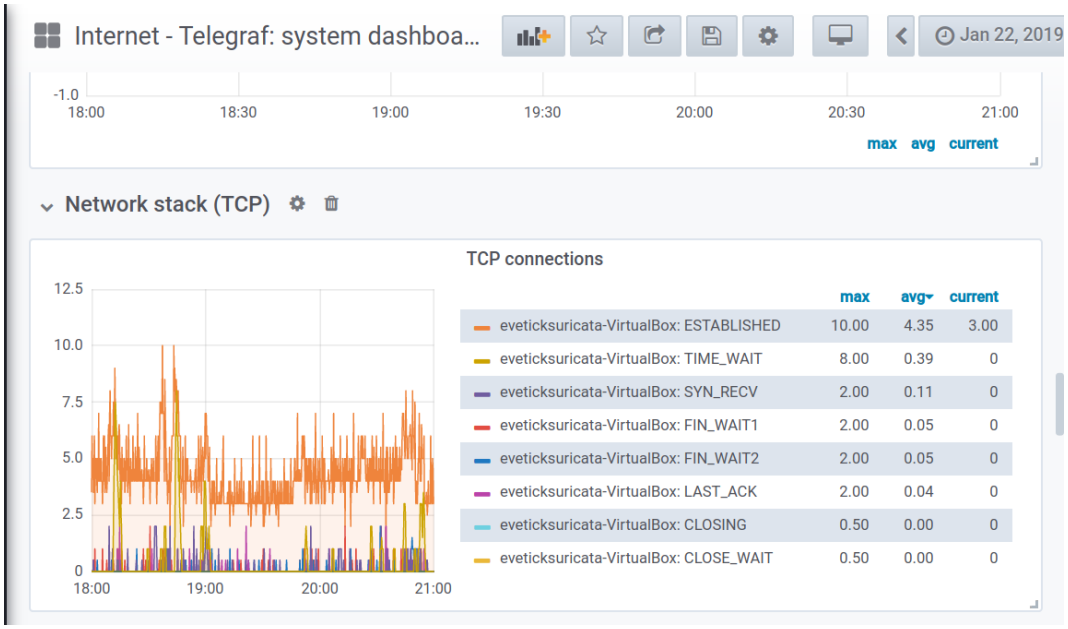

Figura 22 – Análise de desempenho - cenário ii - Câmera IP - Pilha TCP

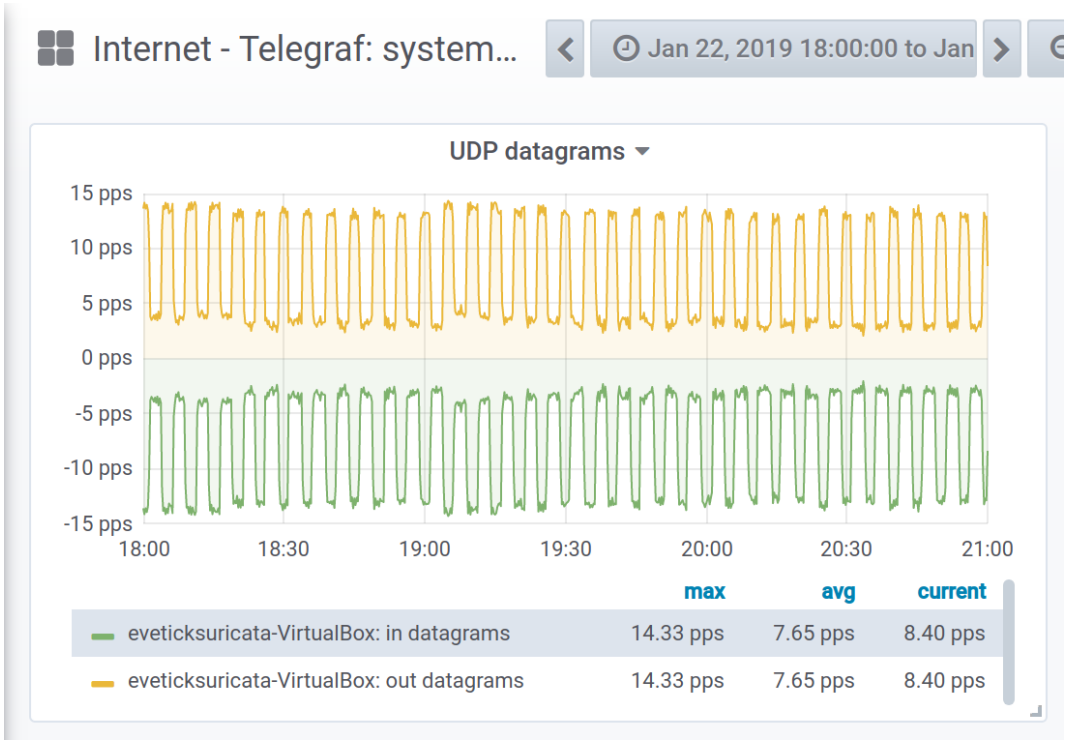

Figura 23 – Análise de desempenho - cenário ii - Câmera IP - datagrama UDP

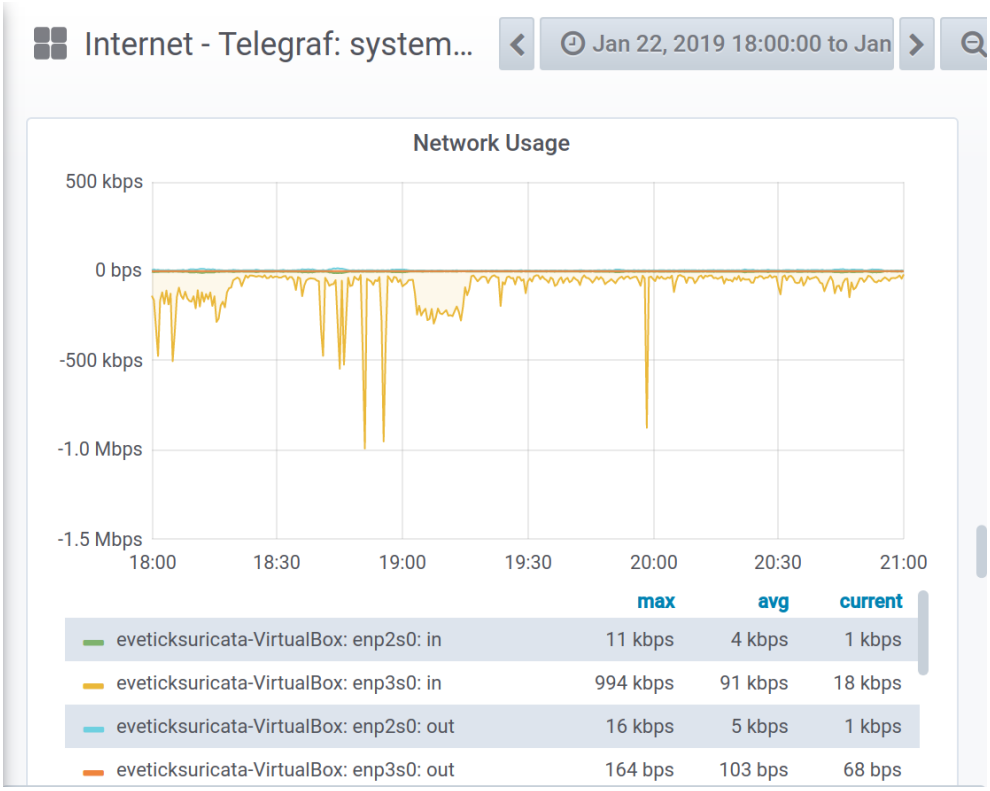

Figura 24 – Análise de desempenho - cenário ii - Câmera IP - tráfego de rede

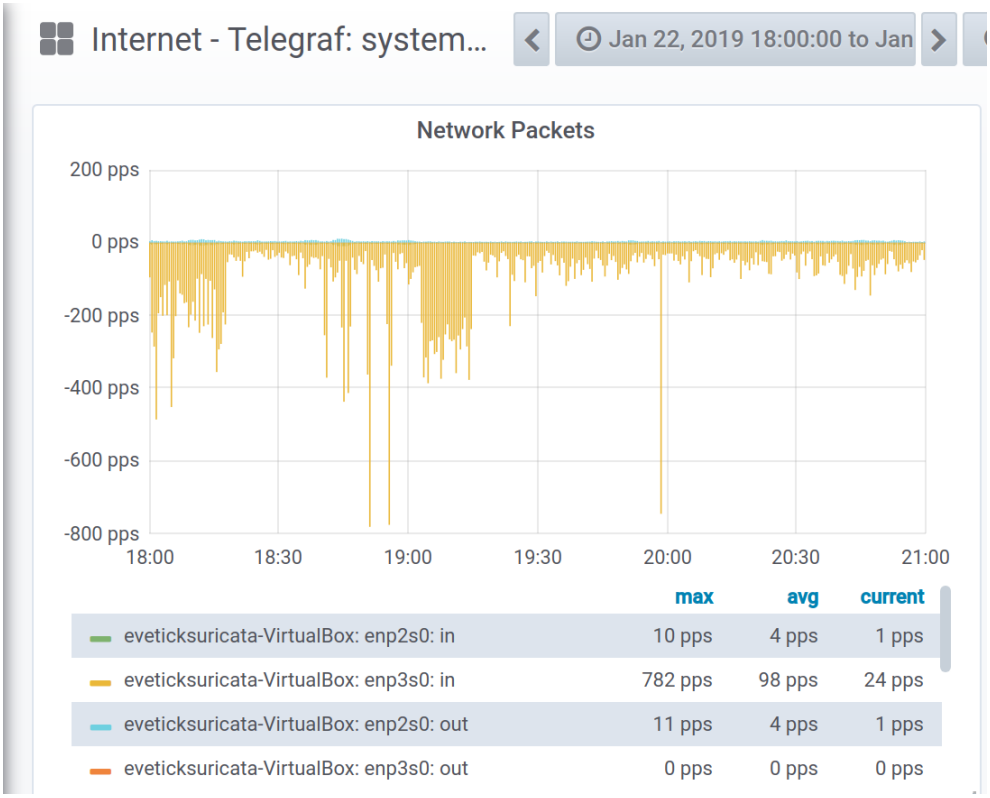

Figura 25 – Análise de desempenho - cenário ii - Câmera IP - tráfego de pacotes

5.5- Cenário ii - Análise consolidada

A partir das informações coletadas no cenário ii extraímos, da monitoração de desempenho, as médias diárias as médias diárias de tráfego de rede de entrada na Figura 26 e de pacotes de rede de entrada na Figura 27. Neste caso, em específico, apenas apresentaremos os dados de entrada, pois, a interface de rede enp3s0 recebe todo o tráfego via espelhamento de portas (SPAN). Extraímos, da monitoração de segurança da informação, o total por dia, de *Alerts* gerados (Figura 28) e de eventos DNS (Figura 29). Verificamos que os picos do dia 22/01/2019 apresentados na seção 5.4 coincidem com os das figuras 28, 29 e colaboram com a validação da solução.

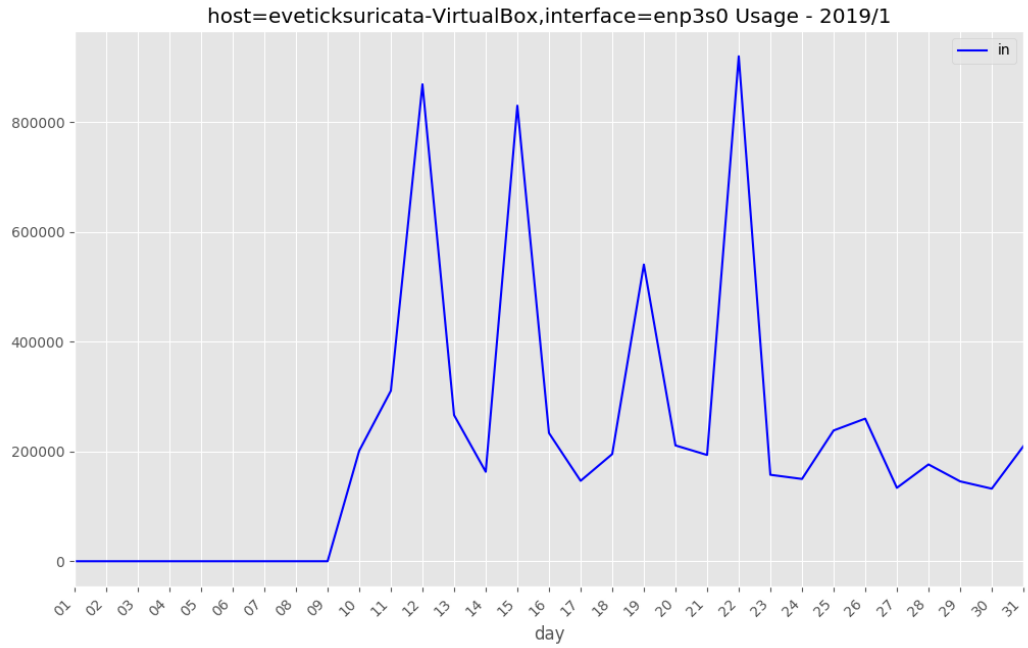

Figura 26 – Vazão de entrada em bits por segundo - Cenário ii - média por dia

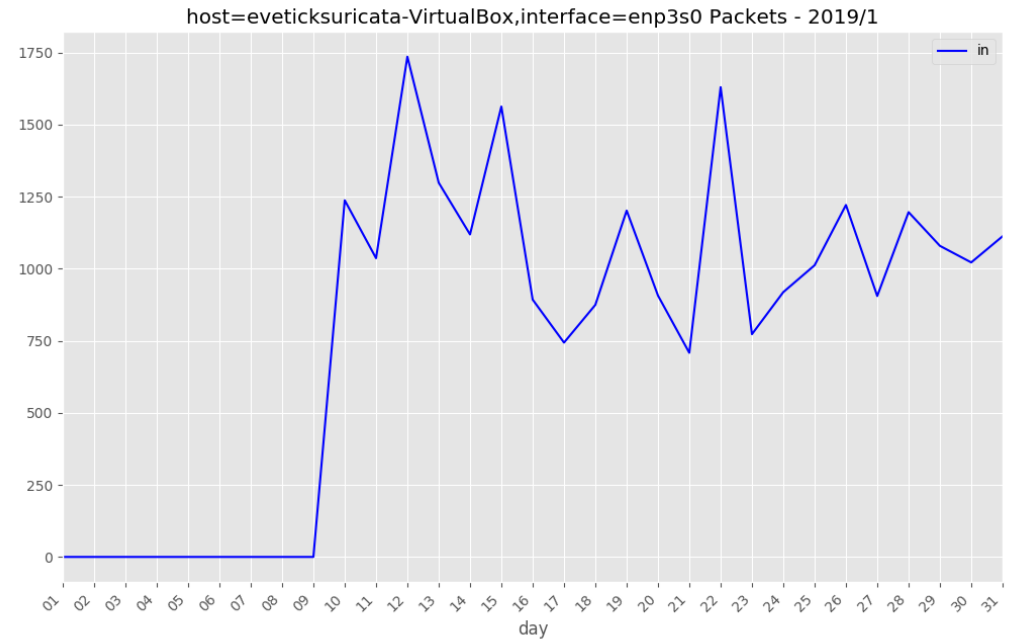

Figura 27 – Quantidade de pacotes de ENTRADA - Cenário ii - média por dia

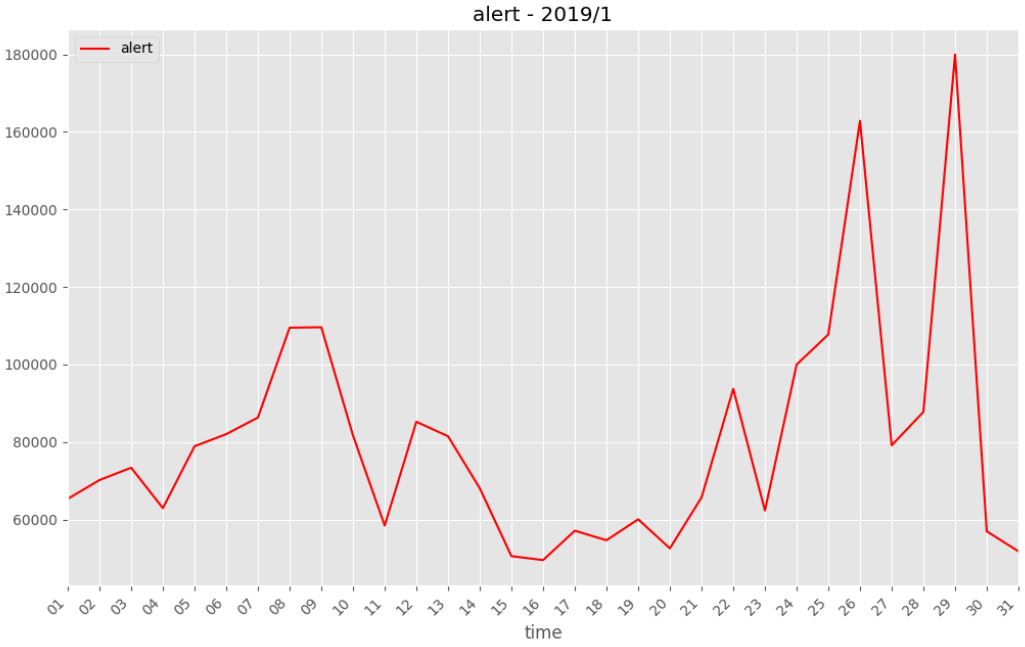

Figura 28 – Quantidade de *Alerts* - Cenário ii - total por dia

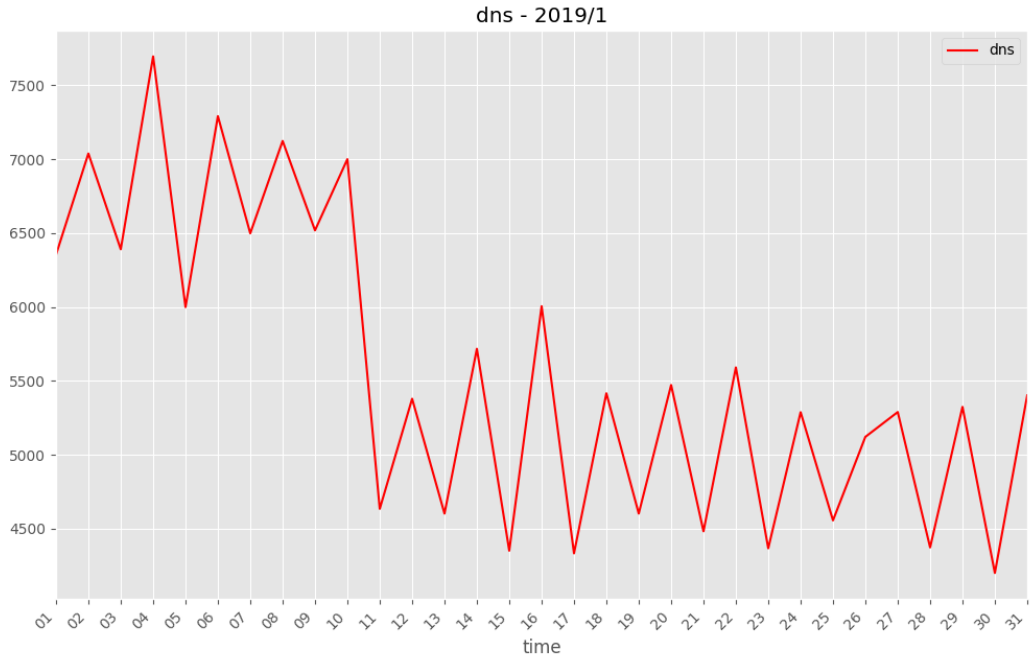

Figura 29 – Quantidade de eventos de DNS - Cenário ii - total por dia

Podemos observar, a partir dos resultados apresentados, que há outros picos de uso de recursos da rede. Contudo, na proposta deste trabalho, as informações de desempenho (uso de recursos da rede) são utilizadas em conjunto com as de segurança da informação, tornando assim o sistema de detecção mais robusto. É por esse motivo que evidenciamos alguns picos de uso de recurso da rede, em que houve também detecção de alertas de segurança. Em outros casos, pode-se observar nos gráficos que há picos ocorridos por outros motivos, mas não houve alerta de segurança, o que mostra a robustez da proposta.

5.6- Análise de escalabilidade da solução

Após as identificações ocorridas nos cenários i e ii, realizamos testes de escalabilidade em nossa solução. Utilizando-se da infraestrutura já configurada no cenário ii, acrescentamos um servidor com solução de virtualização KVM e QEMU. Neste cenário acrescentamos 08 máquinas virtuais e efetuamos o seguinte teste com ataque de negação de serviço:

- 01 máquina virtual a partir das 16:00 em 04/12/2019 com intervalo a cada 01 hora;
- 02 máquinas virtuais a partir das 23:55 em 04/12/2019 com intervalo a cada 01 hora;
- 04 máquinas virtuais a partir das 08:55 em 05/12/2019 com intervalo a cada 5 minutos;
- 08 máquinas virtuais a partir das 17:30 em 06/12/2019 com intervalo a cada 5 minutos;
- Término dos testes com desligamento das máquinas virtuais às 00:00 de 07/12.

Após os testes realizados, extraímos os dados da solução, consolidamos e verificamos, conforme a tabela 3, que ao aumentar a quantidade de máquinas virtuais e diminuir o intervalo do ataque, ou seja, potencializando o ataque, verificamos um aumento do número de alertas assim como um número de pacotes. Além disso, ao se intensificar o teste, verificamos que a plataforma necessitou que os dados coletados fossem transpostos para outro servidor, devido ao volume gerado.

Tabela 3 – Resultados consolidados de teste de escalabilidade da solução

Item	01 Máquina Virtual	02 Máquinas Virtuais	04 Máquinas Virtuais	08 Máquinas Virtuais
Quantidade de alertas (máximas)	2488	2413	1202136	3577547
Quantidade de pacotes (máximas)	2155	2598	3863	11010

Desse modo, podemos observar, através da Tabela 3, que o sistema de detecção proposto neste trabalho conseguiu detectar ataques, aliando dados de segurança da informação e de uso de recursos da rede, mesmo em um cenário com escalabilidade, ou seja, com mais dispositivos conectados à rede. Esta análise faz-se ainda mais importante pensando em cenários de IoT em que o número de dispositivos conectados tende a ser cada vez maior. Contudo, cumpre destacar que a quantidade de dados a serem analisados está diretamente ligada à capacidade de processamento e de armazenamento de um sistema de detecção.

6- Conclusões

Neste capítulo apresentaremos as conclusões sobre nosso trabalho em um Arcabouço de Caixa-Preta para a Detecção de Tráfego Malicioso em Ambientes de TIC. Apresentamos no Capítulo 1 a introdução e motivação de nosso trabalho. No Capítulo 2 estabelecemos os conceitos fundamentais através de ferramentas para implementação do ambiente. Apresentamos, posteriormente, nossa proposta no Capítulo 3 e nossa metodologia experimental no Capítulo 4 com as ferramentas utilizadas. Os resultados obtidos foram apresentados no Capítulo 5, através das coletas e a análise de informações de desempenho e segurança da informação nos ativos de TIC em Teste. Os resultados obtidos foram satisfatórios, tendo sido verificada a validade da metodologia proposta através da capacidade do ambiente em detectar tráfego malicioso, além de não conformidades, e de eventos e alertas de segurança da informação, mesmo em um cenário com escalabilidade.

Os objetivos deste trabalho, validando o ambiente, foram alcançados através dos seguintes resultados:

- Verificamos a presença de tráfego malicioso, primeiramente, através dos eventos apresentados na seção 5.1, com o software de comunicação com banco;
- Ainda no cenário i, apresentamos na seção 5.2 os eventos ocorridos com o decodificador de tv a cabo;
- Apresentamos na seção 5.4, no cenário ii, os testes com a câmera ip, e também encontramos eventos de tráfego malicioso;
- Após o tratamento dos dados dos cenários i e ii, consolidamos as informações e apresentamos nas seções 5.3 e 5.5, onde verificamos que nos dias indicados aos eventos os picos de *alerts*, eventos detectados pelo Suricata de DNS e HTTP, juntamente, com os dados de tráfego de rede e tráfego de pacotes, coincidiam.

Comparativamente com os ambientes citados no capítulo 2, nenhum dos outros ambientes apresentados, até o momento, consegue efetuar tais coletas e consultas a ponto de analisarmos os dados e conferir a identificação de não conformidades, de

eventos ou alertas de segurança da informação em ativos de TIC, juntamente com os dados de desempenho de maneira a validar e assim fazer a correlação entre ambos, o que mostra a robustez do sistema proposto em relação a algum falso-positivo.

Referências Bibliográficas

Acharya, S. and Pandya, V. (2012). Bridge between black box and white box–gray box testing technique. *International Journal of Electronics and Computer Science Engineering*, 2(1):175–185.

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., and Ayyash, M. (2015). Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys Tutorials*, 17(4):2347–2376.

Alaba, F. A., Othman, M., Hashem, I. A. T., and Alotaibi, F. (2017). Internet of things security: A survey. *Journal of Network and Computer Applications*, 88:10–28.

Ameen, R. Y. and Hamo, A. Y. (2013). Survey of server virtualization. *arXiv preprint arXiv:1304.3557*.

Atzori, L., Iera, A., and Morabito, G. (2010). The internet of things: A survey. *Computer Networks*, 54(15):2787 – 2805.

Bellard, F. (2005). Qemu, a fast and portable dynamic translator. In *USENIX Annual Technical Conference, FREENIX Track*, volume 41, page 46.

Blackwelder, B., Coleman, K., Colunga-Santoyo, S., Harrison, J. S., and Wozniak, D. (2016). The volkswagen scandal.

Carlson, T. (2001). Information security management: understanding iso 17799. *Lucent Technologies*.

Cisco, T. (2017). Cisco visual networking index: Global mobile data traffic forecast update, 2016-2021 white paper,. Página . `http://tinyurl.com/zzo6766`, (Acesso em: 04 mar. 2018).

Columbus, L. (2016). Roundup of internet of things forecasts and market estimates (forbes). Página . `http://tinyurl.com/yar5llet`, (Acesso em: 04 mar. 2018).

Cziesla, T., Kemper, J., Muntermann, J., and Sinanaj, G. (2015). Nsa revelations of privacy breaches: Do investors care?

Dash, P. (2013). *Getting started with oracle vm virtualbox*. Packt Publishing Ltd.

DHS (2016). Us department of homeland security: Strategic principles for securing the internet of things (iot).

Droms, R. (1997). Dynamic host configuration protocol. Technical report.

Fernandes, W. A. (2011). O movimento da qualidade no brasil.

Francia III, G. A., Garrett, A., and Brookshire, T. (2012). Virtualization for a cyber-security laboratory. In *Proceedings of the International Conference on Frontiers in Education: Computer Science and Computer Engineering (FECS)*, page 1. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

Glavic, B. and Dittrich, K. (2007). Data provenance: A categorization of existing approaches. *Datenbanksysteme in Business, Technologie und Web (BTW 2007)–12. Fachtagung des GI-Fachbereichs"Datenbanken und Informationssysteme"(DBIS)*.

Goto, Y. (2011). Kernel-based virtual machine technology. *Fujitsu Scientific and Technical Journal*, 47(3):362–368.

Grafana, T. (2018). Grafana labs.

Gupta, U. (2015). Monitoring in iot enabled devices. *arXiv preprint arXiv:1507.03780*.

Hallstensen, C. V. (2017). Multisensor fusion for intrusion detection and situational awareness. Master's thesis, NTNU.

Huang, C., Maltz, D. A., Li, J., and Greenberg, A. (2011). Public dns system and global traffic management. In *2011 Proceedings IEEE INFOCOM*, pages 2615–2623. IEEE.

Inmetro (2012). Avaliação da conformidade. a sociedade demanda, o inmetro faz.

Inmetro (2015). Avaliação da conformidade.

Inmetro (2018). A atividade de avaliação da conformidade.

Ish, J. (2017). EveBox the inbox for your suricata events.

ISO (2015). Iso/iec 15408-1:2009 information technology – security techniques – evaluation criteria for it security – part 1: Introduction and general model. Technical report, https://www.iso.org/standard/50341.html. 00000.

ISO (2018). Iso/iec 19790:2012 information technology – security techniques – security requirements for cryptographic modules. Technical report, https://www.iso.org/standard/52906.html. 00000.

Khan, M. E., Khan, F., et al. (2012). A comparative study of white box, black box and grey box testing techniques. *Int. J. Adv. Comput. Sci. Appl*, 3(6).

Kirtley, J. and Memmel, S. (2018). Rewriting the"book of the machine": Regulatory and liability issues for the internet of things. *Minnesota Journal of Law, Science & Technology*, 19(2):455.

Kumar, M., Singh, S. K., and Dwivedi, R. (2015). A comparative study of black box testing and white box testing techniques. *International Journal of Advance Research in Computer Science and Management Studies*, 3(10).

Lee, I. and Lee, K. (2015). The internet of things (iot): Applications, investments, and challenges for enterprises. *Business Horizons*, 58(4):431 – 440.

Machado, A. F. A. (2018). Utilização de simuladores para a formação de guerreiros cibernéticos. *Universitas: Gestão e TI*, 7(1).

Mendonça, N. M. L. (2015). *Gerador de eventos para testes de configurações de um SIEM*. PhD thesis.

Michael, C. et al. (2013). Black box security testing tools.

Miranda, G. L. et al. (2007). Limites e possibilidades das tic na educacao. *Sísifo. Revista de Ciencias da Educação*, 3:41–50.

Nakao, A., Ozaki, R., and Nishida, Y. (2008). Corelab: An emerging network testbed employing hosted virtual machine monitor. In *Proceedings of the 2008 ACM CoNEXT Conference*, page 73. ACM.

Naqvi, S. N. Z., Yfantidou, S., and Zimányi, E. (2017). Time series databases and influxdb.

nongiach (2018). arm_now is a qemu powered tool that allows instant setup of virtual machines on arm cpu, mips, powerpc, nios2, x86 and more, for reverse, exploit, fuzzing and programming purpose.

(OISF), O. I. S. F. (2018). Suricata engine is capable of real time intrusion detection (ids), inline intrusion prevention (ips), network security monitoring (nsm) and offline pcap processing. Página: `https://suricata-ids.org/`, (Acesso em: 30 set. 2018).

Oracle, V. (2018). Virtualbox.

Orebaugh, A., Ramirez, G., and Beale, J. (2006). *Wireshark & Ethereal network protocol analyzer toolkit*. Elsevier.

Ornaghi, A. and Valleri, M. (2005). Ettercap.

Perrone, G. and Romano, S. (2017). The docker security playground: A hands-on approach to the study of network security. In *2017 Principles, Systems and Applications of IP Telecommunications (IPTComm)*, pages 1–8. IEEE.

Presse, F. (2015). Volkswagen admite que 11 milhões de carros têm software que frauda testes. *G1*.

Restuccia, F., D'Oro, S., and Melodia, T. (2018). Securing the internet of things: New perspectives and research challenges. *arXiv preprint arXiv:1803.05022*.

Scarfone, K. (2011). *Guide to security for full virtualization technologies*. DIANE Publishing.

Sêmola, M. (2003). *Gestão Da Segurança Da Informação*. ELSEVIER EDITORA.

Shinde, A. (2016). Challenges in performance monitoring of hyper connected iot systems. In *Internet of Things and Applications (IOTA), International Conference on*, pages 174–178. IEEE.

Singh, B. and Singh, G. (2018). A study on virtualization and hypervisor in cloud computing. *International Journal of Computer Science and Mobile Applications*, 6(1):17–22.

Sjöberg, J., Zhang, Q., Ljung, L., Benveniste, A., Delyon, B., Glorennec, P.-Y., Hjalmarsson, H., and Juditsky, A. (1995). Nonlinear black-box modeling in system identification: a unified overview. *Automatica*, 31(12):1691–1724.

Stallings, W. (2017). *Cryptography and network security: principles and practice*. Pearson Upper Saddle River, NJ.

team, Q. (2018). QEMU is a generic and open source machine emulator and virtualizer. Página: `https://www.qemu.org/`, (Acesso em: 30 set. 2018).

Uramová, J., Scgeč, P., Moravčík, M., Papán, J., Kontšek, M., and Hrabovskỳ, J. (2018). Infrastructure for generating new ids dataset. In *2018 16th International Conference on Emerging eLearning Technologies and Applications (ICETA)*, pages 603–610. IEEE.

VERAS, M. (2011). *Virtualização*. Brasport.

von Solms, R. (1998). Information security management (3): the code of practice for information security management (bs 7799). *Information Management & Computer Security*, 6(5):224–225.

Welsh, C. (2013). *GNS3 network simulation guide*. Packt Publ.

Whitmore, A., Agarwal, A., and Da Xu, L. (2015). The internet of things—a survey of topics and trends. *Information Systems Frontiers*, 17(2):261–274.

Wilde, E. and Pautasso, C. (2011). *REST: from research to practice*. Springer Science & Business Media.

Wood, M. and Erlinger, M. A. (2007). Intrusion Detection Message Exchange Requirements. RFC, RFC Editor.

Yadav, A. K., Garg, M., et al. (2019). Docker containers versus virtual machine-based virtualization. In *Emerging Technologies in Data Mining and Information Security*, pages 141–150. Springer.

Zhang, Q., Cheng, L., and Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges. *Journal of internet services and applications*, 1(1):7–18.

A- Orientações para instalação e configuração de ambiente de testes

A.1- Introdução

Neste apêndice, sugerimos um exemplo de implantação e instalação para um ambiente de teste de caixa preta em ativos de TIC.

A.1.1 Rede

Para configuração do ambiente, algumas premissas básicas:

1. Conectividade com a Internet
2. Plano de endereçamento IP interno
3. Equipamento para prover conectividade com a Internet e o plano de endereçamento IP
4. Servidor para instalação do Ambiente de Monitoração
5. Servidor para instalação do Ambiente de Virtualização

A definição de plano de endereçamento IP ajudará na identificação dos ativos de TIC que estarão sujeitos a teste e dos que não estarão.

O plano de endereçamento inicial, para rede interna, sugerimos 192.168.1.0/24. Que o equipamento a prover conectividade com a Internet tenha o IP 192.168.1.1/24; o servidor para instalação do Ambiente de monitoração tenha um fixo e como sugestão 192.168.1.2/24; o servidor para instalação do Ambiente de virtualização tenha um fixo e como sugestão 192.168.1.3/24; qualquer outro equipamento que não esteja em teste e

Rede Interna - 192.168.1.0/24		
Nome do ativo de TIC	IP	MAC
Roteador/Modem com Wifi	192.168.1.1	52:54:00:4d:6b:a0
Monitoração	192.168.1.2	52:54:00:4d:6b:b0
Virtualização	192.168.1.3	52:54:00:4d:6b:c0
	...	
Faixa DHCP	192.168.1.50 à 192.168.1.254	

Tabela 4 – Exemplo para documentação de Rede

faça parte de ambiente também tenha um IP fixo e; todos os outros ativos de TIC que estejam em Teste estejam em uma faixa para DHCP dentro da rede definida.

Ainda sobre endereçamento e identificação dos ativos de TIC, sugerimos, a tabela abaixo e que se for possível, ocorra a identificação dos endereços de MAC de todos os ativos de TIC, pois, como será utilizada uma faixa DHCP para os ativos em teste, os mesmos poderão mudar de IP dentro desta faixa. Via de regra, os equipamentos que proveem conectividade têm esta informação.

O endereçamento com a Internet deverá ser fornecido e configurado previamente no equipamento que proverá a conectividade com a Internet. Este deverá estar previamente configurado para o acesso à Internet e com um *firewall* para proteção.

Ainda sobre o endereçamento para rede interna (192.168.1.0/24), este poderá estar junto do equipamento que proverá a conectividade com a Internet. Este equipamento deve ter a capacidade de fornecer um serviço DHCP e estar configurado conforme a sugestão da tabela acima.

Abaixo equipamentos que testamos e que tem as características que necessitamos para conectividade com a Internet:

1. Modem e Roteador Wireless Technicolor TG588V V2 ADSL VDSL
2. Modem e Roteador Wireless Technicolor TD5336
3. Roteador Wireless C3-TECH W-R2000nL
4. Switch Cisco IOS Software, Catalyst 4500 L3
5. Roteador e/ou Modem com Wireless e suporte ao sistema operacional OpenWRT 18.0X

O servidor para instalação do Ambiente de Monitoração, que sugerimos ser configurado na rede interna com IP 192.168.1.2, deve ser mensurado de acordo com o volume de tráfego e dados que devem ser guardados para análise dos Teste de Caixa Preta em ativos de TIC. A configuração mínima para o mesmo sugerida é:

1. 8GB de memória RAM
2. Armazenamento de 500 GB
3. Processador com suporte a no mínimo 04 CPUs
4. Placa de rede com suporte a *FastEthernet* 100Mbps
5. Sistema Operacional Linux Ubuntu 16 ou CentOS 7

Lembrando que caso seja utilizada a opção de *sniffer*, a partir do ambiente de monitoração, é necessária a verificação de suporte e velocidade da placa de rede do servidor.

Ettercap

O pacote Ettercap é instalado a partir *script* (Seção A.1.2), mas, também não é habilitado para iniciar automaticamente. Este pacote ajudará capturar o tráfego de rede caso seja utilizada a técnica *sniffing*. Além disso, este pacote deverá ser executado no mesmo servidor que estará o Suricata IDS e o EveBox.

Para configurar qualquer item do Ettercap, este deve ser configurado no diretório */etc/ettercap* e edite os arquivos do diretório. Estes são os seguintes:

1. etter.conf - Configuração do Ettercap
2. etter.dns - Configuração do plug-in dns_spoof do Ettercap
3. etter.mdns - Configuração do plug-in mdns_spoof do Ettercap
4. etter.nbns - Configuração do plug-in nbns_spoof do Ettercap

Edição de configuração do Ettercap

```
# cd /etc/ettercap
# vi etter.conf
```

Diferente dos outros serviços a execução do Ettercap deve ser manual. Para isso, será iniciado um *screen* e a partir do mesmo será feita a execução. Para exemplos de execução, verifique em *https://github.com/carlos-teles/etsg/blob/master/start-ettercap.sh* . Execute o comando abaixo, como root, onde 192.168.1.1 é o IP de saída para Internet e *gateway* da rede para capturar o tráfego de rede:

Executar Ettercap dentro de screen

```
# screen -S ettercap
# /usr/bin/ettercap -T -M arp:remote /192.168.1.1// ///
```

Após a inicialização do Ettercap deveremos ter a saída abaixo em execução (Figura 30).

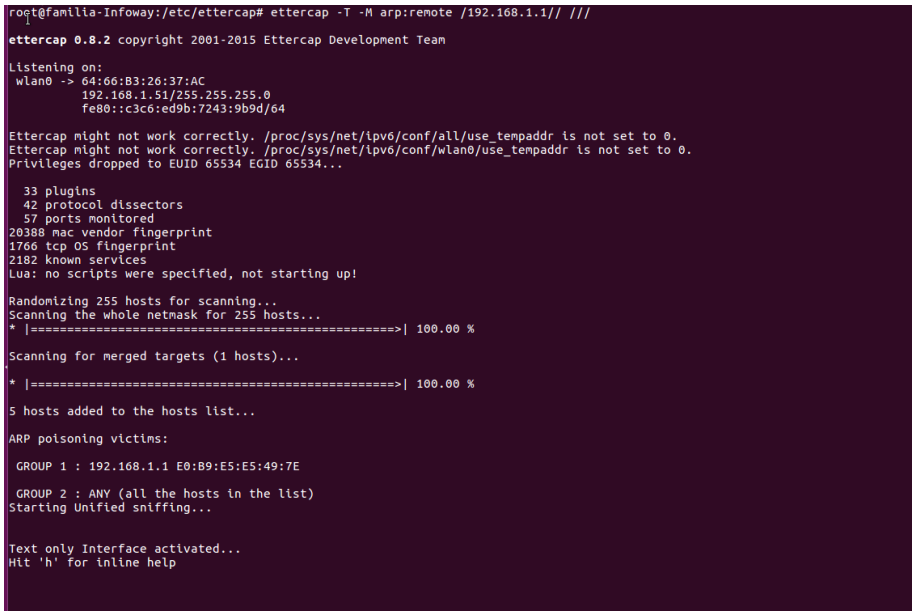

Figura 30 – Tela execução do Ettercap

Espelhamento de portas

O Espelhamento de portas é uma técnica para captura de tráfego de rede. Neste caso, temos duas interfaces de rede. Uma por onde passa todo o tráfego de rede

(entrada e saída) e outra que receberá uma cópia. Esta configuração foi executada em um Switch Cisco IOS Software, Catalyst 4500 L3. As interfaces selecionadas são GigabitEthernet6/21 (Gi6/21) e GigabitEthernet4/39 (Gi4/39). A interface Gi6/21 receberá uma cópia de todo o tráfego da interface Gi4/39, que é por onde passa todo tráfego de entrada e saída que determinamos. Segue abaixo a configuração já aplicada:

Exemplo de configuração de Espelhamento de portas

```
swcore01.rjo#sh run | i monitor session
monitor session 3 source interface Gi4/39
monitor session 3 filter packet-type good rx
monitor session 3 destination interface Gi6/21
```

Para verificar se a configuração de espelhamento de portas ficou correta, execute o comando abaixo:

Verificação de Configuração de Espelhamento de portas

```
swcore01.rjo#show monitor session all
Session 3
---------
Type                   : Local Session
Source Ports           :
Both               : Gi4/39
Destination Ports      : Gi6/21
Encapsulation      : Native
Ingress      : Disabled
Learning : Disabled
Filter Pkt Type        :
RX Only       : Good
swcore01.rjo#
```

A.1.2 Monitoração

O Ambiente de Monitoração deve ser instalado em um Servidor com sistema Operacional Linux. Utilizamos como base as versões de 16.XX do Ubuntu e também as versões de CentOS 7. Em ambos instalamos os pacotes e funcionaram corretamente.

Abaixo algumas das configurações utilizadas.

Hardware do Servidor de Monitoração					
#	CPU	Memória	Armazenamento	Rede	Sistema Operacional
1	Intel(R) Core(TM)2 Quad CPU Q8200 @ 2.33GHz	8GB	148GB	Realtek RTL8111/8168/8411 PCI Express Gigabit Ethernet Controller (rev 02)	Ubuntu 16.04.6 LTS
2	Intel(R) Xeon(R) CPU 5140 @2.33GHz	8GB	126GB	Intel Corporation 82571EB Gigabit Ethernet Controller (rev 06)	Kali GNU/Linux Rolling
3	Intel Pentium 4 @3.4GHz	6GB	500GB	Braodcom NetXtreme BCM5721 Gigabit Ethernet	Linux Mint 18.3

Para facilitar a instalação, desenvolvemos um script que testar a versão de sistema operacional e que instala os pacotes já citamos acima. Este script encontra em https://github.com/carlos-teles/etsg/ e possui o nome ”install-only-etsg.sh”.

Resumidamente, os pacotes a serem instalados são:

1. Softwares básicos caso não estejam instalados como *OpenSSH*, Servidor DHCP e atualizações de Python
2. Suricata IDS
3. EVEBOX
4. Grafana
5. Influxdb e Telegraf
6. Suricata-update (atualização de regras do Suricata IDS)

Uma vez que todos os pacotes estejam instalados é necessário que sejam verificados os estados dos processos que devem estar em execução.

Influxdb e Telegraf

Os pacotes Influxdb e Telegraf são instalados a partir *script* acima (Seção A.1.2), mas não são habilitados para iniciar automaticamente, pois, os mesmos devem ser configurados previamente. Estes pacotes ajudarão a apresentar informações de Desempenho.

O pacote Telegraf, caso queira configurar algum item, este deve ser configurado no diretório */etc/telegraf* e edite o arquivo *telegraf.conf*.

Edição de configuração do Telegraf

```
# cd /etc/telegraf
# vi telegraf.conf
```

Em nosso caso é necessário alterar as configurações, pois, precisamos que todos os dados de tráfego de rede sejam coletados. Assim, sugerimos remover os comentários das linhas abaixo, ficando da seguinte forma:

Edição de configuração do Telegraf

```
# # Read metrics about network interface usage
[[inputs.net]]
#   ## By default, telegraf gathers stats from any up interface (excluding loopback)
#   ## Setting interfaces will tell it to gather these explicit interfaces,
#   ## regardless of status.
#   ##
interfaces = ["eth0"]
#   ##
#   ## On linux systems telegraf also collects protocol stats.
#   ## Setting ignore_protocol_stats to true will skip reporting of protocol metrics.
#   ##
#   # ignore_protocol_stats = false
#   ##
```

É importante lembrar que cada distribuição linux pode apresentar nomes diferentes de interfaces. Em nosso caso, nossa interface é a *eth0*. Podemos ter diversas e que gostaríamos que fossem monitoradas. Desta forma, caso tenhamos em nosso sistema *eth0*, *eth1* e *eth2*, a linha acima ficaria:

Edição de configuração do Telegraf

```
# # Read metrics about network interface usage
[[inputs.net]]
#   ## By default, telegraf gathers stats from any up interface (excluding loopback)
#   ## Setting interfaces will tell it to gather these explicit interfaces,
#   ## regardless of status.
#   ##
interfaces = ["eth0","eth1","eth2"]
#   ##
#   ## On linux systems telegraf also collects protocol stats.
#   ## Setting ignore_protocol_stats to true will skip reporting of protocol metrics.
#   ##
#   # ignore_protocol_stats = false
#   ##
```

Caso queira configurar algum item do Influxdb, este deve ser configurado no diretório */etc/influxdb* e edite o arquivo *influxdb.conf*.

Edição de configuração do Influxdb

```
# cd /etc/influxdb
# vi influxdb.conf
```

Neste caso, para efeito de backup, sugerimos remover o comentário da linha, ficando da seguinte forma:

Edição de configuração do Telegraf

```
bind-address = "127.0.0.1:8088"
```

Assim, agora, poderemos continuar com as verificações. Os pacotes não vêm para serem executados automaticamente. Verificaremos então seus status.

E devem estar com as seguintes saídas:

Verificar status do Telegraf

```
# systemctl status telegraf.service
 telegraf.service - The plugin-driven server agent for reporting metrics into I
Loaded: loaded (/lib/systemd/system/telegraf.service; enabled; vendor preset:
Active: inactive (dead) since Dom 2019-05-05 01:02:09 -03; 9s ago
Docs: https://github.com/influxdata/telegraf
Process: 1274 ExecStart=/usr/bin/telegraf -config /etc/telegraf/telegraf.conf
Main PID: 1274 (code=exited, status=0/SUCCESS)
```

Verificar status do Influxdb

```
# systemctl status influxd.service
 influxdb.service - InfluxDB is an open-source, distributed, time series databa
Loaded: loaded (/lib/systemd/system/influxdb.service; enabled; vendor preset:
Active: inactive (dead) since Dom 2019-05-05 01:02:14 -03; 1min 5s ago
Docs: https://docs.influxdata.com/influxdb/
Process: 1396 ExecStart=/usr/bin/influxd -config /etc/influxdb/influxdb.conf
Main PID: 1396 (code=exited, status=0/SUCCESS)
```

Verificamos assim que ambos os serviços, telegraf e influxdb, estão inativos. Para iniciar e verificar o status destes, execute os comandos abaixo:

Iniciar e Verificar status do Telegraf

```
# systemctl start telegraf.service
# systemctl status telegraf.service
 telegraf.service - The plugin-driven server agent for reporting metrics into I
Loaded: loaded (/lib/systemd/system/telegraf.service; enabled; vendor preset:
Active: active (running) since Dom 2019-05-05 01:07:57 -03; 12s ago
Docs: https://github.com/influxdata/telegraf
Main PID: 9195 (telegraf)
CGroup: /system.slice/telegraf.service
9195 /usr/bin/telegraf -config /etc/telegraf/telegraf.conf -config-
```

Iniciar e Verificar status do Influxdb

```
# systemctl start influxd.service
# systemctl status influxd.service
 influxdb.service - InfluxDB is an open-source, distributed, time series databa
Loaded: loaded (/lib/systemd/system/influxdb.service; enabled; vendor preset:
Active: active (running) since Dom 2019-05-05 01:08:04 -03; 1min 4s ago
Docs: https://docs.influxdata.com/influxdb/
Main PID: 9216 (influxd)
CGroup: /system.slice/influxdb.service
9216 /usr/bin/influxd -config /etc/influxdb/influxdb.conf
```

A partir deste momento, os serviços estão em execução. Para que eles possam iniciar automaticamente, ou seja, caso seja feito um *reboot* do sistema operacional, os seguintes comandos também devem ser executados:

Habilitar Influxdb e Telegraf na inicialização

```
# systemctl enable influxd.service
# systemctl enable telegraf.service
```

Grafana

Durante a instalação do Grafana, o mesmo é habilitado automaticamente. Para se certificar, como root, execute:

Verificar status do Grafana

```
# systemctl status grafana-server

 grafana-server.service - Grafana instance
Loaded: loaded (/usr/lib/systemd/system/grafana-server.service; disabled; ven
Active: active (running) since Dom 2019-05-05 01:20:38 -03; 1s ago
Docs: http://docs.grafana.org
Main PID: 9416 (grafana-server)
CGroup: /system.slice/grafana-server.service
9416 /usr/sbin/grafana-server --config=/etc/grafana/grafana.ini --p
```

Após esta verificação, devemos acessar pelo *browser* o servidor do grafana. Por padrão, este é executado na porta 3000. A URL para acesso é *http://192.168.1.2:3000/login* , caso você tenha utilizado o IP sugerido acima. Ao acessar a seguinte tela aparecerá.

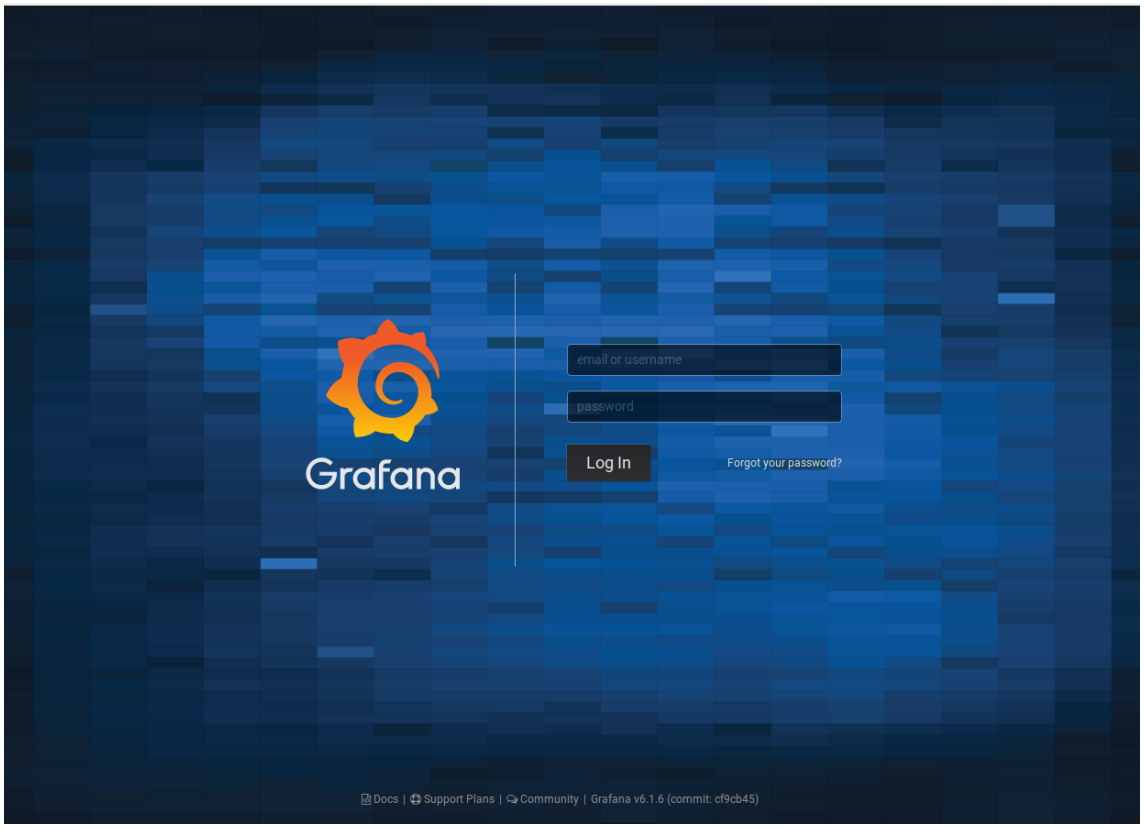

Figura 31 – Tela inicial do Grafana

Neste primeiro acesso o login e senha padrão do sistema são ’admin’ e ’admin’ respectivamente. Será solicitada a troca se senha após o primeiro login e a senha não pode ser ’admin’.

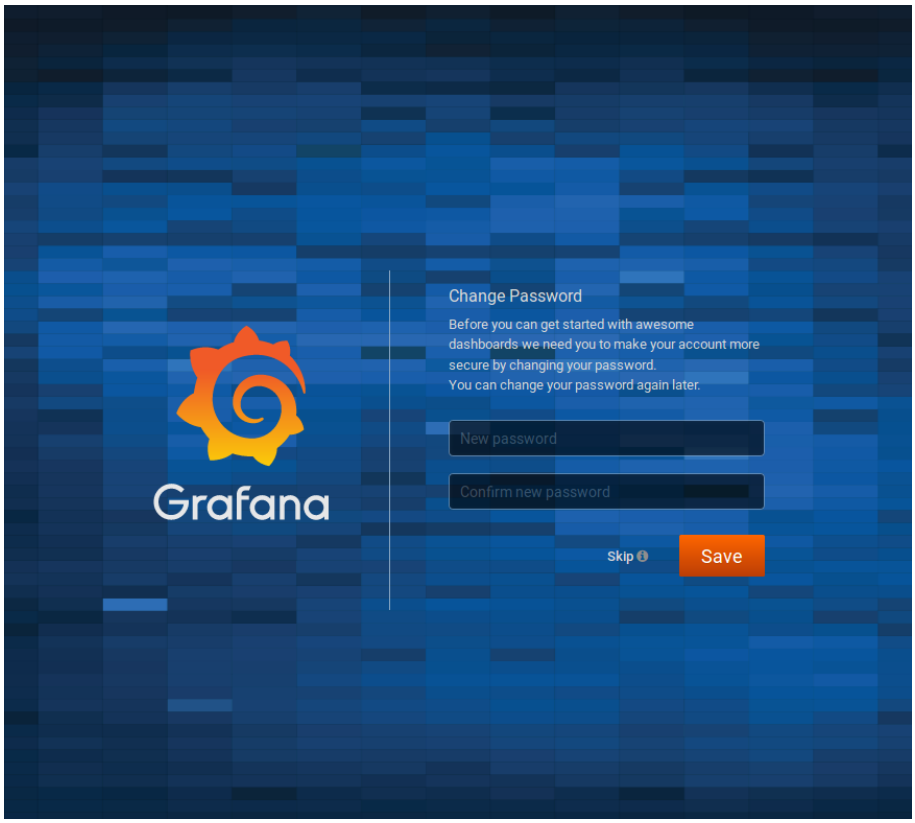

Figura 32 – Troca de senha inicial do Grafana

Após a troca de senha, a tela inicial do Grafana, vide figura 33.

Para mostrar os dados coletados pelo Telegraf e guardados no Influxdb é necessário acrescentar um *data source*. Clique em *Add data source*, que está selecionado na figura acima. Serão mostrados diversos *data sources* suportadas pelo Grafana. Em nosso caso utilizaremos o Influxdb. Clique no ícone relacionado conforme a figura 34.

Após clicar a tela abaixo (Figura 35) será carregada e deverá ser preenchida conforme mostrada.

Depois de preencher os campos, basta clicar no botão *Save & Test*, no rodapé da página (Figura 36). Caso tenha preenchido tudo corretamente a mensagem *Data source is working* aparecerá em verde.

Com o *data source* ’Influxdb’ configurado e testado (Figura 37) é necessário acrescentar um *dashboard* para exibir as informações coletadas. Esta é uma das facilidades do Grafana, pois, é possível pesquisar no site do Grafana, em *https://grafana.com/dashboards*,

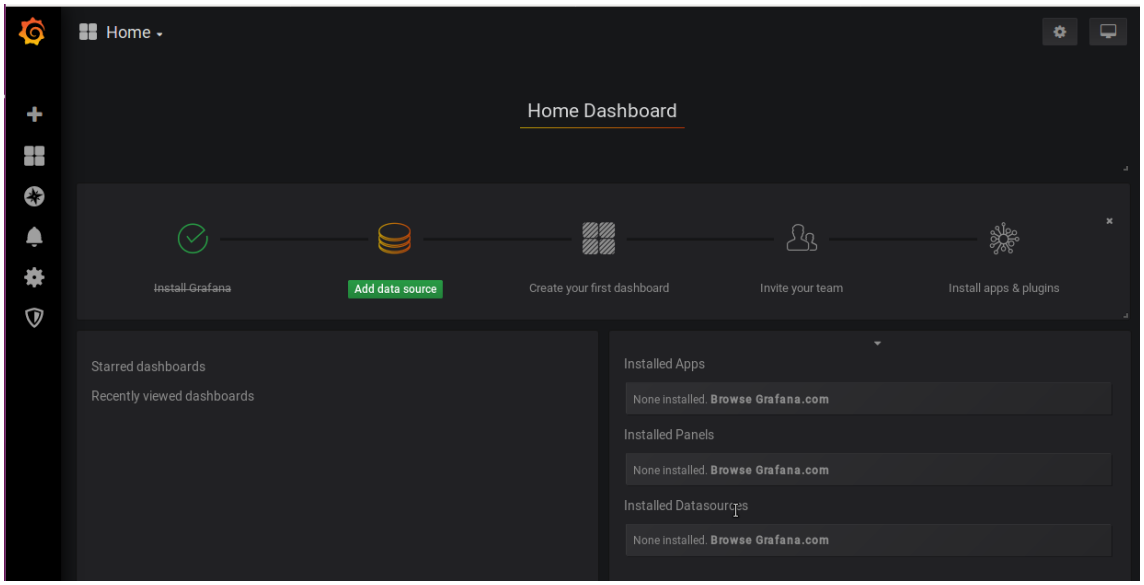

Figura 33 – Tela inicial do Grafana após login

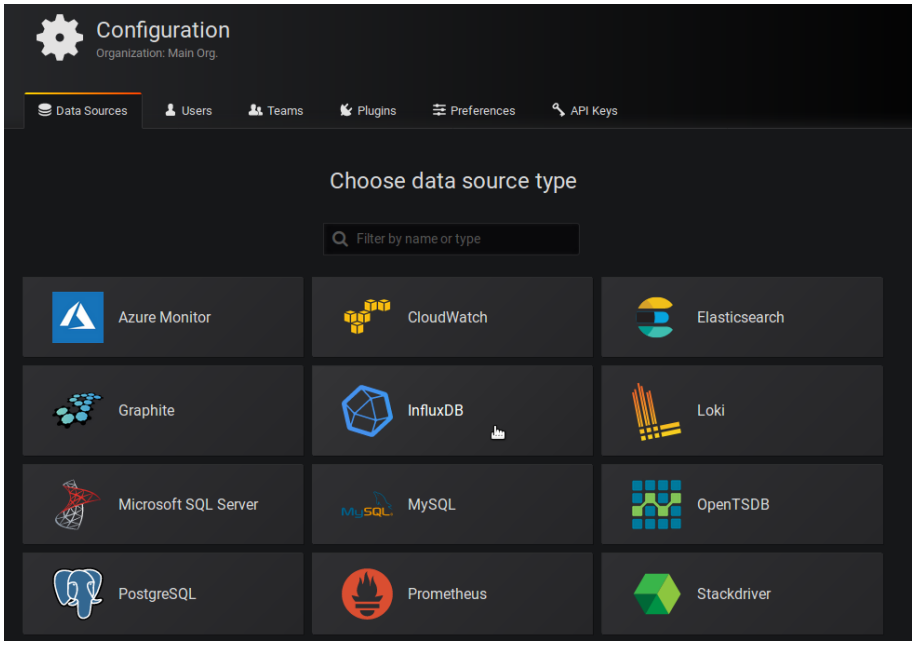

Figura 34 – Tela *Add data source* do Grafana

centenas de *dashboards* disponíveis e facilmente importados para o sistema. Sugerimos que seja importado para o Grafana o *dashboard* com o ID 928. Este possui uma documentação em *https://grafana.com/dashboards/928* .

Para importar no Grafana o *dashboard* com o ID 928 clique no ícone de 'mais', no canto superior esquerdo (Figura 38) e depois em *Import*. Em seguida a tela para importar será carregada (Figura 39).

Na tela carregada (Figura 39), clicar na caixa de texto com título *Grafana.com*

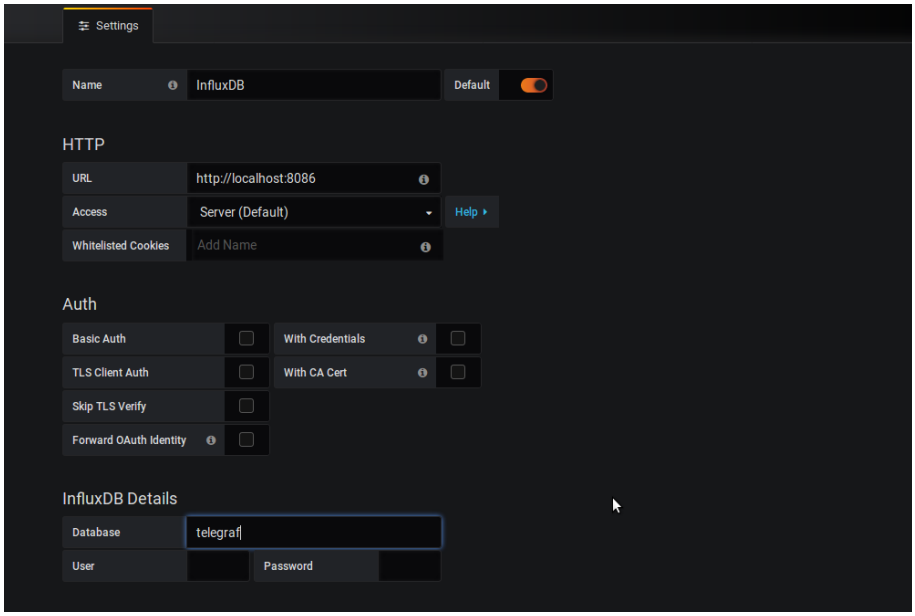

Figura 35 – Configurar *data source* do Grafana

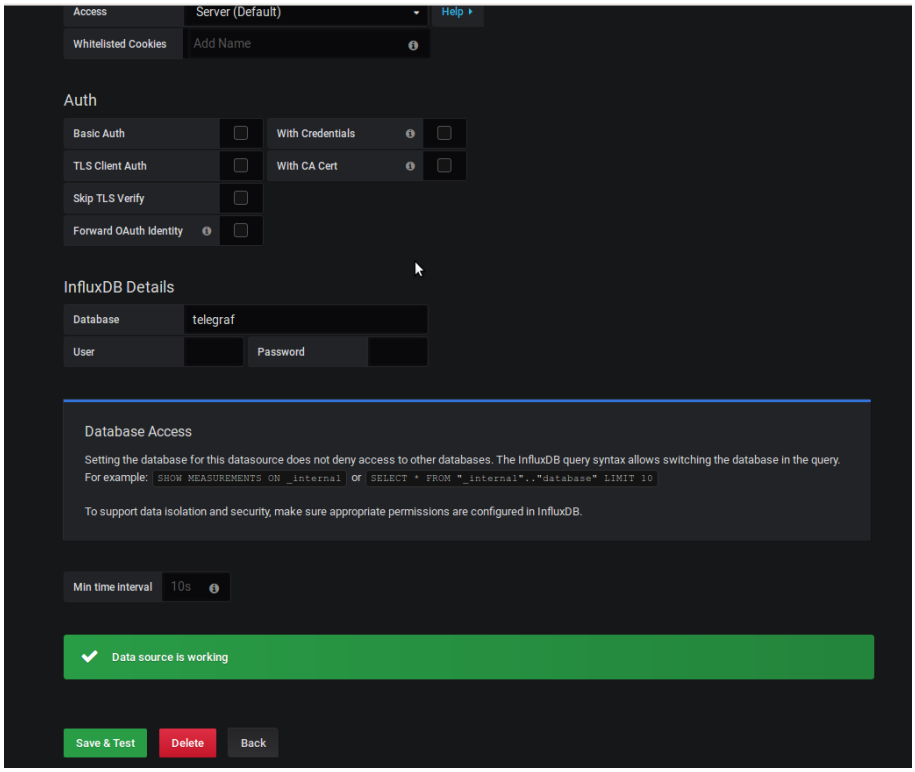

Figura 36 – Salvar configuração do *data source* do Grafana

Dashboard e digite o número 928. Depois clique no botão *Load* em azul para importar o *dashboard*. As configurações do *dashboard* são carregadas (Figura 40), bastando selecionar no campo *InfluxDB telegraf* o *data source* que criamos previamente com o nome de 'InfluxDB' e clicando no botão verde *Import*.

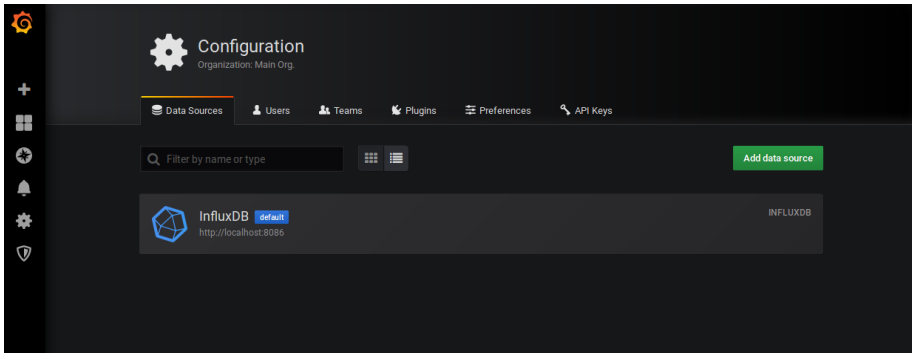

Figura 37 – Inflxudb *data source* do Grafana

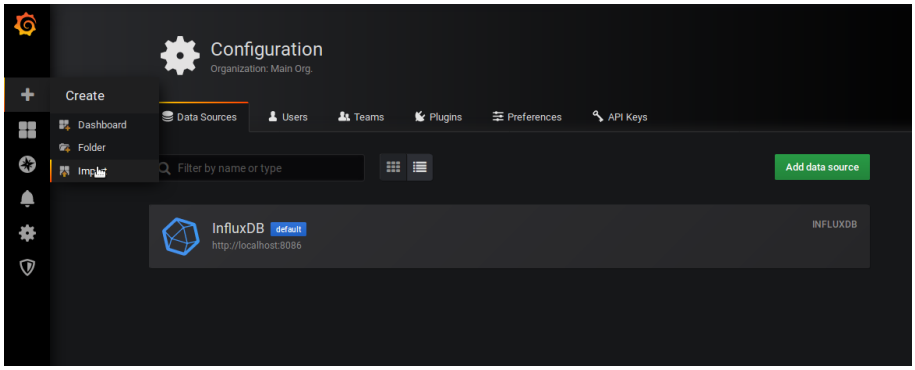

Figura 38 – Como importar o dashboard

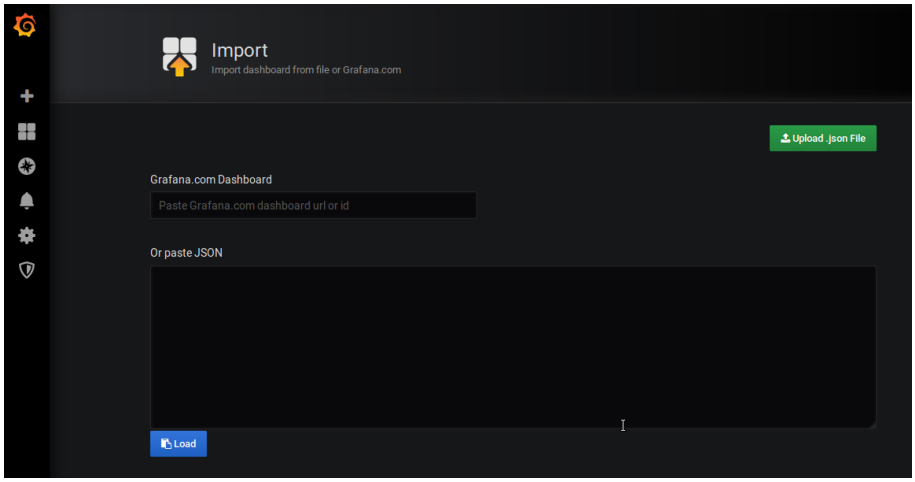

Figura 39 – Importar *dashboard*

Logo após clicar no botão *Import* o Grafana mostrará o *dahsboard* já com dados coletados. Abaixo algumas das informações carregadas.

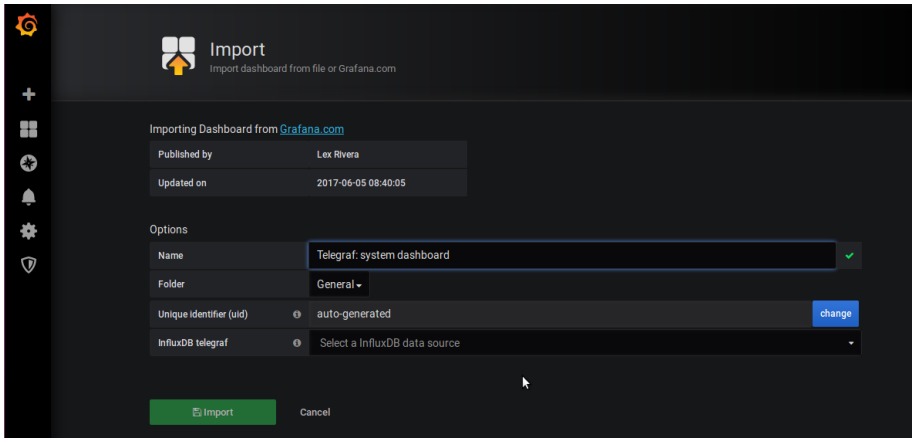

Figura 40 – Dados do *dashboard* 928 - 1

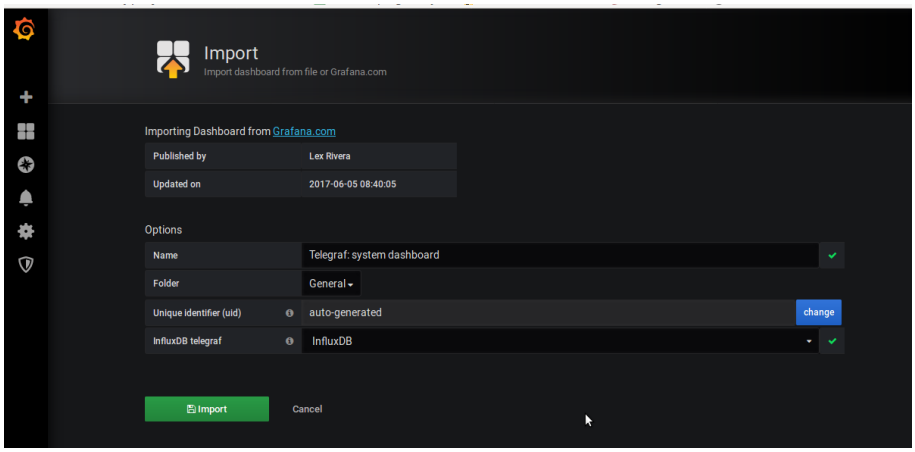

Figura 41 – Dados do *dashboard* 928 - 2

Suricata IDS

O pacote Suricata IDS é instalado a partir *script* (Seção A.1.2), mas, também não é habilitado para iniciar automaticamente. Este pacote ajudará a coletar informações de Segurança.

Para configurar qualquer item do Suricata, este deve ser configurado no diretório */etc/suricata* e edite o arquivo *suricata.yaml*.

Edição de configuração do Suricata

```
# cd /etc/suricata
# vi suricata.yaml
```

O pacote não vem para serem executados automaticamente. Verificaremos então seu status e deve estar com a seguinte saída:

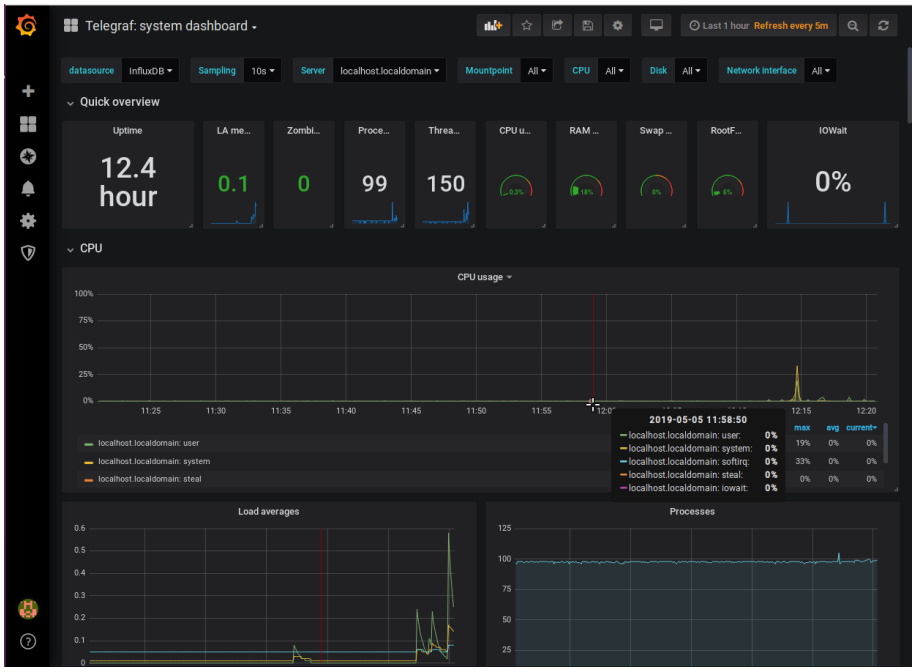

Figura 42 – *Dashboard* 928 - 01

Figura 43 – *Dashboard* 928 - 02

Verificar status do Suricata

```
# systemctl status suricata.service
 suricata.service - LSB: Next Generation IDS/IPS
Loaded: loaded (/etc/init.d/suricata; bad; vendor preset: enabled)
Active: inactive (dead)
Docs: man:systemd-sysv-generator(8)
```

Figura 44 – *Dashboard* 928 - 03

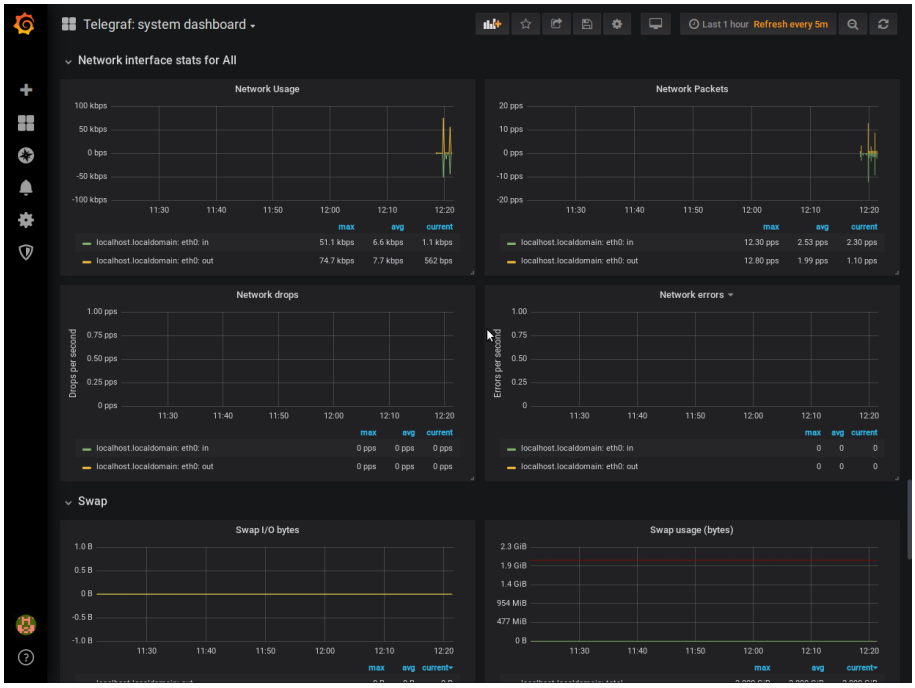

Figura 45 – *Dashboard* 928 - 04

Verificamos assim que o serviço do Suricata está inativo. Para iniciar e verificar o status destes, execute os comandos abaixo:

Iniciar e Verificar status do Suricata

```
# systemctl start suricata.service
# systemctl status suricata.service
 suricata.service - LSB: Next Generation IDS/IPS
Loaded: loaded (/etc/init.d/suricata; bad; vendor preset: enabled)
Active: active (running) since Seg 2019-05-06 00:19:22 -03; 1s ago
Docs: man:systemd-sysv-generator(8)
Process: 22588 ExecStart=/etc/init.d/suricata start (code=exited, status=0/SUC
CGroup: /system.slice/suricata.service
22597 /usr/bin/suricata -c /etc/suricata/suricata.yaml --pidfile /v
```

A partir deste momento o serviço está em execução. Para que possa iniciar automaticamente, ou seja, caso seja feito um *reboot* do sistema operacional, o seguinte comando também deve ser executado:

Habilitar Suricata na inicialização

```
# systemctl enable suricata.service
```

EveBox

O pacote EveBox é instalado a partir *script* (Seção A.1.2), mas, também não é habilitado para iniciar automaticamente. Este pacote ajudará a mostrar as informações de Segurança coletadas.

Para configurar qualquer item do EveBox, este deve ser configurado no diretório */etc/evebox* e faça uma cópia do arquivo *evebox.yaml.example* para *evebox.yaml*.

Edição de configuração do EveBox

```
# cd /etc/evebox
# cp evebox.yaml.example evebox.yaml
# vi evebox.yaml
```

Diferente dos outros serviços, como não utilizaremos neste momento o Elasticsearch, pois, esta é uma abordagem leve para o ambiente, a execução do EveBox será manual. Para isso, será iniciado um *screen* e a partir do mesmo será feita a execução. Execute os comandos abaixo:

Executar EveBox dentro de screen

```
# screen -S evebox
# /usr/bin/evebox -v -D /var/lib/evebox --datastore sqlite --input /var/log/suricata/eve.json
```

Após a inicialização do EveBox deveremos ter a saída abaixo em execução (Figura 46).

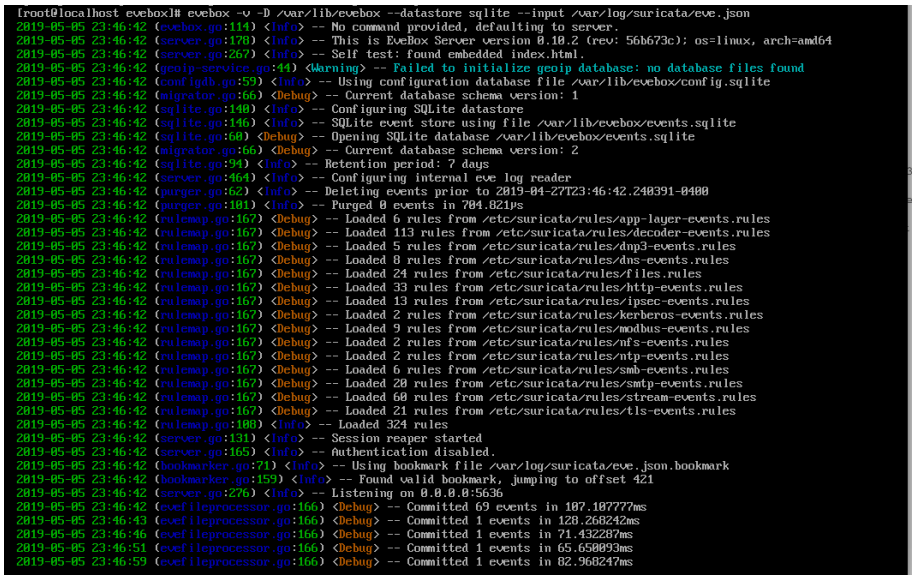

Figura 46 – EveBox em execução

Para sair do *screen* e deixar o EveBox em execução, aperte as teclas ’Ctrl + A’ e depois apenas o ’d’.

Após a execução do EveBox via linha comando é possível acessar pelo *browser* no endereço http://192.168.1.2:5636/. As telas abaixo (Figuras 47 e 48) mostram o sistema e principalmente na Figura 48, dados coletados pelo Suricata IDS, carregados e mostrados via EveBox. A primeira tela (Figura 47) apresenta informações apenas quando há alertas pelo Suricata IDS.

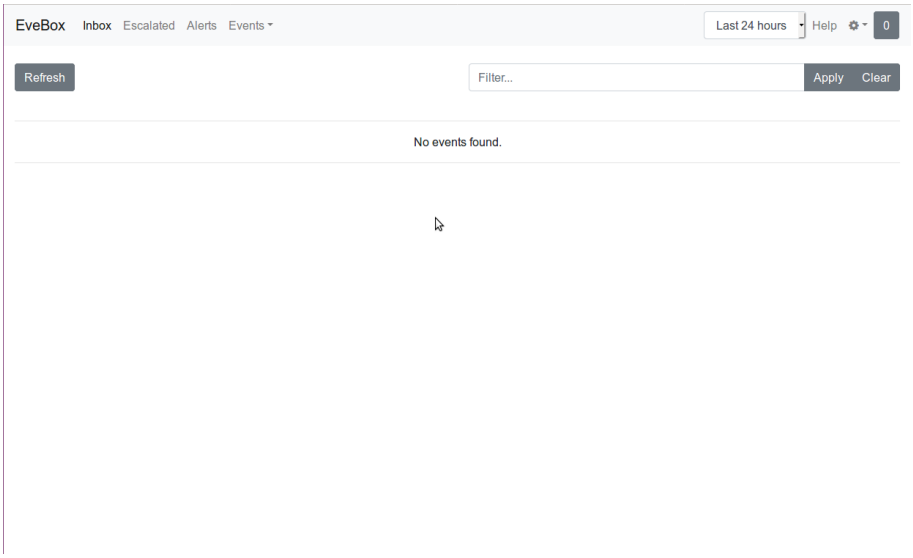

Figura 47 – Dados do EveBox - 01

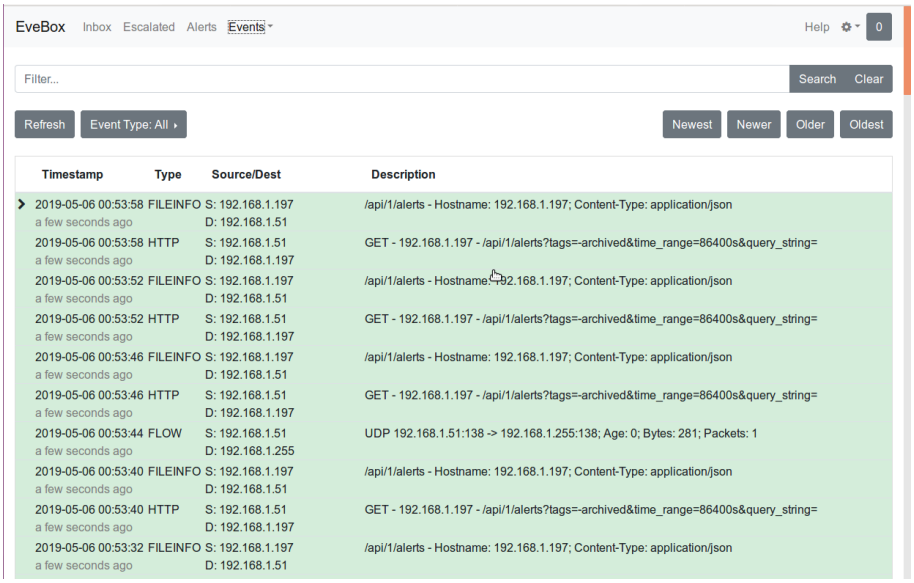

Figura 48 – Dados do EveBox - 02

A.1.3 Virtualização

Para configuração do ambiente de virtualização seguem algumas premissas básicas para o ambiente:

1. Conectividade com a Internet já implementada
2. Servidor instalado do Ambiente de Monitoração

3. Servidor para instalação do Ambiente de Virtualização

Segundo o plano de endereçamento inicial, para rede interna, o servidor para instalação do Ambiente de virtualização terá um fixo e será 192.168.1.3/24.

O Ambiente de Virtualização deve ser instalado em um Servidor com sistema operacional Linux ou Windows. Cada um destes ambientes pode ser utilizado com uma finalidade de virtualização. Para que o ativo de TIC em teste não se considere no mesmo, em nosso caso, utilizamos cada um destes ambientes com os seguintes propósitos:

1. Virtualização de dispositivos para simulação de uma rede real, ou seja, o ativo de TIC pode verificar sua vizinhança e dado que há uma quantidade de outros ativos de TIC, o mesmo não se considera em testes.

2. Virtualização dispositivos em volta para simular acesso ao ativo de TIC e para ambiente de monitoração verificar o tráfego de rede;

3. Virtualização do ativo de TIC em teste, se for possível.

Assim, segue abaixo algumas das configurações utilizadas. É importante lembrar que, no caso dos processadores x86, as tecnologias Intel VT-X e AMD-v orientam o suporte a virtualização. Em todos os casos os servidores tinham 8 GB de memória RAM.

Hardware do Servidor de Virtualização					
#	CPU	Armazenamento	Sistema Operacional	Hypervisor	Propósito
1	Intel(R) Core(TM)2 Quad CPU Q8200 @ 2.33GHz	148GB	Ubuntu 16.04.6 LTS	KVM e QEMU	1 e 2
2	Intel(R) Xeon(R) CPU 5140 @2.33GHz	126GB	Kali GNU/Linux Rolling	KVM e QEMU	1 e 2
3	Intel(R) Core(TM) i7-4790 @3.60GHz	500GB	Windows 10	Oracle VirtualBox	1 e 3

Para identificar que seu processador possui suporte a virtualização execute o comando abaixo, como root, se for Linux:

Comando para identificar suporte a virtualização no processador

```
# lscpu | grep Virtualization
```

Oracle VirtualBox

O Oracle VirtualBox é voltado para virtualização de plataforma x86 e AMD64/Intel64. Pode ser instalado em diversos sistemas operacionais. A lista dos sistemas operacionais suportados e o download pode ser feito em *https://www.virtualbox.org/wiki/Downloads*.

Para instalar o Oracle VirtualBox, no Linux, deve-se saber, previamente, se sua distribuição é suportada. Abaixo exemplos para sistemas baseados em pacotes DEB e RPM. Sugerimos distribuição Linux baseada em RPM o CentOS 7 e como distribuição Linux baseada em DEB o Ubuntu 16.

Para sistemas baseados em pacotes DEB executar:

Exemplo de instalação do Oracle Virtualbox - DEB

```
$ wget -q https://www.virtualbox.org/download/oracle_vbox_2016.asc -O- | sudo apt-key add -
$ wget -q https://www.virtualbox.org/download/oracle_vbox.asc -O- | sudo apt-key add -
$ sudo apt-get update
$ sudo apt-get install virtualbox-6.0
```

Para sistemas baseados em pacotes RPM, como root, executar:

Exemplo de instalação do Oracle Virtualbox - RPM

```
# cd /etc/yum.repos.d
# wget http://download.virtualbox.org/virtualbox/rpm/rhel/virtualbox.repo
# yum install VirtualBox-6.0
```

Exemplos de criação de máquinas virtuais para o Oracle Virtualbox podem ser obtidos em livro, como em ”Getting started with oracle vm virtualbox”[Dash, 2013], além de tutoriais na internet nesta lista:

- Android - https://github.com/carlos-teles/etsg/blob/master/vm/oracle-vm-android.pdf
- Ubuntu - https://github.com/carlos-teles/etsg/blob/master/vm/oracle-vm-ubuntu.pdf
- Windows 10 - https://github.com/carlos-teles/etsg/blob/master/vm/oracle-vm-windows10.pdf

KVM e QEMU

O KVM e o QEMU funcionam em conjunto para virtualização em sistema Linux. Para instalação em sistemas baseados em pacotes RPM, como root, executar:

Exemplo de instalação do KVM e QEMU - RPM

```
# yum install qemu-kvm libvirt libvirt-python libguestfs-tools virt-install qemu-system*
```

Após a instalação, habilite o serviço *libvirtd* e inicie-o com os comandos abaixo como root:

Habilitando serviço libvirtd

```
# systemctl enable libvirtd
# systemctl start libvirtd
```

Para instalação em sistemas baseados em pacotes RPM, como root, executar:

Exemplo de instalação do KVM e QEMU - DEB

```
# apt install qemu-system* qemu-kvm libvirt-bin virtinst bridge-utils cpu-checker
```

Os serviços, como o libvirtd, são habilitados automaticamente.

Para o KVM e QEMU existem diversas formas de se criar máquinas virtuais. Em nosso exemplo, uma das necessidades, era iniciar várias máquinas virtuais dentro da rede de testes dos ativos de TIC. Para esta solução devemos utilizar o conceito de *bridge* (ponte).

Quando iniciamos o KVM e QEMU sem nenhum parâmetro referente a rede, ele inicia com uma configuração padrão, onde se criará uma rede, em que as máquinas virtuais acessam o *host*, a rede e inclusive a internet (caso o *host* também acesse). Mas não acessam as outras máquinas virtuais e nem estarão acessíveis por outras máquinas da rede, ou seja, as máquinas virtuais não se veem na rede e nem os ativos de TIC conectados. Isto fica aquém de nossas necessidades, pois, precisamos criar um ambiente que proporcione algo o mais próximo de um ambiente de rede para que o ativo de TIC em teste não realize que se encontre no mesmo.

Sendo assim, é melhor fazermos uma configuração mais realista, onde a máquina virtual se conecte na rede física, obtendo um IPv4 ou IPv6 do DHCP e se comporte como uma máquina real, além disto proporcionar um melhor desempenho.

Para isso usaremos a configuração TAP e criaremos uma bridge para que as máquinas virtuais se conectem à rede através desta.

O nome da *bridge* será *br0*.

Para verificar se a mesma não existe, execute o comando abaixo para verificar as configurações de rede:

Verificação de configurações de rede

```
# ifconfig

eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
inet 192.168.1.3  netmask 255.255.255.0  broadcast 192.168.1.255
inet6 fe80::2413:f8df:9426:632c  prefixlen 64  scopeid 0x20<link>
ether 20:47:47:ab:52:d3  txqueuelen 1000  (Ethernet)
RX packets 3067  bytes 2912233 (2.9 MB)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 2229  bytes 780577 (780.5 KB)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
device interrupt 20  memory 0xf7c00000-f7c20000

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
inet 127.0.0.1  netmask 255.0.0.0
inet6 ::1  prefixlen 128  scopeid 0x10<host>
loop  txqueuelen 1000  (Loopback Local)
RX packets 589  bytes 45834 (45.8 KB)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 589  bytes 45834 (45.8 KB)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Assim, verificamos que não existe a *bridge* br0 criada no sistema. Para verificar se existe alguma *bridge* no sistema, execute o comando abaixo:

Verificação de existência de *bridge*

```
#brctl show
bridge name bridge id   STP enabled interfaces
```

A saída do comando indica que não existe nenhuma *bridge* criada no sistema. Para criar a *bridge* br0 e verificar sua existência, execute o comando abaixo:

Criação e verificação de *bridge*

```
# brctl addbr br0
# brctl show
bridge name bridge id   STP enabled interfaces
br0   8000.000000000000 no
```

Após as execuções acima devemos acrescentar a interface física eno1 a nossa *bridge* br0 e fazer com que a mesma pegue um endereço do DHCP, para isso, executaremos o seguinte:

Configurção e verificação de *bridge*

```
# brctl addif br0 eno1
# brctl show
bridge name bridge id   STP enabled interfaces
br0   8000.204747ab52d3 no    eno1
# dhclient -v br0
# ifconfig
br0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
inet 192.168.1.101  netmask 255.255.255.0  broadcast 192.168.1.255
inet6 fe80::2247:47ff:feab:52d3  prefixlen 64  scopeid 0x20<link>
ether 20:47:47:ab:52:d3  txqueuelen 1000  (Ethernet)
RX packets 361  bytes 43262 (43.2 KB)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 416  bytes 646930 (646.9 KB)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
inet 192.168.1.3  netmask 255.255.255.0  broadcast 192.168.1.255
inet6 fe80::2413:f8df:9426:632c  prefixlen 64  scopeid 0x20<link>
ether 20:47:47:ab:52:d3  txqueuelen 1000  (Ethernet)
RX packets 9388  bytes 9188783 (9.1 MB)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 7009  bytes 2615262 (2.6 MB)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
device interrupt 20  memory 0xf7c00000-f7c20000

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
inet 127.0.0.1  netmask 255.0.0.0
inet6 ::1  prefixlen 128  scopeid 0x10<host>
loop  txqueuelen 1000  (Loopback Local)
RX packets 755  bytes 60521 (60.5 KB)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 755  bytes 60521 (60.5 KB)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Com a configuração acima criada, já a *bridge* criada e a interface física associada a mesma. Esta configuração fará que ao acrescentarmos uma nova interface, neste caso lógica, para as máquinas virtuais na *bridge*, teremos conectividade com rede.

Para criar uma máquina virtual leve e de forma rápida sugerimos a instalação do pacote *arm_now*.

O *arm_now* é uma ferramenta de virtualização que se utiliza como base o QEMU para uma configuração instantânea de máquinas virtuais [nongiach, 2018].

Para instalação *arm_now* siga os seguintes passos:

Instalação do arm_now

```
# mkdir -p /usr/local/armmow
# cd /usr/local/armmow
# virtualenv -p python3.6 arm-now
# source bin/activate
# pip install https://github.com/nongiach/arm_now/archive/master.zip --upgrade
```

Após a instalação, execute o comando abaixo para verificar se o arm_now foi instalado corretamente:

Verifica arm_now e lista plataformas disponíveis

```
# arm_now list
aarch64
armv5-eabi
armv6-eabihf
armv7-eabihf
m68k-coldfire
microblazebe
microblazeel
mips32
mips32el
mips32r5el
mips32r6el
mips64-n32
mips64el-n32
nios2
powerpc64-e5500
powerpc64-power8
powerpc64le-power8
sh-sh4
x86-64-core-i7
x86-core2
x86-i686
xtensa-lx60
```

Com isso, poderemos instanciar máquinas virtuais com as arquiteturas acima, em nossa rede, de forma que tenhamos tantas quantas possíveis em nosso servidor de virtualização. Para cada máquina virtual deve-se criar um TAPX, onde X é a numeração, e associar o mesmo a *bridge*. Assim abaixo o exemplo de criação do TAP1, associação a bridge *br0*, geração de *MAC Address* aleatório e instancia da máquina virtual do tipo armv5-eabi.

Máquina virtual com arm_now

```
# cd /usr/local/armnow/arm-now
# tunctl -t tap1 -u `whoami`
Set 'tap1' persistent and owned by uid 0
# ifconfig tap1
tap1: flags=4098<BROADCAST,MULTICAST>  mtu 1500
ether 6e:ce:3a:fa:08:a1  txqueuelen 1000  (Ethernet)
RX packets 0  bytes 0 (0.0 B)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 0  bytes 0 (0.0 B)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

# brctl addif br0 tap1
# ip link set up dev tap1
# ifconfig tap1
tap1: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
ether 6e:ce:3a:fa:08:a1  txqueuelen 1000  (Ethernet)
RX packets 0  bytes 0 (0.0 B)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 0  bytes 0 (0.0 B)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

# brctl show
bridge name bridge id   STP enabled interfaces
br0   8000.204747ab52d3 no    eno1
tap1

# printf 'DE:AD:BE:EF:\%02X:\%02X\n' $((RANDOM\%256)) $((RANDOM\%256))
DE:AD:BE:EF:A9:26
# arm_now start armv5-eabi --add-qemu-options="-net nic,model=lan9118,macaddr=DE:AD:BE:EF:A9:26
   -net tap,ifname=tap1,script=no,downscript=no"
...
press ctrl+] to kill qemu

Welcome to arm_now
buildroot login:
```

Caso apareça o prompt acima, basta digitar 'root' e depois enter e a máquina virtual já está ligada. Entretanto, ainda é necessário que a máquina virtual seja conectada à rede. Na máquina virtual, execute os comandos:

Configuração de rede da máquina virtual com arm_now

```
# route del default gw 10.0.2.2
# ifconfig eth0 192.168.1.64
# route add default gw 192.168.1.1
# vi /etc/resolv.conf
# opkg install isc-dhcp-client-ipv4
# dhclient
# ifconfig
eth0      Link encap:Ethernet  HWaddr DE:AD:BE:EF:A9:26
inet addr:192.168.1.225  Bcast:192.168.1.255  Mask:255.255.255.0
UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
RX packets:541 errors:0 dropped:2 overruns:0 frame:0
TX packets:405 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:719261 (702.4 KiB)  TX bytes:26729 (26.1 KiB)
Interrupt:31
```

Este procedimento somente é necessário uma vez, pois, no próximo *boot* da máquina virtual a mesma virá configurada corretamente.