Universidade de São Paulo
Instituto de Matemática e Estatística
Bacharelado em Ciência da Computação

Análise de Desempenho de Computadores de Baixo Custo em um Sistema de Detecção de Intrusão

Lucas Seiki Oshiro

Monografia Final

mac 499 — Trabalho de Formatura Supervisionado

Supervisor: Prof. Dr. Daniel Macêdo Batista

São Paulo
5 de dezembro de 2019

Análise de Desempenho de Computadores de Baixo Custo em um Sistema de Detecção de Intrusão

Lucas Seiki Oshiro

Esta é a versão original da monografia
elaborada pelo candidato Lucas Seiki Oshiro,
tal como submetida à Comissão Julgadora.

Agradecimentos

Agradeço ao professor Daniel Batista, pelo apoio durante os dois anos em que foi meu orientador; a Bruno Aricó, Marcelo Schimitt, Guilherme Souza e Éderson Cássio pelo conhecimento de suma importância para realização deste trabalho; à minha família e aos meus amigos do IME-USP Matheus, Felipe, Bárbara, Rodrigo, Marcos, Ângelo, Gabriel e Raphael pelo apoio que me deram durante a graduação, sobretudo neste ano.

Resumo

Lucas Seiki Oshiro. **Análise de Desempenho de Computadores de Baixo Custo em um Sistema de Detecção de Intrusão**. Monografia (Bacharelado). Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2019.

O baixo custo e o baixo consumo de energia de computadores de placa única (SBCs) em Internet das Coisas levaram ao seu uso em outras aplicações, como na construção de *clusters*. Este trabalho estuda a viabilidade do uso desse tipo de equipamento em um já existente sistema de detecção de ameaças em servidores de aplicações *web*. Através de medições de seu desempenho em diferentes aspectos que possam representar gargalos, comprova-se que é viável e energetamente eficiente o uso de SBCs para execução de partes desse sistema.

Palavras-chave: Computador de placa única. Detecção de ameaças. Desempenho.

Abstract

Lucas Seiki Oshiro. **Análise de Desempenho de Computadores de Baixo Custo em um Sistema de Detecção de Intrusão**. Capstone Project Report (Bachelor). Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2019.

The low cost and the low energy consumption of single-board computers (SBCs) in Internet of Things led to its use in other applications, such as cluster construction. This paper studies the feasibility of using this kind of equipment in an existing threat detection system in web application servers. Through measurements of its performance on different aspects that might be bottlenecks, it is proved that it is feasible and energy-efficient to use SBCs for executing parts of that system.

Keywords: Single-board computer. Threat detection. Performance.

Lista de Abreviaturas

API	*Application Programming Interface*
CPU	*Central Processing Unit*
DNS	*Domain Naming System*
DVWA	*Damn Vulnerable Web Application*
GPIO	*General Purpose Input/Output*
HTML	*Hypertext Markup Language*
HTTP	*Hypertext Transfer Protocol*
LAMP	Linux, Apache, MySQL e PHP
I2C	*Inter-Integrated Circuit*
IP	*Internet Protocol*
RAM	*Random Access Memory*
SGBD	Sistema Gerenciador de Banco de Dados
SQL	*Structured Query Language*
VM	*Virtual Machine*
XSS	*Cross-Site Scripting*

Lista de Figuras

Sumário

Capítulo 1

Introdução

No projeto **GT-BIS – Mecanismos para Análise de Big Data em Segurança da Informação** GT-BIS (2018), foi desenvolvido um sistema para a detecção de ataques em sistemas *web*, tendo como base os *logs* gerados pelos servidores (o servidor *web* Apache e o SGBD MySQL), analisando-os com técnicas de aprendizado de máquina Oshiro e Batista (2019). Uma das sugestões de trabalhos futuros desse projeto foi a análise da viabilidade de utilizar computadores de baixo custo para executar os componentes de software do sistema.

Computadores placa única e de baixo custo, como a Raspberry Pi, obtiveram grande sucesso em aplicações de Internet das Coisas. Como consequência, vários projetos têm buscado usar tais máquinas para outras aplicações, como a conexão de várias unidades para a construção de um *cluster* de baixo custo Pahl *et al.*, 2016. O uso de computadores da placa única poderia ser, portanto, benéfico para utilização do sistema desenvolvido dentro do sistema GT-BIS, caso seja viável Oshiro e Batista (2019).

Este trabalho de conclusão de curso apresenta resultados de uma análise de desempenho de uma Raspberry Pi 3 Model B sendo usada dentro do sistema do projeto GT-BIS, para estudar a viabilidade do uso desse equipamento nesse contexto.

1.1 Objetivo

O objetivo deste trabalho é avaliar a viabilidade do uso de computadores de placa única de baixo custo (no caso, uma Raspberry Pi) em um sistema de detecção de ameaças (no caso, o sistema desenvolvido no projeto GT-BIS). A avaliação toma como critério o comportamento da Raspberry Pi quando nela é executado partes dos sistema existente.

1.2 Resultados alcançados

Este trabalho de conclusão de curso é uma sequência do trabalho de Iniciação Científica iniciado em 2018, cujos resultados foram relatados no artigo Análise Preliminar da Detecção de Ataques Ofuscados e do Uso de Hardware de Baixo Custo em um Sistema para Detecção

de Ameaças Oshiro e Batista (2019), dos mesmos autores, ganhador de menção honrosa no II Workshop de Trabalhos de Iniciação Científica e Graduação, do Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos em 2019.

Para o artigo supracitado, foram feitos ataques de injeção de SQL e XSS à aplicação *web* DVWA. O DVWA é um serviço *web* propositalmente vulnerável para estudos de segurança, e que é executado sobre uma pilha LAMP (Linux, Apache, MySQL e PHP). Os *logs* do servidor *web* Apache 2.0 e do SGBD MySQL 14.14 usados para a execução do DVWA foram capturados e analisados, para a obtenção de evidências da ocorrência dos ataques.

Também nesse trabalho foram feitas medições da vazão de rede da Raspberry Pi, e da temperatura do processador da placa. Para simular um ambiente real, a placa foi estressada com o comando `stress` com diferentes níveis de uso de CPU. Nessas medições, foi constatado que a vazão de rede não se altera com o maior uso de CPU. Também foi observado o crescimento da temperatura conforme o o uso de CPU crescia, além do rápido resfriamento quando o estresse era cessado.

Em adição aos resultados relatados no artigo, neste trabalho de formatura foram feitas medições de rede, CPU, memória, disco, energia e temperatura na Raspberry Pi, com ela agindo como um nó do sistema, quando ele recebe uma grande quantidade de *logs* dos servidores *web*.

1.3 Organização da monografia

Os próximos capítulos desta monografia estão divididos da seguinte forma:

- Capítulo 2: Fundamentação teórica sobre Internet, estado-da-arte no tópico do TCC e trabalhos relacionados;
- Capítulo 3: Conceitos básicos sobre virtualização, ataques cibernéticos e descrição das ferramentas de software e hardware utilizadas;
- Capítulo 4: Descrição do ambiente montado para a realização de experimentos;
- Capítulo 5: Metodologia e resultados dos experimentos;
- Capítulo 6: Conclusões;
- Capítulo 7: Apreciação pessoal e crítica.

Capítulo 2

Fundamentação

Durante a década de 1960, o uso dos computadores tornava-se cada vez mais importante. Era natural, portanto, considerar interligá-los para que eles pudessem ser compartilhados entre usuários distintos e em locais geograficamente diferentes Kurose e Ross (2003). Neste cenário, Lawrence Roberts, um dos líderes do programa de ciência de computadores na ARPA (Advanced Research Projects Agency) publicou um plano para a criação da ARPAnet, a primeira rede baseada em comutação de pacotes, e que viria a ser um ancestral da Internet.

Em meados da década de 1970, a ARPAnet já era composta por aproximadamente 15 nós. Para se conectar a uma máquina dela, era preciso estar conectado também à ARPAnet. Outras redes também surgiram nesse período, como a ALOHAnet, no Havaí, e a Telenet, uma rede comercial. Também nesse período, Vinton Cerf e Robert Kahn desenvolveram o trabalho pioneiro na interconexão de redes, ou seja, uma rede de redes, levando o nome de "internetting". No final da década de 1970, os três protocolos-chave da Internet (TCP, UDP e IP) já estavam em suas primeiras versões, e a ARPAnet já contava com aproximadamente 200 máquinas Kurose e Ross (2003).

Durante a década de 1980, a ARPAnet adotou os protocolos TCP e IP como padrão, e foi desenvolvido o sistema de nomeação de domínios (DNS), para mapear nomes facilmente legíveis por humanos (como `ime.usp.br`) para endereços IP. Neste período, surgiu a Minitel, um plano francês para levar a rede para todos os lares do país, provendo serviços como *home-banking* e listas telefônicas. Em meados da década de 1990, o Minitel estava presente em vinte por cento dos lares franceses Kurose e Ross (2003).

Durante a década de 1990, a ARPAnet foi descontinuada, e a *World Wide Web* foi criada por Tim Berners-Lee no Cern (Conseil Européen pour la Recherche Nucléaire), juntamente com o desenvolvimento das primeiras versões do HTML, do HTTP, de um servidor web e de um navegador Kurose e Ross (2003).

Capítulo 3

Conceitos

Este capítulo apresenta alguns conceitos importantes para o entendimento do que foi desenvolvido neste TCC. São apresentados conceitos e ferramentas relacionados com virtualização pois o ambiente de experimentação criado foi um ambiente virtual. Ferramentas utilizadas para análise de logs e processamento de grandes volumes de dados também são apresentadas. Além das ferramentas de software, as ferramentas de hardware que foram utilizadas são descritas. O capítulo também apresenta os tipos de ataques contra aplicações web detectados pelo sistema de detecção que serviu de base para este TCC. O capítulo encerra com alguns conceitos de redes de computadores necessários para a compreensão da topologia criada para interconectar todos os elementos do sistema.

3.1 Software

3.1.1 Hipervisor Xen

O hipervisor é um processo que gerencia máquinas virtuais, permitindo que uma máquina física as suporte, compartilhando com elas seus recursos físicos VMWare (2019). Existem diversos hipervisores atualmente. No sistema criado no projeto GT-BIS, foi escolhido o hipervisor Xen para gerenciar as máquinas virtuais. O sistema foi criado com base em máquinas virtuais para facilitar que o mesmo escalasse facilmente em função da carga de trabalho.

O Xen é um hipervisor de tipo 1, o único hipervisor *open-source* desse tipo Xen (2019). Hipervisores de tipo 1, também chamados de hipervisores nativos, são executados diretamente sobre o hardware, ao contrário dos hipervisores de tipo 2, que são executados sobre um sistema operacional hospedeiro Desai *et al.* (2013). Exemplos de hipervisores de tipo 2 são o QEMU e o VirtualBox.

O funcionamento do hipervisor Xen está ilustrado na Figura 3.1. Cada instância de máquina virtual é chamada de **domínio**. Em um domínio especial chamado de Domínio 0 (ou Dom0) é executado um sistema operacional que provê os drivers necessários para a comunicação com o hardware, além de outros serviços. As outras máquinas virtuais são chamadas de DomU Xen (2019).

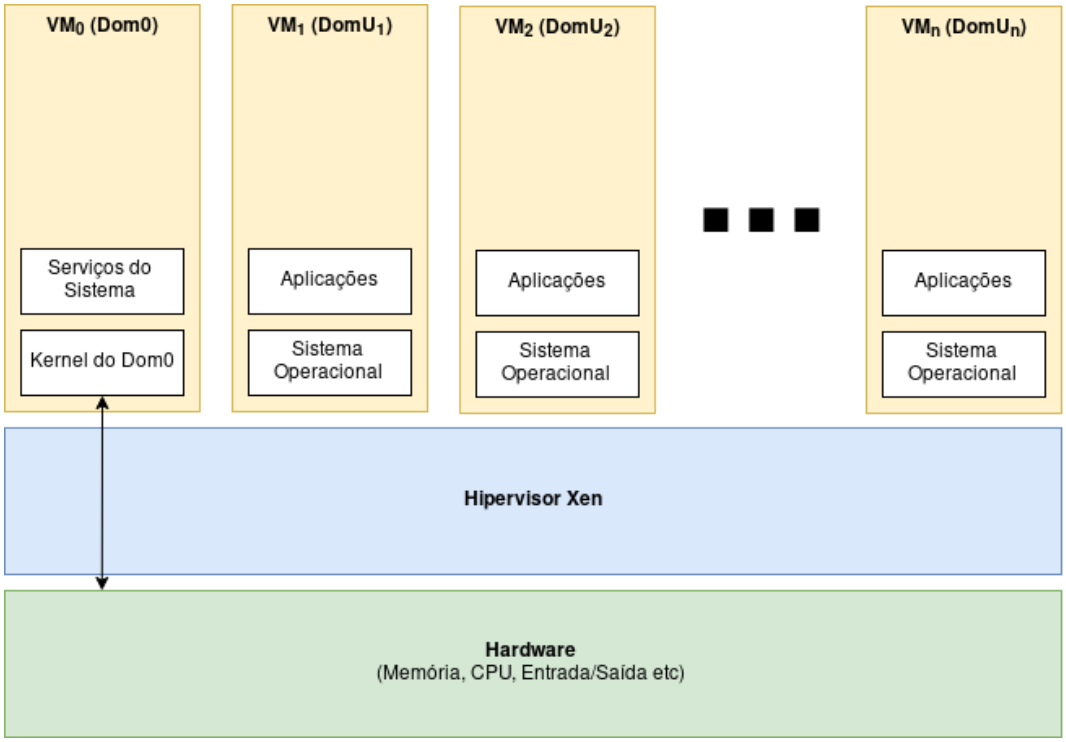


Figura 3.1: *Diagrama do funcionamento do hipervisor Xen, baseado em figura de Xen (2019)*

3.1.2 Ferramentas para análises de *logs* e processamento de grandes volumes de dados

Filebeat

O Filebeat é uma ferramenta que age como um monitor de arquivos de logs, como por exemplo, os logs de servidores web e do sistema operacional. Os dados capturados são centralizados, para serem enviados para outros serviços, como por exemplo, o Logstash Elastic (2019[a]).

Logstash

O Logstash é um motor de coleta de dados, com a capacidade de obter dados de diversas fontes e normalizar para destinos escolhidos Elastic (2019[b]). Os dados agregados podem ser enviados, por exemplo, para o Elasticsearch.

Elasticsearch

O Elasticsarch é um mecanismo de busca e análise de dados, que podem vir de várias fontes, como logs. Os dados são indexados de forma que consultas complexas possam ser feitas neles Elastic (2019[c]).

Kafka

O Apache Kafka é uma plataforma distribuída de fluxo de dados Kafka (2019), coordenando-os e notificando as aplicações conectadas a respeito dos novo registros

GT-BIS (2018).

Spark

O Apache Spark é um arcabouço para o processamento paralelo e massivo de grandes quantidades de dados, com APIs que provém desde processamento de dados básicas até recursos mais avançados como aprendizado de máquina GT-BIS (2018).

3.2 Hardware

3.2.1 Computador de placa única

Computadores de placa única, também conhecidos pelo acrônimo SBC (*single board computers*) são placas de circuito impresso com processador, memória, e entrada e saída de dados Ortmeyer (2014), ou seja, são computadores completos montados em apenas uma placa.

Alguns exemplos de computadores de placa única são a Raspberry Pi Raspberry Pi Foundation (2018) e a Labrador Caninos Loucos (2019).

As principais vantagens destes computadores são seu baixo custo e seu baixo consumo energético. Apesar deles não serem equivalentes a computadores de última linha, as suas configurações permitem a utilização dos mesmos para diversos fins, desde emuladores de vídeo game e *streaming boxes* até desktops simples com navegador web e editores de texto. Outro tipo de utilidade, e que tem relação com este TCC, é como nó de processamento de clusters.

Neste TCC foi utilizado o Raspbery Pi. A Figura 3.2 apresenta uma foto da unidade do Raspbery Pi que foi utilizada. É possível observar a alimentação (cabo branco), a conexão à rede (cabo azul) e a conexão I2C com o sensor de corrente elétrica (fios amarelo e laranja).

3.2.2 Sensor de corrente elétrica

Um sensor de corrente elétrica realiza medições de corrente quando ligado em série ao circuito cujo valor de corrente deseja-se obter.

Um exemplo desse tipo de sensor é o INA219, capaz de medir correntes de até 5A em circuitos de tensão de até 26V. A leitura do INA219 é feita através de uma interface I2C Texas Instruments, 2012. A Figura 3.3 mostra como é feita a ligação do sensor INA219 ao circuito em que a corrente será medida e ao dispositivo que fará sua leitura.

A Figura 3.4 apresenta uma foto do sensor INA219 que foi utilizado para medir a corrente elétrica da placa utilizada no projeto. Este sensor foi necessário para que fosse possível monitorar o consumo de energia da placa em diversas situações de funcionamento do sistema com a placa desempenhando diversos papeis.

O cálculo da potência em um circuito de corrente contínua é feito através da equação $Pot = U * i$, sendo Pot a potência, U a tensão e i a corrente elétrica.

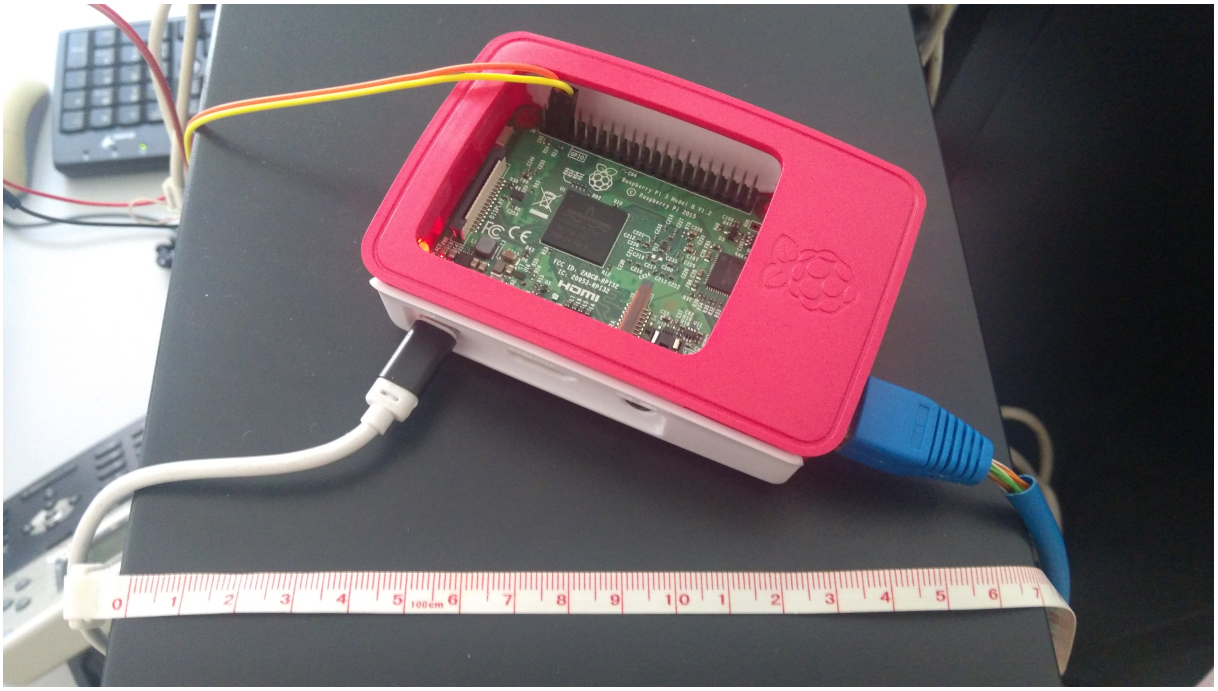


Figura 3.2: *Raspberry Pi*

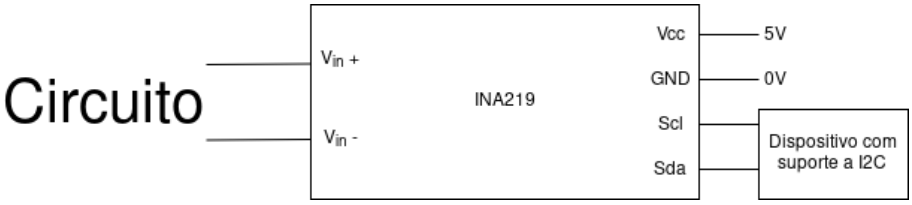


Figura 3.3: *Conexão do sensor INA219*

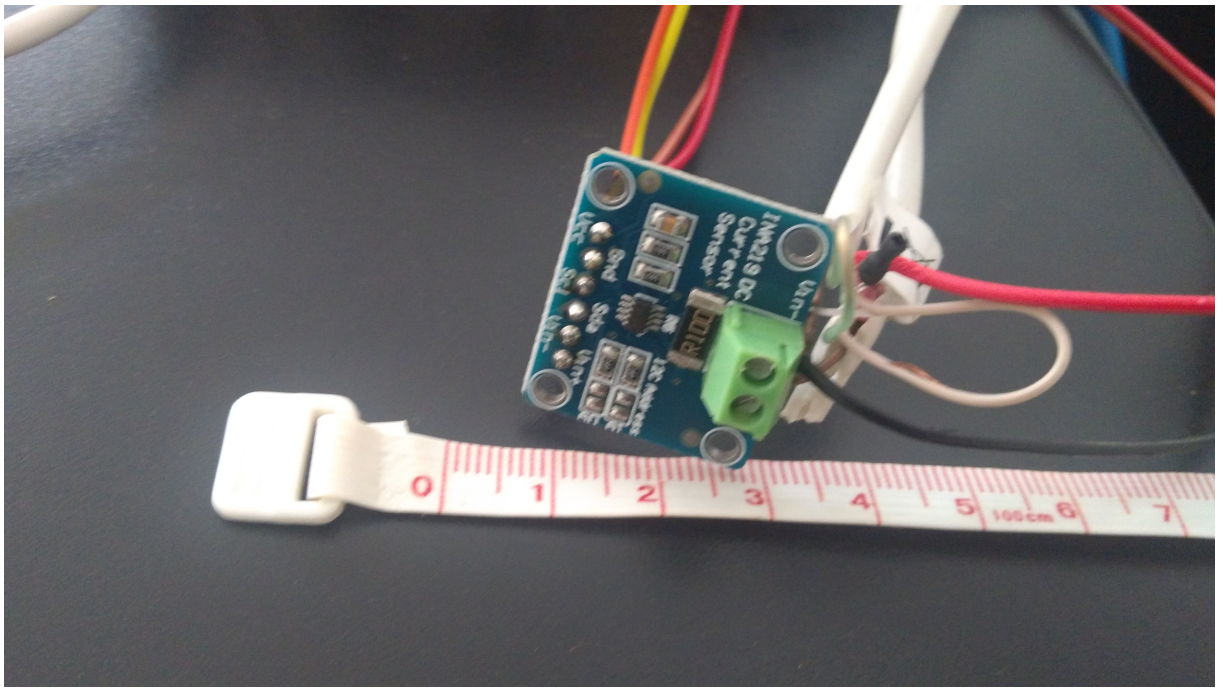


Figura 3.4: *Sensor INA219*

3.3 Ataques contra aplicações web

Ataques contra redes de computadores ocorrem por conta de vulnerabilidades em diversas camadas da arquitetura Internet. Especificamente no caso de aplicações web, aquelas projetadas para serem acessadas principalmente por meio de um navegador web tendo o protocolo HTTP como base, falhas podem levar à negação do serviço, ao acesso indevido a informações ou mesmo à execução de comandos arbitrários do lado do servidor.

Pelo fato do protocolo HTTP ser um protocolo sem estado, boa parte das aplicações web requer a utilização de algum Sistema Gerenciador de Banco de Dados (SGBD). Tais sistemas costumam ser alvos de atacantes em busca de falhas que possam afetar a infraestrutura de uma aplicação web. Outro alvos para a exploração de falhas é a construção de URLs maliciosas.

3.3.1 Injeção de SQL

Um ataque de injeção de SQL consiste de uma inserção de um código SQL através de uma entrada de dados que é aberta para uma aplicação. Uma injeção de SQL bem sucedida consegue fazer consultas no banco de dados, além de poder fazer modificações e alterar suas permissões administrativas OWASP (2019[a]). No caso de aplicações web, esse ataque pode ser utilizado por exemplo adicionando caracteres especiais na URL do sistema web sendo acessado ou em algum campo de um formulário desse sistema.

3.3.2 XSS

Um ataque de XSS (*Cross-Site Scripting*) consiste de uma inserção de código malicioso em uma página *web* benigna OWASP (2019[b]). Existem dois tipos principais de XSS:

- XSS Persistido: O código malicioso é armazenado no servidor, como no banco de dados de um fórum ou um campo de comentários, por exemplo. Quando o conteúdo é enviado para a vítima, o código malicioso é enviado junto OWASP (2019[b]).
- XSS Refletido: O código malicioso não é armazenado no servidor. O código malicioso é inserido na URL, que quando aberta, injeta tal código dentro da página OWASP (2019[b]).

Capítulo 4

Ambiente

Este capítulo descreve a configuração do ambiente em que o trabalho foi desenvolvido. Alguns elementos apresentados foram adquiridos e configurados durante o projeto GT-BIS, porém, a configuração atual foi feita neste trabalho.

O ambiente aqui descrito está fisicamente montado na sala 228 do Centro de Competência em Software Livre do Instituto de Matemática e Estatística da USP. Ele é composto fisicamente por duas máquinas principais, um computador de placa única, e um computador auxiliar.

O objetivo do ambiente era simular ataques contra serviços web e avaliar o desempenho do sistema de detecção de ataques, tanto em termos de uso de recursos das máquinas (CPU, rede e memória) quanto em termos de falsos positivos e de falsos negativos. Em particular, neste TCC, o objetivo era avaliar o desempenho do computador de placa única atuando como um nó do sistema com diferentes funções.

4.1 Máquinas físicas

4.1.1 Máquinas principais

As duas máquinas físicas utilizadas têm a seguinte configuração:

- prestigio
  - Sistema Operacional (Dom0): CentOS Linux 7 (Core) x86_64
  - *Kernel*: 4.9.48
  - Processador: Intel i7-6700K, 8 núcleos, 4,0 GHz
  - Memória RAM: 622 MiB (Dom0)
  - Hipervisor: Xen 4.6.6-3.el7
- talento
  - Sistema Operacional(Dom 0): CentOS Linux 7 (Core) x86_64

  - *Kernel*: 4.9.127
  - Processador: Intel i7-6700K, 8 núcleos, 4,0 GHz
  - Memória RAM: 2944 MiB (Dom0)
  - Hipervisor: Xen 4.8.4.43.ge52ec4b7

Os valores indicados para o tamanho da memória indicam apenas o que é usado exclusivamente para o sistema instalado nas máquinas físicas. As máquinas virtuais têm suas memórias isoladas pelo Xen.

4.1.2 Computador de placa única

O computador de placa única utilizado tem a seguinte configuração:

- Modelo: Raspberry Pi 3 Model B
- Sistema Operacional: Raspbian 9.4 stretch
- *Kernel*: Linux 4.14.50
- Processador: ARMv7 rev 4 (v7l), 4 núcleos, 1,2GHz
- Memória RAM: 927 MiB

O sensor INA219 está conectado aos pinos GPIO desta máquina, além de estar conectados em série a sua alimentação, para fins de medição da corrente elétrica.

4.1.3 Máquina auxiliar

A máquina auxiliar é um computador portátil de uso pessoal, com as seguintes configurações:

- Sistema Operacional: Manjaro Linux x86_64
- Modelo: Dell Inspiron 3542
- *Kernel*: 4.19.66
- Processador: Intel i5-4210U (4 núcleos), 2,7 GHz
- Placa de vídeo: NVIDIA GeForce 610M/710M/810M/820M / GT 620M/625M/630M/720M
- Memória RAM: 7880 MiB
- Hipervisor: VirtualBox 6.0.10

4.2 Máquinas virtuais

O sistema existente era composto por diversas máquinas virtuais.

As máquinas virtuais executadas sobre a máquina física prestigio são os componentes do sistema de detecção de ameaça. São elas:

- elasticsearch
- kafka-spark
- logstash

As máquinas virtuais executadas sobre a máquina física talento são os servidores web vítimas de ataques e que enviam seus logs para o sistema. Elas são as seguintes:

- victim01
- victim02
- victim03
- victim04
- victim05

As máquinas virtuais executadas sobre a talento foram configuradas para usarem 2048 MiB da memória RAM, e executam a distribuição Linux Ubuntu 17.04, com *kernel* 4.10.0.

Além das máquinas virtuais listadas anteriormente, que foram criadas no projeto GT-BIS, também foi criada uma máquina virtual para a simulação de ataques e para executar o *benchmarking* das diferentes partes do sistema.

Essa máquina virtual é executada sobre a máquina física auxiliar. Foram reservados 3072 MiB para ela. O sistema operacional utilizado foi o Kali Linux Rolling Release, com *kernel* Linux 4.16.

4.3 Rede

Originalmente no projeto GT-BIS, as máquinas físicas (prestigio e talento) estavam ligadas à Rede IME, cada uma com um endereço IP estático (143.107.44.125 e 143.107.44.126, respectivamente) e um domínio na Internet (prestigio.ime.usp.br e talento.ime.usp.br, respectivamente). Elas também estavam conectadas a uma rede virtual, sendo seus respectivos endereços IP 10.254.125.1 e 10.254.126.1.

As máquinas virtuais executadas sobre a prestigio estavam conectadas em modo *bridge* à rede virtual. Isto significa que cada máquina virtual era exposta individualmente nessa rede, como se elas fossem várias máquinas físicas diferentes.

As máquinas virtuais executadas sobre a talento estavam conectadas a uma instância do Open vSwitch, um *switch* virtual. Um *switch* é um dispositivo que realiza a comunicação entre máquinas em uma rede, permitindo que um pacote seja enviado apenas da origem ao destino. O *switch* virtual Open vSwitch é um *switch* implementado em software, e que permite configurações automatizadas.

As máquinas virtuais executadas sobre a talento não eram acessíveis pelas máquinas virtuais executadas sobre a prestigio. Essa configuração de rede está ilustrada na Figura 4.1.

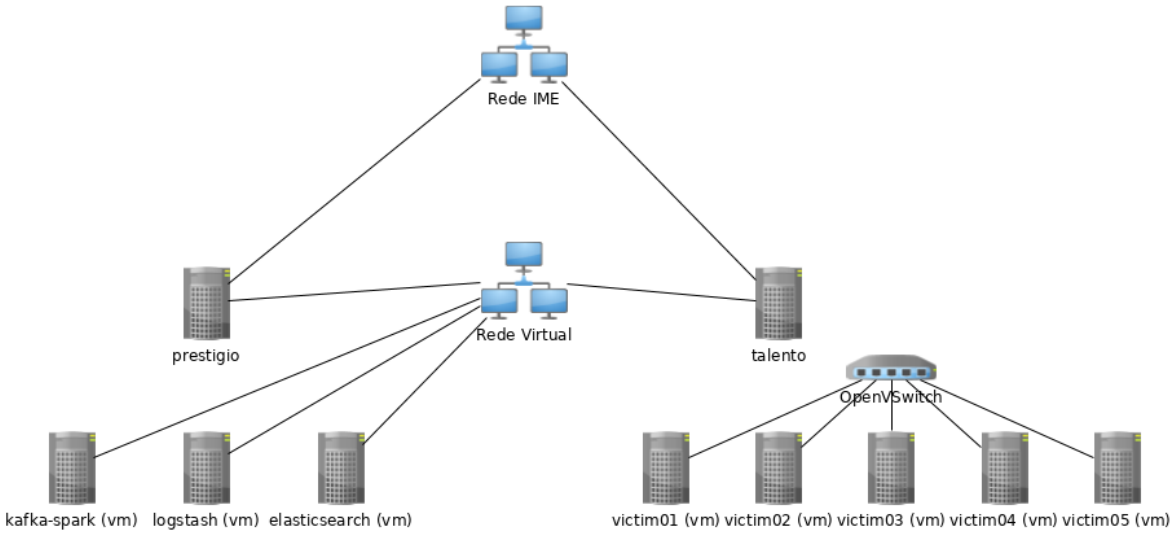


Figura 4.1: *Configuração anterior da rede.*

4.3.1 Configuração atual

A configuração anterior, explicada anteriormente, não permitia uma comunicação entre as vítimas que geravam os logs com as máquinas que os processariam. Optou-se, portanto, por deixar todas as máquinas físicas e virtuais em uma mesma rede.

As máquinas prestigio e talento eram conectadas fisicamente apenas na Rede IME, e uma nova máquina que precisasse ser adicionada a esse sistema precisaria estar conectada a essa rede. A Rede IME é uma rede utilizada por todo o Instituto de Matemática e Estatística, portanto, o grande uso de rede necessitado poderia comprometer o uso dessa rede por terceiros. Além disso, uma nova máquina que precisasse ser adicionada no ambiente necessitaria ser cadastrada pelos administradores da Rede IME. Optou-se, por tais motivos, configurar uma rede interna, de forma que a Rede IME só seria usada para fornecer acesso remoto ao ambiente e para fornecer acesso à Internet para as máquinas.

Um roteador *wi-fi* foi colocado na rede para agir como um *switch*. O endereço IP externo do roteador foi configurado como o usado pela máquina prestigio (143.107.44.125), herdando, portanto, o domínio (prestigio.ime.usp.br) que identificava essa máquina.

As duas máquinas físicas foram desconectadas da Rede IME e foram conectadas no roteador, de forma que ambas estivessem apenas em uma rede local. Os endereços IP das máquinas físicas na rede local foi configurado para ser os que eram utilizados na rede virtual.

Como nessa configuração as máquinas físicas não são acessíveis diretamente por máquinas de fora da rede interna, foi feito um encaminhamento de uma porta do roteador para a porta utilizada pelo ssh na máquina prestígio. Desta forma, foi garantido o acesso ssh para essa máquina, sendo que através dela quaisquer outras que estejam conectadas na rede interna também ficam acessíveis.

As placas de rede das máquinas virtuais foram colocadas em modo *bridge*. Desta forma,

todas as máquinas virtuais se encontram na mesma rede que as máquinas físicas. As máquinas virtuais executadas sobre a prestigio receberam endereços IP estáticos entre 10.254.125.2 e 10.254.125.5, e as máquinas virtuais executadas sobre a talento receberam IPs estáticos 10.254.126.2 até 10.254.126.6.

Além das máquinas existentes, também foi conectada a essa rede a Raspberry Pi adquirida para este trabalho, recebendo o endereço IP 10.254.127.1; também foi conectado a essa rede um computador pessoal, recebendo o endereço IP 10.254.128.1, bem como a máquina virtual com Kali Linux, recebendo o endereço IP 10.254.128.2. Essa configuração de rede está representada graficamente na Figura 4.2.

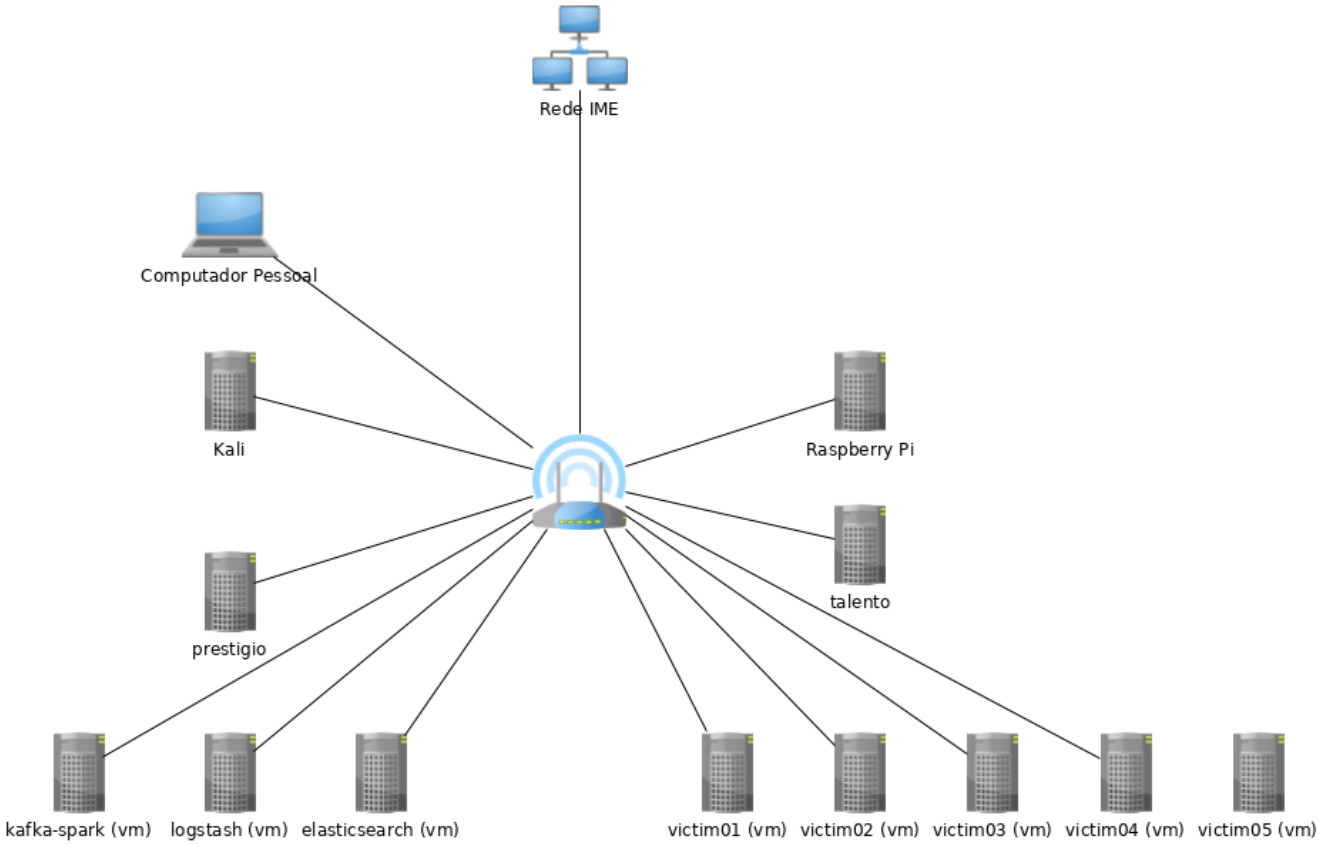


Figura 4.2: *Configuração atual da rede.*

4.4 Fluxo dos dados

O projeto GT-BIS descreve a arquitetura do sistema sistema conforme a figura 4.3. Os “Servidores” descritos nessa figura são as máquinas virtuais que estão na Talento e o “Módulo processamento” são as máquinas virtuais que estão na Prestigio. Os números representam os dados que são passados entre cada componente do sistema, conforme a tabela 4.1 GT-BIS (2018).

Fluxo	Descrição
1	*Logs* de serviços e de sistemas de segurança.
2	Dados normalizados, filtrados e enriquecidos.
3	Fluxos de dados organizados em filas para serem processados.
4	Informações brutas e de dados já processados e comandos de consultas.
5	Dados para serem persistidos ou recuperados.
6	Alertas de ameaças cibernéticas compartilhados pelo sistema central ou parceiros.
7	Dados da interação do administrador com a interface Web.
8	Informações disponíveis nos componentes de Processamento e comandos de consultas.

Tabela 4.1: *Legenda do fluxo de dados, segundo o GT-BIS GT-BIS (2018)*

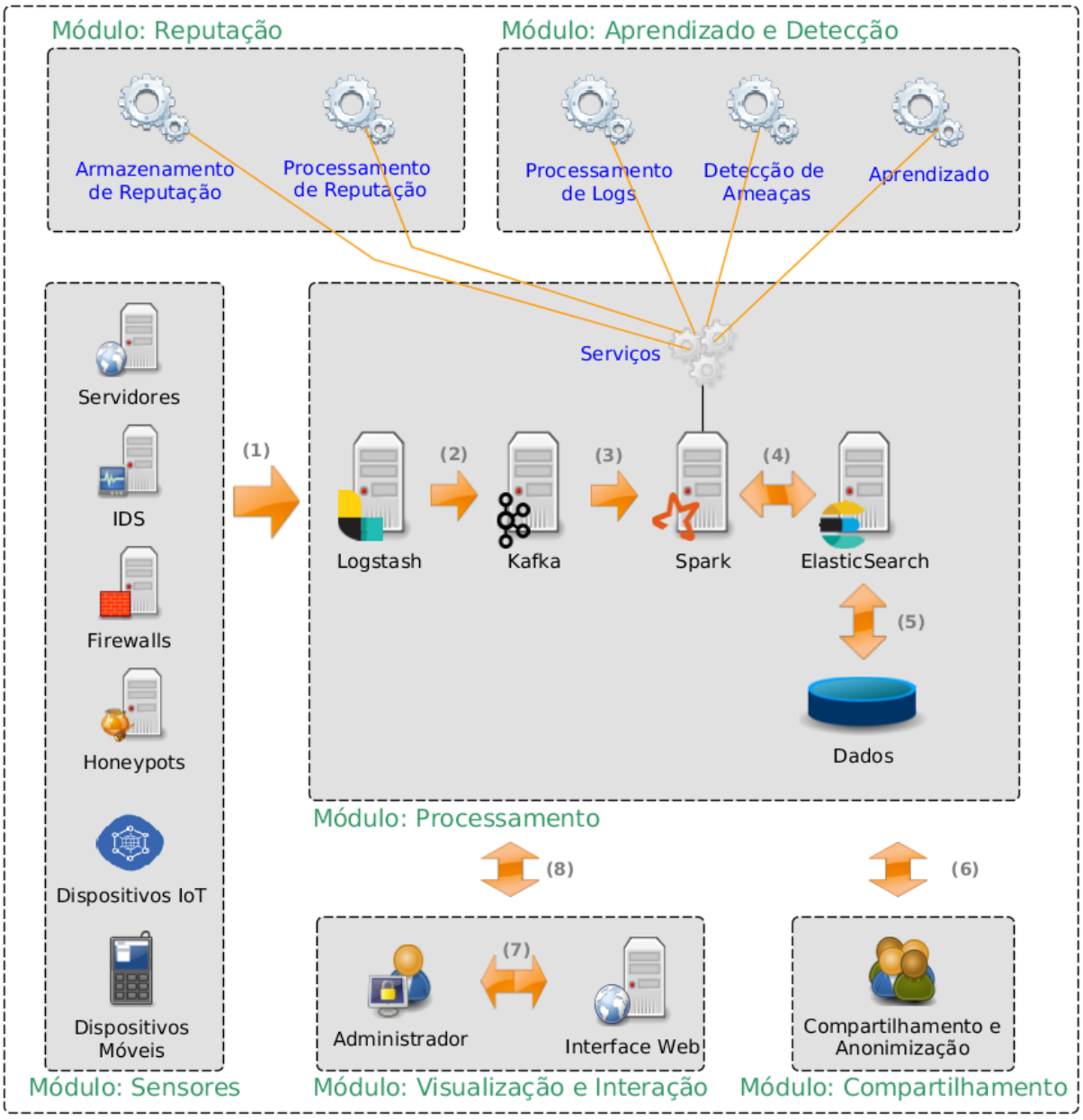


Figura 4.3: *Arquitetura planejada para o sistema desenvolvido no projeto GT-BIS GT-BIS (2018).*

Capítulo 5

Análise de desempenho

5.1 Metodologia

Para fazer a análise de desempenho em cada nó, faz-se as seguintes etapas:

1. Instalação da ferramenta analisada na Raspberry Pi;
2. Início do *script* `raspberry-log` na máquina virtual analisada;
3. Envio de requisições para os servidores *web*;
4. Término da execução do *script* `raspberry-log` na máquina virtual;
5. Configuração do sistema para utilização da Raspberry Pi como nó;
6. Início do *script* `raspberry-log` na Raspberry Pi;
7. Envio de requisições para os servidores *web*;
8. Término da execução do *script* `raspberry-log` na Raspberry Pi;
9. Obtenção das leituras capturadas e sua análise.

O *script* `raspberry-log` foi desenvolvido durante este trabalho. Este *script* captura dados de uso de CPU, memória, rede e disco, e no caso da Raspberry Pi, mede também a temperatura da CPU e potência da placa. Uma descrição desse *script* está no apêndice A.

Os *logs* gerados pelo script `raspberry-log` são posteriormente processados pela biblioteca Pandas. Através dessa biblioteca, e do *software* Jupyter Notebook, é possível fazer observações tendo como base os dados gerados, como por exemplo, o nível de uso da CPU durante os processamento e a variação do uso de energia durante o experimento.

5.2 Experimentos

5.2.1 Controle

O objetivo desse primeiro experimento foi medir o comportamento das máquinas em repouso. Neste primeiro experimento a Raspberry Pi estava apenas com o Logstash instalado, porém, ainda não estava desempenhando nenhum papel dentro do sistema.

Durante aproximadamente 16 horas, o *script* `raspberry-log` foi executado nas máquinas virtuais victim01 e logstash, assim como na Raspberry Pi, enquanto elas estavam em repouso. Os resultados apresentados nesta subseção indicam, portanto, o comportamento das máquinas executando apenas as funções básicas do sistema operacional, além do próprio *script*.

A temperatura média do processador da Raspberry Pi foi de 47.9 °C, com um pequeno desvio padrão de 0.492227 °C, ou seja, a temperatura se manteve bastante estável nesse período.

Nas máquinas victim01, Raspberry Pi e logstash, os usos médios de seus processadores foi de 2.4%, 3.5% e 2.6%, respectivamente e seus desvios padrões foram 2.8%, 1.1% e 2.3%, respectivamente; a quantidade de memória RAM usada foi, em média, de 577 MB, 207 MB e 226 MB, respectivamente, com desvios padrões inferiores a 1 MB.

A escrita em disco se manteve baixa nas três máquinas, na ordem de 3 a 9 KB/s, com alguns picos na ordem de 8 a 16 MB/s, como é possível observar nos gráficos da Figura 5.1. Nesta medição, foi considerada a própria gravação da saída do script no disco.

Apesar de ocorrerem alguns picos de uso de rede, as taxas de *download* e *upload* também se mantiveram baixas, com médias inferiores a 30 *bytes* por segundo com variâncias menores que 300 *bytes*.

A potência da Raspberry Pi durante praticamente todo o perído de repouso se manteve abaixo de 2000mW, como é possível observar no histograma 5.2

5.2.2 Logstash

O objetivo deste experimento é observar o comportamento atual da máquina virtual que executa o logstash, e comparar com o desempenho da Raspberry Pi desempenhando o mesmo papel, possibilitando avaliar a viabilidade de utilizar o computador de placa única como um nó dentre os vários executando o logstash.

Logstash na máquina virtual

O Filebeat da vítima victim01 foi configurado para enviar os *logs* capturados por ele para o Logstash que é executado na máquina virtual dedicada a esse serviço.

O *script* `raspberry-log` foi executado na máquina virtual do Logstash e em uma vítima (victim01). Simultaneamente, na máquina virtual com Kali Linux foi executado o Apache Benchmark através do comando `ab -n 100000 http://10.254.126.2/dvwa`, para que cem mil requisições fossem enviadas para o servidor web da vítima.

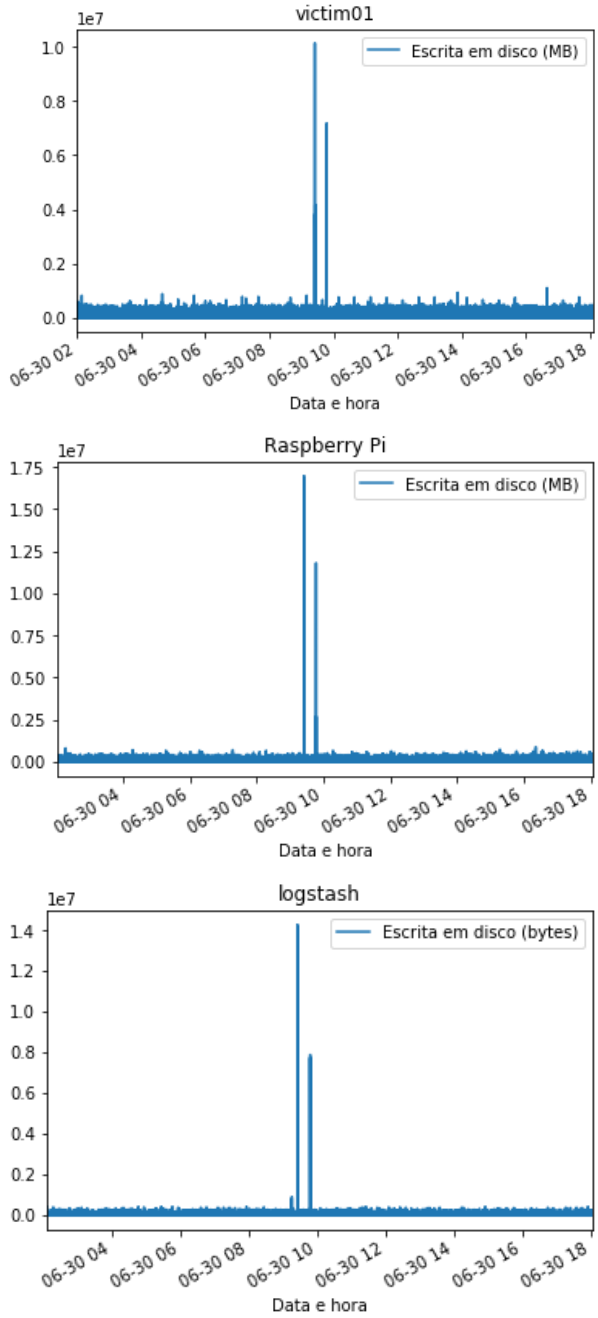


Figura 5.1: *Escrita em disco em repouso*

O grande número de requisições feitos na máquina virtual da vítima gerou um grande volume de *logs* que foram enviados para o Logstash.

Logstash na Raspberry Pi

O Filebeat da vítima victim01 foi configurado para enviar os *logs* capturados para o Logstash que era executado na Raspberry Pi.

De forma análoga ao que foi feito com o Logstash executado na máquina virtual, o *script* `raspberry-log` foi executado em uma vítima e na Raspberry Pi, com a execução

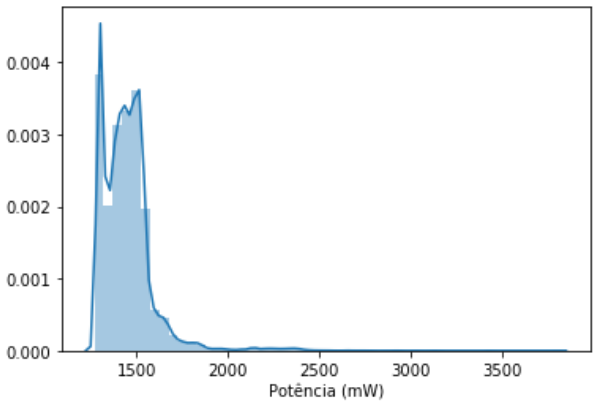


Figura 5.2: *Histograma de potência da Raspberry Pi, em repouso*

simultânea do Apache Benchmark na máquina virtual com Kali Linux através do mesmo comando, para enviar cem mil requisições para o mesmo servidor web. Consequentemente, um grande volume de *logs* foi gerado na vítima, e ele foi enviado para o Logstash.

Resultados

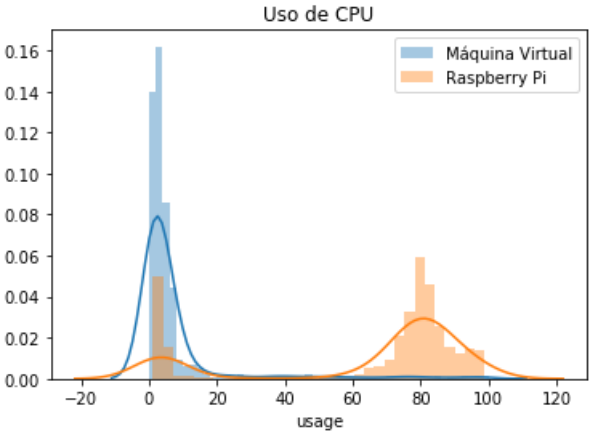
Os usos de CPU pela máquina virtual e pela Raspberry Pi estão representados no histograma da Figura 5.3a. Como no repouso as duas máquinas têm baixo uso de CPU, é possível comparar diretamente o desempenho das duas máquinas: enquanto a máquina virtual consegue executar o Logstash com baixo uso de CPU, a Raspberry Pi utiliza aproximadamente 80% da CPU para desempenhar a mesma tarefa. A Raspberry Pi, portanto, mostra ter maior dificuldade para executar a mesma tarefa que a máquina virtual. Entretanto, essa placa ainda é capaz de realizar o mesmo processamento.

O uso de memória RAM durante o experimento pode ser observado no histograma da Figura 5.4a. Como é possível observar, a Raspberry Pi utiliza aproximadamente 300 MB de memória RAM a menos que a máquina virtual. Considerando os valores obtidos no controle, observa-se que apesar de a Raspberry Pi aparentar ter melhor desempenho nesse caso, os valores se mantiveram próximos aos observados nas máquinas em repouso. Conclui-se que em ambas as máquinas o consumo de memória pelo Logstash não é um problema relevante.

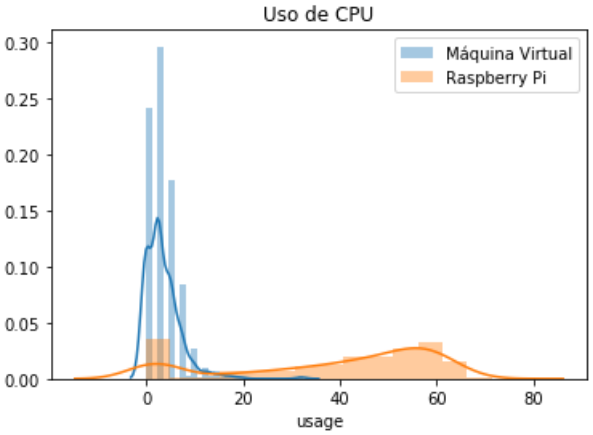
A variação de temperatura do processador da Raspberry Pi durante o experimento pode ser observada no gráfico da Figura 5.5. A temperatura não ultrapassou 80 °C, estando, portanto, dentro do limite operacionais para o processador, que é de até 85°C.

A potência consumida pela Raspberry Pi durante o experimento pode ser observada no histograma da Figura 5.6. Apesar de alguns picos acima de 5000mW, a potência se manteve na maior parte do tempo entre 1500mW e 4000mW.

Esses valores de potência foram bastante baixos para o quanto a placa conseguiu processar. Para efeito de comparação, a potência de uma lâmpada de LED de 800 lúmens, considerado um dos modelos mais econômicos para uso residencial, é de aproximamente 8W, ou seja, o dobro do consumo da Raspberry Pi nos picos durante o processamento.



(a) *Experimento com logs de um servidor*

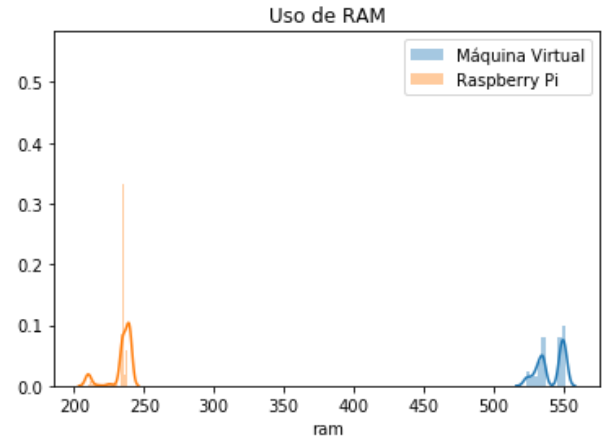


(b) *Experimento com logs de um servidor*

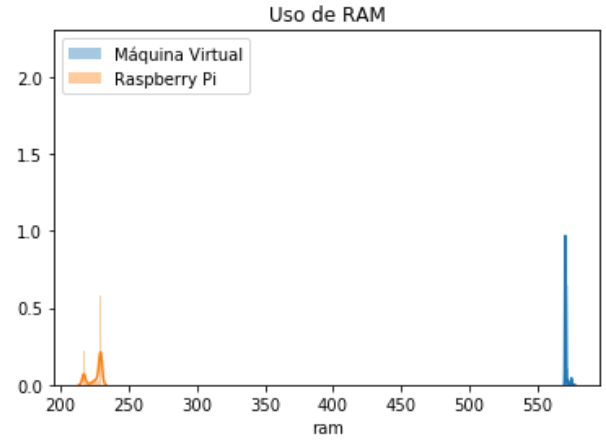
Figura 5.3: *Histogramas da distribuição no tempo do uso de CPU*

5.2.3 Experimentos com mais de uma vítima

A mesma amostragem foi feita com *logs* gerados por 5 máquinas virtuais. Como é possível observar no histograma de uso de CPU na Figura 5.3b, e no histograma de uso de RAM na Figura 5.4b, o comportamento foi próximo ao observado anteriormente, com apenas uma vítima, como é possível observar comparando com os gráficos anteriores.



(a) *Experimento com logs de um servidor*



(b) *Experimento com logs de 5 servidores*

Figura 5.4: *Histogramas da distribuição no tempo do uso de memória, em MB*

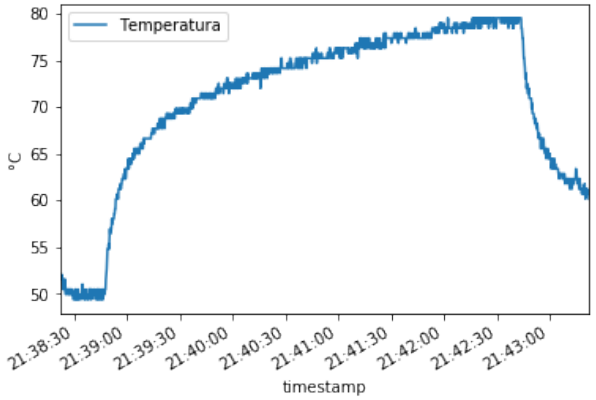


Figura 5.5: *Variação de temperatura da Raspberry Pi*

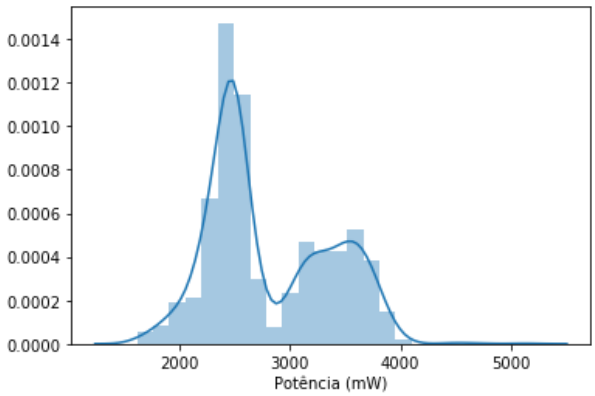


Figura 5.6: *Histograma de potência da Raspberry Pi*

Capítulo 6

Conclusões

A Raspberry Pi mostrou-se capaz de realizar o serviço antes executado pela máquina virtual. Seu desempenho em uso de CPU não foi tão satisfatório quanto as máquinas virtuais, porém, ainda provou que este tipo de computador é viável no sistema de detecção de ameaças.

Alguns componentes do sistema, como o Logstash e o Elasticsearch, podem ser utilizados em ambientes distribuídos. Desta forma, é possível fazer um escalonamento horizontal com outras unidades Raspberry Pi conforme a necessidade.

A potência da Raspberry Pi durante os experimentos foi quase sempre inferior a 4W, bastante reduzido, sendo inferior, por exemplo, a uma lâmpada residencial de LED. Isso mostra que este tipo de equipamento é uma opção viável para ser utilizada em sistemas como o desenvolvido no projeto GT-BIS.

Capítulo 7

Avaliação pessoal e crítica

Gostaria de expressar nesta avaliação pessoal e crítica os pontos bons e ruins do meu trabalho de conclusão de curso, as dificuldades que tive, que aprendi e o que apliquei.

Como este trabalho baseia-se na arquitetura de um projeto anterior, houve uma grande dependência do bom funcionamento da estrutura existente. Tal dependência foi prejudicial, de forma que contratempos envolvendo os equipamentos e a arquitetura de rede levaram a atrasos, à resolução de problemas que não estavam diretamente relacionados com a proposta e ao não cumprimento de todo o trabalho da forma como era desejado.

Alguns dos problemas que ocorreram foram:

1. A rede estava configurada de forma que impedia o funcionamento do sistema como ele fora concebido;
2. A documentação do projeto estava em uma *wiki* hospedada em uma das máquinas virtuais. Era necessário uma conta de usuário para acessá-la, o que eu não tinha, apesar dos membros do projeto GT-BIS indicassem para eu usá-la;
3. O dispositivo de armazenamento usado para uma das partes do sistema estava corrompido, e não havia uma cópia de segurança disponível de seu conteúdo;
4. Uma das máquinas físicas do projeto apresentou problemas em sua inicialização nos últimos meses.

O primeiro problema pôde ser resolvido através de uma nova configuração da rede. Nesse momento, pude aplicar os conceitos expostos na aula de Redes de Computadores em uma situação real. Sendo assim, apesar de não estar previsto para este trabalho, ele foi um grande aprendizado.

O segundo problema foi resolvido com a criação de um usuário manualmente através do Rails, uma vez que eu tinha acesso ao código da aplicação e ao banco de dados.

O terceiro problema não foi possível de resolver, apesar de algumas tentativas de usar ferramentas de checagem de disco para tentar recuperar o sistema de arquivos.

O quarto problema também não foi possível de ser resolvido, pois desde seu início até o presente momento não consegui identificar a causa do problema.

Este trabalho de formatura, portanto, mostrou-se altamente interdisciplinar. Nele apliquei conceitos básicos de elétrica aprendidos no ensino médio; experiência de eletrônica e de sensores adquiridos no grupo de extensão Hardware Livre; gerenciamento de banco de dados e de um sistema *web* feito em Rails aprendidos em Laboratório de Banco de Dados e Técnicas de Programação 2; conhecimentos em redes de computadores aprendidos na disciplina homônima e análise de dados com ferramentas que foram utilizadas em Ciência e Engenharia de Dados. Desta forma, apesar dos diversos problemas que encontrei e de não ter conseguido os objetivos da forma almejada no início do ano, este trabalho foi uma síntese e um exercício da sapiência transmitida a mim ao longo do curso pelos meus professores e colegas do IME-USP.

Apêndice A

Script

O *script* que foi utilizado para a leitura do estado das máquinas foi desenvolvido na linguagem Python 3.5.

Neste apêndice estão transcritos os "sensores" responsáveis pelos seguintes tipos de leitura:

- Corrente e potência
- Uso de CPU
- Temperatura da CPU
- Uso de memória RAM
- Taxas de vazão de rede
- Taxas de acesso ao disco

Cada um dos sensores é feito como uma classe diferente, sem dependência uma de outra. Isto permite que seja possível fazer um tipo de leitura (como por exemplo, o uso de CPU) em um ambiente que não seja possível fazer outro tipo de leitura (como por exemplo, quando não há um sensor INA219 ligado à Raspberry Pi, o que impede o uso do sensor de corrente).

As outras partes do script foram omitidas neste apêndice por serem apenas para interação com o usuário e para a formatação em CSV dos dados capturados. O *script* completo está disponível em https://github.com/lucasoshiro/raspberry-log.

```
1   #!/usr/bin/env python3
2
3   from time import time
4
5   class CurrentSensor:
6       """ Sensor de corrente elétrica. """
7
8       def __init__(self):
9           """ Inicializa o sensor. Só deve ser usado se houver um INA219
10          ligado àGPIO da placa utilizada."""
11
```

```
12            from ina219 import INA219
13            self.ina219 = INA219(0.1)
14            self.ina219.configure()
15
16        def current(self):
17            """ Lê a corrente, em mA. """
18            return self.ina219.current()
19
20        def power(self):
21            """ Lê a potência atual, em mW. Supõe uma tensão de 5V """
22            return self.current() * 5
23
24    class CPUSensor:
25        """ Sensor de porcentagem de uso de CPU. """
26
27        def __init__(self):
28            """ Inicializa o sensor."""
29            from psutil import cpu_percent
30            self._cpu_percent = lambda *_: cpu_percent(*_)
31
32        def usage(self):
33            """ Lê a porcentagem de uso da CPU """
34            return self._cpu_percent(None)
35
36    class TempSensor:
37        """ Sensor de temperatura. Atualmente, compatível apenas com
38        Raspberry Pi"""
39
40        def __init__(self):
41            """ Inicializa o sensor """
42            from gpiozero import CPUTemperature
43            self._cpu_temp = CPUTemperature()
44
45        def temperature(self):
46            """ Devolve a temperatura da CPU, em graus Celsius """
47            return self._cpu_temp.temperature
48
49    class RAMSensor:
50        """ Sensor de uso de RAM. """
51
52        def __init__(self):
53            """ Inicializa o sensor. """
54            from psutil import virtual_memory
55            self._virtual_memory = virtual_memory
56
57        def used(self):
58            """ Lê o uso de memória virtual, em bytes """
59            return self._virtual_memory().used
60
61    class RateSensor:
62        def __init__(self):
63            self.last_sample = {}
64
65        def calculate_rate(self, field):
66            t0, s0, callback = self.last_sample[field]
67            t1, s1 = time(), callback()
```

```
68
69              rate = (s1 - s0) / (t1 - t0)
70
71              self.last_sample[field] = (t1, s1, callback)
72
73              return rate
74
75      class NetSensor(RateSensor):
76          """ Sensor de rede. """
77
78          def __init__(self):
79              from psutil import net_io_counters
80
81              super().__init__()
82
83              n0 = net_io_counters()
84              t = time()
85
86              self.last_sample = {
87                  'down': (t, n0.bytes_recv,
88                           lambda: net_io_counters().bytes_recv),
89                  'up': (t, n0.bytes_sent,
90                           lambda: net_io_counters().bytes_sent)
91              }
92
93          def download_rate(self):
94              """ Lê a taxa de download, em bytes por segundo. """
95              return self.calculate_rate('down')
96
97          def upload_rate(self):
98              """ Lê a taxa de upload, em bytes por segundo. """
99              return self.calculate_rate('up')
100
101     class DiskSensor(RateSensor):
102         """ Sensor de acesso a disco. """
103
104         def __init__(self):
105             from psutil import disk_io_counters
106
107             super().__init__()
108
109             d = disk_io_counters()
110             t = time()
111
112             self.last_sample = {
113                 'read_count': (t, d.read_count,
114                                lambda: disk_io_counters().read_count),
115                 'write_count': (t, d.write_count,
116                                 lambda: disk_io_counters().write_count),
117                 'read_bytes': (t, d.read_bytes,
118                                lambda: disk_io_counters().read_bytes),
119                 'write_bytes': (t, d.write_bytes,
120                                 lambda: disk_io_counters().write_bytes),
121                 'read_time': (t, d.read_time,
122                               lambda: disk_io_counters().read_time),
123                 'write_time': (t, d.write_time,
```

```
124                                 lambda: disk_io_counters().write_time)
125             }
126
127         def read_count_per_sec(self):
128             """ Lê a quantidade de leituras do disco por segundo. """
129             return self.calculate_rate('read_count')
130
131         def write_count_per_sec(self):
132             """ Lê a quantidade de escritas no disco por segundo. """
133             return self.calculate_rate('write_count')
134
135         def read_bytes_per_sec(self):
136             """ Lê a quantidade de bytes lidos do disco por segundo. """
137             return self.calculate_rate('read_bytes')
138
139         def write_bytes_per_sec(self):
140             """ Lê a quantidade de bytes escritos no disco por segundo. """
141             return self.calculate_rate('write_bytes')
142
143         def read_time_per_sec(self):
144             """ Lê a quantidade tempo para a leitura do disco por
145             segundo. """
146             return self.calculate_rate('read_time')
147
148         def write_time_per_sec(self):
149             """ Lê a quantidade tempo para a leitura no disco por
150             segundo. """
151             return self.calculate_rate('write_time')
```

Referências

[Caninos Loucos 2019] Caninos Loucos. *Labrador*. https://wiki.caninosloucos.org.br/index.php/Labrador. Último acesso em 23 de Agosto de 2019. Caninos Loucos, 2019 (citado na pg. 7).

[Desai *et al.* 2013] Ankita Desai, Rachana Oza, Pratik Sharma e Bhautik Patel. "Hypervisor: a survey on concepts and taxonomy". Em: *International Journal of Innovative Technology and Exploring Engineering* 2.3 (2013), pgs. 222–225 (citado na pg. 5).

[Elastic 2019(a)] Elastic. *Filebeat overview*. https://www.elastic.co/guide/en/beats/filebeat/current/filebeat-overview.html. Elastic, 2019 (citado na pg. 6).

[Elastic 2019(b)] Elastic. *Logstash: Collect, Parse, Transform Logs*. https://www.elastic.co/products/logstash. Elastic, 2019 (citado na pg. 6).

[Elastic 2019(c)] Elastic. *What is Elasticsearch*. https://www.elastic.co/pt/what-is/elasticsearch. Elastic, 2019 (citado na pg. 6).

[GT-BIS 2018] GT-BIS. *GT-BIS – Mecanismos para Análise de Big Data em Segurança da Informação*. http://gtbis.ime.usp.br/. Último acesso em 22 de Março de 2019. 2018 (citado nas pgs. 1, 7, 15, 16).

[Kafka 2019] Kafka. *What is Elasticsearch*. https://kafka.apache.org/. Apache, 2019 (citado na pg. 6).

[Kurose e Ross 2003] James F. Kurose e Keith W. Ross. *Redes de Computadores e a Internet*. 1ª ed. Pearson, 2003 (citado na pg. 3).

[Oshiro e Batista 2019] Lucas Oshiro e Daniel Macêdo Batista. "Análise preliminar da detecção de ataques ofuscados e do uso de hardware de baixo custo em um sistema para detecção de ameaças". Em: *Anais Estendidos do XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. SBC. 2019, pgs. 233–240 (citado nas pgs. 1, 2).

[Ortmeyer 2014] Cliff Ortmeyer. "A brief history of single board computers". Em: (2014) (citado na pg. 7).

[OWASP 2019(a)] OWASP. *SQL Injection*. https://www.owasp.org/index.php/SQL_Injection. OWASP, 2019 (citado na pg. 10).

[OWASP 2019(b)] OWASP. *SQL Injection*. https://www.owasp.org/index.php/Cross-site_Scripting_(XSS). OWASP, 2019 (citado na pg. 10).

[Pahl *et al.* 2016] C. Pahl, S. Helmer, L. Miori, J. Sanin e B. Lee. "A Container-Based Edge Cloud PaaS Architecture Based on Raspberry Pi Clusters". Em: *4th IEEE FiCloudW*. Ago. de 2016, pgs. 117–124. doi: 10.1109/W-FiCloud.2016.36 (citado na pg. 1).

[Raspberry Pi Foundation 2018] Raspberry Pi Foundation. *Raspberry Pi – Teach, Learn, and Make with Raspberry Pi*. https://www.raspberrypi.org/. Último acesso em 22 de Março de 2019. Raspberry Pi Foundation, 2018 (citado na pg. 7).

[Texas Instruments 2012] Texas Instruments. *INA219 Zero-Drift, Bidirectional Current/Power Monitor With I2C Interface*. http://www.ti.com/lit/ds/symlink/ina219.pdf. Último acesso em 17 de novembro de 2019. 2012 (citado na pg. 7).

[VMWare 2019] VMWare. *VMware Glossary*. https://www.vmware.com/. Último acesso em 23 de Agosto de 2019. VMWare, 2019 (citado na pg. 5).

[Xen 2019] Xen. *Xen Overview*. https://wiki.xen.org/wiki/Xen_Project_Software_Overview. Xen, 2019 (citado nas pgs. 5, 6).