UNIVERSIDADE FEDERAL DE SERGIPE
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

PROPOSTA DE IDENTIFICAÇÃO DE ATAQUES AO SERVIÇO *SSH* USANDO PADRÕES NO CONSUMO DE CORRENTE EM PLATAFORMAS EMBARCADAS

VICTOR GABRIEL GALVAN

SÃO CRISTÓVÃO/SE
2016

UNIVERSIDADE FEDERAL DE SERGIPE
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

VICTOR GABRIEL GALVAN

PROPOSTA DE IDENTIFICAÇÃO DE ATAQUES AO SERVIÇO *SSH* USANDO PADRÕES NO CONSUMO DE CORRENTE EM PLATAFORMAS EMBARCADAS

Proposta de Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação (PROCC) da Universidade Federal do Sergipe (UFS) como parte de requisito para obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Edward David Moreno Ordonez

Coorientador: Prof. Dr. Ricardo Salgueiro

SÃO CRISTÓVÃO/SE

Novembro 2016

Galvan, Victor Gabriel

G182p Proposta de identificação de ataques ao serviço SSH usando padrões no consumo de corrente em plataformas embarcadas / Victor Gabriel Galvan ; orientador Edward David Moreno Ordonez. – São Cristóvão, 2016.

151f. : il.

Dissertação (mestrado em Ciência da Computação) - Universidade Federal de Sergipe, 2016.

1. Ciência da computação. 2. *RaspberryPi*. 3. *Arduino Uno*. 4. SSH. 5. Medusa. 6. Hydra. 7. Metasploit. I. Ordonez, Edward David Moreno, orient. II. Título

CDU: 004.3

VICTOR GABRIEL GALVAN

PROPOSTA DE IDENTIFICAÇÃO DE ATAQUES AO SERVIÇO *SSH* ATRAVÉS DO CONSUMO DE CORRENTE NA PLATAFORMA EMBARCADA *RASPBERRY PI*

Proposta de Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação (PROCC) da Universidade Federal do Sergipe (UFS) como parte de requisito para obtenção do título de Mestre em Ciência da Computação.

BANCA EXAMINADORA

Prof. Dr. Edward David Moreno Ordonez, Presidente
Universidade Federal de Sergipe (UFS)

Prof. Dr. Ricardo Salgueiro, Coorientador
Universidade Federal de Sergipe (UFS)

Prof. Dra. Edilayne Meneses Salgueiro, Membro
Universidade Federal de Sergipe (UFS)

Prof. Dr. Fabio Dacencio Pereira, Membro
Centro Universitário Eurípides de Marília (UNIVEM)

PROPOSTA DE IDENTIFICAÇÃO DE ATAQUES AO SERVIÇO *SSH* ATRAVÉS DO CONSUMO DE CORRENTE NA PLATAFORMA EMBARCADA *RASPBERRY PI*

Este exemplar corresponde à Dissertação de Mestrado, do mestrando **VICTOR GABRIEL GALVAN** para ser aprovada pela Banca Examinadora.

São Cristóvão - SE, 22 de Novembro de 2016

_____

Prof. Dr. Edward David Moreno Ordonez

Orientador (UFS)

_____

Prof. Dr. Ricardo Salgueiro

Coorientador (UFS)

_____

Prof. Dra. Edilayne Meneses Salgueiro

Universidade Federal de Sergipe (UFS)

_____

Prof. Dr. Fabio Dacencio Pereira

Centro Universitário Eurípides de Marília (UNIVEM)

DEDICATÓRIA

Não alcançam as palavras que explicam esse sentimento dominante em minha vida, por isto, obrigado a todas as pessoas que diretamente e indiretamente estão envolvidas nesta proeza acadêmica e pessoal.

Simplesmente obrigado.

AGRADECIMENTOS

Este trabalho foi possível através da colaboração dos distintos atores envolvidos:

UTN-FRT, OEA, CAPES e UFS e as pessoas que fazem possível tudo isto.

À família, principal força motivadora, aos amigos que acompanham os sonhos e às instituições que materializam as oportunidades desses sonhos.

Obrigado!

RESUMO

Este trabalho apresenta a obtenção de curvas de consumo de corrente elétrica, a partir das respostas geradas por um sistema embarcado de baixo custo *Raspberry Pi 2 Model B* executando o sistema operacional *Linux Raspbian* trabalhando como um servidor de acesso remoto *SSH*, que é avaliado através de diferentes tipos de acessos e ataques de força bruta com dicionários através das ferramentas especializadas *Medusa* e *Hydra*, como também a ferramenta não especializada *Metasploit*. O comportamento energético é interpretado por um sistema de medição de consumo de corrente desenvolvido pela plataforma embarcada de baixo custo *Arduino Uno* que administra um sensor de corrente baseado no chip *ACS721ELC-5A* de efeito *Hall*, que possui a capacidade de coletar as variações geradas pela plataforma de teste em resposta aos eventos produzidos pelos cenários de provas propostos, os dados são processados pelo *Framework Matlab* que coleta, analisa e normaliza por meio do método de *Welch* o sinal de corrente que é interpretado pelo *Arduino Uno*, posteriormente apresenta-se uma curva padrão que caracteriza um determinado evento baseado nos cenários de provas. Os resultados apresentam as diferentes curvas padrões normalizadas, e contextualizadas nos tipos de cenários avaliados, seguidamente apresenta-se um modelo matemático teórico do consumo de corrente proposto, como também as regras ou assinaturas propostas para identificar um ataque através do método de detecção por padrões que utilizada o *IDS Snort*. Essas curvas de corrente facilitam o entendimento e obtenção de um padrão de consumo de corrente para cada acesso e ataque na plataforma embarcada.

Palavras chave: *Raspberry Pi*, *Arduino Uno*, *SSH*, *Medusa*, *Hydra*, *Metasploit*.

ABSTRACT

This paper presents the obtaining of electric power consumption curves, from the responses generated by an embedded low-cost Raspberry Pi 2 Model B system running the Linux operating system Raspbian working as a remote access server SSH, which is assessed through different types of access and brute force attacks dictionaries through specialized tools Medusa and Hydra, as well as the tool Metasploit unspecialized. The energy behavior is interpreted by a current consumption measurement system developed by low embedded platform cost Arduino Uno that runs a current sensor based on ACS721ELC-5A Hall effect chip, which has the ability to collect the variations generated by the platform test in response to events produced by the proposed test scenarios, the data is processed by the framework Matlab that collects, parses and normalizes using the Welch method, the current signal which is interpreted by Arduino Uno subsequently presents a standard curve It features a particular event based on scenarios of evidence. The results show the different curves standard patterns, and contextualized on the types of scenarios evaluated subsequently presents a theoretical mathematical model of the proposed power consumption, as well as rules or signatures proposed to identify an attack using the detection method of standards used IDS Snort. These current curves facilitate understanding and obtaining a pattern of current consumption for each access and attack the embedded platform.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE EQUAÇÕES

LISTA DE SIGLAS

AES	*Advance Encryption Standard*
ARM	*Advanced RISC Machine*
DDoS	*Denial of Service Distributed*
DHCP	*Dynamic Host Configuration Protocol*
DoS	*Denial of Service*
DSA	*Digital Signature Algorithm*
FPGA	*Field Programmable Gate Array*
FTP	*File Transference Protocol*
GPU	*Unit Process Graphics*
GQM	*Goal, Question, Metric*
HTTP	*Hypertext Transfer Protocol*
IANA	*Internet Assigned Numbers Authority*
ICMP	*Internet Control Message Protocol*
IDS	*System Detection Intrusion*
IETF	*Internet Engineering Task Force*
IOT	*Internet of Things*
IP	*Internet Protocol*
IPS	*System Prevention Intrusion*
ISO	*International Organization for Standardization*
LIBSSH	*Library Implementing the SSH2v2 and SSHv1 Protocol*
LIBSSH2	*library implementing the SSH2 protocol*
MAC	*Media Access Control*
MD5	*Message Digest Algorithm*
MHN	*Modern Honey Network*
MMUSIC	*Multiparty Multimedia Session Control*
MYSQL	*Banco de dados Open Source*
NOSQL	*Dados não estruturados*
OPENSSH	*Open Secure Shell*
OSI	*Open System Interconnection*
PAC	*Padrão de Acesso Concedido*
PAH	*Padrão de Ataque Hydra*

PAN	*Padrão de Acesso Negado*
PAM	*Padrão de Ataque Medusa*
PAME	*Padrão de Ataque Metasploit*
PGP	*Pretty Good Privacy*
PSD	*Densidade de Potência Espectral*
PEE	*Padrão de Estado Estável*
PING	*Packet Internet Groper*
RC4	*Rivest Cipher 4*
RDP	*Remote Desktop Protocol*
RFC	*Request for Comments*
RSA	*Cryptographic Algorithm*
SDN	*Network Define Software*
SHA-1	*Secure Hash Algorithm*
SIP	*Session Initiation Protocol*
SMB	*Server Message Block*
SQUID	*Server Proxy*
SSH	*Secure Shell*
TCP	*Transmission Control Protocol*
TELNET	*Telecommunication Network*
UART	*Universal Asynchronous Receiver-Transmitter*
UDP	*User Datagram Protocol*
VNC	*Virtual Network Computing*
VPN	*Virtual Private Network*
ZERO-DAY	*0-day Attack*
3DES	*Triple Data Encryption Standart*

SUMÁRIO

CAPÍTULO 1

INTRODUÇÃO

De acordo com [COELHO et al. 2014], a segurança da informação tem a responsabilidade de proteger a informação sendo determinante para assegurar a competitividade, a lucratividade, e o atendimento aos requisitos legais, preservando a imagem da organização junto ao mercado. A segurança de informação, segundo [GIAVAROTO et al. 2013], não é um produto, é um processo, e por isso, há uma enorme dificuldade em determinar qual é o nível de segurança apropriado.

É necessário investigar os riscos, realizar testes, validar as políticas de segurança e tecnologias utilizadas com o objetivo de atender os preceitos de segurança da informação.

A segurança nas comunicações [STALLINGS 2007] é uma prática que contém como tarefa principal, prevenir o acesso não autorizado a certos recursos informáticos. Esta conotação é mais estrita no sentido de desenho das soluções entre duas entidades que tentam estabelecer um meio de comunicação comum e seguro. Neste contexto, os protocolos de comunicações são os responsáveis de administrar mecanismos necessários, a fim de estabelecer uma ligação entre uma origem e um destino de forma que os dados enviados sejam transmitidos em forma segura [AMATO 2001].

A confidencialidade, segundo os autores [KUROSE et al. 2010], é um conceito que garante a segurança na transmissão de mensagens e devem ser compreendidas somente pelo remetente e pelo destinatário. A forma de afiançar esta segurança é através de técnicas de cifragem de dados combinada com a autenticação do ponto final.

Na atualidade existem soluções em *software* que cifram os canais de comunicação garantindo a confiabilidade dos dados transmitidos, um exemplo são as redes privadas virtuais *VPN* [MASON et al. 2002], arquitetura de comunicação que concede um nível aceitável de segurança. Esta arquitetura de segurança exposta a condições de ataques especializados pode chegar a comprometer sua segurança, desta forma um atacante externo está possibilitado a

interceptar as mensagens enviadas e recebidas com o agravante de manipular as comunicações à sua vontade [ASHIDANI 2009].

O serviço de comunicação *SSH* [SILVERMAN et al. 2001] é o serviço de rede mais difundido e utilizado na administração remota de recursos informáticos. Uma das capacidades que possui *SSH* é estabelecer um canal de comunicação cifra entre uma origem e um destino, este serviço é muito utilizado com o fim de realizar a administração geral de um sistema informático remoto [LAUDON et al. 2013]. Desde o ponto de vista dos administradores de sistemas é a ferramenta por excelência na administração remota, também um ponto de ataque relevante para potenciais intrusões e tudo tipo de ameaças [TABISH et al. 2009].

Hoje em dia se conhecem *Malwares* [INCE 2008] muito sofisticados que automatizam tudo o processo de conexão a um sistema remoto baseado em ameaças conhecidas [MORTENSEN et al. 2013].

Os sistemas embarcados de acordo com os autores [MORENO et al. 2003] são definidos como uma unidade central de processo integrado a um sistema maior com o objetivo de auxiliar o controle e execução de tarefas, ademais os sistemas embarcados em termos gerais são considerados dispositivos eletrônicos de propósitos gerais. Um claro exemplo é o projeto *Raspberry Pi Foundation*[1], esta iniciativa nasceu com o objetivo de estimular o ensino da informática baseado na filosofia *Open hardware e Open software*. Seu desenvolvimento está direcionado em uma economia de escala, frente a um mercado competitivo de sistemas embarcados existentes, tais como: *Oluxino*[2], *Cubieboard2*[3], *Odroid*[4], *Beagleboard*[5], *Raspberry Pi* possui a maior cota de mercado neste segmento, isso se deve à relação que existe entre o preço e as características disponíveis. Na Tabela 1.1 apresentam-se as características técnicas dos sistemas embarcados mencionados anteriormente.

Tabela 1.1: Comparativa de *Raspberry Pi 2 Model B* e seus concorrentes.

	RASPBERRY PI 2	**OLINUXINO (A20)**	**CUBIEBOARD**	**ODROID (XU3)**	**BEAGLEBOARD**
Velocidade (MHZ)	700/1000	1000	1000	2000	1000
Núcleos	QuadCore	Dual Core	Mono Core	OctaCore	Mono Core
Ram (Mb)	1024	512	1024	2048	512
Usb	4	2	2	5	1
Ethernet	100 Mb	100 Mb	100 Mb	100 Mb	100 Mb
Wifi - Bluetooth	Não	Não	Não	Não	Não
Gpio	x40	x160	X67	X30	x96
Câmera	Sim	Não	Não	Não	Não
Preço	$35	$60	$62	$180	$55

[1] www.raspberrypi.org
[2] www.olimex.com
[3] www.cubieboard.org
[4] www.hardkernel.com
[5] www.beagleboard.org

Na Tabela 1.1 a plataforma *Raspberry Pi 2 Model B* possui uma relação de processamento inferior ao seus concorrentes. No entanto analisando desde uma perspectiva de preço final a plataforma *Raspberry Pi 2 Model B* possui uma diferença importante tendo em função preço-característica de produto, este índice posiciona a *Raspberry Pi 2 Model B* como a plataforma mais popular e referente no segmento dos embarcados *Open hardware* e *Open software*.

Está plataforma de baixo custo é ideal para desenvolver *software* focado no paradigma *Linux Embedded* baseado em processadores *ARM*[6], sob este contexto, atualmente se desenvolvem soluções tecnológicas avançadas com custos de *hardware* econômicos, potencializando assim o desenvolvimento deste segmento pertencente à Internet das Coisas (*IoT - Internet of Things*) [PRETZ 2013] [ITU 2015] [PANIAGUA 2015].

O estudo [EVANS 2011] manifesta que uma equipe de investigadores da China estudou os dados de roteamento de *Internet* em intervalos semestrais, ou seja, desde dezembro de 2001 até dezembro de 2006. Os resultados permitem observar que a *Internet* duplica seu tamanho cada 5,32 anos. Este estudo apresenta evidência que o aumento de dispositivos conectados a *World Wide Web* em um curto prazo aumentará em forma gradual crescente, levando a um aumento de ataques informáticos clássicos conhecidos, como também novos tipos de ataques especializados.

1.2 Descrição do Problema

Como afirmam os autores [CHIANG et al. 2008], o código malicioso *Malware* que circula pela rede possui uma conotação importante dentro do âmbito da segurança, tanto por seus meios de ataques, como por sua complexidade. É importante caracterizar estas ameaças e compreender quais são as metodologias de ataques empregadas, neste contexto os autores propõem boas práticas em segurança, como também o desenvolvimento de novas ferramentas que controlem, reduzam e/ou eliminem o código malicioso.

Uma ameaça, segundo os autores [STALLINGS 1999], define-se como um evento potencial indesejável ou real, que pode ser malicioso (como um ataque) ou acidental (falha em um dispositivo de armazenamento).

A modelagem de ameaças é uma atividade planejada para a identificação e avaliação das ameaças e vulnerabilidades nas aplicações, uma vulnerabilidade é compreendida como uma deficiência ou falha na segurança de um sistema em termos gerais.

[6] www.arm.com

Define-se uma política de segurança como o resultado de documentar as expectativas de segurança. O conceito de segurança está relacionado com o comportamento esperado de um sistema. Pode-se afirmar que as políticas de segurança tentam documentar de alguma maneira os conceitos abstratos da segurança.

O *Malware* [MOSER et al. 2007] define-se como um *software* que cumpre uma tarefa deliberada nociva de um atacante, termos como: *Worm, Virus* ou *Trojan* são usados para classificar estes tipos de ameaças. Um exemplo desta ameaça na prática [LUCHI 2013], manifesta que é comum explorar vulnerabilidades de senha ao protocolo *SSH* com técnicas de força bruta utilizando dicionários [VYKOPAL et al. 2009], este processo de *Hacking* segundo os autores é feito através da utilização de uma arquitetura *GPU*[7] [SANDERS et al. 2010]. Existem diversas técnicas, a fim de mitigar estes tipos de ataques.

Os autores [HOFSTEDE et al. 2014] apresentam um algoritmo baseado na análise de fluxo com o alvo de detectar os dispositivos alcançados por ataques de força bruta utilizando dicionários pré-definidos. Assim, podemos afirmar que um ataque bem sucedido que vulnera o acesso ao serviço remoto *SSH*, sob qualquer tipo vulnerabilidade conhecida ou não conhecida, é classificado como crítico pelo fato de comprometer tudo o sistema. A diversidade de técnicas e metodologias a estes tipos de ataques é considerada muito grande, sendo hoje em dia uma das áreas com maior desenvolvimento. Outro ponto importante observado está na importância de melhorar os sistemas de detecção de intrusão *IDS*[8] com o objetivo de mitigar os distintos tipos de ataques praticados a uma infra-estrutura informática *TI*.

O desafio está em desenvolver novas bases de conhecimentos, a partir de distintos meios de informação tais como: *Logs* de acessos, análises do tráfico da rede, falhas de *software*, vulnerabilidades, toda esta informação centralizada e processada contém informação que é utilizada na tomada de decisões em um âmbito de TI.

1.3 Formulação do Problema de Estudo

1. É possível determinar quais são os serviços mais atacados utilizando uma plataforma de teste *MHN Online* baseado em *Honeypots*.

2. É possível desenvolver um sistema eletrônico embarcado de baixo custo baseado na plataforma *Open hardware* e *Open software Arduino Uno* que cumpra os

[7] Unit Process Graphics
[8] Intrusion Detection Systems

seguintes requisitos funcionais: aquisição, processamento e comunicação de parâmetros elétricos provenientes de sensores.

3. É possível detectar uma curva padrão do consumo de corrente gerado pela resposta de uma plataforma de teste embarcada de baixo custo baseada em *Raspberry Pi 2 Model B* avaliada em um conjunto de cenários controlados de ataques ao serviço de acesso remoto *SSH*.

1.4 Objetivos da Dissertação

O objetivo principal desta dissertação é desenvolver um sistema embarcado *Open hardware* e *Open software* de baixo custo, que permita fornecer curvas padrões do consumo de corrente gerado pela resposta de uma plataforma de teste embarcada de baixo custo *Raspberry Pi 2 Model B* executando como um servidor de acesso remoto *SSH* avaliado em cenários de provas controlados.

Para alcançar esse objetivo, podemos destacar os seguintes objetivos específicos:

1. Propor uma plataforma de teste *MHN (Modern Honey Network)* baseado em *Honeypot* com o alvo de conhecer a quantidade de ataques coletados a partir dos serviços disponibilizados;

2. Identificar o serviço com maior porcentagem de ataques recebidos analisando os dados recolhidos pela plataforma de teste *MHN*;

3. Propor uma plataforma de teste baseada em uma plataforma embarcada de baixo *custo Raspberry Pi 2 Model B* executando o serviço de acesso remoto *SSH* avaliada em cenários de provas predefinidos.

4. Propor um sistema embarcado de medição de consumo energético de baixo custo baseado em *Arduino Uno* com alvo de coletar, analisar, processar e prover uma curva padrão de corrente para os principais ataques relacionados ao serviço com maior porcentagem de ataques.

5. Normalizar o consumo de corrente com o método de *Welch*.

1.5 Metodologia Proposta

A metodologia proposta foi organizada nas seguintes atividades:

1. Revisão bibliográfica;

2. Proposta de uma plataforma de teste *MHN* baseado em *Honeypots*;

3. Avaliação dos resultados da plataforma de teste *MHN* com técnicas estatísticas e seleção do serviço com maior quantidade de ataques detectados;
4. Desenvolvimento de um sistema de medição de consumo energético baseado na plataforma embarcada de baixo custo *Arduino Uno* baseada no sensor de corrente *ACS712ELCTR-05B*;
5. Desenvolvimento de uma plataforma de teste embarcada baseada em *Raspberry Pi 2 Model B* executando o serviço de acesso remoto *SSH* avaliada com cenários de tipos de acesso e cenários de ataques de força bruta com dicionário;
6. Definição dos cenários de provas propostos:
   - **6.1** Cenário 1 - Tipos de Acessos: Acesso Permitido.
   - **6.2** Cenário 2 - Tipos de Acessos: Acesso Negado.
   - **6.3** Cenário 3: Ataque de Força Bruta com Dicionário Através da Ferramenta Especializada *Medusa*.
   - **6.4** Cenário 4: Ataque por Força Bruta com Dicionário Através da Ferramenta Especializada *Hydra*.
   - **6.5** Cenário 5: Ataque por Força Bruta com Dicionário Através da Ferramenta Não Especializada *Metasploit*.
7. Análise e interpretação dos dados experimentais obtidos;
8. Obtenção de um padrão de consumo de corrente experimental normalizado através método de *Welch*.

1.6 Justificativa e Delimitação da Investigação

Atualmente os sistemas de proteção da informação em um âmbito de *TI* requerem atualizações constantemente, isto está diretamente relacionado com a grande quantidade de código malicioso que circula na rede. O grau de desenvolvimento do mercado de código malicioso motiva aos atacantes ao desenvolvimento a este tipo de *software*.

O objetivo principal destas aplicações maliciosas é invadir os sistemas de segurança de sistemas especializados como os seguintes: *IDS, IPS, Antivirus*, isto é alcançado a partir de explorar novas vulnerabilidades detectadas que são aproveitadas pelos agressores para comprometer o sistema. Dessa maneira, a abordagem deste projeto está baseada em obter dados experimentais, a partir do comportamento de um modelo do consumo de corrente gerado através do desenvolvimento de uma plataforma de teste, baseado em um *hardware*

embarcado de baixo custo denominado *Raspberry Pi 2 Model B* executando como um servidor de acesso remoto *SSH*, o qual é avaliado sob diferentes cenários de provas propostos.

Pretende-se com este experimento identificar o consumo de corrente incorrido por uma plataforma de teste embarcada baseada em *Raspberry Pi 2 Model B*, avaliada em uma série de cenários de provas com o alvo de encontrar curvas padrões de identificação de ameaças e/ou ataques.

Este experimento é abordado através de um dispositivo embarcado onde é possível obter o consumo de corrente com técnicas *Side Channel Attacks* através de uma plataforma *Arduino Uno*, que através de sensores específicos possibilita a aquisição, processamento e comunicação do consumo energético associado à plataforma de teste *Raspberry Pi 2 Model B*. O limite do escopo da dissertação está definida pelos seguintes sistemas envolvidos:

1. Plataforma de teste baseado *Raspberry Pi 2 Model B* executando o sistema operacional *Linux Raspbian* trabalhando como um servidor dedicado *SSH*;
2. Plataforma de medição de consumo energético baseado em *Arduino Uno* coletando sinais de corrente;
3. Post processamento dos sinais de corrente com o método de *Welch*;
4. Cenários de provas de tipos de acessos e tipos de ataques de força bruta com dicionário;

A delimitação está contida na análise e funcionamento de cada plataforma apresentada, visando uma abordagem em trabalhos correlatos e experiência profissional nas distintas áreas propostas.

1.7 Contribuições da Dissertação

O presente trabalho de pesquisa proposto pode contribuir na melhora dos sistemas de detecção de intrusões através do fornecimento de uma nova fonte de dados de consulta baseado na identificação de ataques, por meio do consumo de corrente gerado. Particularmente neste trabalho se propõe melhorar as regras de detecção de ameaças ao serviço de acesso remoto *SSH*. É importante lembrar que um método de identificação de ameaças é feito através de assinaturas ou padrões de ameaças, estas últimas são armazenadas em um arquivo de regras que se atualizam constantemente e é identificado como fonte de identificação de ameaças externas do sistema.

Usando-se o conceito anterior o trabalho propõe uma nova fonte de dados externas baseado no consumo médio de corrente, a partir de dados experimentais obtidos de um experimento controlado, baseado na execução de cenários de ataques a uma plataforma embarcada de baixo custo *Raspberry Pi 2 Model B* funcionando como um servidor de acesso remoto *SSH*.

Considera-se que a nova fonte de dados externa que associa um ataque com o consumo de corrente de um dispositivo embarcado é propício para desenvolver-se em um ambiente experimental sob um conjunto de cenários de provas controlado, em consequência um *IDS* pode beneficiar-se na melhora da detecção de intrusões através do método de identificação de ameaças por padrões no consumo de corrente dos servidores.

1.8 Organização da Dissertação

Está dissertação está estruturada em sete (7) capítulos como se segue:

O Capítulo 1 mostra o contexto e a definição do problema a abordar, além disso, os objetivos principais, metodologia adotada, e justificativa correspondente. No Capítulo 2 apresentam-se os trabalhos correlatos e plataformas embarcadas usadas na dissertação. No Capítulo 3 apresenta-se a plataforma de coleta *MHN* baseada em *Honeypots*. No Capítulo 4 apresenta-se uma avaliação experimental dos dados coletados pela plataforma *MHN*, a fim de conhecer qual é o serviço identificado com maior taxa de ataques. Já no Capítulo 5 apresenta-se uma descrição detalhada da plataforma de avaliação de baixo custo *Raspberry Pi 2 Model B* executando-se como um servidor dedicado de acesso remoto *SSH*, ademais apresenta-se a plataforma de medição de consumo energético *baseado em Arduino Uno*. No Capitulo 6 apresentam-se uma metodologia de medição de consumo de corrente desenvolvida neste trabalho de dissertação. Capítulo 7 apresentam-se os cenários de provas propostos, analisando os resultados alcançados. Por último, no Capítulo 8, apresenta-se uma análise crítica dos resultados obtidos. Em seguida, são apresentadas sugestões para os trabalhos futuros, conclusões e contribuições desta dissertação.

CAPÍTULO 2

ESTADO DA ARTE

Neste capítulo apresentam-se uma análise das distintas plataformas pesquisadas, a fim de compreender e delimitar o escopo da dissertação proposta. A continuação as plataformas embarcadas pesquisas, âmbitos de utilização, software envolvidos, também as técnicas e metodologias utilizadas na identificação de ataques.

2.1 Fundamentos da Plataforma *Raspberry Pi 2 Model B*

O trabalho [RAGUVARAN et al. 2015] propõe um sistema de monitoração de processos industriais baseado em uma arquitetura cliente-servidor, sob uma plataforma *Open hardware* e *Open software Raspberry Pi Model B+*. O sistema de captura de dados é realizado pelo desenvolvimento de um *hardware* de propósito específico que é administrado por um microcontrolador *PIC 18F4550*, a função principal deste chip é o processamento dos dados enviados pelos sensores de corrente, tensão e temperatura, os dados são transferidos a uma aplicação desenvolvida sob o embarcado *Raspberry Pi Model B+* que apresenta os resultados obtidos em uma interface gráfica.

O trabalho [ABRAHAMSSON et al. 2013] propõe um sistema de computação de alto desempenho com tecnologia *Open hardware Raspberry Pi Model B*. O sistema está fornecido por 300 nodos *Raspberry Pi 2 Model B* executando um sistema operacional baseado em *Linux Raspbian*. A arquitetura de comunicação está baseada em uma rede de topologia em estrela expansível. A arquitetura de *software* de administração do sistema computacional está composta de um nodo *Master* e nodos *Slaves*, e a comunicação é realizada usando-se o paradigma *MPI*[9].

O trabalho [SUDHIR et al. 2014] propõe um sistema de comunicação baseado em uma rede de sensores sem fio (*RSSF*), com uma topologia de distribuição dos nodos em malha (em

[9] Message Passing Interface

inglês, *Mesh*). A plataforma *Open hardware Raspberry Pi Model B* é utilizada como um nodo principal (*Master)* de comunicação entre os distintos nodos sensores (*Slaves*), ademais de fazer o processamento em tempo real das distintas informações enviadas e recebidas pelos sensores, a comunicação entre os nodos sensores e o nodo *Master* é realizada através do protocolo de comunicação *ZigBee* baseado padrão *IEEE 802.15.4*. Os nodos sensores são gerenciados por um *chip ATMEGA324PA* microcontrolador conectado a um módulo de comunicação *XBee* que fornece a capacidade de comunicar-se com outros sensores da *RSSF*[10]. A plataforma *Raspberry Pi* é utilizada com vários objetivos que são: brindar o armazenamento dos dados coletados pelos sensores, cumprir a função de *Gateway*, e por último administrar a capacidade de acesso remoto aos usuários externos.

O trabalho [SURYATALI et al. 2015] propõe um sistema de cobrança de pedágio baseado em reconhecimento de imagens em uma arquitetura de baixo custo Open hardware *Raspberry Pi Model B*. O sistema inclui a funcionalidade de realizar a detecção de veículos usando técnicas de reconhecimento de imagens. O desenvolvimento da aplicação está baseado em biblioteca *OpenCV*[11]. A captura das imagens é através de uma *Webcam* denominada *Iball Face2Face*, onde logo é processado pelo algoritmo de reconhecimento na plataforma *Raspberry Pi*. Acontecido esse processo a imagem é identificada, catalogada, indexada e armazenada em uma base de dados própria do sistema. A avaliação experimental do sistema proposto é realizada sob uma arquitetura *Intel x86* onde os resultados obtidos cumprem seu objetivo.

O trabalho [] propõe um resumo das plataformas embarcadas mais utilizadas no mercado para o desenvolvimento de pesquisas e aplicações nas diversas áreas de estudos. Os amplos desenvolvimentos das plataformas embarcadas atuais possuem configurações muito varias com respeito a sua arquitetura de *hardware* utilizada, porém o software que gerencia estas plataformas possui uma conotação mais flexível, por exemplo, o sistema operacional baseado em *Linux* é amplamente utilizado pelas plataformas deste trabalho apresentado. O custo associado a cada plataforma embarcada destaca-se como um ponto a favor. A continuação apresenta-se as primeiras cinco (5) plataformas embarcadas destacadas nesta pesquisa, são: *Raspberry Pi Model 3*, *Odroid-C2*, *BeagleBoard Black*, *Odroid-XU4* e *Creator Ci40*.

Na Tabela 2.1, é apresenta uma comparação dos distintos trabalhos realizados pelos autores citados anteriormente. Enfatizamos que as características do sistema embarcado *Open*

[10] Rede de Sensores Sem Fio
[11] Open Source Computer Vision

software e *Open hardware Raspberry Pi* é muito desenvolvida no âmbito da pesquisa, com aplicações nas distintas áreas de conhecimento.

Tabela 2.1: Resumo dos trabalhos baseados na plataforma *Raspberry Pi*

Referencias	**Arquitetura**			
	Plataforma		**Parâmetro Medição**	**Aplicação**
	Hardware	**Software**		
[25]	Raspberry Pi Model B+ Baixo Custo Inversão Detalhe Técnico: 1.- Microcontrolador PIC 18F4550 2.- Sensor nível água IC CD4066 3.- Sensor temperatura IC LM35DZ 4.- Sensor brilho transistor BC547	SO Linux baseado Raspbian Linguagem Java e C Compilador MikroC PRO compatível ANSI-C	Temperatura Nível brilho Nível Água	Sistema de processamento, controle e administração de parâmetros específicos SCADA.
[26]	Raspberry Pi Model B Baixo Custo Inversão	SO Linux baseado Raspbian Linguagem de programação C baseada na API de MPI	Custo inversão	Avaliação de um cluster embarcado e eficiência energética.
[27]	Raspberry Pi Model B Baixo Custo Inversão Detalhe Técnico: 1.- Micro controlador ATMEGA324PA 2.- Módulo comunicação XBee série 2	SO Linux baseado Raspbian	Não especifica	Sistema de comunicação entre sensores com tecnologia Zigbee.
[28]	Raspberry Pi Model B Baixo Custo Inversão Detalhe Técnico: 1.- Computador I5 (3,2 Ghz) 2.- Webcam Iball Face2Face	SO Linux baseado Raspbian biblioteca OpenCV	Métricas do sistema: 1.- Entrada 2.- Saída 3.- Reconhecidos Não 4.- Reconhecidos	Sistema de processamento de imagens.
	Raspberry Pi Model 3, Odroid-C2, BeagleBoard Black, Odroid-XU4 e Creator Ci40	SO Linux baseado	-	Sistema de proposito específico
Esta Dissertação	Raspberry Pi 2 Model B	SO Linux baseado Raspbian	Corrente Consumida (mA)	Execução do serviço SSH

2.2 Fundamentos da Plataforma *Arduino Uno*

A continuação na literatura encontra-se várias propostas para compreender o alcance e uso desta plataforma com suas respectivas aplicações desenvolvidas em nossa área de estudo.

O trabalho [FRANSISKA et al. 2013] propõe um sistema embarcado de medição de consumo energético analisando os parâmetros elétricos de corrente, tensão, potência e energia consumida. O desenvolvimento do aplicativo está feito em *Arduino Uno R3 Atmega328* com respeito ao *software* desenvolvido está baseado no uso da ferramenta chamada *Labview* que fornece um ambiente de programação gráfica orientado a objetos. O projeto apresenta um sistema embarcado de baixo custo baseado na plataforma *Arduino Uno* com o objetivo de controlar parâmetros energéticos.

O trabalho [GOMEZ et al. 2012] propõe um sistema embarcado autônomo de medição do consumo de energia em redes sem fio. A solução proposta está baseada em uma plataforma de *Open hardware Arduino Uno* onde a medição de tensão é realizada através do desenvolvimento de um divisor de tensão resistivo. O sensor de corrente empregado consiste em um dispositivo baseado em efeito *Hall*, sua denominação técnica é *ACS712*. A programação de baixo nível dos sensores é realizada com linguagem C/C++.

O trabalho [MAKONIN et al. 2013] propõe um monitor de consumo energético de propósito específico baseado em uma arquitetura *Open hardware* e *Open software Arduino Mega2560*, denominado *APMR*[12]. O monitor apresenta a capacidade de realizar medições de distintos parâmetros elétricos. Uma característica importante é o cálculo do preço da energia consumida em função do tempo. Os resultados são armazenados em um dispositivo externo para seu tratamento posterior, os quais podem ser consultados através de um sistema *online*.

O trabalho [HAIDER-E-KARAR et al. 2015] propõe um sistema de supervisão e aquisição de dados *SCADA*[13] que apresenta a administração de um sistema de geração de energia através de painéis solares. A arquitetura do sistema está composta da seguinte maneira: A lógica de programação está desenvolvida pelo paradigma cliente/servidor na plataforma *Labview* que permite uma integração homogênea com todos os dispositivos de conexão atuais, computadores, telefones inteligentes e aparelhos com tecnologia máquina a máquina *M2M*[14]. A plataforma de *hardware* empregada está baseado em *Arduino Mega 2560* microcontrolador *ATmega2560*. Os sensores utilizados neste projeto são um sensor de

[12] Arduino Power Meter Reader
[13] Supervisory control and data acquisition
[14] Machine to Machine

corrente que está baseado dispositivo *ACS712* e um sensor de tensão baseado em um divisor de tensão resistivo.

O trabalho [ANBYA et al. 2012] propõe um sistema de monitoramento de parâmetros elétricos em tempo real em uma rede de sensores sem fio *RSSF* com topologia em estrela. A plataforma de *hardware* está composta por um chip com capacidade de sensoriamento dos parâmetros elétricos básicos de corrente e tensão, o *módulo* empregado é denominado *IA ADE7753* do fabricante *MicroChip*. Os dados coletados pelo chip são enviados ao *Arduino Duemilanove*. Desta forma o sistema retransmite os dados a uma estação base, por meio do protocolo de comunicação *ZigBee* padrão *IEEE 802.15.4*. A estação base reenvia os dados coletados a um servidor *Web*, está transmissão é realizada através de dois modos de operação: A primeira está baseada em uma conexão *Ethernet*, e a segunda é por meio de uma conexão serial *UART*.

Na Tabela 2.2 apresenta-se um resumo dos distintos trabalhos citados anteriormente. Destaca-se a plataforma *Arduino*, como predominante em um amplo uso de aplicações e processos.

Tabela 2.2: Resumo dos trabalhos baseados na plataforma *Arduino*

Referencias	**Arquitetura**			
	Plataforma		**Parâmetro**	**Aplicação**
	Hardware	**Software**		
[29]	Arduino Uno R3 ATmega328. Microcontrolador Atemega8. 1.- Sensor voltagem baseado divisor de tensão resistivo. Baixo Custo Inversão	Framework de programação Labview	Tensão (V) Corrente (A) Potência (W)	Plataforma *Arduino* de medição de parâmetros energéticos.
[30]	Arduino Uno R3 ATmega328. 1.- Sensor voltagem baseado divisor de tensão resistivo 2.- Sensor corrente ACS712 Baixo Custo Inversão	Linguagem de programação Java Ferramenta Programação (C/C++) baseado biblioteca Wiring Ferramenta simulação: OnNET++	Métricas sistema: Tensão (V) Corrente (A) Potência (W) Métricas simulação: Potência (W) Taxa geração pacotes (MB/s) Tamanho pacote (Bytes)	Plataforma *Arduino* de medição de parâmetros energéticos.
[31]	Arduino Mega 2560 Módulo: 1;- Shield RS-485 (comunicação) 2;- RTC Clock Baixo Custo Inversão	Plataforma Web Armazenamento dados Mysql SO. Linux	Energia (Kw/h)	Plataforma *Arduino* de medição de parâmetros energéticos.
[32]	Arduino Mega 2560 1.- Sensor corrente ACS712 Baixo Custo Inversão	*Framework* de programação Labview – Web	Tensão (V) Corrente (A) Potência (W)	Plataforma *Arduino* de medição de parâmetros energéticos.
[33]	Arduino Duemilanove Módulo: 1. Shield RS-485 (comunicação) 2. Xbee-Pro Baixo Custo Inversão	SO Windows XP Sistema Web baseado ferramenta Xampp	Tensão (V) Corrente (A) Potência aparente (Vah) Potência ativa (Wh) Fator de potência Frequência (Hz)	Plataforma *Arduino* de medição de parâmetros energéticos.
Esta dissertação	Arduino Uno R3	Framework Matlab	Corrente Consumida (mA)	Sistema de medição de consumo de corrente

2.3 Fundamentos das Plataformas de Medição de Consumo Energético

O trabalho [MONTEIRO et al. 2007] propõe um sistema de controle de consumo de energia para dispositivos móveis. A medição é realizada através de um equipamento denominado medida direta de consumo (EMDC). O sistema é avaliado sob uma plataforma de comunicação com tecnologia *Bluetooth*, padrão de fato nas comunicações de hoje em dia em diversos dispositivos eletrônicos.

O trabalho [SOUTO et al. 2016] apresenta técnicas de amostragem para a obtenção do consumo de energia em redes de sensores sem fio, baseado em cenários de simulação com a ferramenta *Tossim* onde os sensores são agrupados por áreas de trabalho. As amostras são procuradas em função de certos sensores, que são representativos do consumo de energia. As técnicas desenvolvidas no trabalho são: Amostragem baseado em cadeia de *Markov* e amostragem estratificada, estas técnicas apresentam os mapas de consumo de energia através de outra técnica chamada *Krigagem* ordinária. Os resultados expõem que é possível obter o mapa de consumo de energia das *RSSF*, com as técnicas de amostragem expostas e também conhecer como é a distribuição do consumo de energia no tempo com a técnica de *Krigagem* ordinária, o uso de modelos estatísticos empregados para a estimação do consumo energético gerado por *RSSF* é devidamente recomendado por seu nível de confiabilidade. Os mapas do consumo energético gerado em uma *RSSF* são de fundamental importância na planificação estratégica de energia.

No trabalho [GUIMARAES et al. 2006], os autores propõem através de um cenário de simulação, como é o comportamento de uma rede de sensores sem fio avaliado por um conjunto de algoritmos criptográficos, os algoritmos propostos são: *SkipJack, RC5, RC6*, *TEA* e *DES*. A plataforma de *hardware* avaliada está baseada no embarcado *MICA2*. Este dispositivo é composto por um micro controlador *ATmega128L*, o qual é programado por uma linguagem de alto nível chamado *nesC*. Os parâmetros avaliados nesse trabalho foram: ciclos do *CPU*, tempo de execução, ocupação memória *RAM*, memória *ROM* e energia consumida. Os resultados encontrados nessa proposta permitem aceitar que é possível a implantação de algoritmos de segurança em uma *RSSF*, com o objetivo de não abater os recursos computacionais e garantir o funcionamento da plataforma.

O trabalho [MORENO et al. 2013] apresenta uma metodologia para o cálculo do consumo de energia através do uso de ferramentas de simulação. O objetivo da pesquisa é conhecer o consumo de energia, exigido pela *CPU* na execução do algoritmo de criptografia

AES[15], o *software* empregado na simulação é: *Mibench* baseado na arquitetura (*CPU*) e *Sim-Panalyzer* baseado na arquitetura (*ARM*). A técnica utilizada na medição de consumo de corrente está baseada em medir a caída de tensão em uma resistência. A avaliação foi feita com a seguinte parametrização: O uso de chaves de 128, 192, e 256 *Bits*, e dois arquivos com tamanhos de 100 e 500 *Kbytes*. Os resultados indicam que quando o tamanho da chave aumenta, o tempo de execução aumenta de forma proporcional, provocando também um aumento do consumo de energia, fato que era esperado.

O trabalho [NASH et al. 2005] propõe uma modelagem de um sistema de detecção de intrusões *IDS* baseado na identificação de perfil de consumo energético, associado a um sistema operacional através de um modelo matemático de regressão linear múltipla. Um sistema *IDS* está constituído por um núcleo central de processamento que avalia dados de entrada procurando padrões conhecidos, essa comparação é feita por meio de uma base de dados que contém assinaturas de ataques conhecidos. Uma assinatura tem capacidade de reconhecer uma cadeia de *Bytes* entre os dados de entrada coletados. O *IDS* compara os dados de entrada com as assinaturas registradas e no caso de coincidência se executa um processo de alerta evidenciando a anormalidade acontecida. A modelagem consiste em caracterizar um conjunto de variáveis conhecidas do sistema operacional, por exemplo: %*CPU*, %Memória, cada variável representará um término do modelo de regressão linear múltipla expressado desta forma: Y = a + b1 * X1 + b2 * X2 + bn * Xn. Está função proporciona uma curva característica, onde se *evidência* a ameaça e/ou ataque detectado. Os resultados alcançados em um nível experimental cumprem com as expectativas contempladas no modelo, baseado em regressão linear múltipla. Na Figura 2.1 apresentam-se os resultados alcançados pelo modelo anteriormente proposto.

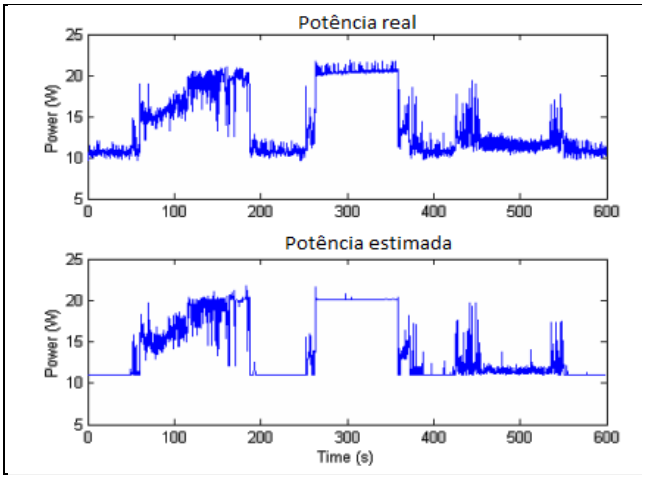


Figura 2.1: Comparação entre a potência real e a estimada [38].

[15] Advanced Encryption Standard

Na Tabela 2.3 apresentam-se as comparações feitas entre as distintas propostas analisadas. Observa-se um amplo uso de hardware multipropósito aplicado no contexto do consumo de energia.

Tabela 2.3: Resumo dos trabalhos baseados em sistema de medições

Referência	**Arquitetura**			
	Plataforma		**Parâmetro**	**Aplicação**
	Hardware	**Software**		
[34]	Adaptador Bluetooth USB classe 1 Chip CSR BlueCore 02	SO Windows	(RRSI) Intensidade sinal recebido (dBm) (GRPR) categoria potência recebido (dBm) (TPL) Potência transmissão (dBm) (LQ) Qualidade do Link. I(f) Variação corrente (mA) (LQ) Qualidade do Link Tamanho pacotes (KB)	Sistema de medição de consumo energético em sensores sem fio.
[35]	Não especificada	Ferramenta Simulação Tossim software R estatístico	Consumo energia (J).	Sistema de medição de consumo energético em RSSF
[36]	Sensor Mica2 baseado Chip ATmega128L	SO TinyOS - Tossim Criptografia MAC algoritmos: 1. SkipJack 2. RC5 3. RC6 4. TEA 5. DES	Consumo CPU (μJ) Tempo CPU (ms) Ciclos CPU Memória ROM (KB) Memória RAM (KB) Vazão mensagens (bytes) Tempo atraso (Seg) Consumo energia (mJ)	Sistema de medição de consumo energético em algoritmos de criptografia em RSSF
[37]	Computador Intel 2.2GHz, 512 RAM Osciloscópio OwonPDS5022S	SO Linux Ferramenta Simulação Mibench baseado (CPU) e Sim-Panalyzer baseado (ARM)	Consumo energia (J) Tamanhos chaves 128-196-256 Bits Tamanhos Arquivos 100 e 500 KB	Sistema de medição de consumo energético em algoritmos de criptografia.
[38]	Não especificada	SO Windows NT 40 Algoritmo criptográfico avaliado AES Ferramenta captura dados: Monitors counter data	Consumo energia (W) Tempo medição (Seg)	Sistema de medição de consumo energético em sensores sem fio.
Esta dissertação	Plataforma Arduino Uno R3Baseado no sensor de corrente	Sistema operacional Windows 7 executando framework Matlab	Consumo de corrente (mA)	Sistema de medição de consumo de corrente

2.4 Fundamentos do Serviço de Acesso Remoto *SSH*

O Autor [LINCK et al. 2016] propõe uma arquitetura de análise de *logs*, baseado em um conjunto de ferramentas de administração de redes, estas ferramentas são: Análise de *logs*

com *IpTables*, análise de *logs* do servidor *DHCP*, análise de *logs* da ferramenta *ArpWatch* e análise de *logs* de um servidor de Acesso Remoto *SSH* atacado por ferramentas de força bruta. O objetivo de analisar os *Logs* do serviço *SSH*, está baseado nas tentativas inconsistentes de acessos registrados como inválidos, identificar e classificar as direções de ataques, a fim de reconhecer os diferentes tipos de ataques. O trabalho apresenta uma arquitetura de caracterização de ameaças, fundamentada em um complexo sistema de análise de *logs*, proveniente de distintas ferramentas utilizadas.

O autor [PONOMAREV et al. 2014] apresenta um esquema de detecção de ameaças, baseado no tempo de inicio de uma conexão, ou seja a análise do tempo de conexão de um cliente valido frente a um cliente não valido, no momento que o servidor recebe uma solicitação de conexão, e até que está última finaliza. Esse tempo transcorrido é alvo de estudo através de uma análise estatística, logrando-se identificar um tipo de ameaça. Na Figura 2.2 apresenta-se o resultado alcançado onde observa-se que o pacote 5 e em menor medida o pacote 10, apresenta um desvio padrão do pacote maior frente a todos os demais, levando a identificar um tipo de ataque.

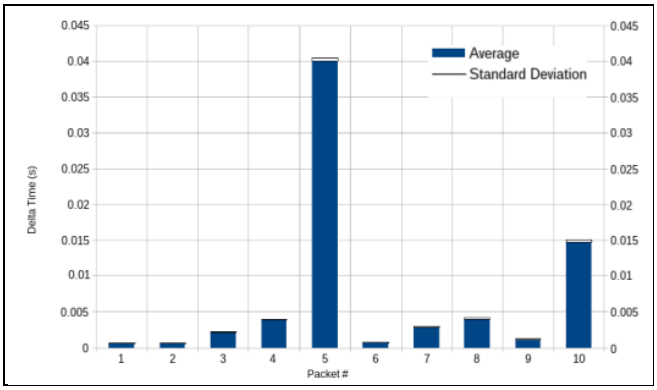


Figura 2.2: Identificação de um ataque através do desvio padrão de pacotes [40].

O autor [VYKOPAL et al. 2009] apresenta um método de detecção de ameaça, baseado em coleta de fluxos de dados associado ao serviço de acesso remoto *SSH*. A detecção é feita através de um algoritmo denominado *SSH-tree*, ou seja um algoritmo fundamentado em tomada de decisões, os fluxos de pacotes são divididos através de n-tuplas da forma n(x,y) onde cada valor de x e y representa uma variável, por exemplo n (*IP* atacante, porta destino) dessa maneira a tupla vai recorrendo a árvore em procura de alguma coincidência. Os resultados apresentam uma identificação tanto de acessos concedidos, como também de acessos negados em termos gerais o algoritmo apresentado cumpre com seu objetivo. A técnica de *NetFlow* utilizada nesse trabalho é uma área de evolução atual nas redes de

comunicações, particularmente o trabalho apresentado avalia uma ameaça ao serviço de acesso remoto *SSH* com a técnica proposta.

O trabalho [HOFSTEDE et al. 2014] apresenta um método de detecção de ameaça ao serviço de acesso remoto *SSH* através da técnica *Netflow*. Está técnica permite identificar padrões de ameaças baseadas em um ataque de força bruta com dicionários. A técnica define um método de captura por pacote que logo é codificado através de um fluxo de transmissão, começo, execução e fim da transmissão. O algoritmo proposto consegue identificar as atividades de dois cenários de avaliação propostos, o cenário 1 está composto de um nível de segurança baixo obtendo um valor de 83% de identificação de ataques validos, já no cenário 2 o nível de segurança é mais elevado, por enquanto identifica-se um valor de 99% de ataques.

O trabalho [VYKOPAL et al. 2009] apresenta uma infra-estrutura *NetFlow* de coleta de ataques baseado em *Honeypots* de alta interação, onde o serviço principal avaliado é um servidor de acesso remoto *SSH* executando-se em um servidor *Vmware*. Tudo o trafego é identificado sob uma tupla da forma x (y,z,w,u) onde cada variável é um atributo do pacote *TCP*. A plataforma de coleta de ataques proposta desempenha as tarefas encomendadas mais não apresentam-se dados quantificáveis de fácil interpretação.

Na Tabela 2.4 apresenta-se uma comparação dos trabalhos analisados anteriormente, observa-se as diferentes técnicas, metodologias e algoritmos utilizados, as métricas de avaliação de cada cenário proposto em função do contexto de aplicação, ou seja os serviços mais vulneráveis e as aplicações mais utilizadas para gerar ataques, assim também as ferramentas de proteção utilizadas. A elevada taxa de ataques identificada ao serviço de acesso remoto *SSH* é alvo de estudo nesta dissertação.

Tabela 2.4: Resumo dos trabalhos baseados no serviço de acesso remoto *SSH*

Referência	Arquitetura					
	Técnicas, metodologias e algoritmos	**Métrica**	**Contexto de aplicação**	**Serviços**	**Aplicações avaliadas**	**Aplicação**
[39]	Análise de log: 1.- IPTABLES 2.- DHCP 3.- ARPWATCH 4.- SSH	Bloqueio IP Bloqueio MAC	Ataque DDNS Contexto: 1.- IDS 2.- IPS	DNS DHCP SMTP SSH	SNORT HONEYPOTS SCAN PORT	Metodologia de detecção de ataques a serviços.
[40]	KMEANS DBSCAN	Pacote por tempo	Tipos de Ataque: IP Spoofing	SSH DHCP V6 HTTP TCP UDP	WIRESHARK TCPDUMP	Proposta de índices de avaliação de ataques a serviços.
[41]	Análise de arquivos log: 1.- NetFlow 2.- Analysis Algoritmo SSH-D-Tree	Falso Positivo Falso Negativo	Tipos de Ataque: dicionário Contexto: 1.- IDS 2.- IPS	SSH	LOGWATCH	Técnica de identificação de ataques ao serviço SSH.
[42]	Análise de arquivos log: 1.- NetFlow IPFIX Analysis	Falso Positivo Falso Negativo V. Positivo V. Negativo	Tipo de Ataque: SSH Dictionary Attack	SSH	EXPECT FAIL2BAN SSHDFILTER SSHBLOCK SSHGUARD HONEYPOTS	Técnica de identificação de ataques ao serviço SSH.
[43]	1.- NetFlow Analysis 2.- FlowMon Traffic Control	Falso Positivo Falso Negativo	Tipo de Ataque: SSH Dictionary Attack	SSH TCP ICMP HTTP	HONEYPOTS SCP PUTTY WINSCP SFTP RSYNC NFDUMP	Técnica de identificação de ataques ao serviço SSH.
Esta dissertação	Ataque de força bruta	Tempo duração do ataque	Ataque de força bruta com dicionário em um ambiente de teste controlado	SSH	MEDUSA HYDRA METASPLOIT	Ataques especializado do serviço SSH

2.5 Fundamentos das Técnicas de Ataques por Canais Laterais

O trabalho [BAYRAK et al. 2013] apresenta o desenvolvimento de um *hardware* criptográfico baseado no algoritmo *AES-128* executado em um dispositivo embarcado. As técnicas baseadas em ataques por canais laterais apresentam métodos com o alvo de obter a chave de cifra conhecendo o algoritmo, isto é possível através do registro de amostras de consumo de potência gerado pelo dispositivo. O funcionamento é baseado no relacionamento que existe entre o consumo e as instruções executadas pelo dispositivo. O *hardware* de proteção proposto possui a capacidade de gerar ruído, com o alvo de distorcer o consumo de energia e desta forma pode-se dificultar o conhecimento da senha de encriptação.

O trabalho [THANH-HA et al. 2008] apresenta as técnicas mais utilizadas em tipos de ataques por canais laterais, as técnicas classificadas em duas classes são: a primeira técnica é chamada ataques sem referência ao dispositivo que apresentam duas variações: *DPA*[16] e *CPA*[17]; A segunda técnica é chamada ataques com referência ao dispositivo que apresentam duas variações: *Template Attack*, *Stochastic Attack*. O Ataque por análise de consumo de potência desenvolve um método que coleta amostras do consumo de potência quando o dispositivo está realizando uma operação de cifra de dados. O ataque por análise de potência diferencial utiliza métodos estatísticos para vulnerar uma chave de cifra em base aos dados previamente coletados. O ataque por análise de correlação de potência desenvolve, um modelo real de consumo em base à amostra de potência frente a outro modelo ideal já feito, desta forma pode-se descobrir a senha de encriptação.

O trabalho [MERLO et al. 2014] apresenta duas técnicas, para realizar a medição de consumo de potência de um *Smartphone* que possui um sistema operacional baseado em *Android*. O primeiro método é através de um *software*, que interatua com uma *API* do sistema operacional. O segundo método denominado de baixo nível trabalha em uma camada inferior do sistema operacional, ou seja no núcleo do sistema que está baseado no *Kernel* de *Linux*. Os resultados alcançados apresentam evidências que o segundo método proposto obtém melhores resultados frente ao primeiro. O primeiro método utiliza os recursos do sistema operacional, com o fim de obter o consumo de potência logrando reproduzir tudo o processo de execução de uma aplicação normal. Em termos de consumo energético, a própria execução da aplicação proposta modifica o resultado final. Um segundo método possui um caráter invasivo respeito ao primeiro e está baseado em obter o consumo de potência diretamente desde o próprio *Kernel* do sistema operacional, obtendo desta forma um valor de consumo de potência mais exata.

O trabalho [MERLO et al. 2016] apresenta dois cenários de avaliação, a partir de um dispositivo móvel com o objetivo de obter o consumo de potência em função de um ataque chamado *ScanPort*, seguidamente realiza-se um teste de aplicações de segurança perseguindo o mesmo objetivo, com um *software Antivirus* e um *Firewall*. O primeiro cenário utiliza um ataque *Scanport* que realiza uma varredura de portas, a fim de obter os serviços que estão executando no dispositivo. O segundo cenário está baseado no uso de um *software Antivirus* e *Firewall* incorporado ao sistema operacional. Os resultados apresentam no primeiro cenário, um incremento de potência baseado no ataque *Scanport* no dispositivo de rede como também

[16] Differential Power Analysis
[17] Correlation Power Analysis

um incremento no *CPU* do dispositivo. O segundo cenário avalia o uso de aplicações que mitiguem as ameaças implicando um impacto positivo com respeito à segurança do dispositivo a nível do sistema operacional, no entanto observa-se um incremento no consumo de energia que impacta negativamente no rendimento.

Na Tabela 2.5 é apresentado um resumo dos trabalhos mencionados anteriormente destacando as técnicas de ataques identificadas, as métricas de avaliação mais utilizadas, as arquiteturas de *hardware* empregadas, os algoritmos propostos e finalizamos com as ferramentas utilizadas nos cenários de provas propostos.

Tabela 2.5: Trabalhos baseados em metodologias de ataques por canais laterais

Referência	**Arquitetura**					
	Ataque Proposto	**Métrica Avaliação**	**Arquitetura**	**Algoritmo**	**Ferramenta**	**Aplicação**
[44]	Single Power Analysis	Consumo energia	ASIC FPGA	AES-128	EDA Tools Synopsys Design Compiler	Metodologia variação de energia
[45]	SPA DPA Template Attack Stochastic Attack	Peak Número de sinais	-	-	-	Metodologia variação de energia
[46]	Medição de potência com API Kernel	Potência (mW)	Smartphones	-	Skype - Ping-Flood Powertutor	Metodologia variação de energia
[47]	Scanport	Potência (mW)	Smartphones	-	Powertutor	Metodologia variação de energia.
Esta dissertação	Ataque de força bruta ao serviço SSH	Consumo corrente (mA)	Raspberry Pi	-	Medusa Hydra Metasploit	Metodologia variação de energia

Na Tabela 2.6 apresenta-se um resumo das plataformas de hardware, software, dispositivos embarcados, técnicas e metodologias de medição de consumo energético no contexto de identificação de ataques. Desta maneira apresentamos o alcance desta dissertação, onde observa-se o uso das distintas plataformas embarcadas, no contexto de segurança de informação.

Tabela 2.6: Resumo geral do alcance da dissertação

Contexto	Plataforma Raspberry Pi	Plataforma Arduino	Plataforma de consumo energético	Serviço de Rede Avaliado SSH	Metodologia de Ataques por Canais Laterais
Esta Dissertação	Raspberry Pi 2 Model B SO Linux baseado Raspbian	Arduino Uno R3 Framework Matlab	sensor de corrente *ACS712ELCTR-05B*	Ataque de força bruta com dicionário em um ambiente de teste controlado Ao serviço SSH	Sistema de medição de consumo de corrente (mA)

2.6 Considerações Finais do Capítulo 2

A continuação apresenta-se um resumo baseado nos trabalhos e correlatos, visados anteriormente com o objetivo de fundamentar as bases teóricas e práticas que levam à eleição dos distintos sistemas intervenientes neste trabalho de dissertação, fundamentado em obter as curvas padrões do consumo de corrente empregando uma plataforma embarcada de baixo custo *Raspberry Pi 2 Model B*, trabalhando como um servidor de acesso remoto *SSH* avaliada com cenários de tipos de acesso e tipos de ataques de força bruta com dicionário através de ferramentas especializadas como *Medusa* [MEDUSA 2016] e *Hydra* [HYDRA 2016], assim também com ferramentas não especializadas como *Metasploit* [METASPLOIT 2016].

Seleção da Plataforma de *Raspberry Pi 2 Model B*

Neste ponto apresenta-se a plataforma embarcada *Open software* e *Open hardware* de baixo custo *Raspberry Pi 2 Model B*, como o sistema principal embarcado que cumpre a função de executar o sistema operacional *Linux Raspbian*[18]. A seleção desta plataforma embarcada foi selecionada principalmente por seu custo baixo de investimento de aproximadamente R$150 reais, além disso, por oferecer suporte ao sistema operacional por parte da comunidade desenvolvedora.

Considera-se o sistema operacional *Linux Raspbian* pelo fato que está baseado no *kernel* da conhecida distribuição de *Linux Debian*[19], também pelo suporte desta plataforma de software para sistemas embarcados. A justificativa destas plataformas é baseada nos trabalhos propostos apresentados anteriormente.

Seleção da Plataforma *Arduino Uno*

Neste ponto apresenta-se a plataforma embarcada de propósito específico *Arduino Uno-R3 ATmega328* como a plataforma base para o desenho de um sistema de aquisição, controle e comunicação de parâmetros elétricos baseado nos trabalhos propostos anteriormente.

A seleção desta plataforma está baseada no baixo custo de investimento de aproximadamente R$100 reais, além disso, por um amplo suporte de diversos tipos de sensores, e a capacidade de integração com um variado conjunto de ferramentas de análise e processamento de sinais digitais, particularmente este trabalho utiliza o *Framework Matlab*[20].

[18] www.raspbian.org
[19] www.debian.org
[20] www.matlab.com

Seleção do Serviço de Acesso Remoto *SSH*

Analisando os trabalhos propostos anteriormente no âmbito do serviço de acesso remoto *SSH*, foi identificado um tipo de ataque denominado ataque por força bruta com dicionário, este vetor de ataque está baseado em ferramentas especializadas para vulnerar um acesso a um sistema determinado, deve se destacar que este tipo de ataque está condicionado a duas questões, a primeira às senhas contidas no dicionário utilizado, e a segunda aos usuários avaliados nas provas de acesso.

Os ataques baseados em dicionários consistem em *Script* e ferramentas automatizadas que intentam adivinhar os usuários e as senhas no formato de *Login* (usuário:senha). As ferramentas propostas nos cenários de provas avaliados são: *Medusa* e *Hydra* que possuem um desenvolvimento otimizado para estes tipos de ataque, e pertencem ao contexto de ferramentas especializadas, no entanto apresentamos ao *software Metasploit* que é considerado um *Framework* de múltiplos ataques baseado no contexto de ferramentas não especializadas.

A importância de desenvolver novas ferramentas ou melhorar as existentes no contexto da segurança de serviços críticos como *SSH*, motiva a desenvolver novas técnicas e metodologias utilizadas em outros âmbitos de aplicações, por exemplo, na área de *hardware* reconfigurável temos as *FPGA*[21] e as *NetFPGA*[22], já no *software* temos as redes definidas por *software* conhecidas pelo acrônimo *SDN*[23].

Seleção das Técnicas de Mapeamento de Energia

Neste ponto apresentamos uma abordagem técnica de como realizar um mapeamento energético de um dispositivo embarcado através de canais não convencionais ou canais laterais.

A técnica empregada consiste em mapear o consumo de corrente do dispositivo embarcado através do uso de sensores específicos, neste caso utilizamos um sensor de corrente baseado no efeito *Hall* modelo *ACS712ELCTR-05B* que funciona na plataforma embarcada de medição de consumo energético baseado em *Arduino Uno*.

Outra aplicação é mapear o consumo de corrente de um dispositivo criptográfico no momento de criptografar uma mensagem, desta forma logra-se obter uma curva de consumo

[21] Field Programmable Gate Array
[22] http://netfpga.org/
[23] Network Define Software

que logo é processada por diferentes métodos com o alvo de descobrir sua senha de encriptação.

Em função desse principio propõe-se nesta dissertação mapear o consumo de corrente gerado pela plataforma de teste *Raspberry Pi 2 Model B* executando cenários de ataques definidos ao serviço de acesso remoto *SSH*.

Na Figura 2.3 é apresentada a arquitetura proposta que apresenta os sistemas e tecnologias utilizadas. O nível 1 possui a tarefa de fornecer os ataques através das ferramentas especializadas *Medusa* e *Hydra* e não especializadas *Metasploit* propostas, além disso, se define o dicionário recomendado com senhas de 30 caracteres equivalente a 240 *Bits*, está senha proposta está baseada tendo em conta que uma senha fraca é considerada de um tamanho de 128 Bits, porém uma senha forte é baseada em 256 Bits, assim o tamanha selecionado de 240 bits proposto nesta dissertação, é considerada uma senha de termo meio.

O nível 2 está formado pelas plataformas embarcadas *Raspberry Pi 2 Model B* com a função de executar o servidor de acesso remoto *SSH* na porta *22* do sistema operacional *Linux Raspbian*. Por outro lado, temos a plataforma embarcada *Arduino Uno* que funciona coletando as variações de corrente ocasionadas pela plataforma de teste avaliada sob os cenários de ataques definidos. Por último no nível 3 apresenta-se o *Framework Matlab* que possui a tarefa de interpretar as variações de corrente interpretadas pelo sensor de corrente *ACS712ELCTR-05B* do *Arduino Uno* que posteriormente são normalizadas como o método de *Welch* para sua interpretação gráfica.

Na Figura 2.4 apresentam-se os dispositivos de *hardware* embarcado utilizados nas provas experimentais efetuadas. A Figura 2.4 (a) apresenta no primeiro plano a plataforma *Raspberry Pi 2 Model B* com um tela *Touch Screen* táctil de 7 polegadas, em segundo plano observa-se a plataforma *Arduino Uno* de menor tamanho. A Figura 2.4 (b) apresenta no primeiro plano o computador encarregado de coletar os dados através do *Framework Matlab*, e também contém executando uma máquina virtual com o sistema operacional *Kali Linux 2.0* designado para fornecer os diferentes cenários de ataques propostos. Já a última Figura 2.4 (c) uma vista geral dos dispositivos embarcados utilizados nos testes.

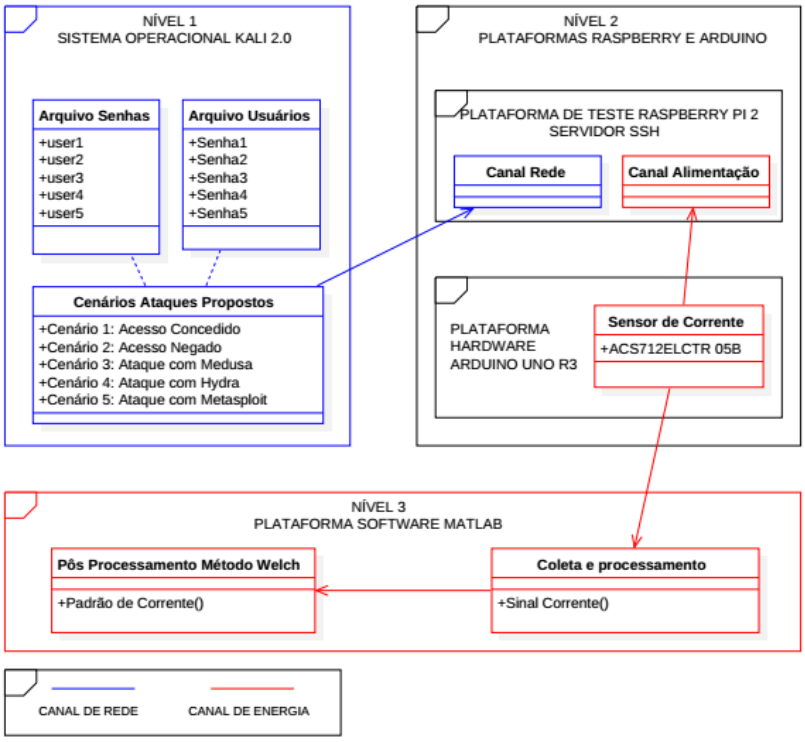
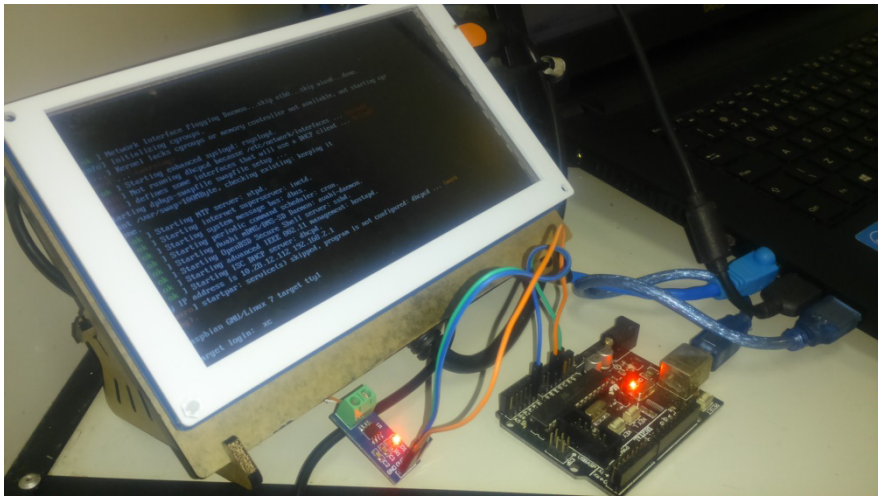
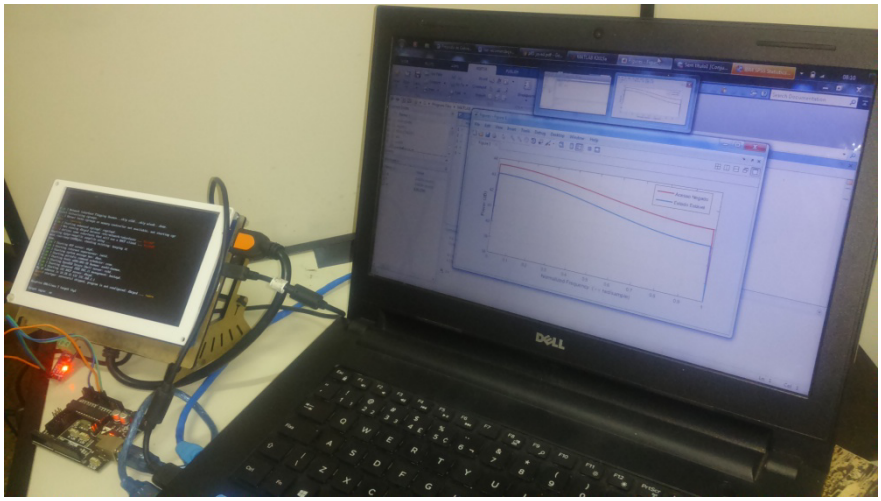


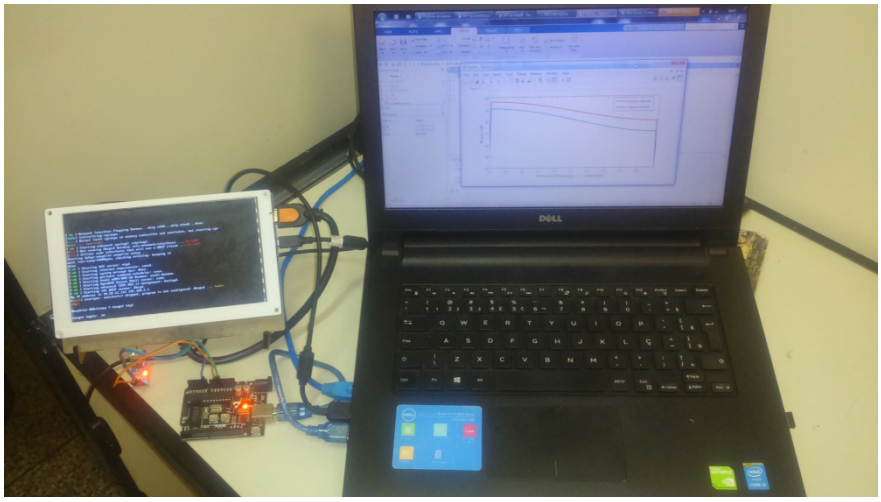
Figura 2.3: Arquitetura de medição proposta.



(a)



(b)



(c)

Figura 2.4: Dispositivos embarcados propostos.

CAPÍTULO 3

FUNDAMENTOS DE *HONEYPOT*

3.1 Fundamentação

Segundo o autor [48] define-se um *Honeypot* como um recurso computacional altamente monitorado o qual deseja-se que seja provado, atacado e comprometido.

Os *Honeypots* na atualidade se utilizam como mecanismo de segurança na rede, isto serve para detectar quando se produz uma intrusão e analisar as ações realizadas e executadas pelos atacantes. Estes sistemas simulam para o exterior como se tratassem de sistemas vulneráveis, no entanto, para os administradores resulta de vital importância observar as ações realizadas e compreender as estratégias utilizadas pelos atacantes na intrusão ao sistema [SPITZNER 2002].

A tradução inglesa da palavra *Honeypot* é compreendida como "potes de mel", que são iscas para invasores. Os *Honeypots* são conjuntos de aplicações ou computadores com a capacidade de atrair invasores, simulando sistemas vulneráveis [MOHAMMADZADEH et al. 2012] [MARCELO et al. 2003]. Neste contexto, uma configuração deficiente tanto em instalação como em administração é um exemplo válido.

Os *Honeypots*, em sua forma mais básica são sistemas contendo informação falsa, localizado estrategicamente em um mapa na rede, alem disso com um viés imposto que priorize a informação como classificada ou de um valor importante para a organização. Esta exposição de forma deliberada se faz atrativa para um intruso em busca de um objetivo de atacar. Em geral um sistema baseado em *Honeypots* possui outro supra-sistema que possibilita o registro de todas as atividades, com isto os administradores têm a capacidade de realizar uma análise posterior detalhando os movimentos efetuados pela intrusão, compreendendo as técnicas e metodologias utilizadas para vulnerar um sistema, aplicações e/ou serviços.

3.2 Tipos de *Honeypots*

A continuação apresenta-se os distintos níveis de aplicação dos tipos de *Honeypots* avaliados.

3.2.1 Baixa Interação

Este tipo de configuração limita o nível de interação entre o invasor e o *Honeypot*, ou seja o sistema está limitado se é uma aplicação; às funcionalidades disponíveis, se é um serviço; o nível de execução sob o sistema operacional. A vantagem fundamental desta característica é atribuída principalmente pela simplicidade de instalação e configuração, como ponto negativo, o nível de emulação é muito limitado em alguns casos torna-se ineficaz.

3.2.2 Alta Interação

Estes tipos de *Honeypots* são considerados sistemas e/ou aplicações reais ao contrário do parágrafo anterior, está disposição permite ao atacante fazer uma intrusão completa ao sistema e/ou serviço disponibilizado.

A vantagem principal com respeito ao nível de informação coletada é considerada altamente valiosa, como exemplo este tipo de desenho permite realizar uma detecção de um ataque do tipo *Zero-day* [RATINDER et al. 2014] vulnerabilidade extremamente importante no âmbito da segurança. Os recursos podem apresentar-se como aplicações ou serviços que possuem configurações por defeito ou mal feitas pelo administrador, ou também como versões com falhas de segurança que não possui as atualizações correspondentes.

A desvantagem do sistema ao não incluir os níveis de segurança adequados gera uma possível falha de segurança que um atacante pode ganhar, ou seja o sistema que contém o *Honeypot* pode chegar a ser comprometido.

Os *Honeypots* deste tipo possuem um grau de dificuldade maior em sua instalação e configuração, neste ponto estabelecer medidas de acordo com as necessidades de segurança é imperativo, em alguns casos deve-se delimitar e/ou demarcar perfeitamente entre um ataque e uma ação lícita ao sistema.

3.2.3 Sistemas Híbridos

Estes tipos de *Honeypots* combinam as características de baixa e alta interação, na Tabela 3.1 é apresenta uma análise de vantagem e desvantagem deste sistema.

Tabela 3.1: Vantagem e desvantagem dos *Honeypots*

Honeypots	
Baixa Interação	**Alta Interação**
Emulação da pilha protocolos TCP/IP	Sistemas operacionais, serviços e aplicações
Baixo risco	Alto risco
Nível baixo de instalação e administração	Nível complexo de instalação e administração
Coleta de dados menos detalhada	Coleta de dados mais detalhada
Baixa detecção de ataques	Detecção de novos ataques
Recursos hardware baixos	Recursos hardware altos

3.3 Classificação Segundo Propósito

Os sistemas baseados em *Honeypots* são empregados geralmente em dois tipos de cenários. O primeiro é utilizado com o objetivo de monitorar uma rede, além disso, como mecanismo de defesa. O segundo cenário persegue um objetivo puramente de pesquisa centrado na coleta e análise de ataques perpetrados na rede de estudo. O segundo cenário é avaliado neste trabalho de dissertação.

3.4 *Honeypots* de Pesquisa

Dentro de um entorno de investigação, os pesquisadores procuram descobrir novas técnicas e metodologias utilizadas pelas ameaças e/ou ataques, nesse contexto é possível classificar dois tipos de ameaças principais, as ameaças baseadas em ataques atribuídos a pessoas físicas, ou seja pessoas que são conhecidas com o nome de *Cracker* ou também como *Hacker*, está última caracterização segundo o autor deste trabalho possui certo grau de aceitação, e depende do nível de conhecimento ou formação que tenha uma pessoa sob o significado desta palavra. A continuação se justifica esta controvérsia generalizada por meios e pessoas pouco experientes nesta área de conhecimento.

Define-se um *Hacker* ético como uma pessoa com altos conhecimentos na área da informática em geral, que regrado sob princípios profissionais e éticos possui as capacidades de investigar, potenciais falhas de segurança em sistemas informáticos, com o objetivo de descobrir uma vulnerabilidade. Está definição apresenta duas questões:

A primeira está relacionada com a informação que apresentamos ao responsável da área do sistema vulnerado. A segunda está relacionada com os desenvolvedores do sistema e/ou aplicação, sob essas duas questões o autor justifica o correto uso da palavra *Hacker* ético, sem embargo é de amplo conhecimento pela comunidade, que existem pessoas que perseguem outros valores que não são éticos. Nessa concepção classifica-se a toda pessoa que através de seu conhecimento, explora e explode uma vulnerabilidade informática com atos delitivos, roubo de informação, acesso a dados privados ou qualquer outra asseveração que coloque em

risco um sistema informático, neste contexto é considerado um *Hacker* não ético. Conclui-se que um *Hacker* não ético é a antíteses do que conhecemos como *Hacker* ético.

A segunda categoria de ameaça que fazemos referência é pelas ameaças baseadas em *software*, capazes de vulnerar ou colocar em risco um sistema informático.

3.5 Plataforma de Teste Avaliada *MHN* (*Modern Honey Network*)

Apresenta-se uma plataforma de teste baseado em *Honeypots*. A plataforma de teste está baseada em um *software* denominado *MHN* (*Modern Honey Network*) que possui um conjunto de aplicações que são utilizadas como *Honeypot***.** A plataforma proposta está composta da seguinte maneira, na Figura 3.1 um diagrama de blocos que apresenta todos os componentes da arquitetura.

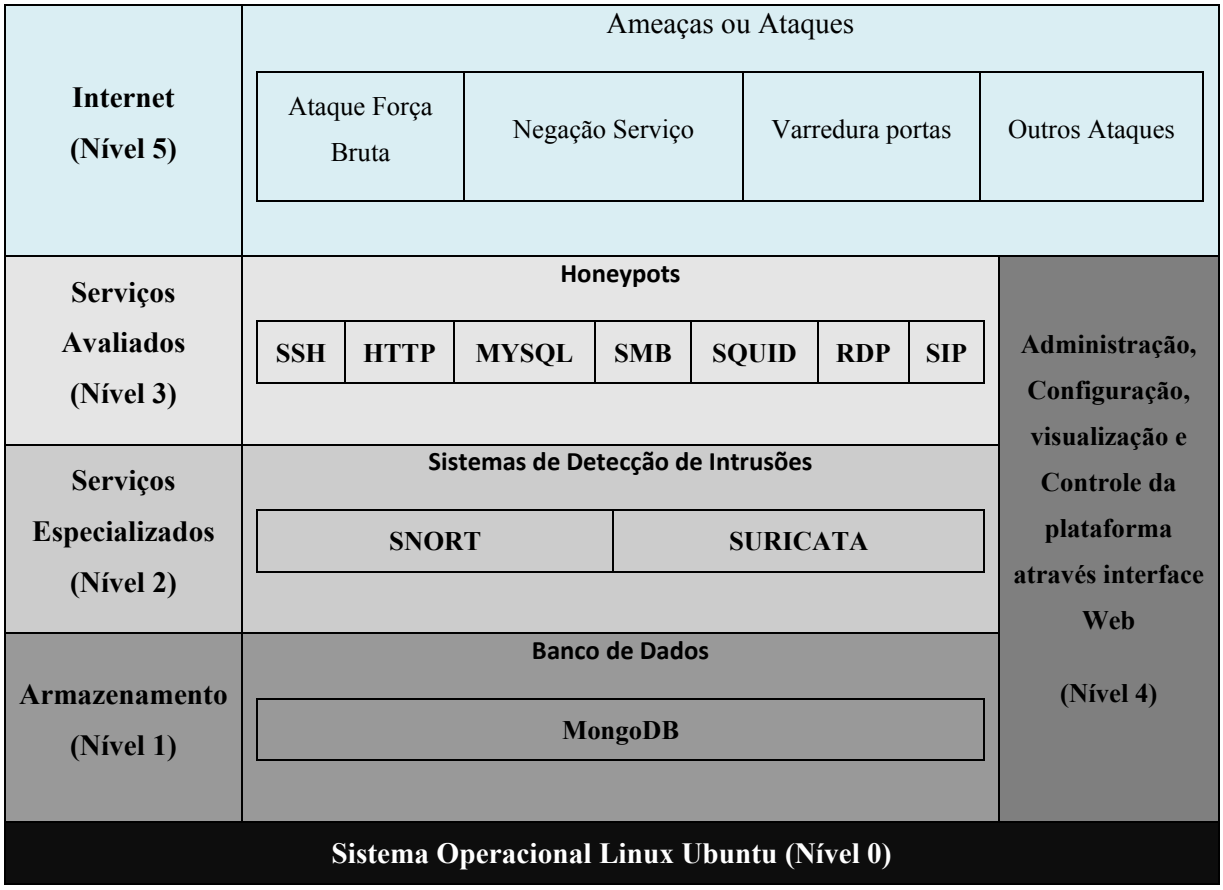


Figura 3.1: Plataforma de teste proposta baseada em *MHN*.

No (Nível 0) é a base que sustenta toda a plataforma e é baseada no sistema operacional *Linux*, particularmente para este dissertação é utilizada a versão de *Ubuntu 14*, este sistema se encontra alocado em um servidor virtual alugado na nuvem pelo serviço da empresa *DigitalOcean*[24]

[24] https://www.digitalocean.com/.

No (Nível 1) está formado por um banco de dado que realiza a função de administração do armazenamento das ameaças e/ou ataques coletados pelos *Honeypots*. O gestor da base de dados empregado é *mongoDB*[25], que utiliza um paradigma de trabalho baseado em dados não estruturados (*NoSQL)*.

No (Nível 2) está formado por serviços especializados que são instalados em função das necessidades especificas de cada teste a realizar particularmente, neste trabalho realizou-se a instalação dos serviços que formam parte da família dos sistemas de detecção de intrusões, estes *softwares* são: *Snort* [SAGALA 2015] e *Suricata* [ALBIN et al. 2012].

Um sistema de detecção de intrusões (*IDS*) possui a função de identificar uma ameaça baseada em regras padrões. Essa detecção origina uma alarme e registra a ocorrência através de um arquivo *Log*, fornecendo uma resposta. Os métodos de detecção utilizados pelos *IDS*, a fim de identificar uma ameaça segundo os autores [BUTUN et al. 2014] [CHAUHAN et al. 2012] são:

1. Detecção de padrões (*Pattern Matching*);
2. Detecção de padrões que mantém o estado (*Stateful Pattern Matching*);
3. Análise de protocolos (*Protocol decode*);
4. Análise heurística (*Heuristic analysis*);
5. Análise das anomalias (*Anomaly analysis*).

O escopo abordado neste trabalho está baseado na primeira opção, detecção de ameaças por reconhecimento de padrões. O método funciona identificando pacotes de dados através de atributos com alvo de descobrir tráfico malicioso. A regra padrão definida como regra de identificação só é testada para determinado serviço, determinada porta, determinada combinação de *FLAGS* e determinado inicio e fim de seu conteúdo. Um exemplo do funcionamento de uma detecção feita com este método consiste em acionar uma alarme se o pacote é *IP*, tipo de protocolo é *TCP,* porta é número 22 e *FLAGS* é *SYN*, está regra pode identificar um ataque ao serviço de acesso remoto *SSH*.

Estas regras são definidas em um arquivo principal de assinaturas de ameaças utilizadas pelos *IDS*, com o alvo de detectar ameaças e/ou ataques. No Quadro 3.1 mostra-se uma regra padrão extraída do arquivo principal de regras utilizadas pelos *IDS Snort* e *Suricata* na plataforma de teste *MHN*.

Quadro 3.1: Regra de identificação de um ataque ao serviço *SSH*

```
ALERT TCP $EXTERNAL_NET ANY -> $HOME_NET 22 (MSG:"ETSCAN POTENTIAL SSH SCAN"; FLAGS:S,12)
```

[25] https://www.mongodb.com/

Feita uma aclaração do funcionamento de um sistema de detecção de intrusões, apresentam-se os *IDS* avaliados na plataforma de teste *MHN*.

O *IDS Snort*[26] é um detector de intrusos baseado em rede, possui um motor de detecção de ataques e/ou ameaças que permite registrar tudo o tráfico malicioso em uma rede de dados, a arquitetura de funcionamento é baseado em *Single-Threaded*.

O *IDS Suricata*[27] apresenta o mesmo conceito que *Snort*, a diferença principal está na arquitetura de funcionamento que é baseado em *Multi-Threaded*.

A continuação na Figura 3.2 apresenta-se a arquitetura de funcionamento de *Snort*, formada por um módulo de *Network Backbone* (rede), *Sniffer*, *Preprocessador*, Mecanismo de detecção, e por último um módulo de alertas e armazenamento.

Network Backbone (rede): É a rede que deseja analisar.

Sniffer: É o primeiro passo de captura, de decodificação de um pacote de dados que armazena e classifica todos os pacotes, por exemplo, *IP* de origem e destino.

Preprocessador: É uma camada que permite acrescentar o funcionamento do detector através de módulos adicionais.

Mecanismo de detecção: É uma camada que inspeciona o conteúdo do pacote para comparar com os padrões ou regras de assinaturas (conjunto de regras ou rules).

Alertas e armazenamento: É a última camada de processamento de *Snort* que quando detecta um pacote suspeito, por uma decisão do preprocessador ou através do mecanismo de detecção, este módulo gera uma alarme, que pode ser armazenado em um arquivo de texto ou em um banco de dados externo.

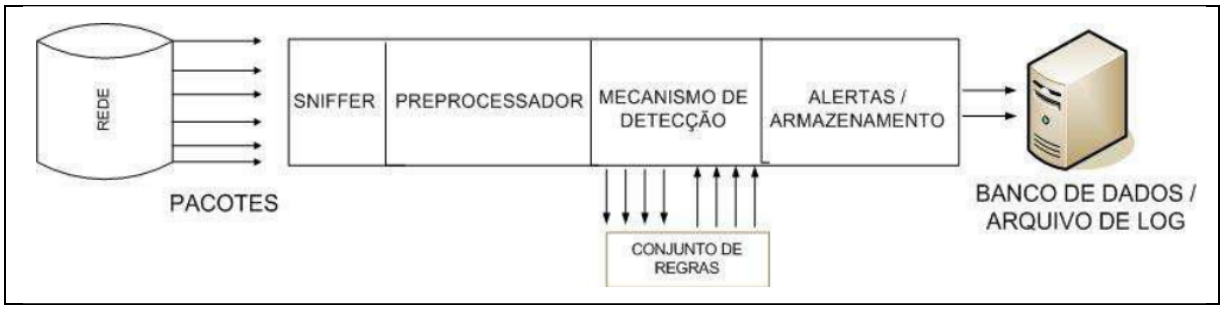


Figura 3.2: Arquitetura de funcionamento do *IDS Snort*.

No (Nível 3) está formado por uma camada que contém os diferentes *Honeypots* a avaliar, este grupo está identificado pelos seguintes serviços: *SSH* (22), *HTTP* (80), *MYSQL* (3306), *SMB* (445), *SQUID* (3128), *RDP* (3389), *SIP* (5060). Particularmente na plataforma

[26] https://www.snort.org
[27] https://suricata-ids.org

de teste *MHN*, foi instalado e configurado o *Honeypot Dionaea* que possui todos os serviços anteriormente mencionados.

No (Nível 4) está formado por uma camada que é responsável pela administração de toda a plataforma de teste proposto *MHN*. O alcance deste nível compreende tudo o domínio ou âmbito do sistema, ou seja a instalação, administração, eliminação de *Honeypots*, tudo isto é possível através de uma interface *Web* que o sistema disponibiliza. Na Figura 3.3 apresenta-se a interface *web* da plataforma de teste *MHN*.

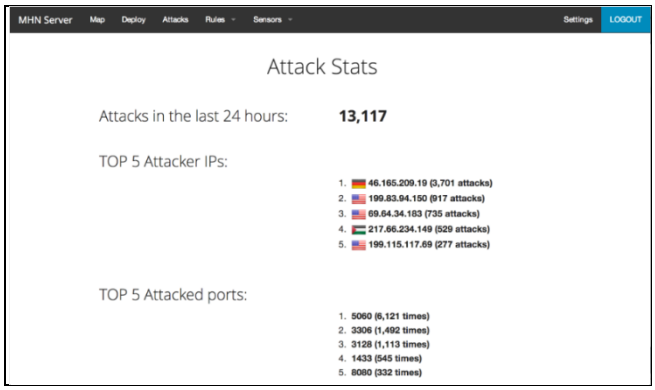


Figura 3.3: Interface *Web* da plataforma de teste *MHN*.

No (Nível 5) apresenta-se um camada denominada *Internet* onde o universo de discurso presumível corresponde a ataques e ameaças conhecidas *DDos*, Varreduras de portas, Ataques de Força bruta, e os ataques desconhecidos atribuídos a *ZeroDay*.

Apresentada a plataforma de teste *MHN* baseada em *Honeypots*, a continuação uma explicação dos serviços de redes avaliados, com o objetivo de compreender quais são os serviços com maior taxa de ataques. Os serviços disponibilizados na plataforma *MHN* são identificados a partir do acrônimo e o número de porta por defeito: *SSH* (22), *HTTP* (80), *MYSQL* (3306), *SMB* (445), *SQUID* (3128), *RDP* (3389), *SIP* (5060). A continuação um resumo dos serviços disponibilizados.

O serviço *SSH* é um protocolo de comunicação que através de aplicações especificas de *software*, por exemplo, *OpenSSH* cifra o canal de comunicação entre uma origem e um destino, o serviço brinda a possibilidade de administrar um sistema em forma parcial ou completa, dependendo do tipo de privilégio outorgado através de um intérprete de comandos, a porta por defeito utilizada na plataforma de teste *MHN* é a porta 22 **(RFC-4250/4251/4252/4253/4254)**.

O serviço *HTTP* é um protocolo de transferência de hipertexto utilizado na *web,* este serviço facilita a sintaxe e semântica que utilizam os distintos servidores e clientes *web* para

comunicar-se entre si, a porta por defeito utilizada na plataforma de teste *MHN* é porta 80 **RFC-2616**.

O serviço *MYSQL* está associado ao servidor de banco de dados chamado pelo mesmo nome, este banco de dados é considerado o sistema mais popular no ambiente *Open Source*, a porta por defeito utilizada na plataforma de teste *MHN* é porta 3306.

O serviço *SMB* é um protocolo de rede que permite compartilhar arquivos, impressoras entre computadores em um ambiente *Microsoft Windows*, a porta por defeito utilizada na plataforma de teste *MHN* é porta 445.

O serviço *Squid* é um servidor *Proxy* para *Web* com *cache*, é uma das aplicações mais populares em ambiente *Linux*, como característica principal suporta um amplo número de protocolos conhecidos, *HTTP*, *FTP*. A porta por defeito utilizada na plataforma de teste *MHN* é porta 3128.

O serviço *RDP* é um protocolo de comunicação que permite conectar um cliente remoto a um servidor, por meio, de uma interface gráfica desenvolvido por *Microsoft*. A informação gráfica que gera o servidor é convertida a um formato próprio *RDP* e enviada, através da rede ao terminal que interpreta a informação contida no pacote do protocolo para reconstruir a imagem, e apresentar na tela do terminal. Este serviço é utilizado para o acesso remoto na administração de sistemas, a porta por defeito utilizada na plataforma de teste *MHN* é porta 3389.

O serviço *SIP* é um protocolo desenvolvido pelo grupo de trabalho *Mmusic* do *IETF* com ideia de ser um estandarte para a iniciação, modificação e finalização de sessões interativas de usuários onde existem elementos de multimídia como a voz, vídeo, mensagem instantânea, jogos on-line e realidade virtual. A porta por defeito utilizada na plataforma de teste *MHN* é porta 5060 **RFC-3261**.

3.6 Considerações Finais do Capítulo

Propõe-se ministrar um experimento controlado fundamentado na coleta de ameaças através da plataforma *MHN* com os seguintes serviços: *SSH* (22), *HTTP* (80), *MYSQL* (3306), *SMB* (445), *SQUID* (3128), *RDP* (3389), *SIP* (5060).

O objetivo do experimento é compreender em termos quantitativos quais são as taxas de ataques relacionadas diretamente aos serviços mencionados anteriormente.

Está avaliação de serviços através do experimento proposto é considerado prioritário na pesquisa, pois foi possível delinear os tipos de ataques identificados justificando assim a definição dos cenários de provas propostos nesta dissertação.

CAPÍTULO 4

IDENTIFICAÇÃO DE ATAQUES ATRAVÉS DA PLATAFORMA *MHN*

A metodologia selecionada para a projeto experimental é o proposto por [WOHLIN et al. 2000], esta seção se concentra na definição do objetivo e planejamento do experimento.

4.1 Objetivos

O objetivo foi formalizado com o modelo *GQM* (*Goal, Question, Metric*) proposto por [SOLINGEN et al. 1999]: Analisar qual é a quantidade de ataques detectados, pelos diferentes sistemas de detecção de intrusões *Snort* e *Suricata*, desde o ponto de vista dos administradores no contexto da segurança de informação.

4.2 Formulação da Hipótese

As questões de pesquisa proposta para este experimento estão focadas em responder às seguintes questões quantitativas e avaliadas através das métricas sugeridas.

Características quantitativas:

1. Qual é a quantidade de ataques reconhecida pelas plataformas *Snort* e *Suricata* em uma análise global.
2. Qual é a quantidade de ataques reconhecida pelas plataformas *Snort* e *Suricata* em uma análise por serviço.

Métrica propostas:

1. A quantidade total de ataques entre as plataformas avaliadas.
2. A quantidade total de ataques entre as plataformas avaliadas por serviço.

HIPÓTESE 1: Quantidades de ataques.

H_0: A quantidade de ataques reconhecida pela plataforma *Snort* é igual à quantidade de ataques reconhecida pela plataforma *Suricata*.

$$H_0: \mu_{Snort} = \mu_{Suricata}$$

H_1: A quantidade de ataques reconhecida pela plataforma *Snort* é diferente à quantidade de ataques reconhecida pela plataforma *Suricata*.

$$H_1: \mu_{Snort} \neq \mu_{Suricata}$$

Para a presente hipótese, a H_0 é a que deseja-se rejeitar. Para averiguar qual será rejeitada serão consideradas as seguintes variáveis dependentes e independentes apresentadas na Figura 4.1.

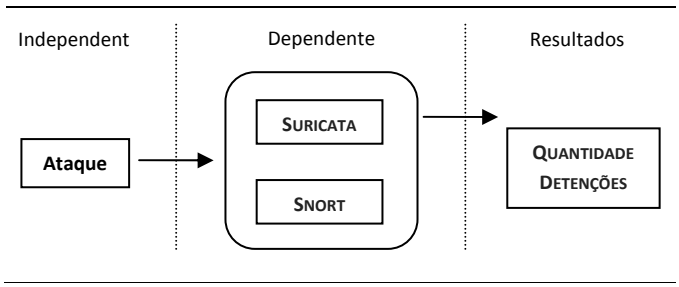


Figura 4.1: Variáveis do experimento

4.3 Seleções de Variáveis

Nesta seção são selecionados os tipos de variáveis [BASILI et al. 1988]. Dentro do experimento identificamos dois tipos de variáveis, que são parte do processo de análise. Consideram-se estas variáveis como únicas neste contexto de trabalho. As variáveis identificadas são apresentadas no seguinte ponto.

4.3.1 Variáveis Independentes

Estas variáveis são aquelas que podem ser controladas e alteradas dentro do experimento, e são críticas para a abordagem de cenários experimentais, dentro deste grupo definimos as seguintes variáveis:

- *Destination IP*: Direção *IP* do sistema atacado.
- *Destination Port*: Apresenta as portas onde se identificam os distintos serviços disponíveis da plataforma de teste *MHN*.

4.3.2 Variáveis Dependentes

Estas variáveis representam a saída de um processo ou sistema e são críticas para a tomada de decisão, para este experimento propõem-se as seguintes variáveis.

- *Source IP*: Direção *IP* atacante.
- *Source Port*: Porta associada *IP* atacante.
- *TCP Flags*: Atributo do pacote *TCP* que representa um estado particular.

- *Protocol*: Tipo de protocolos comuns, *TCP*, *UDP* e *ICMP*.
- *Signature*: Padrão de reconhecimento de ataques.
- *Channel*: Apresenta o tipo de *IDS* utilizado.

4.4 Seleção de Amostra

A plataforma de teste *MHN avaliada* possui aproximadamente (N = 60.000) registros coletados em diversos tipos de ataques, desta população procede-sea tomar uma amostra aleatória com uma porcentagem de 10% contabilizando então (n = 6.000) registros validos. A amostra anterior está equilibrada na quantidade de registros avaliados pelas plataformas *IDS Snort* e *Suricata*, ou seja os dados não apresentam nenhuma tendência ou desvio para um *IDS* em particular.

4.5 Escolha Nível de Significância

O nível de significância escolhido é de 99%, portanto o $\alpha = 0,05$.

4.6 Instrumentação

A plataforma de teste *MHN* foi inicialmente instalada e configurada sob um ambiente controlado em uma rede local *in-situ,* desta forma avaliamos o sistema em sua totalidade garantindo o normal funcionamento dos serviços propostos, superada esta etapa a plataforma de teste *MHN* foi trasladada para um serviço chamado *Simple Cloud Computing* que provém um *Virtual Private Server* (*VPS*) da empresa *DigitalOcean*, desta forma a plataforma de teste *MHN* está *Online* na Internet coletando ataques no seguinte endereço (*https://malware.ufs.singapure*), a plataforma de teste *MHN* esteve coletando ataques por 90 dias, desde o dia 1 de Setembro até 1 de Dezembro do 2015.

4.7 Instrumentação de Objetos

Instrumentação em nível de sistema operacional e ambiente de trabalho.

Uma breve descrição da configuração do sistema onde o experimento será executado.

- Plataforma de *hardware* utilizada possui 1GB Memória, 1 Core de processamento, 30 GB HD. Alocação da plataforma de teste.
- Instalação do sistema operacional: Distribuição *Linux Ubuntu 14-04 LTS 64Bit*.
- Instalação da Plataforma de teste *MHN*.
- Instalação e configuração do *HoneyPot Dionaea*, *Snort* e *Suricata*.

4.8 Guias de Instrumentação

A continuação no Quadro 4.1 apresentam-se a ordem que inicia o processo de instalação da plataforma de teste *MHN* através do seguinte comando (*Sudo ./install*), os seguintes parâmetros de configuração que seguem forma parte da configuração geral do sistema.

Quadro 4.1: Processo de instalação da plataforma *MHN*

```
$cd /opt/
$sudo apt-get install git -y
$sudo git clone https://github.com/threatstream/mhn.git
$cd mhn/
$ sudo ./install.sh
================================================
MHN Configuration
================================================
Do you wish to run in Debug mode?: y/n n
Superuser email: usuário@usuário.com
Superuser password:xxxxxx
Server base url ["http://104.131.170.135"]:
Path for log file ["mhn.log"]
```

O Quadro 4.2 apresenta-se o serviço do *Honeypot Dionaea* executando-se corretamente na plataforma de teste *MHN*, com o número pid 28343 que identifica o processo no sistema operacional e funcionando *Online* por 20 minutos aproximadamente.

Quadro 4.2: Execução do serviço *Dionaea* na plataforma *MHN*

```
root@:104.131.170.135/opt/mhn/scripts$ sudo supervisorctl status
Dionaea                          RUNNING   pid 28343, uptime 0:20:10
```

A continuação apresentam-se os passos de configuração do *HoneyPot Dionaea*, que permite ministrar os serviços avaliados neste trabalho, através do uso da plataforma de teste *MHN* baseado em *HoneyPots*. O *Honeypot Dionaea* é instalado e configurado através do sistema de administração web. Feito o processo de instalação procede-se à comprovação do funcionamento, por meio do seguinte arquivo de configuração chamado *dionaea.conf*. Os parâmetros básicos de configuração são apresentados na Tabela 8, e está composta pelo nome do serviço, sua porta lógica e o endereço *IP* público.

Tabela 4.1: Configuração dos parâmetros do *HoneyPot Dionaea*

SERVICES	**PORT**	**HOST IP**
SSH	22	104.131.170.135
HTTP	80	
SMB	445	
MSSQL	1433	
SQUIT	3128	
RDP	3389	
SIP	5060	

Nesta etapa da pesquisa o sistema *MHN* está em funcionamento coletando dados.

Terminada a etapa da coleta de dados, procede-se executar um procedimento de migração dos dados armazenados no sistema. No Quadro 4.3 apresenta-se o comando executado com o objetivo de exportar os dados coletados a um arquivo separado por vírgulas.

Quadro 4.3: Comando de manipulação dos dados armazenados em *MongoDB*

```
Mongoexport db mnemosyne collection session fields=source_ip, destination_ip csv > nome_arquivo.csv
```

O Quadro 4.4 apresenta os parâmetros utilizados no comando anterior.

Quadro 4.4: Atributos do comando *Mongoexport*

```
mongoexport: comando para exportar dados.
--db nome da base de dados.
--collection nome da Tabela que contém os dados.
--fields nome dos campos que precisamos.
--csv formato do arquivo a gerar.
```

4.9 Avaliação da Validade

A fim de garantir a validade dos dados no experimento e seguindo as recomendações de [SHADISH et al. 1979], consideramos que o âmbito de nossos dados do estudo corresponde à validade interna.

A plataforma de teste *MHN* foi avaliada com o objetivo de descobrir erro de configuração e/ou funcionamento.

Os testes propostos de *software* são:

- Funcionamento da plataforma de teste *MHN*.
- Funcionamento de gestor base de dados *mongoDB*.
- Prova de funcionamento da plataforma de teste *MHN* ao serviço *SSH* com usuário e senha não válida. Este evento provoca a criação de um registro de acesso não valido por parte do servidor de acesso remoto *SSH* e é armazenado em um arquivo log do sistema operacional.

4.10 Operação

Nesta fase procede-seà execução do experimento proposto por parte da plataforma de teste *MHN* baseada em *Honeypots*, que avaliam um grupo de serviços selecionado que se apresento anteriormente. Este processo de disponibilizar a plataformas de teste consta das seguintes três etapas:

1. Preparação: Este processo abrange os objetos envolvidos no desenvolvimento do experimento. O provedor do serviço *VPS* onde se executa nossa aplicação, a instalação e configuração da plataforma de teste *MHN* baseada em *HoneyPots*.

2. Execução: O objetivo é garantir o funcionamento correto pela plataforma de teste *MHN*. Propõe-se uma execução de intentos de acessos não validos através do serviço *SSH,* este evento gera um registro de acesso não valido que é armazenado em um arquivo *Log* dentro do sistema operacional. Na Tabela 4.2 apresenta-se um exemplo de um registro de acesso não valido extraído do arquivo *Log* gerado pelo serviço acesso remoto *SSH*.

Tabela 4.2: Registro de acesso não valido ao serviço *SSH*

Tempo	**ID**	**Mensagem**
11/02/2016 13:35:51	[SSHService *SSH*-userauth on HoneyPotTransport,24, IP=59.45.79.39]	root trying auth password
11/02/2016 13:35:51	[SSHService *SSH*-userauth on HoneyPotTransport,24, IP=59.45.79.39]	login

3. Validação: A validação proposta está baseada nas funções internas do sistema operacional, os registros de ameaças são armazenados em arquivos *Logs* pelas plataformas *Snort* e *Suricata*. Na Tabela 4.3 apresentam-se alguns registros gerados pelos *IDS* avaliados, os *IDS Snort* e *Suricata* identificam um evento irregular, *Snort* particularmente identifica um acesso suspeito ao serviço *MYSQL* na porta 3306, identificando o endereço *IP* remoto que realiza a conexão 115.28.56.26 na porta de origem 6000, identificando ademais os parâmetros do pacote de dados chamado *TTL* (latência do pacote na rede), *SEQ* (seqüência ou ordem do pacote), e *Flags* (Bandeira de controle que está com o valor *SYN*). Já o *IDS Suricata* identifica do mesmo modo, uma ação suspeita ao serviço *MYSQL* identificando só o endereço *IP* remoto 20.145.122.44 na porta de origem 43874. Neste caso em particular observa-se que o *IDS Snort* realiza uma melhor identificação de este tipo de ameaça frente ao *IDS Suricata*.

Tabela 4.3: Arquivos de *Logs* dos *IDS Snort* e *Suricata*

Arquivos Logs	
Snort	**Suricata**
[1:2010937:2] ET POLICY Suspicious inbound to mySQL port 3306	[1:2010936:2] ET POLICY Suspicious inbound to mySQL port 3306
[Classification: Potentially Bad Traffic] [Priority: 2]	[Classification: Potentially Bad Traffic] [Priority: 2]
10/04-15:35:34.167246 115.28.56.26:6000 -> 104.131.170.135:3306	10/04/2015-15:49:57.340372 {TCP} 20.145.122.44:43874 -> 104.131.170.135:1521
TCP TTL:104 TOS:0x0 ID:256 IpLen:20 DgmLen:40	-
******S* Seq: 0x667F0000 Ack: 0x0 Win: 0x4000 TcpLen: 20	-

4.11 Análise e Interpretação

A continuação os principais testes estatísticos propostos.

4.11.1 Teste de Normalidade

Na Tabela 4.4 apresentam-se os resultados obtidos, a partir do teste de normalidade *Kolmogorov-Smirnov*, que proporciona o tipo de comportamento dos dados avaliados, particularmente para este caso de estudo, observa-se um (p-valor = 0) da variável *Destination Port*, indica que os dados avaliados não se comportam como dados normais, o intervalo de confiança proposto é de 99%, levando o valor de $\alpha = 0{,}01$.

Tabela 4.4: Resultado do Teste de normalidade proposto

Kolmogorov-Smirnov	
Variável	**p-valor**
Destination Port	0

4.11.2 Teste para Dados Não Normais *Mann-Whiteney*

A Tabela 4.5 apresenta os resultados do teste de *Mann-Whiteney* para duas amostras independente sendo o objetivo deste ensaio, tentar encontrar diferenças entre as amostras através da diferenças da média. A variável a avaliar corresponde a *Destination Port*, que representa o conjunto de serviços disponíveis nas plataformas *IDS* avaliadas *Snort* e *Suricata*. Observa-se que o valor de (p-valor = 0), por tanto temos que rejeitar a hipótese nula de que as amostras são iguais, ou seja rejeitamos a hipótese H_0. Por conseguinte aceitamos a seguinte hipótese H_1: (A quantidade de ataques reconhecida pela plataforma *Snort* é diferente à quantidade de ataques reconhecida pela plataforma *Suricata)*.

Tabela 4.5: Avaliação de duas amostras independentes

	Destination Port
U de Mann-Whitney	3350062,500
Wilcoxon W	7501583,500
Z	-15,409
p-valor	0

Na Figura 4.2 apresentam-se as diferenças entre as duas amostras avaliadas, em termo de quantidades de ataques identificados pelas plataformas *IDS Snort* e *Suricata*. No eixo horizontal se observam a quantidade de ataques identificados, já no eixo vertical se encontram os serviços avaliados. Em uma análise global pode-se concluir que o *IDS Snort* identifica uma quantidade melhor de ameaças e/ou ataques frente ao *IDS Suricata*, Já em uma análise

específica por cada tipo de serviço disponível, observa-se que *IDS Suricata* possui uma taxa de identificação maior que *IDS Snort*. Esclarece-se que as regras de assinaturas utilizadas pelos *IDS Snort* e *Suricata* são as mesmas, a fim de não desviar os resultados obtidos a um *IDS* em particular.

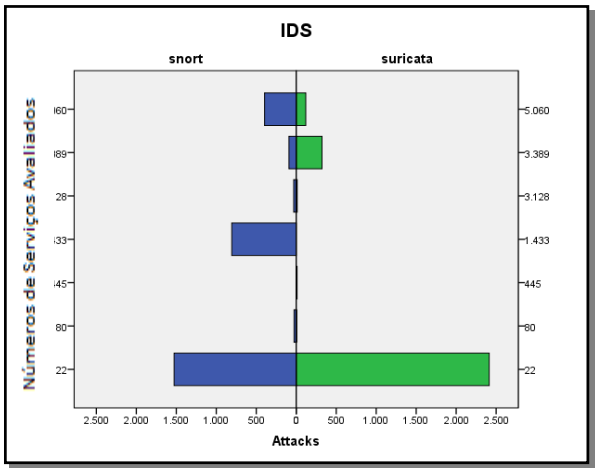


Figura 4.2: Desempenho das plataformas *IDS* avaliadas

Na Tabela 4.6 apresentam-se os resultados obtidos pelas detecções feitas dos *IDS* avaliados. Os resultados não proporcionam um relacionamento entre as detecções feitas pelos *IDS* no contexto dos serviços disponibilizados, ou seja as diferentes plataformas possuem distintos mecanismos para identificar um ataque. Desta forma não podemos assegurar que ambos *IDS* obtenham os mesmos valores em quantidade de detecção de ataques. O foco do experimento é oferecer informação quantitativa, que permita evidenciar sob a quantidade de ataques, identificado por cada serviço disponibilizado na plataforma de teste *MHN*. Observa-se em termos quantitativos que a soma de detecção de ameaças por parte de cada *IDS* avaliado apresenta um valor de aproximadamente 5761 identificações, com um erro do 4% de ameaças não identificadas pelos *IDS*. Na Figura 4.3 apresenta-se a quantidade de ataques identificados por serviço disponibilizado. Particularmente nesta dissertação estamos interessados nos resultados obtidos pelo serviço de acesso remoto *SSH*, que possui uma alta taxa de ataques por força bruta proveniente de ferramentas especializadas.

Tabela 4.6: Quantidade ataques reconhecidos pelos *IDS*

Serviço	Porta	Snort	Suricata	Total de Ataques	% Concentração de Ataques
SSH	***22***	***1528***	***2413***	***3941***	***65,68***
MSSQL	1433	807	0	807	13,45
SIP	5060	396	119	515	8,58
RDP	3389	91	322	413	6,88
SQUIT	3128	31	12	43	0,72
HTTP	80	27	4	31	0,52
SMB	445	1	11	12	0,20
Totais Detecções		2881	2881	5762	95,32

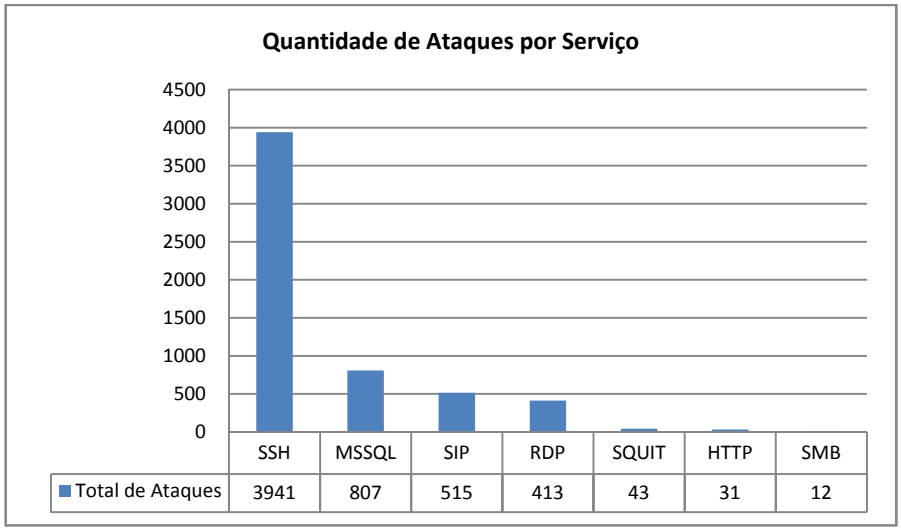


Figura 4.3: Quantidade de ataque identificado por serviço avaliado

4.12 Ameaças do Experimento

A continuação apresenta-se um resumo das ameaças detectadas na avaliação do experimento controlado baseado na plataforma teste de coleta de ataques *MHN*.

O contexto de execução do experimento está baseado na captura de ameaças reais que acontece na própria *Internet*, e estas ameaças estão relacionadas diretamente com os serviços disponibilizados pela plataforma de teste *MHN*. Este fato exige incluir um planejamento na segurança do ambiente de teste, a fim de garantir o normal desenvolvimento do experimento.

As possibilidades de ser atacado ou invadido estão diretamente relacionado a nível de segurança que adotemos em nossa plataforma de teste, por este motivo realizou-se uma comprovação da segurança do ambiente e os serviços disponibilizados, a fim de minimizar os eventos externos, que possam comprometer o normal funcionamento da plataforma de teste *MHN*. Na Tabela 4.7 apresenta-se um modelo de registro das ameaças detectadas na execução das distintas provas efetuadas na avaliação do experimento.

Tabela 4.7: Modelo de avaliação de ameaças proposto

Pontos de Análises: Identificação do problema
Grau do problema: Critico (precisa solucionar no momento, para continuar com as provas) ou Leve (não precisa solucionar no momento, para continuar com as provas)
Problema: Descrição breve do potencial problema.
Origem do Problema: Descrição breve da origem do problema.
Solução Proposta: Descrição breve da possível solução.

O planejamento proposto consta de verificar os seguintes pontos: Na Tabela 4.8 o funcionamento dos serviços avaliados. Na Tabela 4.9 o funcionamento da interface de

administração *Web*. Na Tabela 4.10 o funcionamento do Banco de Dados *MongoDB*. Na Tabela 4.11 o funcionamento geral do sistema operacional base.

Tabela 4.8: Análise de serviços

Pontos de Análises: Serviços.
Grau do problema: Critico.
Problema: Em reiteradas ocasiões se identifico problemas do serviço do *Honeypot Dionaea* que administra os demais serviços, o erro detectado apresentava que o serviço estava *OFFLINE.*
Origem do Problema: O serviço principal *Dionaea* associado a *MHN* apresenta um erro de armazenamento para o serviço *SSH*, o arquivo *log* de consulta está localizado neste rota no sistema operacional *Linux* /var/dionaea/dionaea.log
Solução Proposta: Se redefine o tamanho do arquivo *log* de *SSH* a um tamanho maior no Arquivo de configuração de dionaea.conf

Tabela 4.9: Análise de interface *Web*

Pontos de Análises: interface de administração *Web.*
Grau do problema: Critico.
Problema: Em reiteradas ocasiões se identifico problemas ao aceder à pagina principal de acesso à plataforma de teste *MHN*.
Origem do Problema: O serviço Apache que administra as conexões *Web* da plataforma de teste *MHN*, apresenta erro, por motivos de disponibilidade de memória *RAM*. Lembre-se que o servidor que aloca a plataforma de teste possui 1 GB de Memória *RAM* disponível.
Solução Proposta: Pesquisou-se o problema e contatou-se que o problema é originado pela falta de memória *RAM* do servidor *VPS* alugado. Procede-se a realizar uma modificação no arquivo de configuração do servidor *Apache*.

Tabela 4.10: Análise do banco de dados

Pontos de Análises: Funcionamento do Banco de Dados
Grau do problema: ----------
Problema: Não se identifico problemas.
Origem Problema: ----------
Solução Proposta: ----------

Tabela 4.11: Análise do sistema operacional

Pontos de Análises: Sistema Operacional.
Grau do problema: Leve.
Problema: Identificou-se problemas de acesso ao serviço em reiteradas ocasiões, tanto pela interface *Web*, como pelo acesso remoto *SSH*, Esclarece-se que este acesso no tem relacionamento com o serviço que se avalia por parte do *Honeypot Dionaea*.
Origem do Problema: Nestá instância o erro é atribuído à falta de memória *RAM* disponível.

Solução Proposta: Acontecido o problema, e ao não poder ingressar ao sistema pelos meios mencionados anteriormente, o provedor do serviço *DigitalOcean* possui uma interface próprio *Web* que permite administrar o servidor virtual. Este método de acesso ingressa à imagem virtual do sistema e força o reinicio do sistema, um método arriscado de praticar com um possível dano do sistema operacional e a correspondente perda de informação.

A conclusão nesta seção, manifesta que o experimento proposto apresenta diversos problemas relacionados com o sistema operacional base e a plataforma de teste *MHN*. No entanto se realizou tudo o processo corretivo necessário que garantiu o normal funcionamento do experimento proposto. Destaca-se ademais o conhecimento técnico da plataforma operacional *Linux Ubuntu* do administrador que possibilitou efetuar em pouco tempo a correção às falhas identificadas, logrando desta maneira obter um tempo de *Offline* menor a uma hora, em todos os casos apresentados anteriormente.

4.13 Considerações Finais do Capítulo 4

A partir do experimento efetuado que está baseado na coleta de ataques pela plataforma de teste *MHN*, fundamentada em *Honeypots*, *evidência*-se que qualquer serviço que seja disponibilizado na *Internet* está propenso a sofrer ataques. Observa-se que o serviço com maior quantidade de ataques identificados neste experimento corresponde ao serviço de acesso remoto *SSH* executando-se na porta 22 do sistema operacional, baseado nas avaliações feitas pelos *IDS Snort* e *Suricata*, este resultado é avaliado através de modelos estatísticos com fortes evidências nos fatos constatados.

A plataforma de teste baseada em *Honeypots MHN* possui a capacidade de ser instalada em uma rede real de dados, a fim de ser alvo de ataques por intrusos, desta forma se espera lograr o desvio dos ataques aos serviços disponibilizados na rede. O futuro analises destas ameaças coletadas é de vital importância para entender as metodologias de ataques e as ferramentas utilizadas, o conhecimento interpretado ajuda ao administrador da rede a compreender como deve planejar as políticas de segurança e táticas defensivas com o alvo de proteger a infra estrutura crítica do sistema geral.

Determina-se através de uma análise de *logs* gerado pelo servidor de acesso remoto *SSH* que há uma alta taxa de ataques derivado de ataques por força bruta com dicionário fornecido,

por meio de ferramentas especializadas como *Medusa*[28], *Hydra*[29], *Brutus*[30] e *John the Ripper*[31].

Os resultados alcançados neste capítulo justificam a eleição do serviço de acesso remoto *SSH*, a ser avaliado na plataforma de teste baseado em *Raspberry Pi 2 Model B*, como também, os tipos de ataques de força bruta com dicionário a efetuar através das ferramentas especializadas *Medusa* e *Hydra*, e também ferramentas não especializadas como *Metasploit*.

[28] http://foofus.net/
[29] http://www.thc.org/
[30] http://www.hoobie.net/brutus/
[31] http://www.openwall.com/john/

CAPÍTULO 5

PLATAFORMAS DE AVALIAÇÃO

Neste capítulo apresentam-se as distintas plataformas embarcadas de *hardware* utilizadas neste trabalho de dissertação, começando pela plataforma *Raspberry Pi*, *Arduino*, e as características técnicas associadas. Por outro lado, expõem-se as aplicações de *software* que possibilitam o desenvolvimento dos cenários de provas propostos, os fundamentos teóricos são abordados em cada ponto de análise, justificando, métodos, abordagens e implementações utilizadas.

5.1 Plataforma de Teste Proposta Baseada *Raspberry Pi 2 Model B*

Nesta seção apresenta-se uma plataforma de teste baseada em *Open software* e *Open hardware* de baixo custo *Raspberry Pi 2 Model B*. A eleição da plataforma de teste é baseada nos trabalhos apresentados Capítulo 2, onde se detalham as características tecnológicas que identificam está plataforma. Lembra-se no capítulo anterior que através do experimento controlado baseado em *Honeypots* identificou-se o serviço alvo de estudo correspondente ao serviço de acesso remoto *SSH*. Este serviço cumprirá a função de um servidor de acesso remoto *SSH* executando-se na plataforma de teste de baixo custo baseada em *Raspberry Pi 2 Model B*.

5.1.1 Características Técnicas Da Plataforma

Na Tabela 5.1 apresentam-se um resumo das características técnicas do *hardware* embarcado *Raspberry Pi 2 Model B*. Na Figura 5.1 uma representação gráfica do dispositivo analisado.

Tabela 5.1: Detalhe técnico da plataforma *Raspberry Pi 2 Model B*

Fornecedor	Fundação Raspberry Pi
Tipo	Placa Computador (SBC)
Sistema Operacional	Linux Raspbian
Alimentação	3,5 W
CPU	Bcm2836 900mhz ARM Cortex-A7 Quad-Core
GPU	Broadcom Video Core Iv
Memória	1GB LPDDR2 SDRAM
Armazenamento	Cartão
Saida Video	Saída De Vídeo HD 1080p
Rede	Rj45 Ethernet 10/100
Usb	X4
Gpio	40 Pinos

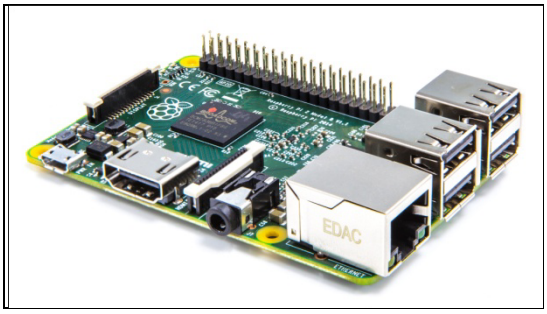


Figura 5.1: Dispositivo embarcado *Raspberry Pi 2 Model B*

5.1.2 Conceitos Teóricos e Instalação do Serviço *SSH*

Nesta subseção apresentam-se os conceitos principais de funcionamento e instalação do serviço de acesso remoto *SSH*. Neste ponto estamos em condições de instalar o serviço de acesso remoto *SSH* que é alvo de estudo. O serviço *SSH* disponibilizado nas provas propostas, está baseado em *OpenSSH*[32] que é um conjunto de aplicações que permite realizar comunicações cifradas através da rede utilizando o protocolo *SSH*, ou seja, a aplicação proposta é a implementação e desenvolvimento final dos lineamentos especificados no protocolo *SSH*. Além disso, entenda-se o acrônimo *SSH* como a aplicação instalada no sistema operacional *Linux Raspbian* executando-se na plataforma de teste *Raspberry Pi 2 Model B*. As aplicações que formam parte da aplicação são:

SSH: Cliente *SSH*.

SCP e *SFTP*: Comandos de transferências de arquivos.

SSHD: Servidor *SSH* executando-se como um deamon no sistema operacional *Linux*.

SSH-AGENT e *SSH-ADD*: Ferramentas que administra a autenticação do usuário.

SSH-KEYSCAN: Ferramenta de busca de senhas atribuídas ao cliente.

[32] Open Secure Shell

SSH-KEYGEN: Ferramenta que gera e administra chaves *RSA* e *DSA*, que são utilizadas pelo método de autenticação de usuários.

A continuação apresenta-se os métodos de criptografia mais utilizado, no entanto existem outras implementações que não é alvo de estudo neste trabalho, os métodos apresentados são:

- Chave Simétrica: *AES* **RFC-3962**, *3DES* **RFC-1851**, *RC4* **RFC-7465**.
- Chave Assimétrica: *RSA* **RFC-2437**, *DSS* , *PGP* **RFC-3156/4880**.
- *Hash*: *MD5* **RFC-1321**, *SHA-1* **RFC-3174**.

A continuação, os passos desenvolvidos que apresentam a instalação e configuração do serviço *SSH*, no sistema operacional *Linux Raspbian* da plataforma *Raspberry Pi 2 Model* B, os comandos utilizados são:

1. sudo apt-get install *SSH* - Apresenta a instalação do servidor *SSH*.
2. sudo /etc/init.d/*SSH* start - Apresenta o funcionamento do servidor *SSH*.
3. sudo update-rc.d *SSH* defaults - Atualiza inicio *SSH* no sistema operacional.

A continuação apresentam-se as sequências de passos para fornecer uma conexão entre o cliente, e o servidor *SSH* executando-se na plataforma *Raspberry Pi 2 Model B:*

1. O cliente solicita uma conexão *TCP* à porta 22 do servidor.
2. O cliente e o servidor acordam a versão do protocolo a utilizar de acordo às configurações efetuadas, neste ponto esclarece-seque é utilizada a configuração por defeito no servidor *SSH*.
3. O servidor possui um par de chaves pública e privada gerada pelo algoritmo *RSA* (*Rivest*, *Shamir* e *Adleman)* [RIVEST et al. 1978] [DIFFIE et al. 1976] **RFC-3447**. O servidor envia ao cliente sua chave pública.
4. O cliente compara a chave pública do servidor com a chave que possui o cliente, para verificar sua autenticidade. Se não a conhece previamente, solicita confirmação ao usuário para que seja válida.
5. O cliente gera uma chave de sessão aleatória e seleciona um algoritmo de cifra simétrico.

6. O cliente envia uma mensagem contendo a chave de sessão e o algoritmo selecionado cifra com a chave pública do servidor usando o algoritmo *RSA*.

7. Em adiante, para o resto da comunicação utilizar-se-á o algoritmo de cifra simétrico selecionado e a chave compartilhada de sessão.

8. Logo realiza-se a autenticação do usuário por um método denominado (Autenticação com senha ou método de autenticação).

9. Finalmente se inicia a conexão ao servidor *SSH*, fornecimento do túnel cifra.

O ponto 8 utiliza um método de autenticação por senha (chave de sessão), senha que é utilizada pelo sistema operacional *Linux* para autenticar um usuário. Este método de autenticação está baseado no uso de um algoritmo *Hash SHA-512* [NIST 2015][AUMASSON et al. 2015] **RFC-5754**. No Quadro 5.1 apresenta-se a primeira línha do arquivo *Shadow* identificando o usuário (*root*) seguido do símbolo (**$6**) que representa o uso do algoritmo *SHA-512* para cifrar senhas do sistema operacional *Linux Raspbian*. Os distintos métodos de autenticação são apresentados com detalhe no Apêndice C.

Quadro 5.1: Identificação do algoritmo de *Hashing SHA-512*

root:**$6**$XHxtN5iB$5WOyg3gGfzr9QHPLo.7z0XIQIzEW6Q3/K7iipxG7ue04CmelkjC51SndpOcQlxTHmW4/AKKsKew4f3cb/.BK8/:16828:0:99999:7:::

5.2 Plataforma De Medição Baseada em *Arduino Uno*

Arduino Uno é uma plataforma de prototipagem baseada em Open software e Open hardware de fácil uso. A placa *Arduino Uno* contém um microcontrolador *ATmega328*, comandado por uma linguagem de programação de alto nível. Está plataforma facilita o desenho e modelagem de soluções de diferentes áreas de desenvolvimento, também é capaz de atualizar-se em suas distintas configurações de *hardware*, ou seja contém a capacidade de expansão baseado em módulos *Shield*. Os módulos estão suportados por um amplo conjunto de protocolos de comunicações e sensores de propósitos específicos, alguns exemplos de aplicações tecnológicas atuais são: sistemas de controle e aquisição de dados, sistemas embarcados de propósito específico, sistemas baseados *IoT*, sistemas de comunicações com diversas tecnologias.

5.2.1 Arquitetura do *Hardware*

Na Tabela 5.2 apresentam-se os detalhes técnicos da arquitetura de hardware do embarcado de baixo custo *Arduino Uno*. Na Figura 5.2 apresenta-se um gráfico do dispositivo embarcado proposto.

Tabela 5.2: Detalhe técnico da plataforma *Arduino Uno*

Microcontrolador	Atmega328p
Tensão Operação	5v
Tensão Entrada (Recomendada)	7-12v
Tensão Saída (Limite)	6-20v
Digital I/O Pinos	14 (Dos Quais 6 Fornece PWM Saídas)
PWM Digitais I/O Pinos	6
Entrada Pinos Analógica	6
Corrente Por I/O Pinos	20 Ma
Corrente Para 3.3v Pinos	50 Ma
Memória Flash	32 Kb (Atmega328p)
SRAM	2 Kb (Atmega328p)
EEPROM	1 Kb (Atmega328p)
Velocidade Do Clock	16 Mhz
Comprimento	68.6 Mm
Largura	53.4 Mm
Peso	25 G

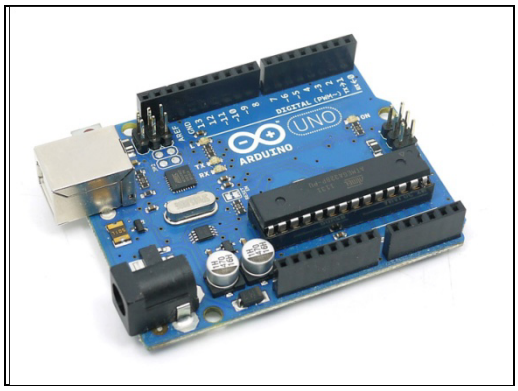


Figura 5.2: Plataforma embarcada *Arduino Uno*

5.2.2 Sensor de Corrente *ACS721ELC-5A*

O *ACS712* é um sensor de corrente que funciona, tanto em corrente contínua como alternada, de acordo com o fabricante existem três versões diferentes que variam na corrente suportada: *ACS712ELCTR-05B* corrente máxima de 5A, *ACS712ELCTR-20A* corrente máxima de 20A e *ACS712ELCTR-30A* corrente máxima de 30A. O sensor proposto nesta pesquisa é *ACS712ELCTR-05B* que administra uma corrente máxima de 5A, e dado que estamos mapeando correntes no contexto de miliampères está proposta é a melhor que se adapta aos requerimentos técnicos da pesquisa e ao custo associado do dispositivo, o custo estimado deste sensor é de aproximadamente R$15 reais.

O funcionamento deste dispositivo é detectar e controlar o fluxo de corrente de outro dispositivo ou sistema baseado no *Chip ACS712* da empresa *Allegro Microsystem*[33], que pode detectar com precisão a corrente alternada e contínua, seu funcionamento está baseado no efeito *Hall* detectando o campo magnético gerado pela passagem da corrente. O sensor é conectado ao *Arduino Uno* através da porta analógica de entrada onde é transferido ao conversor analógico digital, dispositivo que interpreta os valores analógicos enviados pelo sensor e o transforma em sinal digital. Particularmente, este conversor no *Arduino Uno* possui uma capacidade de amostragem de 1023 valores possíveis, ou seja, uma resolução de 10 Bits, através de este mapeamento logra-se obter uma série de pontos que apresentam os valores de corrente no tempo. Neste trabalho utilizou-se este dispositivo, a fim de conhecer os valores de correntes fornecidos pela plataforma de teste baseada no *Raspberry Pi 2 Model B* executando um servidor de acesso remoto *SSH*. Na Tabela 5.3 apresentam-se as características técnicas do dispositivo *ACS712ELCTR-05B*, seguidamente na Figura 5.3 o sensor proposto.

Tabela 5.3: Detalhe técnico do sensor modelo *ACS712ELCTR-05B*

Faixa de Medição	-5a A +5a
Alimentação	5v
Tempo de Resposta	5µs
Saída de Tensão por Ampère	185 Mv/A
Erro %	7--10
Dimensões	30 X 12 X 12mm

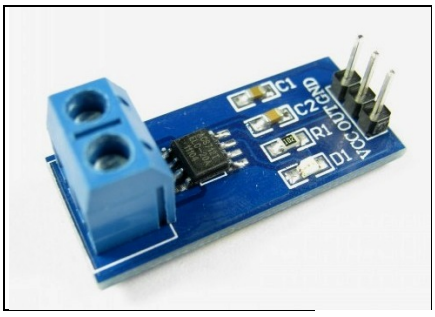


Figura 5.3: Sensor de corrente proposto *ACS712ELCTR-05B*

5.2.3 Plataforma de *Software Matlab*

A plataforma de *software* utilizada para realizar a captura e processamento do sinal obtido através do *Arduino Uno*, é feita com o *software* cientifico *Matlab*. Apresentando toda a lógica de programação convencional conhecida, ademais incorpora um ambiente de processamento e aquisição de sinais, destacando a inter operabilidade com distintas

[33] http://www.allegromicro.com/

plataformas embarcadas do mercado, permitindo atingir a coleta de dados e pós-processamento requerido.

5.2.4 Código de Coleta Proposto

O Quadro 5.2 apresenta o código utilizado no *Framework Matlab* que coleta os dados provenientes do sensor *ACS712ELCTR-05B* administrado pelo *Arduino Uno*. Os dados coletados pela plataforma de medição de consumo energético são armazenados em um arquivo de texto plano para um posterior processamento.

```
% Conenctando ao Arduino no COM7:
a = arduino('COM3');
% periodo de amostragem [s]
ts = 0.1;
c=1;
while c <=1
tempo = input('Quantidade de amostras: ');
total_amostras = tempo / ts;
for ii = 1:total_amostras
    h1=subplot(3,1,2);
%Cálculo da corrente
%(valorSensor-(Vcc/2))*tenção_por_unidade/sensibilidade_de_tenção
    I1(c,ii) = ((a.analogRead(1)-(1023/2))*0.004887586/0.185)*1000
    h1=plot(I1,'b');
    hold on
    title('Corrente');
    ylabel('Corrente (mA)');
    xlabel('Amostras');
    pause(ts);
End
c=c+1;
End
```

Quadro 5.2: Código de coleta de corrente proposto em *Matlab*

CAPÍTULO 6

METODOLOGIA DE MEDIÇÃO DE CONSUMO

6.1 Metodologia Proposta de Medição de Consumo Energético

Neste ponto da pesquisa estamos em condições de utilizar a plataforma de teste baseada em *Raspberry Pi 2 Model B* que disponibiliza o serviço de acesso remoto *SSH*, por outro lado, foi utilizada a plataforma de medição de consumo energético *baseado em Arduino Uno* que interpreta as respostas do embarcado *Raspberry Pi 2 Model B* através do consumo de corrente elétrica reconhecida pelo sensor *ACS712ELCTR-05B*.

As duas plataformas apresentadas anteriormente formam parte do projeto experimental proposto, que consiste em mapear as respostas energéticas geradas pela plataforma de teste *Raspberry Pi 2 Model B*, executando o serviço de acesso remoto *SSH* avaliado com cenários de provas.

Os cenários de avaliação propostos correspondem a dois grupos bem definidos que são:

O primeiro grupo corresponde aos tipos de acessos, os cenários avaliados são:

Cenário 1: Possui como objetivo apresentar os fatos interveniente no processo de acesso ao sistema através de um nome de usuário valido e sua senha válida

Cenário 2: Possui como objetivo apresentar os fatos interveniente no processo de acesso ao sistema através de um nome de usuário valido e suã senha não válida.

O segundo grupo corresponde aos ataques de força bruta com dicionário através de ferramentas especializadas e não especializadas, os cenários avaliados são:

Cenário 3: Ataque de força bruta com dicionários através da ferramenta especializada *Medusa*.

Cenário 4: Ataque de força bruta com dicionários através da ferramenta especializada *Hydra*.

Cenário 5: Ataque de força bruta com dicionários através da ferramenta não especializada *Metasploit*.

O vetor de ataque, denominado de ataque de força bruta com dicionário, possui a capacidade de vulnerar um sistema de acesso, que utilize um meio de validação por usuário e senha, está técnica emprega um arquivo ou comumente denominado também um dicionário que contém uma lista dos usuários e senhas mais comuns.

Lembre-se que a maioria dos sistemas empregados na atualidade, possui usuários e senhas por defeito com níveis de execução e privilégios elevados, pois são estes usuários os alvos de ataques selecionados, por exemplo, no sistema *Linux* um usuário com altos privilégios corresponde ao super usuário (*Root*).

Em função dos cenários propostos, deseja-se obter as respostas geradas pela plataforma de teste *Raspberry Pi 2 Model B* a nível de sistema operacional mapeadas em um contexto de consumo energético, onde a métrica selecionada corresponde à corrente elétrica consumida no sistema. Os grupos propostos são fundamentais para o desenvolvimento da pesquisa, pois se procura reproduzir um fato de comum conhecimento entre os administradores e usuários do serviço de acesso remoto *SSH*. Particularmente, neste trabalho propõem-se dois cenários que são alvo de estudos. O primeiro cenário está relacionado ao evento que acontece no servidor quando um usuário ingressa ao sistema com credenciais válidas, seguidamente o segundo cenário está relacionado ao evento que acontece no servidor quando um usuário intenta ingressar ao sistema com credenciais de acesso não válidas, ou seja nome de usuário valido e senha de acesso não válida.

O terceiro cenário proposto está relacionado ao ataque de força bruta com dicionário através da ferramenta especializada *Medusa*.

O quarto cenário proposto está relacionado ao mesmo tipo de ataque apresentado no cenário anterior, a diferença está na ferramenta proposta que é baseada em *Hydra*.

Por último o quinto cenário está formado por uma ferramenta não especializada que tecnicamente pode realizar o ataque proposto, porém o método de ataque não está aperfeiçoado, a ferramenta proposta é denominada *Metasploit* que é um *Framework* de análise de vulnerabilidades, a justificação desta ferramenta é contrastar os resultados dos ataques especializados frente aos não especializados.

Nesta conjuntura analisada pretende-se obter uma curva padrão característica do consumo de corrente, que possa identificar um tipo de acesso ao sistema operacional frente a outro, ademais de poder interpretar um tipo de ataque de força bruta otimizado, frente a outro ataque não otimizado. Na Figura 6.1 apresentam-se os cenários de avaliação propostos.

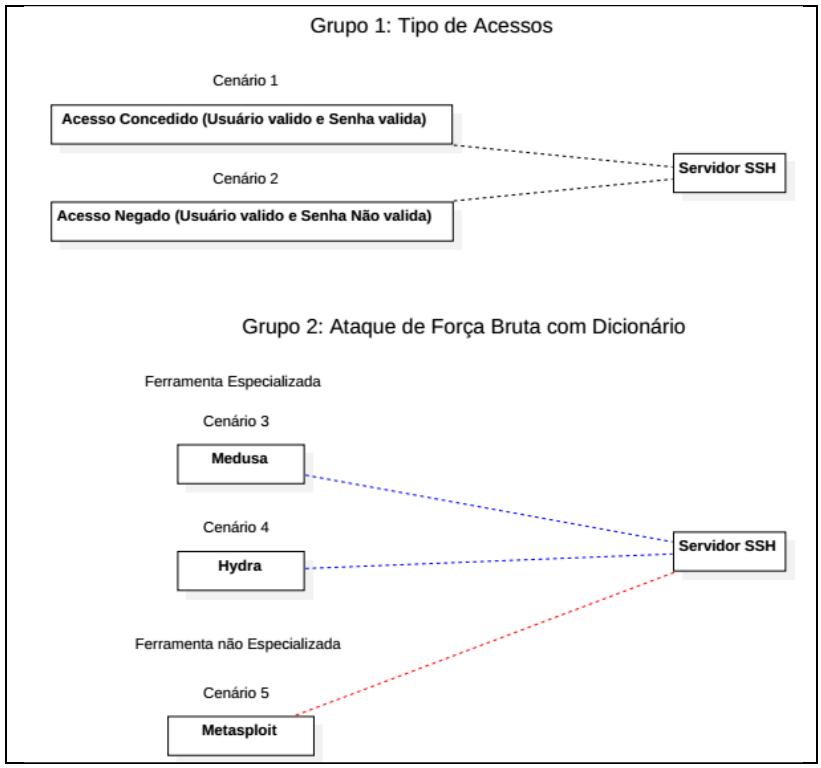


Figura 6.1: Cenário de avaliação propostos

6.2 Metodologia de Avaliação Estatística dos Sinais de Corrente

A seguinte metodologia de avaliação científica proposta, apresenta os lineamentos básicos a desenvolver, explicando passo a passo os dados a pesquisar, os métodos de análises a empregar, tudo isto com o fim de recolher, processar e interpretar os resultados obtidos que oferecem informação importante para a tomada de decisão.

A continuação os passos a desenvolver na metodologia proposta são:

P1: Define-se em cada cenário de prova avaliado um sinal denominada Padrão de Estado Estável (PEE) que representa o estado do sistema em um estado estável referenciado pelo funcionamento da plataforma de teste embarcada *Raspberry Pi 2 Model B* executando o serviço de acesso remoto *SSH*. A seguir se define outro sinal padrão própria do cenário avaliado, portanto se obterão dois sinais por cenário avaliado, e que são alvo de análise e comparação, que permitirá apresentar as evidências de identificar um tipo de acesso ou um tipo de ataque, frente a um estado normal de funcionamento da plataforma de teste avaliada no contexto de análise energético. A continuação os sinais propostos por cenários avaliados são:

O cenário 1 está composto por um sinal Padrão de Estado Estável (PEE) e Padrão de Acesso Concedido (PAC).

O cenário 2 está composto por um sinal Padrão de Estado Estável (PEE) e Padrão de Acesso Negado (PAN).

O cenário 3 está composto por um sinal Padrão de Estado Estável (PEE) e Padrão de ataque por força bruta através da ferramenta especializada *Medusa* (PAM).

O cenário 4 está composto por um sinal Padrão de Estado Estável (PEE) e Padrão de ataque por força bruta através da ferramenta especializada *Hydra* (PAH).

O cenário 5 está composto por um sinal Padrão de Estado Estável (PEE) e Padrão de ataque por força bruta através da ferramenta especializada *Metasploit* (PAME).

P2: Define-se um estado estável, a um sinal onde as variáveis que definem seu comportamento respeito ao tempo permanecem invariantes, outra definição de estado estável é quando as variações do sinal são periódicas e se repetem de maneira idêntica em cada período de tempo. Particularizando a definição anterior, consideramos que o estado estável do sinal de corrente elétrica, é atribuído ao funcionamento normal do sistema embarcado *Raspberry Pi 2 Model B* executando o sistema operacional *Linux Raspbian* com o serviço *SSH*. Esta definição afiança o conceito de padrão de consumo de corrente baseado em seu estado estável, premissa necessária para a estimação de qualquer atividade desenvolvida na plataforma de teste embarcada *Raspberry Pi 2 Model B*. Na prática considera-se um estado estável ao valor médio de corrente consumido pelo *hardware* embarcado executando-se o sistema operacional em condições normais de funcionamento, ou seja sem receber eventos externos que alterem sua invariante. O fato de ter um consumo médio de energia por parte do dispositivo embarcado é devido ao relacionamento de *hardware* e *software* que envolve o funcionamento mínimo operativo do sistema tratado como um tudo. O sinal de estado estável de funcionamento proposto é calculado para todos os cenários avaliados.

P3: Define-se a quantidade de amostras a coletar em 10 unidades, por cada cenário avaliado, lembrar que cada cenário possui dois sinais a analisar, ou seja 10 sinais do Padrão de Estado Estável (PEE), e 10 sinais do Padrão de Acesso Concedido (PAC) para o cenário 1 avaliado, consequentemente a mesma lógica para os demais cenários propostos.

P4: Procede-se a realizar uma análise das amostras coletadas identificando o desvio padrão do sinal de estado estável, como também, o sinal identificado na avaliação do tipo de acesso ou ataque. O desvio padrão apresenta quanto de variação ou dispersão existe em relação à média, com essa medida selecionamos a amostra que menor desvio tenha, desta forma obtemos um sinal com baixa porcentagem de variação de corrente.

Este sinal com desvio padrão mínimo se converte no sinal de referência do cenário avaliado, ou seja o sinal padrão do modelo.

P5: Feita a coleta das diferentes amostras de cada cenário avaliado procede-serealizar um teste de normalidade de *Shapiro-Wilk* [ANDERSON et al. 2003], para cada cenário de maneira independente por sinal. O objetivo é contrastar a normalidade do conjunto de sinais coletadas. Feita está operação conheceremos qual teste utilizar para encontrar as diferenças entre os sinais. Os testes propostos para encontrar as diferenças entre os sinais são:

Dados normais: Utiliza-se o teste *T Student* [KING et al. 2004].

Dados não normais: Utiliza-se o teste não paramétrico *Mann-Whitney* [KAZMIER et al. 2004].

Os dois testes propostos são utilizados para encontrar diferença entre as médias das amostras avaliadas, por exemplo, no cenário 1 se compara a média de (PAC) contra a média de (PEE).

P6: Procede-se a efetuar o teste de comparação entre o sinal padrão do cenário proposto com o sinal padrão do estado estável (PEE) através do teste apresentado no ponto 5 desta metodologia.

Hipótese 1: Valor Médio de Corrente

H_0: O valor médio de corrente do padrão associado ao cenário avaliado é igual ao valor médio de corrente do padrão de estado estável (PEE).

$$H_0: \mu_{P\ \text{CENÁRIO AVALIADO}} = \mu_{PEE\ \text{CENÁRIO AVALIADO}}$$

H_1: O valor médio de corrente do padrão associado ao cenário avaliado é distinto ao valor médio de corrente do padrão de estado estável (PEE).

$$H_1: \mu_{P\ \text{CENÁRIO AVALIADO}} \neq \mu_{PEE\ \text{CENÁRIO AVALIADO}}$$

Esclarece-se que $\mathbf{H_o}$ corresponde à hipótese e é a que deseja-se rejeitar.

A variável que avaliará está hipóteses é o consumo médio de corrente estimado do sinal.

O nível de significância escolhido para avaliar a hipótese é de 95%. $\alpha = 0{,}05$.

A interpretação de aceitar $\mathbf{H_o}$ é que não se produz diferença entre os sinais avaliados, por tanto, não logra provar o fato de se existe ou não um tipo de acesso ou ataque, porém se rejeitamos $\mathbf{H_o}$ estamos aceitando $\mathbf{H_1}$ que apresenta as evidências que existe uma diferença nos sinais avaliados no contexto estatístico.

P7: Neste último ponto, já contamos com uma curva padrão característica que identifica um tipo de acesso ou um tipo de ataque dependendo do cenário avaliado, no entanto desde um aspecto gráfico procedemos a efetuar uma normalização do sinal obtido com o método de *Welch*, visando apresentar uma curva padrão característica ajustado a um gráfico de interpretação mais representativo.

6.3 Identificação de Curva Padrões com o Método de *Welch*

Nesta seção apresenta-se uma interpretação gráfica da curva padrão obtida para cada cenário avaliado. Está curva representa um sinal padrão de um cenário de normal funcionamento, como também um padrão de tipo de acesso ou tipo de ataque baseado no contexto do consumo de corrente. Está representação gráfica é necessária pelo fato de uma rápida e clara interpretação dos resultados alcançados para cada cenário avaliado. A Figura 6.2 apresenta um sinal padrão de consumo de corrente em função do tempo que representa a resposta de um evento de acesso negado. Observa-se que a figura não expõe uma representação compreensível que possa ser utilizada como um instrumento de comparação visual efetivo.

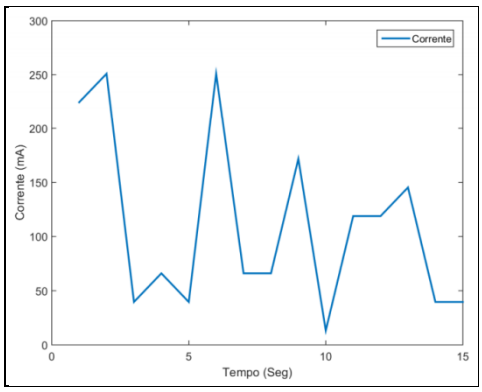


Figura 6.2: Exemplo de uma curva de consumo de corrente.

6.4 A Transformada de *Fourier*

Em geral a caracterização no domínio da frequência ou análise espectral tem seu fundamento na transformada de *Fourier*, a qual é capaz de proporcionar uma transformação reversível entre o domínio do tempo, e o da frequência, e vice-versa. No entanto, não se fornece nenhuma informação nova pelo fato de transformar um sinal do domínio do tempo ao da frequência. Sem embargo a distribuição da energia ou potência do sinal densidade espectral de potência (PSD), que proporciona a transformada de *Fourier* no domínio da frequência, permite uma análise mais simplificada das características da frequência do sinal.

A transformação matemática, que leva à teoria de *Fourier* de demonstrar que um sinal pode ser decomposto como uma soma infinita de funções senoidais com seus correspondentes termos de frequência e amplitude. Por isso, o espectro de um sinal não é mais que a representação, sobre um eixo de frequências dos termos de amplitude e fase, de cada uma das sinusoidais que compõem o sinal.

6.5 Estimação Espectral

O objetivo da Estimação Espectral é descrever a distribuição da frequência de energia contida em um sinal baseado em um conjunto finito de amostras. Esse tipo de análise possibilita a extração de informações sobre a dinâmica do processo, na medida em que o sinal é estudado em termos de unidades de frequência ao invés de unidades de tempo [RANGAYYAN et al. 2002]. A base da análise espectral está baseada na análise de *Fourier*.

6.6 Densidade de Potência Espectral

A densidade de potência espectral PSD tem como função descrever como a variância de um processo aleatório está distribuída em relação às suas frequências [MARPLE 1987], onde a área sob a curva representa a energia ou potência do sinal. A estimação da PSD pode ocorrer de forma Paramétrica e Não-Paramétrica. Os métodos paramétricos são aqueles nos quais os dados são modelados como saídas de um sistema linear com entrada modelada como um ruído branco.

Nos métodos não-paramétricos, a PSD é estimada diretamente do sinal, sendo que existem duas abordagens clássicas: o método indireto realiza inicialmente uma estimação da sequência de auto-correlação no conjunto de dados, desta forma a transformada de *Fourier* calcula a PSD estimada, chamada de método pelo Correlograma; o método direto obtém a PSD ao extrair a magnitude quadrática da transformada de *Fourier* de uma sequência de dados com a apropriada média estatística, chamada de Periodograma modificado.

6.7 Periodograma Modificado de *Welch*

O periodograma é a estimativa do espectro de potência de um sinal, dado pelo quadrado da magnitude do resultado da transformada discreta de *Fourier* das amostras do processo, conforme mostra a equação 6.1, para um dado sinal x[n] de tamanho N.

$$Pxx(f) = \frac{|X(f)|^2}{F_s N}, \text{ onde } X(f) = \sum_{n=0}^{N-1} x[n] e^{-j\,2\,\pi\,f\,n/f_s} \qquad \textbf{(6.1)}$$

Equação 6.1: Definindo o periodograma

O periodograma modificado de *Bartlett* realiza a média entre múltiplos PSDs produzidos, a partir de segmentos do sinal original, sendo que a estimativa da PSDs melhora com o aumento na quantidade de segmentos utilizados, desde que sejam eles estatisticamente independentes, mas se perde em resolução espectral, caso os segmentos não contenham amostras suficientes. Na Figura 6.3 apresenta o resultado do método de *Bartlett* para um sinal, que representa o Padrão de Acesso Concedido (PAC) do cenário avaliado chamado tipos de acessos.

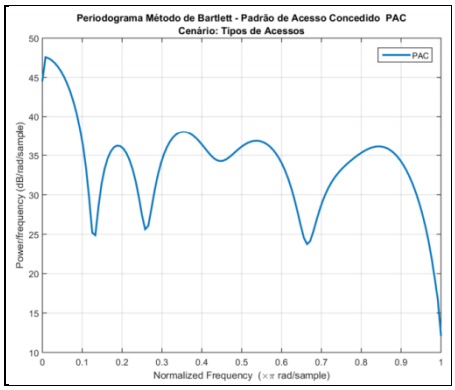


Figura 6.3: Periodograma do Método de *Bartlett* - PAC

Finalmente, o periodograma modificado de *Welch* amplia o periodograma de *Bartlett* na medida em que permite a sobreposição entre segmentos e também introduz janelas no sinal.

Welch propôs uma versão semelhante ao periodograma de *Bartlett*, que utilizou janelas temporais e a possibilidade de sobreposição dos intervalos de estimação dos espectros de ordem K. Se a série temporal $S \leq L$ for divida em L segmentos com N = KxL amostras, cada segmento está atrasado em relação ao anterior de [(N - L)/S + 1] amostras, pois o número de segmentos L é igual à parte inteira de $X^{(K)}[n] = w[n]\ x[n+KS]$. No periodograma de *Welch* o segmento de ordem K é $\omega\ [n]$ onde $\hat{P}_w\ (\omega)$ é a função janela.

Pode-se notar que o procedimento de *Welch* permite, para um mesmo valor de N, uma maior estabilidade porque L é mais elevado mantendo uma resolução constante, o valor de N = KxL. Prova-se, além disto, que uma sobreposição de 50% oferece o melhor compromisso de estabilidade à resolução, para valores de N e N = KxL fixos [MARPLE 1987][KESHAB et al. 2014].

A resolução em frequência do espectro é aproximadamente $1/NT_s$ onde N é o número de amostras por segundo. Ao tomar um segmento pequeno se aumenta o número de segmentos, por tanto se aumenta o suavizado no espectro estimado.

A sobreposição associado ao periodograma calculado via método de *Welch* ajuda a melhorar a correlação dos dados, melhorando suas propriedades estatísticas e a confiabilidade

da estimação. Na Figura 6.4 apresenta-se o método de *Welch* aplicado ao sinal Padrão de Acesso Concedido (PAC).

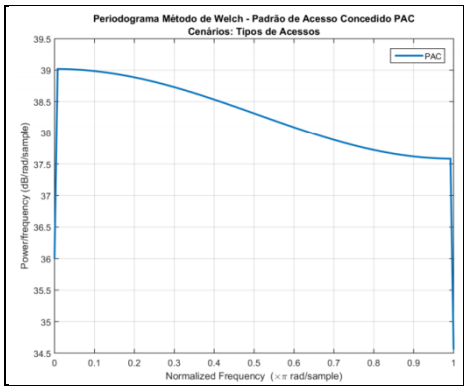


Figura 6.4: Periodograma do Método de *Welch* - PAC

A continuação na Figura 6.5 apresenta-se uma comparação dos dois métodos avaliados anteriormente.

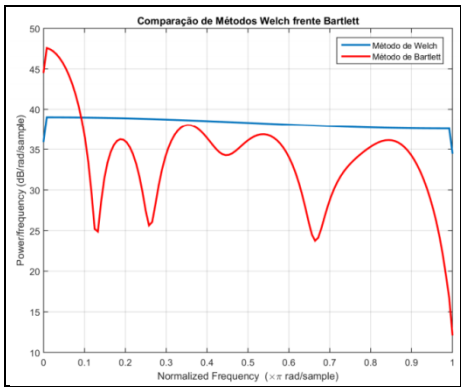


Figura 6.5: Comparação do sinal com método de *Welch* em relação *Bartlett*

Observa-se que o método de *Welch* permite suavizar o espectro de um sinal na medida em que permite diminuir a variância entre os estimadores. Os métodos de representação gráficos propostos anteriormente nesta subseção, sugerem que o método de *Welch* possui a melhor representação dos sinais padrões obtidos como resultado da execução dos cenários de avaliação propostos.

6.8 Modelo Matemático de Estimação do Consumo de Corrente

Nesta seção apresenta-se uma proposta matemática teórica do cálculo da corrente consumida pelo dispositivo de *hardware* embarcado *Raspberry Pi 2 Model B*.

6.8.1 Fundamentação Teórica

O ponto de começo é situando-se no contexto energético onde a corrente consumida pela plataforma de teste baseada *Raspberry Pi 2 Model B*, está relacionada diretamente a duas

questões, a primeira é baseada em uma abordagem de quantidade de pacotes enviados e recebidos (transmissão), e a segunda nas intrusões vinculadas ao sistema operacional *Linux Raspbian* que interpreta e executa.

A abordagem baseada em pacotes explica o consumo de corrente identificado na plataforma de teste proposta. Esta justificativa está fundamentada na arquitetura de funcionamento do modelo de referência *OSI* [3].

Um detalhe importante a destacar no uso do modelo *OSI* frente ao modelo *TCP/IP*, é que o modelo *OSI* define como as regras governam a sintaxe, a semântica e a sincronização da comunicação. No segundo caso o modelo *TCP/IP* é o protocolo desenvolvido seguindo as regras do modelo de referência *OSI*.

Na Figura 6.6 apresenta-se o modelo *OSI* onde estão descritas as diferentes camadas que a compõem. A seguir apresentamos as camadas alvo de estudo que explicam a abordagem baseada em pacotes.

A camada de Rede (3) apresenta toda a capacidade de comunicação entre duas entidades que desejam estabelecer uma comunicação, os endereços lógicos denominados *IPs*, e o encaminhamento ou roteamento do pacote, como também a fragmentação de pacotes, um exemplo de uso desta camada é através do protocolo *ICMP* **RFC-792**.

Este protocolo é utilizado para o controle e notificação de erro do protocolo *IP* **RFC-791**, entre outras tarefas logra identificar o estado do serviço avaliado (disponível ou não disponível), identificar falhas em rotas de encaminhamento.

A camada de Aplicação (7) possui a função de administrar todas às aplicações do usuário final, ademais de proporcionar os protocolos de nível superior para que uma comunicação seja feita. Por exemplo, *SSH*, *HTTP, FTP* **RFC-765***, TELNET* **RFC-854**.

Dados	(7) Aplicação	Fornece serviços às aplicações.
	(6) Apresentação	Cifra e compressão de dados. Assegura a compatibilidade entre camadas de aplicação de sistemas diferentes.
	(5) Sessão	Controla (estabelece, faz a gestão e termina), as sessões entre aplicações.
Segmentos	(4) Transporte	Controle de fluxo de informação, segmentação e controle de erros.
Pacotes	(3) Rede	Encaminhamento (*routing*) de pacotes e fragmentação. Esquema de endereçamento lógico.
Quadros	(2) Enlace de Dados	Controla o acesso ao meio físico de transmissão. Controle de erros da camada física
Bits	(1) Física	Define as características do meio físico de transmissão da rede, conectores, interfaces, codificação ou modulação de sinais.

Figura 6.6: Modelo de referência *OSI*

Apresentado o fundamento teórico das camadas do modelo *OSI* alvo de estudo, pode-se explicar que o fato de enviar pacotes de uma origem a um destino conhecido, é uma atividade que trabalha com a camada de Rede (3) do modelo *OSI*. Esta ação de troca de pacotes onde estão associadas as camada Física (1) até a camada de Rede (3), gera uma série de eventos interpretados pelo sistema operacional que tecnicamente gera um consumo energético explicado pelo aumento de consumo de corrente do *hardware* embarcado *Raspberry Pi 2 Model B* ou plataforma de teste. Na Figura 6.7 apresenta-se um exemplo do uso das três primeiras camadas do modelo *OSI* através do uso do protocolo *ICMP*, por meio da ferramenta *Ping* que envia mensagens de petição *ECHO* e recebe respostas *ECHO* para determinar se um destino está disponível.

Na parte superior do gráfico se identificam dois pacotes (No 1, No 2), seguidamente os respectivos endereços *IPs* de origem (*Source*) e destino (*Destination*) e finalmente o protocolo associado *ICMP*. Na parte inferior do gráfico se visualiza (4) linhas onde a primeira identificada pelo nome *Frame* referência ao pacote (No 1), com um tamanho de 96 *Bytes*. A segunda linha identificada pelo nome *Ethernet II* é a camada de Enlace de Dados (2) onde apresentam-se os endereços *MAC* (*Media Access Control*) do nó origem e nó destino. A terceira linha corresponde à camada de Rede (3) onde identifica-se o endereço origem e endereço destino dos nós que fornecem o canal de comunicação. Por último, na quarta linha identifica-se o protocolo de mensagem *ICMP* que trabalha sob o protocolo *IP*.

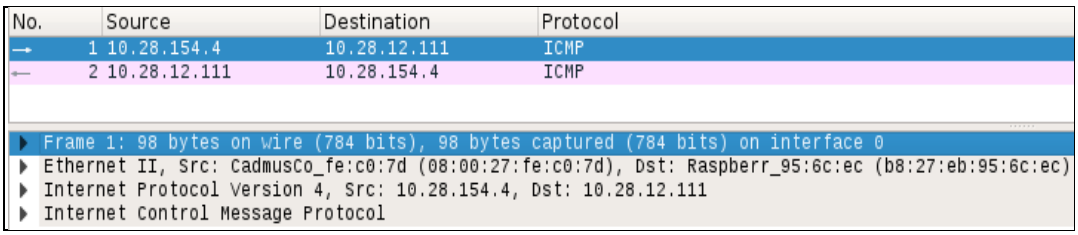


Figura 6.7: Pacote *ICMP* gerado através da ferramenta *Ping*

Por outro lado, o fato de estabelecer um canal de comunicação através do protocolo *SSH*, que recorre todas as camadas do modelo *OSI*, ou seja desde a camada Física (1) até camada de Aplicação (7), apresenta administrar e controlar um conjunto maior de eventos gerados por cada camada. Esta interpretação, sem dúvida, provoca um desenvolvimento mais de recursos e processos por parte do sistema operacional, em forma análoga interpretado no contexto energético, produz um aumento de consumo de corrente por parte do *hardware* embarcado *Raspberry Pi 2 Model B*, executando o serviço de acesso remoto *SSH*.

Na Figura 6.8 apresenta-se um exemplo do funcionamento de baixo nível, do serviço *SSH* no contexto de transmissões de pacotes. Na parte superior do gráfico observa-se os

endereços *IPs* de origem (*Source*) e destino (*Destination*) e número de pacotes (No 4-6-8-11), já na parte inferior observa-se as camadas correspondentes ao protocolo *SSH*, onde na primeira linha identifica-se ao pacote através do nome (*Frame*), na segunda linha se identificada por nome *Ethernet II* é a camada de Enlace de Dados (2) onde apresentam-se os endereços *MAC* do nó origem (01:00:5e:00:00:fb) e nó destino (b8:27:eb:95:6c:ec), a terceira linha corresponde à camada de Rede (3) onde identifica-se o endereço origem (10.28.154.4) e endereço destino (10.28.12.111) do canal de comunicação, já na quarta linha identifica-se o protocolo *TCP* trabalhando na camada de Transporte (4) onde identifica-se a porta de origem (*Src Port = 54180*) e a porta de destino (*Dst Port = 22*), por último na linha quinta identifica-se o serviço *SSH* executando-se na camada de aplicação (7).

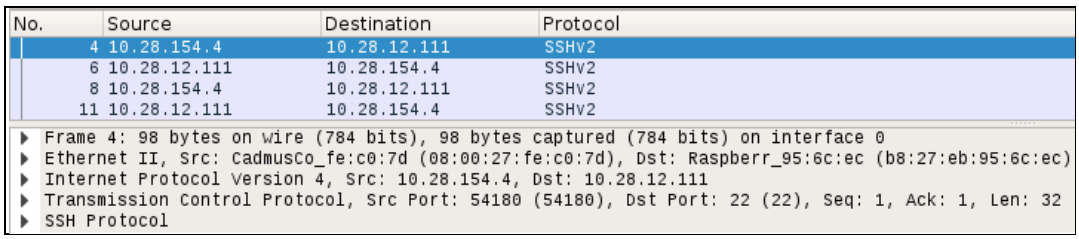


Figura 6.8: Análise de um pacote *SSH*

6.8.2 Modelo Matemático do Consumo de Corrente *1-Thread*

Em base ao conhecimento teórico e prático adquirido a partir dos experimentos desenvolvidos nos cenários proposto, apresentados nas seções posteriores. A continuação se propõe um modelo matemático teórico que apresenta as pautas necessárias para o processo de cálculo e estimação do consumo energético, gerado pela plataforma de teste baseado no sistema embarcado *Raspberry Pi 2 Model B*, executando o sistema operacional *Linux Raspbian* trabalhando como um servidor de acesso remoto *SSH*. O modelo teórico proposto do cálculo e estimação do consumo de energia está dividido em duas abordagens que são:

Abordagem 1: Pacotes Transmitidos (Camada Física e Camada de Rede do modelo *OSI*): Está abordagem considera todos os recursos associados aos pacotes de redes enviados e recebidos (transmissão) por um nó origem e nó destino. A justificação está fundamentada através do modelo de referência *OSI* que descreve as distintas camadas pela qual um pacote transita desde uma origem até seu destino final. O alcance desta abordagem está baseado desde que se inicia uma solicitação de conexão na Camada (7) de Aplicação do nó origem (Cliente) até que a solicitação é processada pela Camada (7) de aplicação nó destino (servidor).

Abordagem 2: Intrusões Executadas no Sistema Operacional (Camada de Aplicação do modelo *OSI*): Está abordagem considera tudo o processo executado só na Camada (7) de Aplicação do modelo de referência *OSI* utilizado pelo protocolo *SSH*, para estabelecer um canal de comunicação seguro ou túnel cifra para o intercâmbio de informação.

Ao analisarmos o fluxo de execução do *Login* (Usuário:Senha) o primeiro passo é avaliar o (Usuário) potencial na base de dados do sistema operacional, se o (Usuário) existe, seguidamente realiza-se o processo de *Hashing* com a (Senha) proposta, e posteriormente se compara o resultado obtido com a senha armazenada no sistema operacional. Este processo causa que o tempo de comprovação do usuário se some ao tempo de *Hashing* da (Senha), portanto vinculando o conceito anterior a um contexto energético, logra-se identificar uma resposta associada à variação de consumo de corrente do sistema embarcado *Raspberry Pi 2 Model B*.

A abordagem 2 apresentada anteriormente possui três variações com respeito a seu funcionamento, apresentadas no Apêndice C onde a Figura C.1 é o método principal utilizado no tipo de acesso com nome de usuário e senha válida, a Figura C.2 é denominado Abordagem 2 - Caso especial C.2 que tecnicamente possui o nome de usuário valido e a senha não válida, seguidamente a Figura C.3 é denominada Abordagem 2 - Caso especial C.3 que tecnicamente possui o nome de usuário não valido e a senha válida, estas variações são considerados métodos especiais pelo fato de não fornecer uma conexão válida ao servidor *SSH*.

A continuação apresenta-se o desenvolvimento do modelo matemático teórico proposto, que calcula e estima o consumo de corrente gerado pela plataforma de teste embarcada *Raspberry Pi 2 Model B*.

Entenda-se o procedimento lógico de *Login* (Usuário:Senha), que em forma simplificada apresenta o conceito de como aceder ao serviço de acesso remoto *SSH*, através de um nome de usuário ou *ID*(identificador), e a senha associadas, dentro da perspectiva do *software*. No entanto, o conceito anterior interpretado desde uma perspectiva prática de testes definidos, onde intervierem as respostas do *hardware* que é mapeado, através de uma plataforma de medição de consumo energético, e posteriormente realizado tudo o processamento do sinal obtido, como resultado se alcança, um valor médio estimado de consumo de corrente elétrica (mA).

Definição 1: Apresentadas duas funções definidas por f e g, denota-se a soma de (f + g), como [LEITHOLD 2007]:

$$(f + g)(x) = f(x) + g(x) \quad \textbf{(6.2)}$$

Equação 6.2: Definição da soma de uma função

Particularizando a equação 6.2 da seguinte forma, x = abordagem 1, e y = abordagem 2, e (f + g) (x) é igual ao consumo estimado da corrente (mA). O modelo matemático teórico proposto do cálculo e estimação do consumo de corrente é apresentado na Equação 6.3.

$$\text{Consumo (mA)} = f(\text{Abordagem 1}) + g(\text{Abordagem 2}) \quad \textbf{(6.3)}$$

Equação 6.3: Modelo de cálculo e estimação do consumo de corrente

Na Equação 6.4 apresenta o modelo matemático proposto, com uma interpretação mais abstrata.

$$\text{Consumo (mA)} = \text{Login (Usuário:Senha)} \quad \textbf{(6.4)}$$

Equação 6.4: Modelo matemático abstrato proposto

Na Figura 6.9 apresenta-se um diagrama conceitual desde uma perspectiva energética.

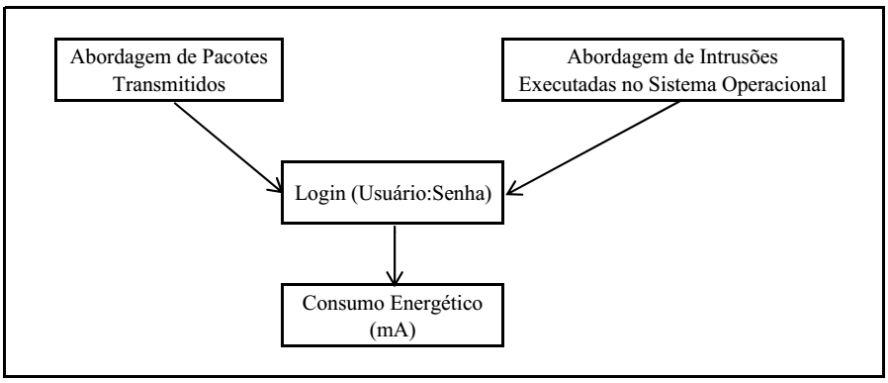


Figura 6.9: Modelo conceitual do consumo energético proposto

Os distintos esquemas de validação e/ou autenticação de usuários, mencionados anteriormente, ajudam a explicar o conceito de Threads [TANENBAUM 2003] de execução.

Um hilo de execução é a unidade de processamento menor que pode ser planejada pelo sistema operacional. O conceito de hilo de execução está mais relacionado ao processamento concorrente, os Threads permitem a execução concorrente de varias sequências de instruções associadas a diferentes funções dentro do mesmo processo, compartilhando um mesmo espaço de endereços, e as mesmas estruturas de dados do *Kernel*.

Os Threads se distinguem dos processos, em que os processos são geralmente independentes, possuem uma ampla variedade de estados, e interatuam através de mecanismos de comunicação dados pelo sistema. Entretanto os Threads compartilham outros

recursos de maneira mais direita. Os Threads são gerados a partir do fornecimento do processo, ou seja um processo é um hilo de execução, conhecido também como monohilo. A vantagem dos Threads apresenta-se quando falamos de *Multithread* que é quando um processo possui muitos Threads de execução as quais realizam atividades diferentes, que podem ser cooperativas entre se mesma ou não.

Uma vantagem a destacar está focada na programação *MultiThread*, que acelera a execução dos programas, em sistemas que possuem múltiplas *CPUs*, já que os Threads estão preparados para executar-se concorrentemente em distintos núcleos.

Do mesmo modo, apresenta-se o conceito de um *Sockets* de comunicação [GONZÁLES 2015].

Sockets é definido como um canal de comunicação, entre dois programas e/ou aplicações baseado no protocolo *TCP/IP*. Desde o ponto da programação um *Sockets* é definido como um arquivo acedido de forma especial, trabalhando com operações de leitura *read()* e escrita *write()*. Consideram-se dois tipos de *Sockets* comumente utilizados, o primeiro é denominado *Socket TCP* orientado a conexão, e o segundo *Socket UDP* não orientado a conexão.

Estes conceitos estão associados aos protocolos *TCP* e *UDP*. É importante lembrar que *TCP* garante a transmissão dos dados enviados *TCP*, porém o protocolo *UDP* realiza a transmissão dos dados, mais não garante uma recepção completa. Os *Sockets* constituem o mecanismo para a entrega de pacotes, proveniente de uma origem (processo ou *thread*), até um destino, previa definição de dois endereços *IP* e as portas associadas. Na Figura 6.10 apresenta-se um gráfico de uma comunicação entre um nó cliente e um nó servidor através de *Sockets*, as funções que formam parte do processo de inicio e fim da comunicação são:

Socket (): Função que fornece um *Socket*.

Bind (): Função que associa o endereço *IP* do servidor com uma porta local.

Listen (): Função que informa ao sistema operacional que o *Socket* foi fornecido corretamente através do endereço e porta. O *Socket* encontra-se operacional.

Connet (): Função que fornece uma conexão ativa entre o cliente e o servidor, está função utiliza como parâmetro o endereço do servidor e a porta onde recebe conexões.

Accept (): Função que recebe conexões no servidor.

Recv (): Função que recebem dados através do *Socket*.

Send (): Função que transmitem dados através do *Socket*.

Close (): Função que termina um *Socket* de comunicação.

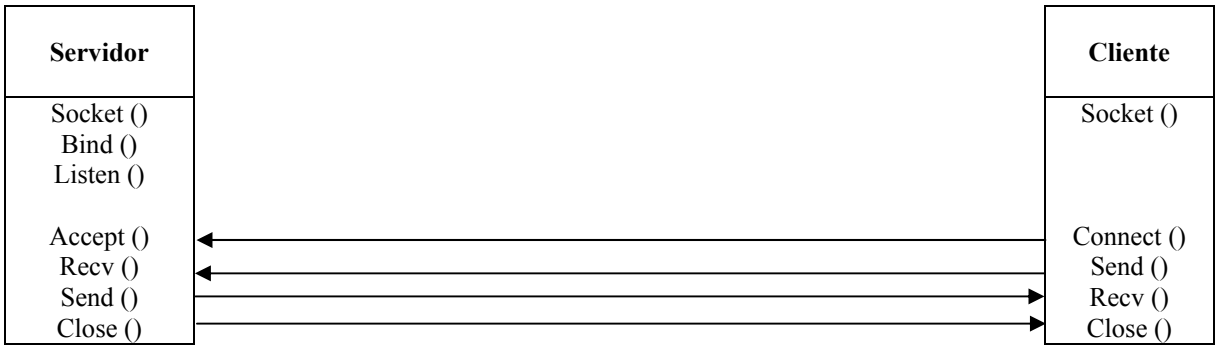


Figura 6.10: Processo de comunicação através de Sockets

6.8.3 Modelo Matemático do Consumo de Corrente para *N-Thread*

A continuação apresenta-se o modelo matemático teórico proposto, particularizado para N-Threads executando-se simultaneamente. O seguinte modelo está baseado na Equação 5.3 onde se define um modelo matemático teórico que calcula e estima o consumo de corrente dos cenários avaliados neste trabalho de dissertação.

Define-se o somatório do consumo de corrente, contido entre o limite inferior (m) até um limite superior (n), que representa a quantidade de usuários que enviam uma solicitação de acesso através do processo ou método de autenticação denominado *Login* (Usuário:Senha) ao sistema de forma concorrente. Lembre-se da relação *Login* (Usuário:Senha) = Consumo (mA).

A anterior operação se define, como a soma parcial de cada acesso concedido ou acesso negado, particularizado para cada usuário com sua correspondente senha de acesso predefinida [72]. O resultado esperado, corresponde ao consumo total de corrente (mA), causado pela resposta aos eventos externos (ameaças e/ou ataques), que interpretado pelo sistema operacional *Linux Raspbian*, desenvolve uma série de respostas, como conceder o acesso, negar o acesso. O anterior desde um contexto do domínio do *software* onde o alcance está limitado só aos cenários avaliados.

Por outro lado, se trasladamos esse conceito ao contexto do *hardware*, e particularmente ao conceito de consumo de energia, podemos lograr estimar o consumo de corrente, como as respostas aos eventos identificados pelo sistema operacional. A identificação desses eventos no contexto energético é realizado, por meio, de uma plataforma de medição de consumo de corrente desenvolvida no *hardware* embarcado *Arduino Uno*. Na Equação 6.5 apresenta-se o modelo matemático proposto.

$$\sum_{i=m}^{n} a_i = \mathbf{a_m + a_{m+1} + a_{m+2} + \ldots + a_n} \qquad \mathbf{onde\ m \leq n;\ a_n = login(usuário: senha)(mA)} \qquad (6.5)$$

Equação 6.5: Modelo matemático teórico de *N-Thread*

Feita a definição do modelo matemático teórico proposto, procedemos a realizar a particularização da Equação 6.5. O intervalo (m) até (n) representa o universo de discurso dos usuários com credenciais válidas de acesso, como também as credenciais não válidas. Lembra-se que um ataque por força bruta está formado por acessos não validos e acessos validos, este último caso é considerado um evento favorável, se o usuário e a senha de acesso este contida no dicionário utilizado pela ferramenta que está praticando o ataque.

O intento de acesso observado desde um ponto de vista geral deve-se interpretar como o processo de *login*. Por outro lado, desde um ponto de vista técnico o processo anterior é interpretado como método de autenticação de usuários, as duas acepções são válidas.

O conceito anterior está intrinsecamente associado ao conceito de consumo de corrente, interpretado no contexto energético. Por analogia, podemos estimar o consumo de corrente por intento de acesso, através da plataforma de teste *Raspberry Pi 2 Model B*. Na Equação 6.6 apresenta-se o modelo particularizado geral para os cenários proposto neste trabalho.

$$\sum_{i=m}^{n} \boldsymbol{corrente\ consumida} = \mathbf{login1(user1: pass1) + login2(user2: pass2) + \cdots + login_n(user_n: pass_n)}$$

$$\forall\ \mathbf{(Usuarío: Senha)\ contido\ no\ diccionario\ de\ usuários\ e\ senhas.}$$

Equação 6.6: Modelo matemático proposto

6.8.4 Modelo Hipotético de Identificação de Ameaças Baseado em Assinatura

Nesta subseção apresenta-se um modelo hipotético de identificação de ameaças baseado em assinaturas. Considera-se o funcionamento do Sistema de detecção de intrusão *IDS Snort*, que identifica ameaças, por meio de um arquivo de assinaturas ou regras padrões definidas. No Quadro 6.1 apresenta-se uma regra extraída do arquivo padrão de *Snort*, que identifica uma ameaça ou ataque, relacionado ao serviço de acesso remoto *SSH*.

O processo de identificação, por meio de assinatura de identificação, só é testado para determinado serviço, determinada porta, determinada combinação de *FLAGS* e determinado inicio e fim de seu conteúdo. Um exemplo do funcionamento de uma detecção feita com este método, consiste em acionar uma alarme se o pacote é *IP*, tipo de protocolo é *TCP,* porta é

número 22 e *FLAGS* é *SYN*. Esta regra alcança identificar um ataque ao serviço de acesso remoto.

Quadro 6.1: Assinatura padrão de identificação de ameaça

```
ALERT TCP $EXTERNAL_NET ANY -> $HOME_NET 22 (MSG:"ETSCAN POTENTIAL SSH SCAN"; FLAGS:S,12)
```

Uma interpretação teórica da regra padrão anteriormente proposta é apresentada na Equação 6.7.

Equação 6.7: Modelo matemático teórico de assinaturas proposto

$$Ataque = f(x, y, z, w, u) \quad \text{onde}$$

$$x=\text{EXTERNAL\_NET ANY};\ y=\$\text{HOME\_NET 22};\ z=\text{MSG:"ETSCAN POTENTIAL SSH SCAN"};\ w=\text{"FLAGS:S"};\ u=\text{"S,12"} \tag{6.7}$$

A regra apresentada anteriormente está formada por valores correspondentes, ao pacote *IP* (endereço origem, e endereço destino), seguidamente ao segmento *TCP* (porta origem e porta destino, *Bits Flags*). No Apêndice A apresentam-se os campos que definem um segmento *TCP*, seguidamente se define a composição do pacote *IP*.

O modelo proposto na Equação 6.8, é utilizado para realizar uma analogia, e definir um modelo matemático hipotético que possa identificar uma ameaça ou ataque, através da análise de varias variáveis intervenientes no processo de identificação.

A continuação apresenta-se o modelo proposto, que define uma assinatura padrão de identificação de um ameaça ou ataque na Equação 6.8 apresenta-se o modelo proposto.

$$\boldsymbol{Ataque = f(x, y, z, w, u, v)} \quad \textbf{onde}$$

$$\textbf{x=EXTERNAL\_NET ANY;}$$

$$\textbf{y=\$HOME\_NET 22;}$$

$$\textbf{z= MSG:"ETSCAN POTENTIAL SSH SCAN";}$$

$$\textbf{w="FLAGS:S";}$$

$$\textbf{u= "S,12";}$$

$$\textbf{v = CONSUMO CORRENTE (mA)} \tag{5.8}$$

Equação 6.8: Modelo hipotético de assinatura proposto

Observa-se que cada variável independente apresenta uma característica distinta relacionada ao ataque e/ou ameaça a padronizar, este fato é importante porque não é possível identificar uma ameaça ou ataque através de observar só uma variável isolada.

O modelo hipotético de detecção de ameaça e/ou ataque baseado em assinatura proposto, encontra-se particularizado para os cenários experimentais avaliados neste trabalho de dissertação.

6.9 Considerações Finais do Capítulo

O foco deste capítulo apresentado foi propor um modelo matemático teórico que calcula e estima o consumo de corrente gerado pela plataforma de teste embarcada *Raspberry Pi 2 Model B*, trabalhando como um servidor de acesso remoto *SSH*, avaliado sob um conjunto de cenário de provas definidos.

O consumo gerado pela plataforma de teste está relacionado às instruções que administra o sistema operacional, e que estão associadas aos eventos internos e externos do sistema. Entenda-se por eventos internos, por exemplo, interrupções ou chamadas ao sistema através de primitivas *fork* ou *wait*. Entenda-se por eventos externos aos associados, por exemplo, ao fornecimento de chamadas de procedimento remoto *RPC* **RFC-1057/1831**.

Particularmente se destaca que os eventos a avaliar são os cenários de provas apresentados anteriormente.

Conclui-se que a plataforma *Raspberry Pi 2 Model B* proporciona um nível de confiabilidade aceitável, que cumpre com o objetivo principal desta dissertação, que é funcionar como um servidor de acesso remoto *SSH*. Esclarece-se neste ponto da pesquisa que o serviço de acesso remoto *SSH*, está operativo e destacamos que a plataforma não apresenta problema.

CAPÍTULO 7

CENÁRIOS DE PROVAS PROPOSTOS

Considerações Gerais dos Cenários Avaliados

Nesta seção apresentam-se os cenários alvos de estudo baseados no serviço de acesso remoto *SSH*, avaliado em diferentes tipos de acessos, e diferentes ataques por força bruta com dicionário através de ferramentas especializadas e não especializadas.

Os resultados coletados são avaliados através de processos estatísticos, a fim de obter um grau de confiabilidade aceitável, este processo garante as evidências necessárias que possam discernir entre os diferentes tipos de ações ou eventos que ocorrem em resposta a estímulos externos, os cenários propostos são:

- Cenário 1 - Tipos de Acessos: Acesso Permitido;
- Cenário 2 - Tipos de Acessos: Acesso Negado;
- Cenário 3: Ataque de Força Bruta com Dicionário Através da Ferramenta especializada *Medusa*;
- Cenário 4: Ataque por Força Bruta com Dicionário Através da Ferramenta especializada *Hydra*;
- Cenário 5: Ataque por Força Bruta com Dicionário Através da Ferramenta não especializada *Metasploit*.

A seguir as considerações gerais utilizadas nas definições de todos os cenários avaliados, fato que apresenta um grau de detalhe específico para cada cenário, tanto a nível teórico de funcionamento, como em nível de implementação, os pontos propostos são:

1. As ferramentas utilizadas nos diferentes ataques por força bruta avaliados utilizam o mesmo dicionário ou arquivo, tanto para usuários como também para as senhas. O alvo desta regra é não desviar um resultado para alguma ferramenta especifica. No Quadro 7.1 apresentam-se as configurações utilizadas em todos os cenários propostos, ou seja os usuários com suas correspondentes senhas de acesso, o cumprimento das

senhas propostas é de 30 caracteres ou o equivalente a 240 Bits. Esclarece-se que é utilizado um arquivo para usuários e um arquivo para senhas em forma independente.

Quadro 7.1: Arquivo de usuários e arquivo de senhas

Arquivo de usuários	Arquivo de senhas
a. user1	&m<Vr>K9QpF.X1,7)#:w{T2!$ae+e $
b. user2	Q?E3[#rcVRyogj!6OckD9uHo^v&b^=
c. user3	:;EAV!!Q%3&{3ppZ3Y0|O,JniuJwtW
d. user4	pYO$A1/cxmZ;v6VWj}g:&RDOOSa7$F
e. user5	2fL:~YK<-j+3vAiX6N;[/kbB?qucH*

2. O serviço de acesso remoto *SSH* é o único serviço avaliado.
3. A análise de baixo nível de pacotes é feito e garantido pelo analisador de protocolo denominado *Wireshark*[34], ferramenta *Open software* que coleta e calcula todas as métricas correspondentes à transmissão de pacotes.
4. Os cenários de ataques propostos são executados através de um sistema baseado na distribuição de *Kali*[35] *Linux*, executando-se em uma máquina virtual através da ferramenta *VirtualBox*[36].
5. O tempo de avaliação do cenário está em função do tipo de acesso ou ataque efetuado.
6. Propõe-se uma curva padrão de estado estável (PEE) de consumo que é calculada no mesmo período do tempo necessário em avaliar um cenário específico proposto.
7. Os cenários de avaliação propostos estão configurados no ambiente isolado de rede, ou seja em uma topologia ponto a ponto, isto se justifica pelo fato que se a plataforma de teste embarcada está ligada a uma rede de dados local, está em condições de receber pacotes de outros nós, por exemplo, pacotes de *broadcast*. Esclarece-se que baixo este ambiente isolado o retardo do tempo se considera depreciável na comunicação.

7.1 Cenário 1 - Tipos de Acessos: Acesso Permitido

Neste cenário de avaliação, se propõe identificar a curva padrão de acesso concedido (PAC) do consumo de corrente baseado no processo de acesso, com usuário valido e senha válida, este é um exemplo básico de efetuar um acesso remoto através de *SSH*. Lembrar que este tipo de acesso é apresentado no Apêndice C Figura C.1 denominado diagrama de sequência do método de autenticação de *SSH* através de usuário valido e senha válida.

[34] www.wireshark.com
[35] www.kali.org
[36] www.virtualbox.org

7.1.1 Desenvolvimento da Prova

Neste ponto se fornece um *Script* parametrizado com o nome de usuário e sua senha válida de acesso. Esclarece-se que o tempo de *login* ao serviço *SSH* no tipo de cenário que estamos avaliando é baixo, por tanto se prevê executar comandos básicos do sistema operacional *Linux Raspbian* com o alvo de gerar carga no *CPU*.

Uma solução pratica é fornecer um *Script* onde se define, um usuário valido com senha válida que possa realizar um *login* ao servidor *SSH*, a continuação executar a criação de um arquivo de texto, apresentar a hora do sistema, e a memória disponível no servidor. No Quadro 7.2 apresenta-se o *Script* utilizado neste cenário proposto.

Quadro 7.2: Comandos utilizados no *Script* proposto do Cenário 1

sshpass -p '&m<Vr>K9QpF.X1,7)#:w{T2!$ae+e$' *SSH* user1@10.28.156.58 "touch user1.txt & users>>user1.txt & date & free -m & w"

A continuação os comandos proposto para o desenvolvimento do cenário atual avaliado são:

1. *sshpass*: Executa uma conexão remota do tipo *SSH* com um usuário e senha válida ao servidor @10.28.156.58.
2. *Touch*: Realiza a criação de um arquivo de texto plano com o nome de usuário (*touch user1.txt*) dessa sessão.
3. *Date*: Apresenta o dia e hora do sistema operacional nesse momento.
4. *Free*: Apresenta a memória livre do sistema operacional.

7.1.2 Teste de Normalidade - Tipos de Acesso PAC e PEE

Na Tabela 7.1 apresentam-se os resultados alcançados do teste de normalidade efetuado com os seguintes sinais avaliados: Padrão de Acesso Concedido (PAC) e Padrão de Estado Estável (PEE). Os resultados proporcionam *evidência* que os dados avaliados não se ajustam à distribuição normal, observa-se no (p-valor) obtido que é menor que o nível de significância definido, ou seja $\alpha = 0,05$. Na Figura 7.1 apresentam-se os histogramas associados aos sinais analisadas anteriormente, onde observa-se que os dados não ajustam a uma distribuição normal.

Tabela 7.1: Teste de normalidade proposto PAC e PEE

Teste de Normalidade Shapiro-Wilk	
Variável	p-valor
PAC	0,03
PEE	0,04

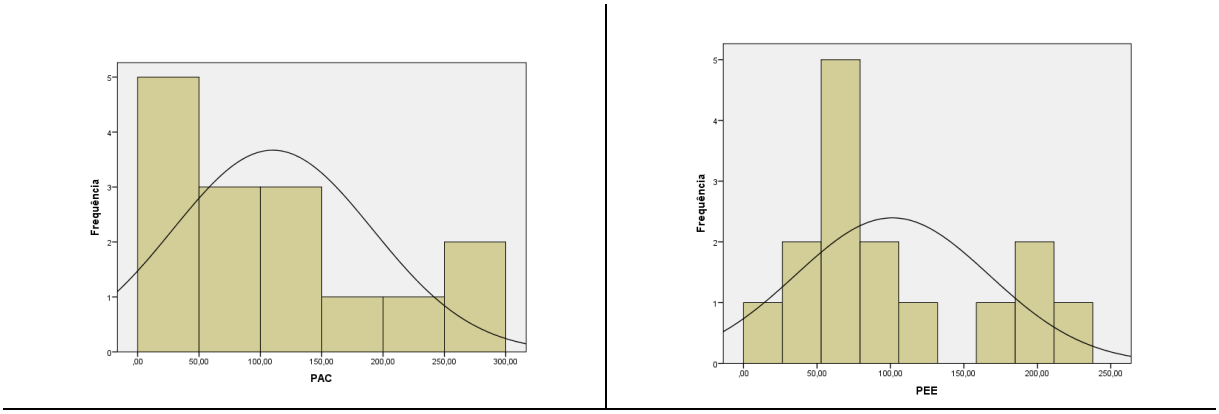


Figura 7.1: Distribuição de dados dos sinais PAC e PEE

7.1.3 Identificando um Acesso Concedido

A continuação apresenta-se o teste de *Mann-Whiteney* para duas amostras independentes ou dois sinais. O objetivo do ensaio é tentar encontrar diferenças entre as amostras propostas, ou seja o padrão de acesso concedido PAC frente ao padrão de estado estável PEE. Na Tabela 7.2 o resultado alcançado pelo teste efetuado apresenta evidência que existe diferença entre as amostras avaliadas. Isto é observado no valor de (p-valor = 0,01) rejeitando a hipótese nula H_0 que as amostras são iguais, portanto aceitamos a hipótese alternativa H_1 que existe um consumo médio de corrente distintos dos sinais PAC e PEE avaliados.

Tabela 7.2: Avaliação das amostras PAC e PEE

Acesso Concedido	
p-valor	0,01

O evento de interação entre o sistema operacional *Linux Raspbian* da plataforma de teste *Raspberry Pi 2 Model B*, e um usuário valido PAC apresenta evidências que existe um consumo de energia diferente com respeito ao estado estável PEE. Estes sinais são avaliados estatisticamente e representada graficamente através do modelo de *Welch*. Os valores de corrente para este cenário são apresentados na Tabela 7.3, pois na Figura 7.4 apresenta-se uma interpretação gráfica do comportamento energético das curvas padrões avaliadas PAC e PEE.

Conclui-se enfatizando que o fato de realizar uma conexão de acesso remoto ao serviço *SSH* com usuário valido e senha válida, gera uma curva de consumo de corrente maior com respeito ao contexto de estado estável do sistema, está diferença de consumo é de aproximadamente 8,81 mA.

Tabela 7.3: Comparação de curvas PAC frente PEE

Dados das Curvas Padrão de Acesso Concedido frente Padrão Acesso Estável	
Padrão Estado Estável (PEE)	101,27 (mA)
Padrão Acesso Concedido (PAC)	110,08 (mA)
Diferença de Corrente entre PAC e PEE	8,81 (mA)

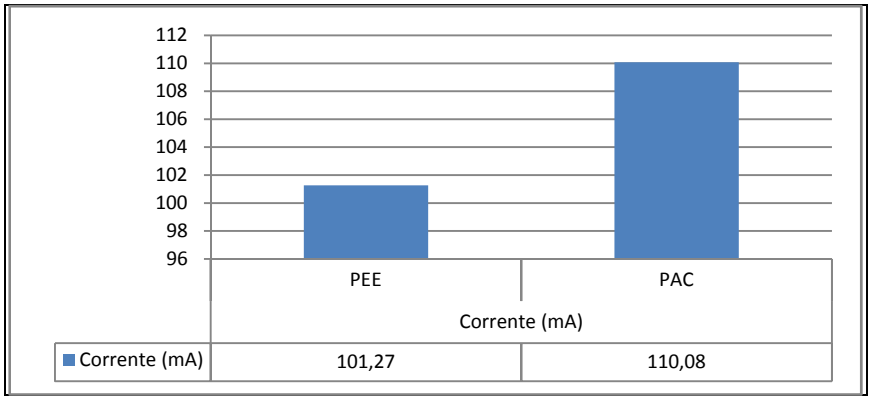


Figura 7.2: Diferença de corrente entre PAC frente PEE

O cálculo e estimação do consumo de corrente para o cenário 1 de acesso concedido, desde um ponto de vista teórico, é representada através da Equação 7.3 por tratar-se de um único *1-Thread* de execução. Além disso, o cenário apresenta um contexto de alcance onde as duas abordagens propostas na fundamentação teórica são abordadas, por conseguinte o cálculo e estimação proposto é apresentado na Equação 7.1.

$$\text{Corrente consumida (acesso concedido)} = f(\text{Abordagem 1}) + g(\text{Abordagem 2})$$
$$\text{Corrente consumida} = 8.81\ (\text{mA}) \quad \textbf{(7.1)}$$

Equação 7.1: Consumo de corrente para 1-Threads do Cenário 1

Apresenta-se na Figura 7.3 o método de *Welch* com o objetivo de realizar uma análise espectral do comportamento energético dos sinais PAC e PEE, a fim de identificar graficamente o acesso concedido frente ao estado estável da plataforma de teste embarcada baseada em *Raspberry Pi 2 Model B*.

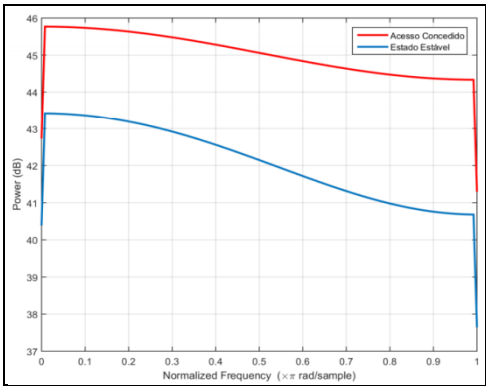


Figura 7.3: Padrão de consumo energético baseado em PAC e PEE

Observa-se na Figura 7.4 está dividida em 5 Figuras, a Figura 7.4 (a) apresenta o padrão (PAC) sem processamento, a Figura 7.4 (b) apresenta o padrão (PAC) normalizado com o método de *Welch*, a Figura 7.4 (c) apresenta o padrão (PEE) sem processamento, a Figura 7.4 (d) apresenta o padrão (PEE) normalizado com o método de *Welch*, finalmente a Figura 7.4 (e) apresenta uma combinação dos padrões (PAC) e (PEE) normalizado com o método de *Welch*, observa-se a uma diferença entre as duas curvas padrões avaliadas com respeito ao consumo gerado pela plataforma de teste *Raspberry Pi 2 Model B* em distintas provas feitas.

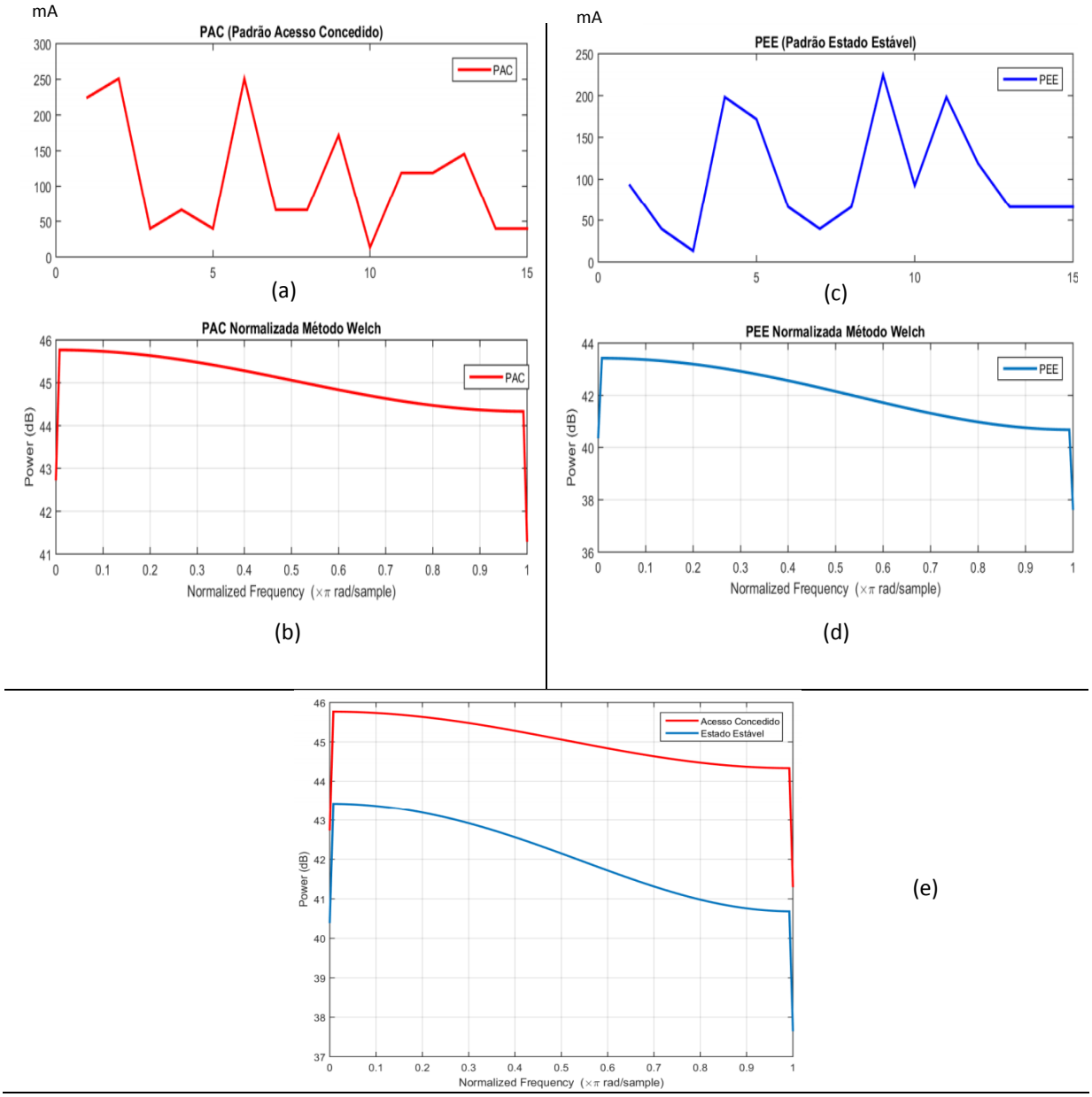


Figura 7.4: Padrões de consumo do Cenário PAC

Na Tabela 7.4 apresentam-se as seguintes métricas principais analisadas no cenário de Acesso Permitido. A métrica denominada tempo de prova (segundos), apresenta o tempo total

utilizado desde que é enviado o primeiro pacote até a recepção do último, o valor obtido é de 6 segundos, seguidamente a métrica denominada quantidade de pacotes transmitidos, apresenta o fluxo ou intercâmbio de pacotes entre o nó cliente e o nó servidor, o valor obtido é de 42 pacotes, seguidamente apresenta-se a métrica dados transmitidos (dados enviados e recebidos), obtendo um valor de 7611 *bytes*. Por último, a métrica quantidade de Threads possui um valor de 1 (um), que representa neste caso particular o processo de execução que fornece o *Socket* de comunicação entre o nó cliente e o nó servidor *SSH* para a transmissão de dados, se destaca que cada *Socket* está associado só a um hilo ou Threads de execução.

Tabela 7.4: Métricas avaliadas no cenário de acesso permitido

Tempo da Prova (segundos)	6
Quantidade de Pacotes Transmitidos	42
Dados Transmitidos (bytes)	7611
Quantidade de Threads	1

Um resultado a destacar do seguinte cenário está baseado em forçar a utilização e/ou execução de comandos do sistema operacional, intrinsecamente no mesmo instante que se produz o acesso ao sistema utilizando o túnel cifra *SSH*. Desta forma, podemos garantir um post processamento logrando executar intrusões no sistema operacional com o alvo de gerar carga no *CPU*. Desta forma o sensor de corrente da plataforma de medição de consumo energético baseada em *Arduino Uno*, logra identificar esta variação de corrente elétrica.

No Quadro 7.3 apresenta-se um modelo teórico de identificação de ameaça ou ataque fundamentado na Equação 6.7 apresentada no Capítulo 6 Subseção 6.8.4, particularizada para o cenário de acesso permitido, identificando um acesso ao serviço de acesso remoto *SSH* executando-se em um plataforma embarcada *Raspberry Pi 2 Model B*. Esclarece-se que a regra proposta não identifica um acesso concedido ao serviço *SSH*, pela combinação de *FLAGS:S,12* que mostra o *Bits SYN* ativado, junto aos *Bits* especiais *CWR* e *ESE*, está informação foi apresentada no Capítulo 6 Subseção 6.8.4 Quadro 6.1.

Quadro 7.3: Assinatura de identificação de PAC proposta

```
ALERT TCP $EXTERNAL_NET ANY -> $HOME_NET 22 (MSG:"ETSCAN POTENTIAL SSH SCAN"; FLAGS:S,12, 8.81)
```

A forma correta da assinatura proposta é apresentada no Quadro 7.4, substituindo o *FLAGS* pela variável *FLOW*, com o valor (*TO_SERVER, ESTABLISHED*), que indica o sentido de fluxo da comunicação entre o cliente e o servidor, ou seja tudo o tráfico recebido da rede externa e canalizado na rede interna pela porta 22, onde fluxo de comunicação entre o

nó cliente e nó servidor este estabelecido, e consumo de corrente está em 8,81, então ativar a alarme.

Quadro 7.4: Assinatura PAC modificada

```
ALERT TCP $EXTERNAL_NET ANY -> $HOME_NET 22  (MSG:"ETSCAN POTENTIAL SSH SCAN"; FLOW:TO_SERVER, ESTABLISHED; 8.81)
```

7.1.4 Resumo do Cenário de Acesso Permitido

Nesta seção se interpretam os resultados alcançados do cenário 1 de Acesso Permitido, logrando-se identificar uma curva padrão característica, do consumo de corrente baseado na proposta da Abordagem 1 (Pacotes Transmitidos) e abordagem 2 (Intrusões Executadas no Sistema Operacional) apresentada no Capítulo 6 Subseção 6.8.1.

O consumo estimado para o PEE é aproximadamente 101,27 mA, frente ao PAC com um valor aproximando de 110,08 mA, a diferença obtida de 8,81 mA, é atribuída à transmissão de pacotes entre o cliente e o servidor totalizando 42 pacotes transmitidos, somando também o custo de *CPU* utilizado pelo sistema operacional *Linux Raspbian* em tudo o processo de autenticação.

7.2 Cenário 2 - Tipos de Acessos: Acesso Negado

Neste cenário de avaliação se propõe identificar a curva padrão de consumo de corrente, baseado no processo de acesso negado ao serviço de acesso remoto *SSH*, com nome de usuário valido e senha não válida.

7.2.1 Desenvolvimento da Prova

Neste ponto se fornece um *Script* parametrizado com um usuário valido e sua correspondente senha não válida. No Quadro 7.5 apresenta-se o *Script* proposto onde o parâmetro (-p) identifica a senha não válida (ATTACK) de 6 caracteres alfabéticos ou o equivalente a 48 *Bits*. Esclarece-se que desde um ponto de vista da segurança a senha proposta é considerada vulnerável por duas causas, a primeira pelo comprimento e a segunda pelo uso de caracteres do alfabeto em maiúscula. No entanto não afeita ao desenvolvimento do cenário proposto.

Quadro 7.5: Acesso Negado

```
sshpass -p 'ATTACK' SSH user1@10.28.156.58
```

O comando de *sshpass* executa um acesso ao usuário chamado *user1* com a senha de acesso não válida '*ATTACK*' ao servidor @10.28.156.58. O servidor *SSH* rejeita a conexão de acesso ao usuário user1 com a senha não válida (*ATTACK*). Lembrar que este tipo de acesso não valido, possui duas variações com respeito ao método de autenticação utilizado pelo servidor *SSH*. Neste caso particular a interpretação do método de usuário valido e senha não válida, é apresentado no Apêndice C Figura C.2 denominado diagrama de sequência do método de autenticação de *SSH* através de usuário valido e senha não válida.

O seguinte Cenário de acesso negado foi executado normalmente, a resposta deste Cenário avaliado no contexto de consumo de energia foi coletada exitosamente pela plataforma de medição.

7.2.2 Teste de Normalidade - Tipos de Acesso PAN e PEE

Apresentam-se na Tabela 7.5 os resultados alcançados do teste de normalidade efetuado com os seguintes sinais avaliados: Padrão de Acesso Negado (PAN) e Padrão de Estado Estável (PEE). Os resultados proporcionam evidência que os dados avaliados não se ajustam à distribuição normal, isto é observado através do valor do p-valor que apresenta um valor menor ao nível de significância adotado de $\alpha = 0{,}05$. Na Figura 7.5 apresentam-se os histogramas associados aos sinais analisadas anteriormente, onde observa-se que os dados não ajustam a uma distribuição normal.

Tabela 7.5: Teste de normalidade proposto PAN e PEE

Teste de Normalidade Shapiro-Wilk	
Variável	p-valor
PAN	0,04
PEE	0,03

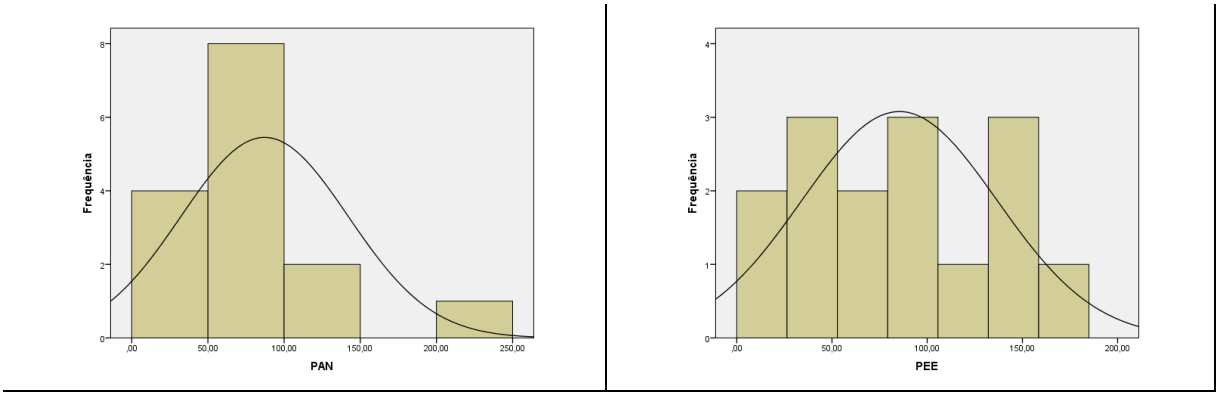


Figura 7.5: Distribuição de dados dos sinais PAN e PEE

7.2.3 Identificando um Acesso Negado

A continuação apresenta-se o teste de *Mann-Whiteney* para duas amostras independentes o objetivo do ensaio é tentar encontrar diferenças entre as amostras propostas, ou seja o padrão de acesso negado PAN frente ao padrão de estado estável PEE. Na Tabela 7.6 apresenta-se o resultado alcançado pelo teste efetuado que evidência a diferença entre as amostras avaliadas, isto é observado no valor de (p-valor = 0,01) rejeitando a hipótese nula H_0 que as amostras são iguais. Portanto aceitamos a hipótese alternativa H_1 que existe um consumo médio de corrente distintos dos sinais PAN e PEE avaliados.

Tabela 7.6: Avaliação das amostras PAN e PEE

Acesso Negado	
p-valor	0,01

O evento de interação entre o sistema operacional e um acesso não valido PAN apresenta evidência que existe um consumo de corrente diferente com respeito ao estado estável PEE. Esta última é avaliada estatisticamente e representada graficamente através do modelo de *Welch*. Os valores de corrente para este Cenário são apresentados na Tabela 7.7, pois na Figura 6.6 apresenta-se uma interpretação gráfica do comportamento energético das curvas padrões avaliadas PAN e PEE. Conclui-se enfatizando que o fato de realizar uma conexão de acesso com usuário valido e senha não válida, gera uma curva de consumo de corrente maior com respeito ao contexto de estado estável do sistema, está diferença de consumo é de aproximadamente 1,76 mA.

Tabela 7.7: Comparação de curvas PAN frente PEE

Dados das Curvas Padrão de Acesso Negado frente Padrão Acesso Estável	
Padrão Estado Estável (PEE)	85,42 (mA)
Padrão Acesso Negado (PAN)	87,18 (mA)
Diferença de Corrente entre PAN e PEE	1,76 (mA)

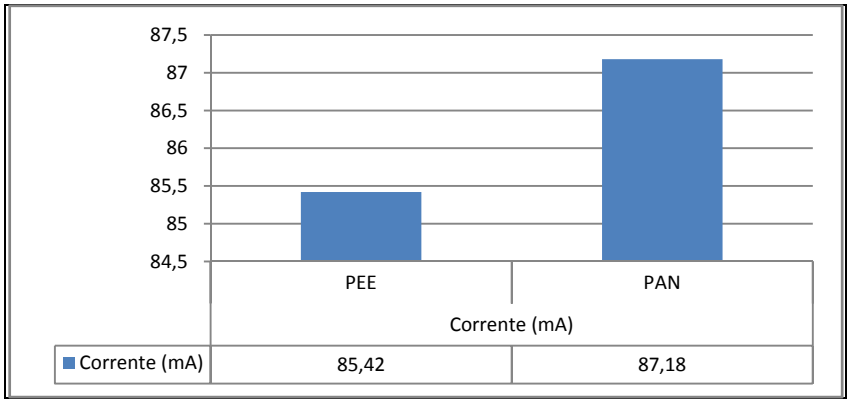


Figura 7.6: Diferença de corrente entre PAN frente PEE

O cálculo e estimação do consumo de corrente, para o Cenário 2 de acesso negado, desde um ponto de vista teórico, é representado através da Equação 6.3 por tratar-se de um único *1-Thread* de execução. Além disso, o Cenário apresenta um contexto de alcance de duas abordagens propostas, ou seja Abordagem 1 e Abordagem 2 - Caso especial C.2. Sem embargo, a abordagem 2 é um caso especial pelo fato que o processo de autenticação utiliza o algoritmo de *Hashing SHA-512* para cifrar a senha, em seguida se prossegue a comparar a senha cifrada, com a senha armazenada na base dados do sistema operacional, neste caso ao tratar-se de uma senha não válida, se impede o acesso ao sistema, tudo este processo possui como consequência, o uso do poder de computo do *CPU* através do algoritmo de *Hashing*, por conseguinte se justifica o aumento do consumo de corrente. O detalhe do processo de autenticação é apresentado no Apêndice C Figura C.2.

O processo de cálculo e estimação do consumo de corrente proposto, é apresentado na Equação 7.2.

$$\text{Corrente consumida (acesso negado)} = \text{f(Abordagem 1)} + \text{g(Abordagem 2)}$$
$$\text{Corrente consumida} = 1{,}76\ (\text{mA}) \qquad (7.2)$$

Equação 7.2: Consumo de corrente para 1-Threads do Cenário 2

Na Figura 7.7 apresenta-se o método de *Welch* com o objetivo de realizar uma análise espectral do comportamento energético dos sinais PAN e PEE, a fim de identificar graficamente o acesso negado frente ao estado estável avaliado.

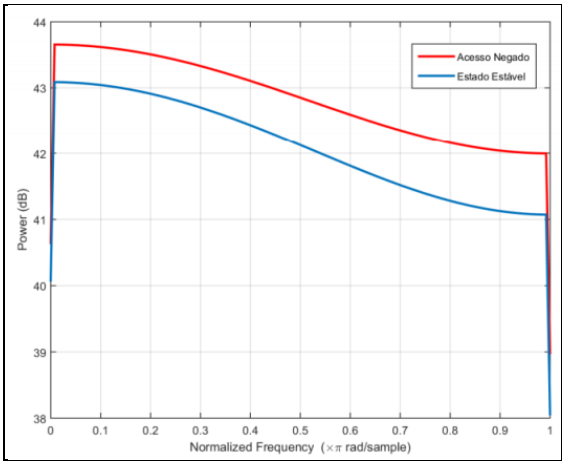


Figura 7.7: Padrão de consumo energético baseado em PAN e PEE

Observa-se na Figura 7.8 está dividida em 5 Figuras, a Figura 7.8 (a) apresenta o padrão (PAN) sem processamento, a Figura 7.8 (b) apresenta o padrão (PAN) normalizado com o método de *Welch*, a Figura 7.8 (c) apresenta o padrão (PEE) sem processamento, a Figura 7.8

(d) apresenta o padrão (PEE) normalizado pelo método de *Welch*, finalmente a Figura 7.8 (e) apresenta uma combinação dos padrões (PAN) e (PEE) normalizado com o método de *Welch*.

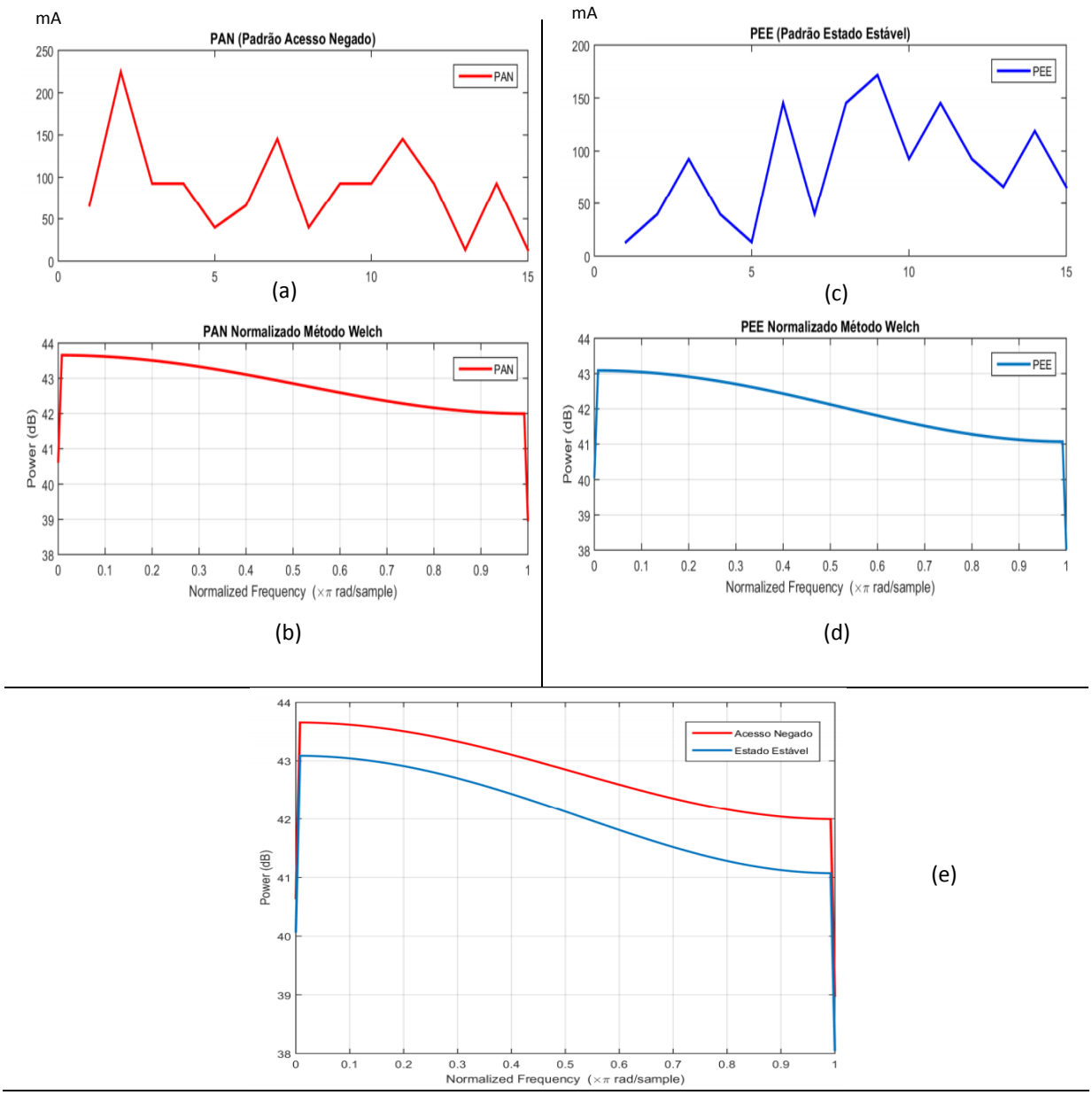


Figura 7.8: Padrões de consumo do Cenário PAN

Na Tabela 7.8 apresentam-se as seguintes métricas avaliadas do Cenário de acesso negado. O tempo total utilizado desde que se inicia o envio do primeiro pacote até a recepção do último pacote, o valor obtido é 6 segundos, seguidamente a quantidade de pacotes transmitidos nesse instante de tempo é de 27 pacotes, seguidamente a quantidade de dados transmitidos alcança um valor de 5549 *bytes*. Finalmente a quantidade de Threads presentes nesta prova é de 1 (um), responsável pelo fornecimento do único *Socket* de comunicação.

Tabela 7.8: Métricas avaliadas no Cenário PAN

Tempo da Prova (segundos)	7
Quantidade de Pacotes Transmitidos	27
Dados Transmitidos (bytes)	5549
Threads	1

No Quadro 7.6 apresenta-se um modelo teórico de identificação de ameaça ou ataque fundamentado na Equação 6.7 apresentada no Capítulo 6 Subseção 6.8.4, particularizada para o Cenário de acesso negado, identificando um acesso não permitido ao serviço de acesso remoto *SSH* executando-se em um plataforma embarcada *Raspberry Pi 2 Model B*. No Quadro 7.6 apresenta-se a assinatura modifica para a identificação do evento de acesso negado.

Quadro 7.6: Assinatura PAN modificada

ALERT TCP$EXTERNAL_NET ANY -> $HOME_NET 22(MSG"ACESSO NEGADO"; FLOW:TO_SERVER, !ESTABLISHED, 1,76

Função da regra: Identifica qualquer host que apresente o fluxo de tráfico não estabelecido entre cliente e o servidor *SSH* através da porta 22 do protocolo *TCP* e identifique um consumo de 1,76.

7.2.4 Resumo do Cenário Acesso Negado

Nesta seção se interpretam os resultados alcançados do Cenário de Acesso Negado, logrando-se identificar uma curva padrão característica, do consumo de corrente baseado na proposta da Abordagem 1 e da Abordagem 2 - Caso especial C.2 apresentada na Subseção 6.8.1 do Capítulo 6.

O consumo estimado para o PEE é aproximadamente 85.42 mA, frente ao PAC com um valor aproximando de 87.18 mA. A diferença obtida de 1,76 mA, é atribuída à transmissão de pacotes entre o cliente totalizando 27 pacotes transmitidos, somando também a execução parcial do processo de autenticação apresentado no Abordagem 2 - Caso especial C.2 denominado método de autenticação com senha não válida.

7.3 Cenário 3: Ataque de Força Bruta Através de *Medusa*

Neste Cenário de avaliação se propõe identificar a curva de consumo de corrente baseado no processo de ataque por força bruta com dicionário através da ferramenta especializada *Medusa* [72]. Um ataque por força bruta é uma técnica de busca de chaves de acesso provando todas as combinações possíveis até descobrir aquela que permite o acesso,

isto é feito através de dicionários também denominados arquivos onde se armazenam os usuários e senhas mais comuns. A ferramenta especializada *Medusa* é um *software* que fornece ataques de força bruta a um variado conjunto de serviços e/ou protocolos disponível. No Quadro 7.7 apresentam-se os serviços disponíveis que são alvos de ataques, particularmente neste trabalho o serviço a ser estudado, é o serviço de acesso remoto *SSH* proposto.

A versão da ferramenta *Medusa* utilizada é baseada na v2.2, a arquitetura de funcionamento é baseada em *thread* permitindo executar ataques a simultâneos destinos e também definir Threads de execução em função da arquitetura do hardware que fornece o ataque. Os arquivos de armazenamento de usuários e senhas podem estar alocados no mesmo arquivo padrão ou em arquivos diferentes, fato importante na hora de programar um ataque a um alvo selecionado.

Quadro 7.7: Serviços disponíveis no software *Medusa*

```
root@kali:/home/prueba/test# medusa -d
Medusa v2.2 [http://www.foofus.net] (C) JoMo-Kun / Foofus Networks <jmk@foofus.net>
Available modules in "/usr/lib/medusa/modules" :
+ cvs.mod : Brute force module for CVS sessions : version 2.0
+ ftp.mod : Brute force module for FTP/FTPS sessions : version 2.1
+ http.mod : Brute force module for HTTP : version 2.1
+ imap.mod : Brute force module for IMAP sessions : version 2.0
+ mssql.mod : Brute force module for M$-SQL sessions : version 2.0
+ mysql.mod : Brute force module for MySQL sessions : version 2.0
+ nntp.mod : Brute force module for NNTP sessions : version 2.0
+ pcanywhere.mod : Brute force module for PcAnywhere sessions : version 2.0
+ pop3.mod : Brute force module for POP3 sessions : version 2.0
+ postgres.mod : Brute force module for PostgreSQL sessions : version 2.0
+ rexec.mod : Brute force module for REXEC sessions : version 2.0
+ rlogin.mod : Brute force module for RLOGIN sessions : version 2.0
+ rsh.mod : Brute force module for RSH sessions : version 2.0
+ smbnt.mod : Brute force module for SMB (LM/NTLM/LMv2/NTLMv2) sessions : version 2.1
+ smtp-vrfy.mod : Brute force module for verifying SMTP accounts (VRFY/EXPN/RCPT TO) : version 2.1
+ smtp.mod : Brute force module for SMTP Authentication with TLS : version 2.0
+ snmp.mod : Brute force module for SNMP Community Strings : version 2.1
+ ssh.mod : Brute force module for SSH v2 sessions : version 2.1
+ svn.mod : Brute force module for Subversion sessions : version 2.1
+ telnet.mod : Brute force module for telnet sessions : version 2.0
+ vmauthd.mod : Brute force module for the VMware Authentication Daemon : version 2.0
+ vnc.mod : Brute force module for VNC sessions : version 2.1
+ web-form.mod : Brute force module for web forms : version 2.1
+ wrapper.mod : Generic Wrapper Module : version 2.0
```

7.3.1 Desenvolvimento da Prova

Neste ponto se fornece um dicionário composto por 5 usuários que são utilizados para testar as conexões de acesso ao serviço *SSH*. As senhas utilizadas neste Cenário são armazenadas em um arquivo de texto plano, seguido de outro arquivo que armazena os usuários avaliados. A aplicação utiliza estes arquivos como parâmetros de configuração para realizar o ataque ao serviço de acesso remoto *SSH*. No Quadro 7.8 apresenta-se a sintaxe da ferramenta especializada *Medusa* empregada neste Cenário de prova.

Este tipo de cenário de avaliação foi executado normalmente.

Quadro 7.8: Sintaxe da ferramenta *Medusa*

Medusa -h 10.28.12.111 -t 4 -U user.txt -P pass.txt -M *SSH* -O *Medusa*_salida.txt
-h: Representa o host a ser atacado (servidor *SSH*).
-t Quantidade de *Thread* de execução concorrentes
-U: Arquivo que contém os usuários a provar.
-P: Arquivo que contém as senhas a provar.
-M: Serviço a atacar.

No Quadro 7.9 observa-se a saída do comando executado anteriormente, identifica-se ousuário user1 com a senha de acesso (&m<Vr>K9QpF.X1,7)#:w{T2!$ae+e$) que possui um cumprimento de 30 caracteres equivalente a 240 Bits. Se destaca que está combinação de usuário e senha é válida para aceder ao servidor *SSH*. Por outro lado, o servidor *SSH* rejeita o acesso aos usuários não autorizados, e está resposta interpretada em consumo de energia é coletada exitosamente pela plataforma de medição de consumo energético baseada no *Arduino Uno*.

Quadro 7.9: Arquivo de saída do comando *Medusa*

Medusa v.2.2 (2016-08-01 08:36:52)
Medusa -h 10.28.12.111 -U user.txt -P pass.txt -M *SSH* -O *Medusa*_salida.txt
ACCOUNT FOUND: [*SSH*] Host: 10.28.12.1111
User: user1 Password: &m<Vr>K9QpF.X1,7)#:w{T2!$ae+e$ [SUCCESS]
Medusa has finished (2016-08-01 08:37:03).

7.3.2 Teste de Normalidade - Tipos de Ataques PAM e PEE

Na Tabela 7.9 apresenta-se os resultados alcançados do teste de normalidade efetuado com os seguintes sinais avaliados: Padrão de Ataque *Medusa* (PAM) e Padrão de Estado Estável (PEE). Os resultados proporcionam evidência que os dados avaliados não se ajustam à distribuição normal, isto é observado através do valor do p-valor que apresenta um valor menor ao nível de significância adotado de $\alpha = 0,05$. Na Figura 7.9 apresentam-se os histogramas associados aos sinais analisadas anteriormente, onde observa-se que os dados não ajustam a uma distribuição normal.

Tabela 7.9: Teste de normalidade proposto PAM e PEE

Teste de Normalidade Shapiro-Wilk	
Variável	p-valor
PAM	0,02
PEE	0,04

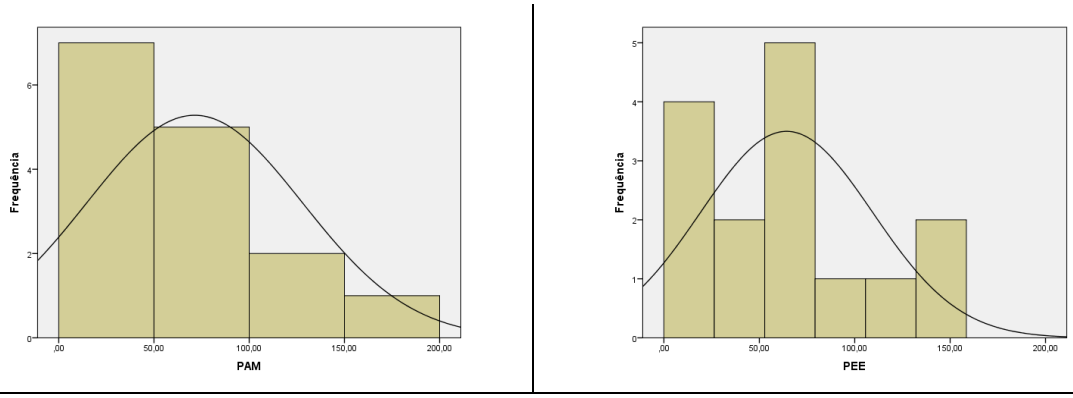


Figura 7.9: Distribuição de dados dos sinais PAM e PEE

7.3.3 Identificando um Ataque com *Medusa*

A continuação apresenta-se o teste de *Mann-Whiteney* para duas amostras independentes, o objetivo do ensaio é tentar encontrar diferenças entre as amostras propostas, ou seja o padrão de ataque *Medusa* PAM frente ao padrão de estado estável PEE. Na Tabela 7.10 apresenta-se o resultado alcançado pelo teste efetuado que evidência a diferença entre as amostras avaliadas, isto é observado no valor de (p-valor = 0,03) rejeitando a hipótese nula H_0 que as amostras são iguais. Portanto aceitamos a hipótese alternativa H_1 que existe um consumo médio de corrente distintos dos sinais PAM e PEE avaliados.

Tabela 7.10: Avaliação das amostras PAM e PEE

Medusa	
p-valor	0,03

O evento de interação entre o sistema operacional e um ataque por força bruta com dicionário através da ferramenta especializada *Medusa* ao serviço de acesso remoto *SSH*, apresenta evidências que existe um consumo de corrente diferente com respeito ao estado estável PEE. Esta última é avaliada estatisticamente e representada graficamente através do modelo de *Welch*. Os valores de corrente para este cenário são apresentados na Tabela 7.11, pois na Figura 7.10 apresenta-se uma interpretação gráfica do comportamento energético das curvas padrões avaliadas PAM e PEE. Conclui-se enfatizando que o fato de realizar um ataque por força bruta com dicionário através da ferramenta especializada *Medusa*, gera uma curva de consumo de corrente maior com respeito ao contexto de estado estável do sistema, está diferença de consumo é de aproximadamente 7,04 mA.

Tabela 7.11: Comparação de curvas PAM frente PEE

Dados das Curvas Padrão de Ataque *Medusa* frente Padrão Acesso Estável	
Padrão Estado Estável (PEE)	64,29 (mA)
Padrão Ataque *Medusa* (PAM)	71,33 (mA)
Diferença de Corrente entre PAM e PEE	7,04 (mA)

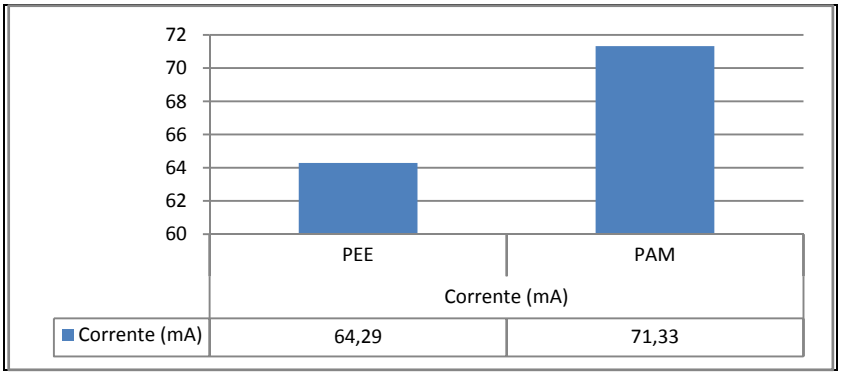


Figura 7.10: Diferença de corrente entre PAM frente PEE

O cálculo e estimação do consumo de corrente, para o cenário 3 de ataque por força bruta com dicionário através da ferramenta especializada *Medusa*, desde um ponto de vista teórico, é representada através da Equação 6.5 por tratar-se de duas ou mais conexões (N-Threads), particularmente neste cenário corresponde a 4-Threads. Além disso, o cenário apresenta um contexto de alcance onde as duas abordagens propostas estão na fundamentação teórica e são abordadas, por conseguinte o cálculo e estimação do consumo de corrente da plataforma embarcada *Raspberry Pi 2 Model B* proposto é apresentado na Equação 7.3.

Lembre-se que o seguinte ataque desde um ponto de vista teórico representa a soma de um acesso concedido e um acesso negado com senha não válida, conforme ao uso da

ferramenta que possui como alvo, vulnerar o acesso ao serviço *SSH* através da ferramenta *Medusa*.

$$\sum_{i=1}^{4} \text{corrente consumida (medusa)} = \text{login1(user1: pass1)} + \text{login2(user2: pass2)} = 7{,}04\ \text{mA} \qquad (7.3)$$

Equação 7.3: Consumo de corrente para 4-Threads do Cenário 3

Na Figura 7.11 apresenta-se um esquema de operação, desde a criação do Thread de execução (Thread x) até a combinação de usuários (m) e senhas (n) disponíveis no arquivo de usuários e no arquivo de senhas, seguidamente o contexto de ataque está limitado ao serviço *SSH,* onde o ataque é representado da seguinte forma Threads (x) = [User(m) x Pass(n)] que apresenta a combinações possíveis de ataques.

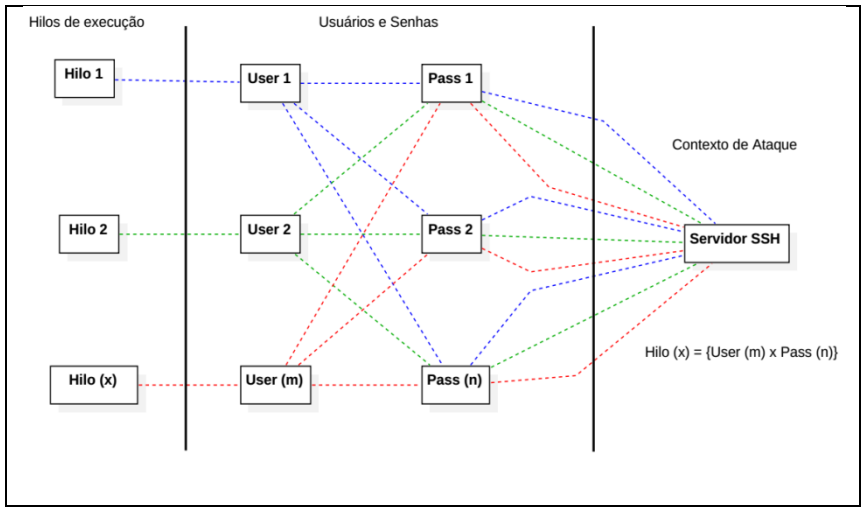


Figura 7.11: Combinação de Threads de execução com usuários e senhas

Na Figura 7.12 apresenta-se o método de *Welch* com o objetivo de realizar uma análise espectral do comportamento energético dos sinais PAM e PEE, a fim de identificar graficamente o ataque de força bruta com dicionário através da ferramenta especializada *Medusa* frente ao estado estável avaliado.

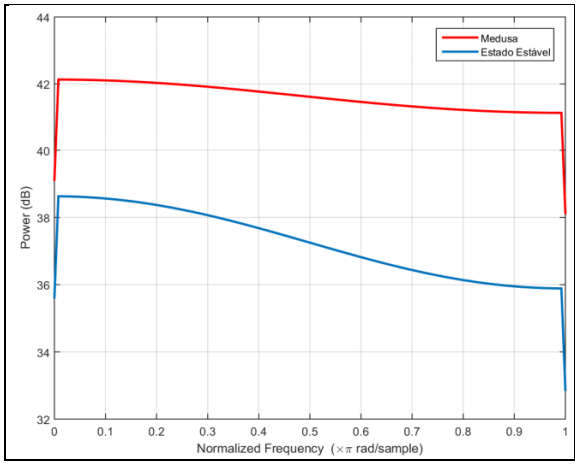
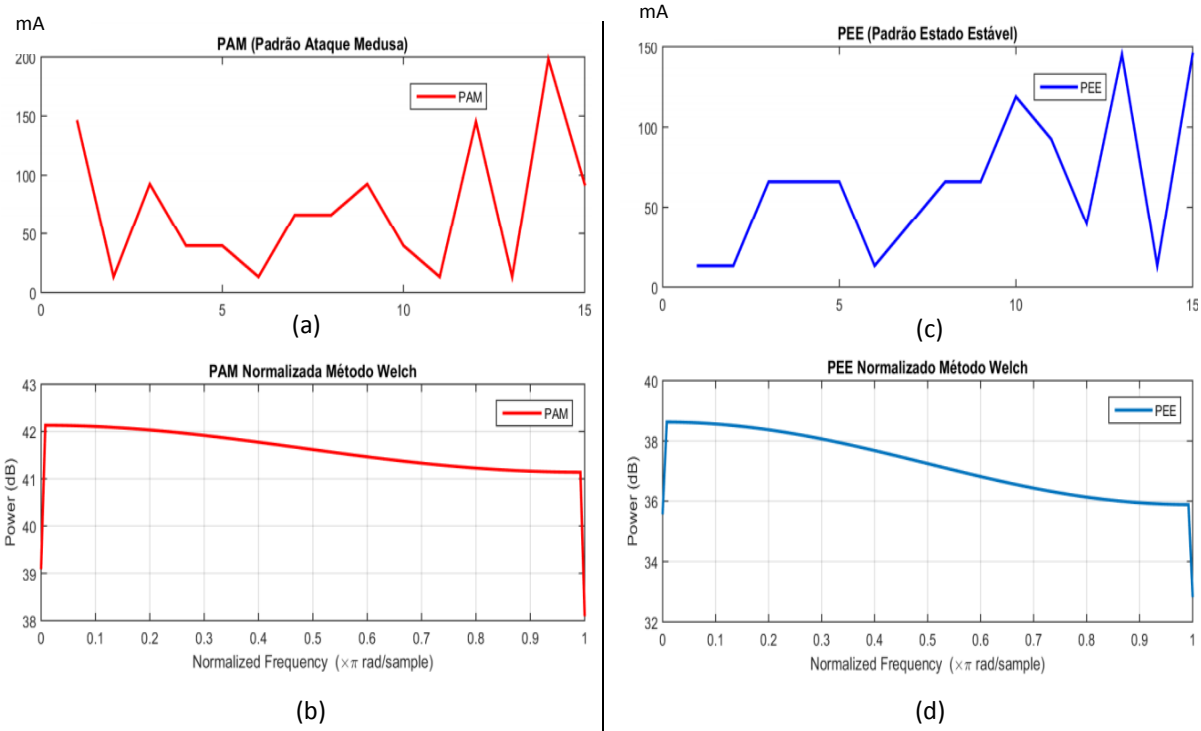
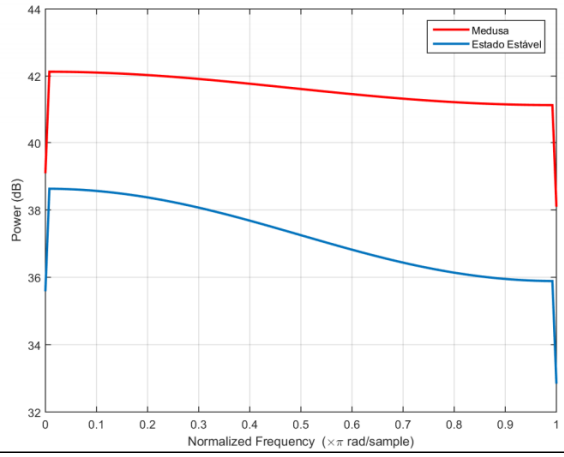


Figura 7.12: Padrão de consumo energético baseado em PAM e PEE

A continuação apresenta-se o resultado final alcançado no cenário de ataque por força bruta com dicionário através da ferramenta especializada *Medusa*. Observa-se na Figura 7.13 está dividida em 5 Figuras, a Figura 7.13 (a) apresenta o padrão (PAM) sem processamento, a Figura 7.13 (b) apresenta o padrão (PAM) normalizado com o método de *Welch*, a Figura 7.13 (c) apresenta o padrão (PEE) sem processamento, a Figura 7.13 (d) apresenta o padrão (PEE) normalizado com o método de *Welch*, finalmente a Figura 7.13 (e) apresenta uma combinação dos padrões (PAM) e (PEE) normalizado com o método de *Welch*.





(e)

Figura 7.13: Padrões de consumo do cenário PAM

Na Tabela 7.12 apresentam-se as seguintes métricas avaliadas do cenário de ataque por força bruta com dicionário através da ferramenta especializada *Medusa*. O tempo total utilizado desde que se inicia o envio do primeiro pacote até a recepção do último pacote é de 7,5 segundos, seguidamente a quantidade de pacotes transmitidos nesse instante de tempo é de 112 pacotes, seguidamente a quantidade de dados transmitidos possui um valor de 20892 bytes. Finalmente a quantidade de Threads presentes nesta prova é de 4 (quatro), responsável pelo fornecimento de 4 *Socket* de comunicação.

Tabela 7.12: Métricas avaliadas no cenário PAM

Tempo da Prova (segundos)	7,5
Quantidade de Pacotes Transmitidos	112
Dados Transmitidos (bytes)	20892
Threads	4

A continuação apresenta-se o cálculo da quantidade de pacotes transmitidos, estimado através do modelo matemático teórico proposto na Equação 6.5, a métrica quantidade de pacotes possui um grau de confiabilidade avaliado pelo software *Wireshark* que garante o valor obtido. Considerando que este cenário de ataque utiliza 4 Threads de execução para realizar o ataque atribuído a vulnerar a senha de acesso para um usuário específico. A continuação se estima a quantidade de pacotes transmitidos em função dos resultados alcançados, tanto no cenário de acesso permitido e o cenário de acesso negado abordado anteriormente. Consideraram-se estes 4 Threads, onde 3 Threads estão associados ao cenário de acesso negado, seguidamente 1 Thread é considerado um acesso permitido, a Equação 7.4 apresenta o cálculo proposto.

Pacotes (Medusa) = Acesso Negado * 3 Threads + Acesso Concedido

Pacotes (Medusa) = (27 * 3) + 40

Pacotes (Medusa) = 121

Resultado alcançado do teste 112 pacotes (7.4)

Equação 7.4: Estimação da quantidade de pacotes para o Cenário 3.

A diferença obtida de 9 pacotes no cálculo representa o 8 % mais de pacotes estimados transmitidos, isto de deve a que cada ferramenta utiliza uma implementação diferente de biblioteca que está diretamente relacionada com a arquitetura da ferramenta especializada *Medusa*, particularmente observa-se que a biblioteca utilizada é denominada *libssh2*[37] que é uma implementação do protocolo *SSHv2*, as características desta biblioteca são apresentadas no Apêndice B. Portanto podemos dizer, que o modelo teórico proposto baseado na análise de pacotes transmitidos apresenta uma taxa de confiabilidade de 92%, considerando que os 121 pacotes identificados por *Wireshark* correspondem ao 100% dos pacotes coletados.

No Quadro 7.10 apresenta-se um modelo teórico de identificação de ameaça ou ataque fundamentado na Equação 6.7, apresentada no Capítulo 6 Subseção 6.8.4, particularizada para este cenário, identificando um ataque por força bruta com a ferramenta *Medusa* ao serviço de acesso remoto *SSH*, executando-se em um plataforma embarcada *Raspberry Pi 2 Model B*.

Quadro 7.10: Assinatura PAM modificada

ALERT TCP $EXTERNAL_NET ANY -> $HOME_NET 22 (MSG:"ATAQUE MEDUSA";threshold: type both, track by_dest, count 5, seconds 5 ; 7,04)
Função da regra: Identifica qualquer host que apresente 5 intentos de conexão ao serviço *SSH* através da porta 22 do protocolo *TCP*, em qualquer dispositivo da rede interna, no intervalo de 5 segundos e identifique um consumo de 7,04.

7.3.4 Resumo do Cenário de Ataque com *Medusa*

Nesta subseção se interpretam os resultados alcançados do cenário 3 denominado Ataque por Força Bruta com Dicionário através da Ferramenta Especializada *Medusa*, logrando-se identificar uma curva padrão característica, do consumo de corrente gerado pela plataforma de teste *Raspberry Pi 2 Model B*, baseado na proposta da Abordagem 1(Pacotes Transmitidos) e Abordagem 2 - Caso especial C.2 apresentada no Capítulo 6 Subseção 6.8.1.

O consumo estimado para o PEE é aproximadamente 64,29 mA, frente ao PAM com um valor aproximando de 71,33 mA, a diferença obtida de 7,04 mA, é atribuída à transmissão

[37] https://www.libssh2.org

de pacotes entre o cliente e o servidor totalizando 112 pacotes transmitidos, somando também o custo de *CPU* utilizado pelo sistema operacional *Linux Raspbian* em tudo o processo de autenticação de usuário com senha não válida.

7.4 Cenário 4: Ataque de Força Bruta Através de *Hydra*

Neste cenário de avaliação se propõe identificar a curva de consumo de corrente baseado no processo de ataque por força bruta com dicionário através da ferramenta especializada *Hydra* [73]. O conceito de ataque é similar ao *software* ou ferramenta especializada *Medusa*, a diferença está em que cada *software* possui uma arquitetura diferente de funcionamento, a diferença mais notável é baseada na utilização das bibliotecas de manipulação do protocolo *SSH*, onde a ferramenta *Hydra* utiliza uma implementação da biblioteca *Libssh*, frente à *Medusa* que utiliza *Lissh2*, no Apêndice B apresenta-se uma comparação de cada biblioteca apresentada. A versão da ferramenta *Hydra* utilizada nesta avaliação é v3.0.

7.4.1 Desenvolvimento da Prova

Neste ponto se emprega a mesma configuração de parâmetros utilizados no cenário de prova baseado na ferramenta especializada *Medusa*, ou seja o arquivo de usuário e arquivo de senhas não teve modificação alguma. No Quadro 7.11 apresenta-se a sintaxe da ferramenta especializada *Hydra* empregada neste cenário de prova.

Este cenário de avaliação foi executado normalmente.

Quadro 7.11: Sintaxe da ferramenta *Hydra*

Hydra -h 10.28.156.58 -s SSH 22 -L -t 4 user.txt -P pass.txt -L: Arquivo que contém os usuários a provar. -P: Arquivo que contém as senhas a provar. -t: Quantidade de Threads de execução concorrentes. -s: Serviço atacar. -h: Representa o host a ser atacado.

No Quadro 7.12 observa-se a saída do comando executado anteriormente, identifica-se ao usuário user1 com a senha de acesso (&m<Vr>K9QpF.X1,7)#:w{T2!$ae+e$) que possui um cumprimento de 30 caracteres equivalente a 240 *bits*. O servidor *SSH* rejeita o acesso aos usuários não autorizados, está resposta é interpretada em função do consumo de corrente e foi

coletada exitosamente pela plataforma embarcada de medição de consumo energético baseada em *Arduino Uno*.

Quadro 7.12: Saída do comando *Hydra*

[DATA] attacking service *SSH* on port 22 [22][*SSH*] host: 10.28.156.58 login: user1 password: &m<Vr>K9QpF.X1,7)#:w{T2!$ae+e$ 1 of 1 target successfully completed, 1 valid password found

7.4.2 Teste de Normalidade - Tipos de Ataques PAH e PEE

Na Tabela 7.13 apresenta-se os resultados alcançados do teste de normalidade efetuado com os seguintes sinais avaliados: Padrão de Ataque *Hydra* (PAH) e Padrão de Estado Estável (PEE). Os resultados proporcionam evidência que os dados avaliados não se ajustam à distribuição normal, isto é observado através do valor do p-valor que apresenta um valor menor ao nível de significância adotado de $\alpha = 0,05$. Na Figura 7.14 apresentam-se os histogramas associados aos sinais analisadas anteriormente, onde observa-se que os dados não ajustam a uma distribuição normal.

Tabela 7.13: Teste de normalidade proposto PAH e PEE

Teste de Normalidade Shapiro-Wilk	
Variável	p-valor
PAM	0,02
PEE	0,04

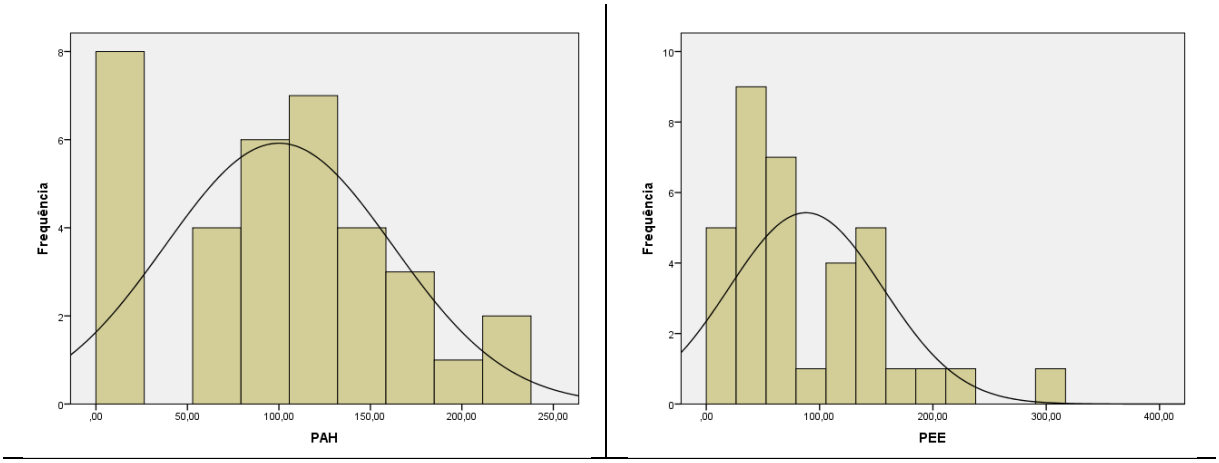


Figura 7.14: Distribuição de dados dos sinais PAH e PEE

7.4.3 Identificando um Ataque com *Hydra*

A continuação apresenta-se o teste de *Mann-Whiteney* para duas amostras independentes o objetivo do ensaio é tentar encontrar diferenças entre as amostras propostas,

ou seja o padrão de ataque *Hydra* PAH frente ao padrão de estado estável PEE. Na Tabela 7.14 apresenta-se o resultado alcançado pelo teste efetuado que evidência a diferença entre as amostras avaliadas, isto é observado no valor de (p-valor = 0,03) rejeitando a hipótese nula H_0 que as amostras são iguais. Portanto aceitamos a hipótese alternativa H_1 que existe um consumo médio de corrente distintos dos sinais PAH e PEE avaliados.

Tabela 7.14: Avaliação das amostras PAH e PEE

Hydra	
p-valor	0,03

O fato de realizar um ataque por força bruta com dicionário através da ferramenta especializada *Hydra* ao serviço de acesso remoto *SSH*, gera uma curva de consumo de corrente maior com respeito ao contexto de estado estável do sistema, está diferença de consumo é avaliada estatisticamente e representada graficamente através do modelo de *Welch*.

Os valores de corrente para este cenário são apresentados na Tabela 7.15, e na Figura 7.15 uma interpretação gráfica do comportamento energético das curvas padrões é apresentada.

Tabela 7.15: Comparação de curvas PAH frente PEE

Dados das Curvas Padrão de Ataque Hydra frente Padrão Acesso Estável	
Padrão Estado Estável (PEE)	87,94 (mA)
Padrão Ataque Hydra (PAH)	100,02 (mA)
Diferença de Corrente entre PAH e PEE	12,08 (mA)

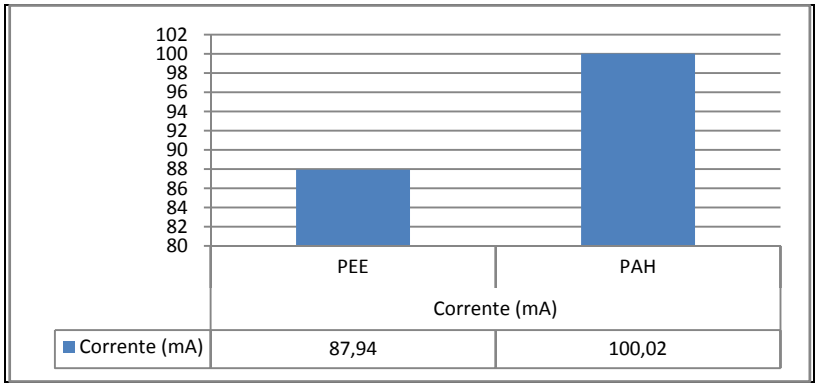


Figura 7.15: Diferença de corrente entre PAH frente PEE

O cálculo e estimação do consumo de corrente, para o cenário 4 de ataque por força bruta com dicionário da ferramenta especializada *Hydra*, desde um ponto de vista teórico, é representada através da Equação 6.5 por tratar-se de duas ou mais conexões (N-Threads), particularmente para este cenário se identificam 9-Threads. Além disso, o cenário apresenta um contexto de alcance onde as duas abordagens propostas estão na fundamentação teórica e

são abordadas, por conseguinte, o cálculo e estimação do consumo de corrente da plataforma embarcada *Raspberry Pi 2 Model B* proposto é apresentado na Equação 7.5. Lembre-se que o seguinte ataque desde um ponto de vista teórico, representa a soma de um acesso concedido e um acesso negado, conforme ao uso da ferramenta *Hydra*, que possui como alvo, vulnerar o acesso ao serviço *SSH*.

$$\sum_{i=1}^{9} corrente\ consumida\ (Hydra) = \text{login1(user1: pass1)} + \cdots + \text{login11(user11: pass11)} = 12{,}08\ \text{mA} \qquad (\mathbf{7.5})$$

Equação 7.5: Consumo de corrente para *9*-Threads do Cenário 4

Na Figura 7.16 apresenta-se o método de *Welch* com o objetivo de realizar uma análise espectral do comportamento energético dos sinais PAH e PEE, a fim de identificar graficamente o ataque de força bruta com dicionário através da ferramenta especializada *Hydra* frente ao estado estável avaliado.

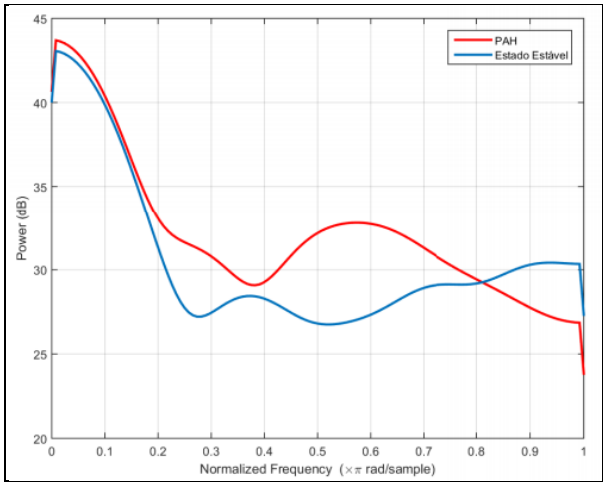


Figura 7.16: Padrão de consumo energético baseado em PAH e PEE

Nesta seção apresenta-se o resultado final alcançado do cenário de ataque por força bruta com dicionário através da ferramenta especializada *Hydra* proposta. Observa-se na Figura 7.17 está dividida em 5 Figuras, a Figura 7.17 (a) apresenta o padrão (PAH) sem processamento, a Figura 7.17 (b) apresenta o padrão (PAH) normalizado com o método de *Welch*, a Figura 7.17 (c) apresenta o padrão (PEE) sem processamento, a Figura 7.17 (d) apresenta o padrão (PEE) normalizado pelo método de *Welch*, finalmente a Figura 7.17 (e) apresenta uma combinação dos padrões (PAH) e (PEE) normalizado com o método de *Welch*.

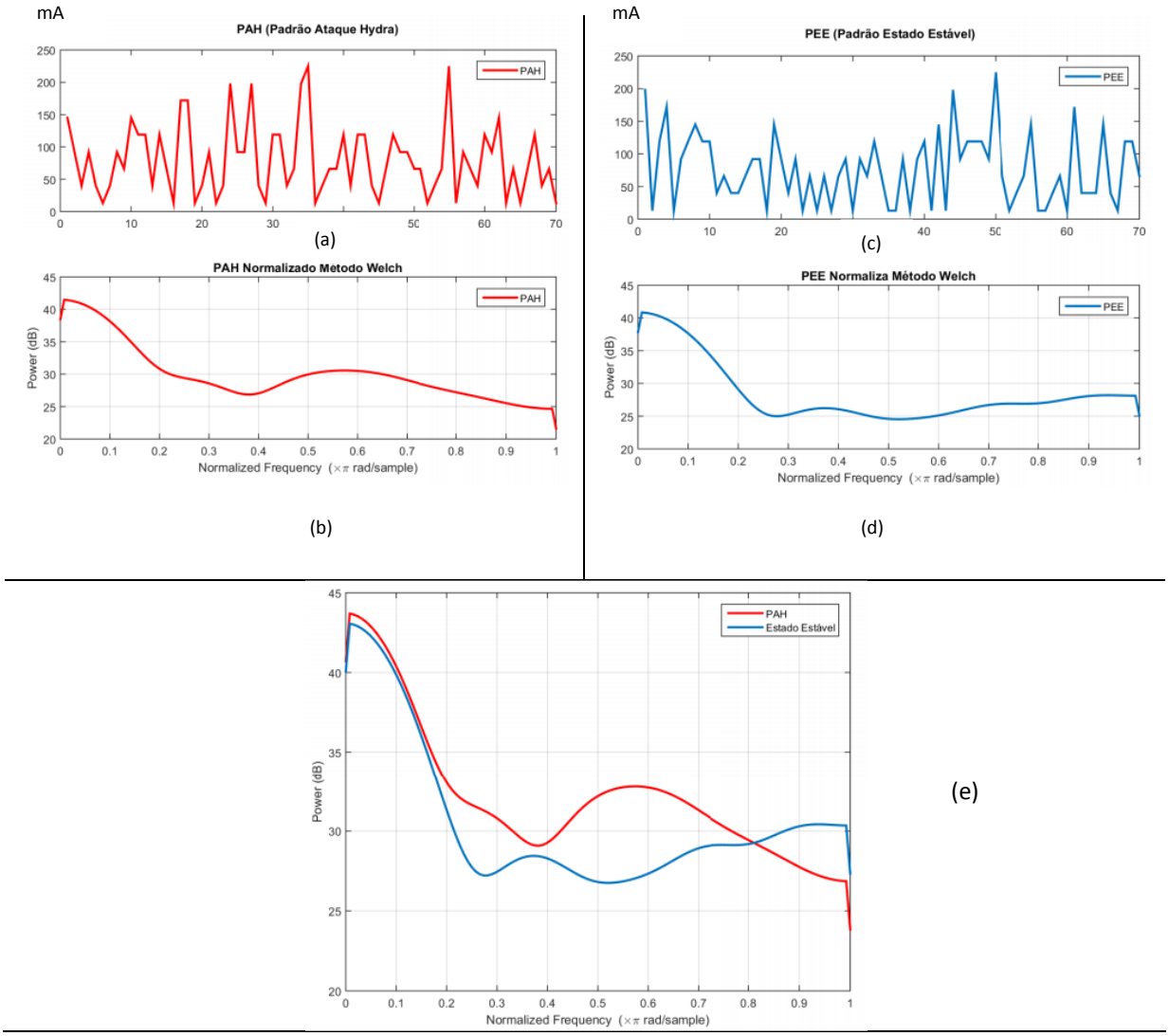


Figura 7.17: Padrões de consumo do cenário PAH

Na Tabela 7.16 apresentam-se as seguintes métricas avaliadas do cenário de ataque por força bruta com dicionário através da ferramenta especializada *Hydra*. O tempo total utilizado desde que se inicia o envio do primeiro pacote até a recepção do último pacote é de 12 segundos, seguidamente a quantidade de pacotes transmitidos nesse instante de tempo é de 253 pacotes, seguidamente a quantidade de dados transmitidos possui um valor de 38382 *bytes*. Finalmente a quantidade de Threads presentes nesta prova é de 9 (nove), responsável pelo fornecimento de 9 Sockets de comunicação.

Tabela 7.16: Métricas avaliadas no cenário PAH

Tempo da Prova (segundos)	12
Quantidade de Pacotes Transmitidos	253
Dados Transmitidos (bytes)	38382
Threads	9

Considerando que este cenário de ataque utiliza 9 *Thread* de execução para realizar o ataque atribuído a vulnerar a senha de acesso para um usuário específico, a continuação se estima a quantidade de pacotes transmitidos, em função dos resultados alcançados, tanto no cenário de acesso permitido e o cenário de acesso negado tratados anteriormente, se consideramos estes 9 Threads, onde 8 Threads estão associados ao cenário de acesso negado, seguidamente 1 hilo é considerado um acesso concedido, a Equação 7.6 apresenta o cálculo proposto.

$$\text{Pacotes (Hidra)} = \text{Acesso Negado} * 8 \text{ Threads} + \text{Acesso Concedido}$$
$$\text{Pacotes (Hydra)} = (27 * 8) + 40$$
$$\text{Pacotes (Hydra)} = 256$$
$$\text{Resultado alcançado do teste } 253 \qquad (7.6)$$

Equação 7.6: Estimação da quantidade de pacotes para o Cenário 4

A diferença obtida de 3 pacotes no cálculo, representa o 1,1 % mais de pacotes estimados transmitidos, isto de deve a que cada ferramenta utiliza uma implementação diferente de biblioteca, particularmente a ferramenta especializada *Hydra* utiliza a biblioteca *libssh*[38]. Portanto podemos dizer que o resultado alcançado em termos de pacotes transmitidos, se encontra dentro dos valores aceitáveis, cumprindo o objetivo de vulnerar um acesso ao serviço *SSH*. Portanto podemos dizer que o modelo teórico proposto baseado na análise de pacotes transmitidos apresenta uma taxa de confiabilidade de 98,08%, considerando que os 256 pacotes identificados por *Wireshark* correspondem ao 100% dos pacotes coletados. No entanto o tempo de duração do ataque é de 12 segundos, valor muito excessivo frente à ferramenta *Medusa* que totalizou um tempo de Hacking da senha em apenas 7,5 segundos, comparação válida só para ferramentas especializadas.

No Quadro 7.13 apresenta-se um modelo teórico de identificação de ameaça ou ataque fundamentado na Equação 6.7 apresentada no Capítulo 6 Subseção 6.8.4, particularizada para este cenário, identificando ataque com a ferramenta *Hydra* ao serviço de acesso remoto *SSH* executando-se em um plataforma embarcada *Raspberry Pi 2 Model B*.

Quadro 7.13: Assinatura PAH modificada

ALERT TCP $EXTERNAL_NET ANY -> $HOME_NET 22 (MSG:"ATAQUE HYDRA";threshold: type both, track by_dest, count 5, seconds 5 ; 12,08)
Função da regra: Identifica qualquer host que apresente 5 intentos de conexão ao serviço *SSH* através da porta 22 do protocolo *TCP*, em qualquer dispositivo da rede interna, no intervalo de 5 segundos e identifique um consumo de 12,08.

[38] https://www.libssh.org

7.4.4 Resumo do Cenário de Ataque com *Hydra*

Nesta subseção se interpretam os resultados alcançados do Cenário 4 de Ataque por Força Bruta com Dicionário através da Ferramenta Especializada *Hydra*, logrando-se identificar uma curva padrão característica, do consumo de corrente gerado pela plataforma de teste *Raspberry Pi 2 Model B*, baseado na proposta da Abordagem 1 (Pacotes Transmitidos) e Abordagem 2 - Caso especial C.2 apresentada no Capítulo 6 Subseção 6.8.1

O consumo estimado para o PEE é aproximadamente 87,94 mA, frente ao PAM com um valor aproximando de 100,02 mA. A diferença obtida de 12,08 mA, é atribuída à transmissão de pacotes entre o cliente e o servidor totalizando 253 pacotes transmitidos, somando também o custo de *CPU* utilizado pelo sistema operacional *Linux Raspbian* em tudo o processo de autenticação.

Destaca-se que o tempo de *Hacking* da senha de acesso avaliado neste cenário possui um valor de 12 segundos, este valor comparado com o alcançado pela ferramenta *Medusa* de 7,5 segundos é 60% mais lento, esta diferença é fundamentada nas bibliotecas implementadas pelas distintas ferramentas, a presente comparação é só válida para as ferramentas otimizadas para ataque de força bruta com dicionário.

7.5 Cenário 5: Ataque de Força Bruta Através de *Metasploit*

Neste cenário de avaliação se propõe identificar a curva de consumo de corrente baseado no processo de ataque por força bruta com dicionário através da ferramenta não especializada *Metasploit* [74], está ferramenta explora informação sobre as vulnerabilidades de segurança em distintos níveis de execuções de um sistema. Entenda-se isto como vulnerabilidades de falhas de *software*, e falhas de *hardware*, que derivam em uma falha de segurança. Neste trabalho se utiliza um *módulo* chamado *SSH_login* proporcionado pelo *Framework Metasploit*, este módulo efetua um ataque por força bruta similar às ferramentas avaliadas anteriormente, se faz uso de um arquivo de usuários como também de um arquivo onde estão armazenadas as senhas.

A avaliação deste cenário proposto desde um ponto de vista de um ataque por força bruta é similar aos cenários avaliados anteriormente, a diferença principal é que se trata de uma ferramenta não especializada para ataques de força bruta com dicionário, por isto a biblioteca que utilizada a ferramenta não está otimizada para fornecer este tipo de ataque, que como no caso de *Medusa* e *Hydra* as bibliotecas estão otimizadas.

A eleição de uma ferramenta não especializada é para contrastar as métricas avaliadas em cenários de ferramentas especializadas, e compreender a importância de definir uma ferramenta que aperfeiçoe um ataque definido, em tempo execução e uso de recursos computacionais. A versão utilizada de Metasploit é v4.12.

7.5.1 Desenvolvimento da Prova

Neste ponto apresenta-se a configuração utilizada no *Framework Metasploit*, com o alvo de efetuar um ataque por força bruta ao servidor de acesso remoto *SSH*, executando-se na plataforma de teste *Raspberry Pi 2 Model B*. No Quadro 7.14 apresenta-se a sintaxe do *Framework Metasploit* empregada neste cenário de prova.

Este cenário de avaliação foi executado normalmente.

Quadro 7.14: Arquivo de configuração de *Metasploit*

PASS_FILE = Arquivo que contém as senhas
RHOSTS = 10.28.156.58 Servidor *SSH*
RPORT 22 = Porta do Servidor
USER_FILE = Arquivo que contém os usuários
Quantidade de *Thread* = 4

No Quadro 7.15 a saída do *Framework Metasploit* executando um ataque de força bruta com dicionário, o resultado identifica ao usuário *user1* com a senha de acesso (&m<Vr>K9QpF.X1,7)#:w{T2!$ae+e$)) que possui um cumprimento de 30 caracteres equivalente a 240 *bits*. O *Framework* fornece uma conexão ativa ao sistema, isto é, identificado através da seguinte linha "*Command shell session 1 opened (10.28.154.4:40686 -> 10.28.156.58:22) at 2016-07-29 15:36:52 -0400*", onde o endereço IP e a porta do sistema que vulnera o acesso é identificado com *10.28.154.4:40686* e o sistema vulnerado é identificado através do endereço *IP* e porta *10.28.156.58:22.*

A diferença desta ferramenta frente às avaliadas anteriormente é que o *Framework Metasploit* uma vez que realiza o *Hacking* da sessão do servidor *SSH* e ganha privilégios de acesso, fornece uma sessão ativa com o servidor *SSH*.

As ferramentas como *Medusa* e *Hydra* só realizam a comprovação do usuário e a senha, sem fornecer uma conexão ativa. O exposto anteriormente supõe uma diferença de consumo de corrente por parte deste esquema de trabalho desde um ponto de vista energético, pelo contrário desde um ponto de vista funcional, o *Framework Metasploit* oferece um conjunto de utilidades como uma *Shell* de comandos, que permite a execução de *Scripts* e comandos

arbitrários, ademais possui um *Meterpreter* que é uma utilidade que permite aos usuários controlar a tela do dispositivo vulnerado mediante um serviço *VNC* RFC-6143.

Quadro 7.15: Saída da ferramenta *Metasploit*

```
[*] Command shell session 1 opened (10.28.154.4:40686 -> 10.28.156.58:22) at 2016-07-29 15:36:52 -0400
[-] 10.28.156.58:22 SSH - Failed: ':&m<Vr>K9QpF.X1,7)#:w{T2!$ae+e$'
[-] 10.28.156.58:22 SSH - Failed: ':Q?E3[#rcVRyogj!6OckD9uHo^v&b^='
[-] 10.28.156.58:22 SSH - Failed: '::;EAV!!Q%3&{3ppZ3Y0|O,JniuJwtW'
[-] 10.28.156.58:22 SSH - Failed: ':pYO$A1/cxmZ;v6VWj}g:&RDOOSa7$F'
[-] 10.28.156.58:22 SSH - Failed: ':2fL:~YK<-j+3vAiX6N;[/kbB?qucH*'
[*] Scanned 1 of 1 hosts (100% complete)
```

7.5.2 Teste de Normalidade - PAME (Padrão Ataque *Metasploit*)

Na Tabela 7.17 apresentam-se os resultados alcançados do teste de normalidade efetuado com os seguintes sinais avaliados: Padrão de Ataque *Metasploit* (PAME) e Padrão de Estado Estável (PEE). Os resultados proporcionam evidência que os dados avaliados não se ajustam à distribuição normal, isto é observado através do valor do p-valor que apresenta um valor menor ao nível de significância adotado de $\alpha = 0{,}05$. Na Figura 7.18 apresentam-se os histogramas associados aos sinais analisadas anteriormente, onde observa-se que os dados não ajustam a uma distribuição normal.

Tabela 7.17: Teste de normalidade proposto PAME e PEE

Teste de Normalidade Shapiro-Wilk	
Variável	p-valor
PAME	0,04
PEE	0,04

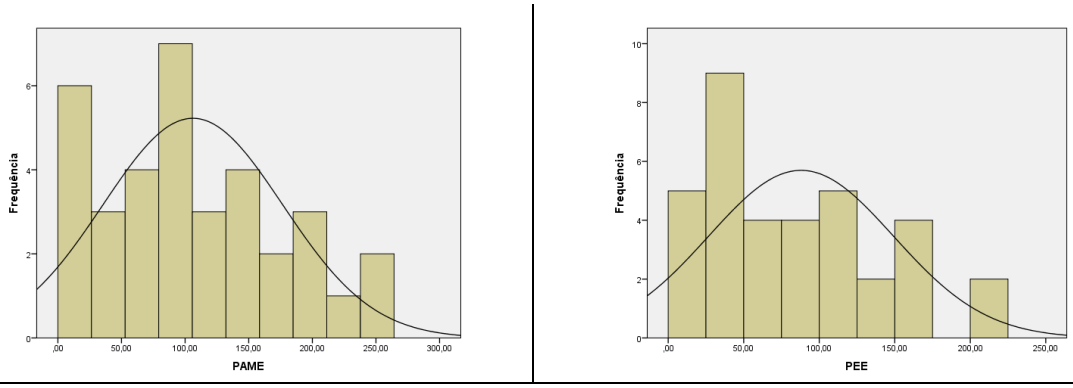


Figura 7.18: Distribuição de dados dos sinais PAME e PEE

7.5.3 Identificando um Ataque com *Metasploit*

A continuação apresenta-se o teste de *Mann-Whiteney* para duas amostras independentes o objetivo do ensaio é tentar encontrar diferenças entre as amostras propostas, ou seja o padrão de ataque *Metasploit* PAME frente ao padrão de estado estável PEE. Na Tabela 7.18 apresenta-se o resultado alcançado pelo teste efetuado que evidência a diferença entre as amostras avaliadas, isto é observado no valor de (p-valor = 0,03) rejeitando a hipótese nula H_0 que as amostras são iguais. Portanto aceitamos a hipótese alternativa H_1 que existe um consumo médio de corrente distintos dos sinais PAME e PEE avaliados.

Tabela 7.18: Avaliação das amostras PAME e PEE

Metasploit	
p-valor	0,03

O fato de realizar um ataque por força bruta com dicionário através da ferramenta não especializada *Metasploit* ao serviço de acesso remoto *SSH* gera uma curva de consumo de corrente maior com respeito ao contexto de estado estável do sistema, esta diferença de consumo é de aproximadamente 18,11 mA. Esta diferença é avaliada estatisticamente e representada graficamente através do modelo de *Welch*. Os valores de corrente para este cenário são apresentados na Tabela 7.19 e na Figura 7.19 uma interpretação gráfica do comportamento energético das curvas padrões avaliadas.

Tabela 7.19: Comparação de curvas PAME frente PEE

Dados das Curvas Padrão de Ataque *Metasploit* frente Padrão Acesso Estável	
Padrão Estado Estável (PEE)	87,94 (mA)
Padrão Ataque *Metasploit* (PAME)	106,02 (mA)
Diferença de Corrente entre PAME e PEE	18,11 (mA)

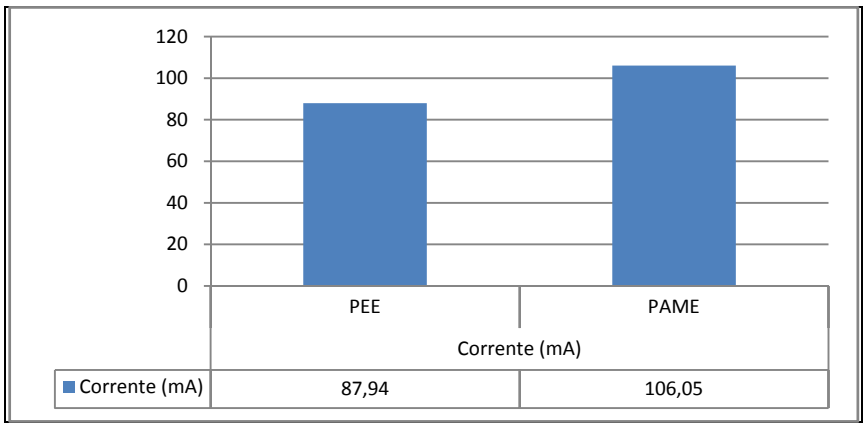


Figura 7.19: Diferença de corrente entre PAME frente PEE

O cálculo e estimação do consumo de corrente, para o Cenário 5 de ataque por força bruta com dicionário através da ferramenta não especializada *Metasploit*, desde um ponto de

vista teórico, é representada através da Equação 6.5 por tratar-se de duas ou mais conexões (N-Threads), particularmente de 5-Threads. Além disso, o cenário apresenta um contexto de alcance onde as duas abordagens propostas na fundamentação teórica são abordadas, por conseguinte, o cálculo e estimação do consumo de corrente da plataforma embarcada *Raspberry Pi 2 Model B* proposto é apresentado na Equação 7.7.

Lembre-se que o seguinte ataque desde um ponto de vista teórico, representa a soma de um acesso concedido e um acesso negado, conforme ao uso da ferramenta não especializada *Metasploit*, que possui como alvo vulnerar o acesso ao serviço *SSH*, e fornecer um canal de comunicação persistente com o nome de usuário e senha vulnerado.

$$\sum_{i=1}^{5} corrente\ consumida\ (Metasploit) = \text{login1(user1: pass1)} + \cdots + \text{login4(user4: pass4)} = 18{,}11\ \text{mA} \qquad (7.7)$$

Equação 7.7: Consumo de corrente para 5-Threads do Cenário 5

Neste ponto apresenta-se na Figura 7.20 o método de *Welch* com o objetivo de realizar uma análise espectral do comportamento energético dos sinais PAME e PEE, a fim de identificar graficamente o ataque de força bruta com dicionário através da ferramenta não especializada *Metasploit* frente ao estado estável avaliado.

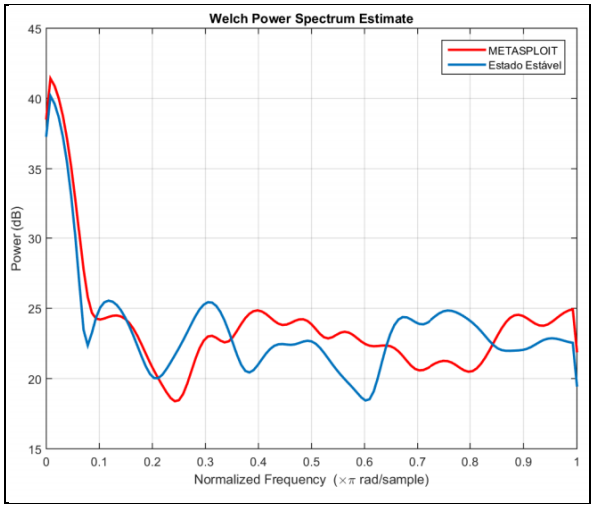


Figura 7.20: Padrão de consumo energético baseado em PAME e PEE

Nesta seção apresenta-se o resultado final alcançado do cenário de ataque por força bruta com dicionário através da ferramenta não especializada *Metasploit*. Observa-se na Figura 7.21 está dividida em 5 figuras, a Figura 7.21 (a) apresenta o padrão (PAME) sem processamento, a Figura 7.21 (b) apresenta o padrão (PAME) normalizado com o método de *Welch*, a Figura 7.21 (c) apresenta o padrão (PEE) sem processamento, a Figura 7.21 (d) apresenta o padrão (PEE) normalizado pelo método de *Welch*, finalmente a Figura 7.21 (e)

apresenta uma combinação dos padrões (PAME) e (PEE) normalizado com o método de *Welch*.

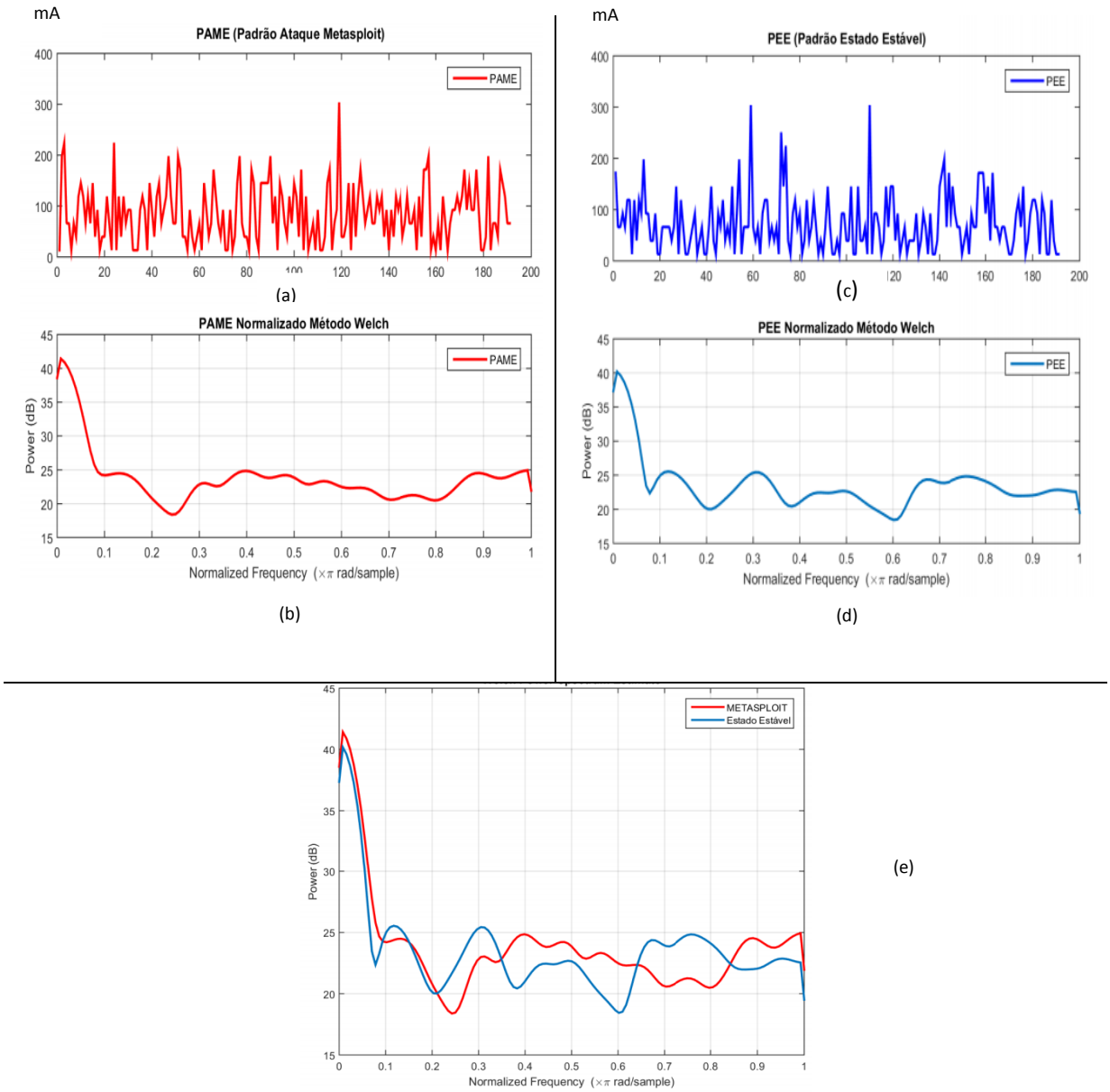


Figura 7.21: Padrões de consumo do cenário PAME

Na Tabela 7.20 apresentam-se as seguintes métricas avaliadas do cenário de ataque por força bruta com dicionário através da ferramenta não especializada *Metasploit*. O tempo total utilizado desde que se inicia o envio do primeiro pacote até a recepção do último pacote é de 20 segundos, seguidamente a quantidade de pacotes transmitidos nesse instante de tempo é de 135 pacotes, seguidamente a quantidade de dados transmitidos possui um valor de 24550 bytes. Finalmente a quantidade de Threads presentes nesta prova é de 5 (cinco), responsável pelo fornecimento de 5 *Socket* de comunicação.

Tabela 7.20: Métricas avaliadas no cenário PAME

Tempo da Prova (segundos)	20
Quantidade de Pacotes Transmitidos	135
Dados Transmitidos (bytes)	24550
Threads	5

Considerando que este cenário de ataque utiliza 5 Threads de execução para realizar o ataque atribuído a vulnerar a senha de acesso para um usuário específico, a continuação se estima a quantidade de pacotes transmitidos, em função dos resultados alcançados, tanto no cenário de acesso permitido e o cenário de acesso negado tratados anteriormente.

Consideraram-se estes 5 Threads, onde 4 Threads estão associados ao cenário de acesso negado, seguidamente 1 Thread é considerado um acesso concedido, a Equação 7.8 apresenta o cálculo proposto.

$$\begin{aligned} \text{Pacotes (Metasploit)} &= \text{Acesso Negado} * 4 \text{ Threads} + \text{Acesso Concedido} \\ \text{Pacotes (Metasploit)} &= (27 * 4) + 45 \\ \text{Pacotes (Metasploit)} &= 153 \\ \text{Resultado alcançado do teste } &135 \end{aligned} \tag{7.8}$$

Equação 7.8: Estimação da quantidade de pacotes para o Cenário 5

A diferença obtida de 18 pacotes no cálculo representa o 14 % mais de pacotes estimados transmitidos. O resultado estimado está baseado dois supostos, o primeiro ponto está fundamentado no tipo de ferramenta não especializada *Metasploit* que utiliza bibliotecas de comunicação genéricas do protocolo *SSH*. O segundo ponto está relacionado à quantidade de pacotes transmitidos em função do consumo de corrente estimado, na Tabela 7.21 está discrepância do consumo de corrente da ferramenta *Metasploit*, apresenta-se mais clara, ou seja que o consumo de 18,11 mA relacionado à quantidade de pacotes transmitidos 135, não está entre os intervalos aceitáveis de consumo de corrente das ferramentas *Medusa* e *Hydra*, que são usadas como o limite inferior e o limite superior, tendo em conta que os intervalos propostos correspondem a valores das ferramentas especializadas.

Este erro detectado através da relação entre pacotes transmitidos e consumo de corrente, é ocasionado pelo sensor de corrente *ACS712ELCTR-05B* da plataforma de medição de consumo energético baseado em *Arduino Uno*.

Portanto podemos dizer que o modelo teórico proposto baseado na análise de pacotes transmitidos apresenta uma taxa de confiabilidade de 86%. Considerando que os 135 pacotes identificados por *Wireshark* correspondem ao 100% dos pacotes coletados.

Tabela 7.21: Compreensão do erro de leitura de *Metasploit*

Métricas	Ataques		
	Especializados		Não Especializados
	Medusa	Hydra	Metasploit
Tempo da Prova (segundos)	7,5	12	20
Quantidade de Pacotes Transmitidos	112	253	135
Dados Transmitidos (bytes)	20892	38382	24550
Threads	4	9	5
Consumo Corrente (mA)	7,04	12,08	18,11

No Quadro 7.16 apresenta-se um modelo teórico de identificação de ameaça ou ataque fundamentado na Equação 7.7 apresentada no Capítulo 6 Subseção 6.8.4, particularizada para este cenário, identificando um ataque com a ferramenta *Metasploit* ao serviço de acesso remoto *SSH* executando-se em um plataforma embarcada *Raspberry Pi 2 Model B*.

Quadro 7.16: Assinatura PAME modificada

ALERT TCP $EXTERNAL_NET ANY -> $HOME_NET 22 (MSG:"ATAQUE METASPLOIT";threshold: type both, track by_dest, count 5, seconds 5 ; 18,11)

Função da regra: Identifica qualquer host que apresente 5 intentos de conexão ao serviço *SSH* através da porta 22 do protocolo *TCP*, em qualquer dispositivo da rede interna, no intervalo de 5 segundos e identifique um consumo de 18,11.

7.5.4 Resumo do Cenário de Ataque com *Metasploit*

Nesta subseção se interpretam os resultados alcançados do cenário 5 Ataque por Força Bruta com Dicionário através da Ferramenta Não Especializada *Metasploit*, logrando-se identificar uma curva padrão característica do consumo de corrente baseado na proposta da Abordagem 1(Pacotes Transmitidos) e Abordagem 2 - Caso especial C.2 apresentada no Capítulo 6 Subseção 6.8.1

O consumo estimado para o PEE é aproximadamente 87,94 mA, frente ao PAME com um valor aproximando de 106,02 mA, a diferença obtida de 18,11 mA, é atribuída à transmissão de pacotes entre o cliente e o servidor totalizando 135 pacotes transmitidos, somando também o custo de processamento do *CPU* utilizado pelo sistema operacional *Linux Raspbian* em tudo o processo de autenticação.

Destaca-se que a relação entre o consumo obtido e a quantidade de pacotes identificados na transmissão, apresenta valores de consumo superiores em comparação aos cenários avaliados anteriormente, o erro é produzido na coleta do Padrão de Estado Estável (PEE) atribuído ao sensor de corrente, especificamente no valor médio de corrente calculada de 87,94 mA.

7.6 Resumo dos Cenários de Provas

Apresenta-se um resumo dos cenários avaliados produto das provas efetuadas.

A continuação apresenta-se um Figura 7.22 com o método de *Welch* aplicado aos diferentes cenários avaliados anteriormente, as curvas correspondem ao Cenário 1 - PAC, Cenário 2 - PAN, Cenário 3 - PAM, Cenário 4 - PAH e Cenário 5 - PAME, a figura apresenta o comportamento de cada tipo de acesso e tipo de ataque, onde claramente se encontram as diferenças de consumo de corrente.

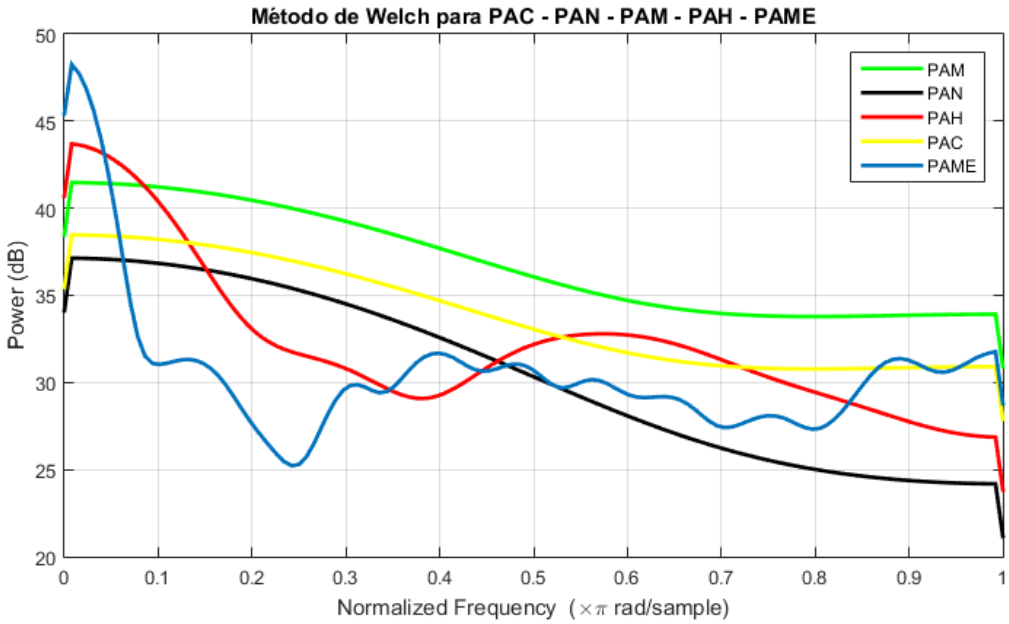


Figura 7.22: Método de *Welch* dos cenários avaliados

As métricas utilizadas para avaliar os conjuntos de cenários propostos estão baseadas no relacionamento que possuem umas de outras, o consumo de corrente por parte da avaliação de um cenário de prova apresenta o consumo meio de corrente na base da quantidade de tempo que dura o teste, a quantidade de bytes enviados e recebidos pelo canal de comunicação e quantidade de pacotes que representa a métrica anterior, para finalizar se apresente o consumo de corrente diretamente relacionado às métricas anteriores, assim estas justificam porque acontece o consumo obtido apresentando uma relação diretamente proporcional entre o consumo e quantidade de pacotes no intervalo de tempo definido da prova.

Neste ponto são apresentados três grupos que formam parte das avaliações propostas. O primeiro grupo corresponde a uma avaliação por tipos de acessos, onde intervierem o cenário 1 Acesso Permitido com usuário e senha válida, frente cenário 2 Acesso Negado com usuário valido e senha não válida. O segundo grupo corresponde a uma avaliação por tipos de ataques

de força bruta com dicionário, onde intervierem o cenário 3 Ataque de Força Bruta com Dicionário através da Ferramenta Especializada *Medusa*, frente ao cenário 4 Ataque de Força Bruta com Dicionário através da Ferramenta Especializada *Hydra*. Por último, o terceiro grupo corresponde a uma avaliação de Ataques por Força Bruta com Dicionário através da comparação das ferramentas especializadas baseada no cenário 4 frente ao cenário 5 Ataque por Força Bruta através da Ferramenta Não Especializada *Metasploit*.

Apresenta-se uma análise do primeiro grupo: Tipos de Acessos.

A Figura 7.23 está formada por 4 Figuras onde realiza-se a comparação do Cenário 1 Acesso Permitido frente ao Cenário 2 Acesso Negado.

A Figura 7.23 (a) apresenta a comparação do tempo de duração dos tipos de acesso avaliados no servidor de acesso remoto *SSH*, onde observa-se o Cenário 1 com um valor de 6 segundos, frente ao Cenário 2 que possui um valor de 7 segundos, a diferença de tempo está baseada no tipo de procedimento utilizado pelos cenários, ou seja o Cenário 1 utiliza o método de autenticação de usuário valido com senha válida, apresentado no Apêndice C Figura C.1, porém o Cenário 2 utiliza o método de autenticação de usuário valido e senha não válida que é atribuído ao Apêndice C Figura C.2, está sutil diferença entre métodos torna-se uma diferença de 1 segundo.

A Figura 7.23 (b) apresenta os Bytes transmitidos no canal de comunicação entre nó cliente e nó servidor *SSH*, observa-se que o Cenário 1 possui um valor de 7611 *Bytes* frente ao Cenário 2 com um valor 5549 *Bytes*, isto acontece porque o Cenário 1 ao realizar tudo o processo de autenticação com usuário e senha válida, o servidor de acesso remoto *SSH* permite o acesso com essa tupla de usuário e senha, posteriormente os comandos executados na sessão são: comando *Touch* que realiza a criação de um arquivo de texto, comando *Date* que apresenta à hora do sistema, por último o comando *Free* que apresenta a memória disponível do sistema operacional, os comandos propostos permitem gerar carga no *CPU* da plataforma de teste *Raspberry Pi 2 Model B* executando o servidor *SSH*, que através da plataforma embarcada de medição de consumo de corrente logra-se mapear.

A Figura 7.23 (c) apresenta os pacotes transmitidos onde o Cenário 1 possui 42 pacotes frente ao Cenário 2 com um valor de 27 pacotes, isto acontece pelo fato que a execução de comandos remotos no servidor *SSH* ocasiona maior transmissão de pacotes, frente ao Cenário 2 que só realiza um intento de acesso não valido.

A Figura 7.23 (d) apresenta o consumo de corrente (mA) dos cenários avaliados, onde o Cenário 1 apresenta um valor de 8,81 mA relacionado ao processo de autenticação com usuário e senha válida, seguido da execução de comandos no servidor *SSH*, porém o Cenário

2 com um valor de 1,76 mA só está relacionado ao processo de autenticação com usuário valido e senha não válida, lembrar que Cenário 1 possui uma quantidade de pacotes superior ao Cenário 2, os fatos apresentados anteriormente justificam o consumo de corrente do Cenário 1 maior ao Cenário 2.

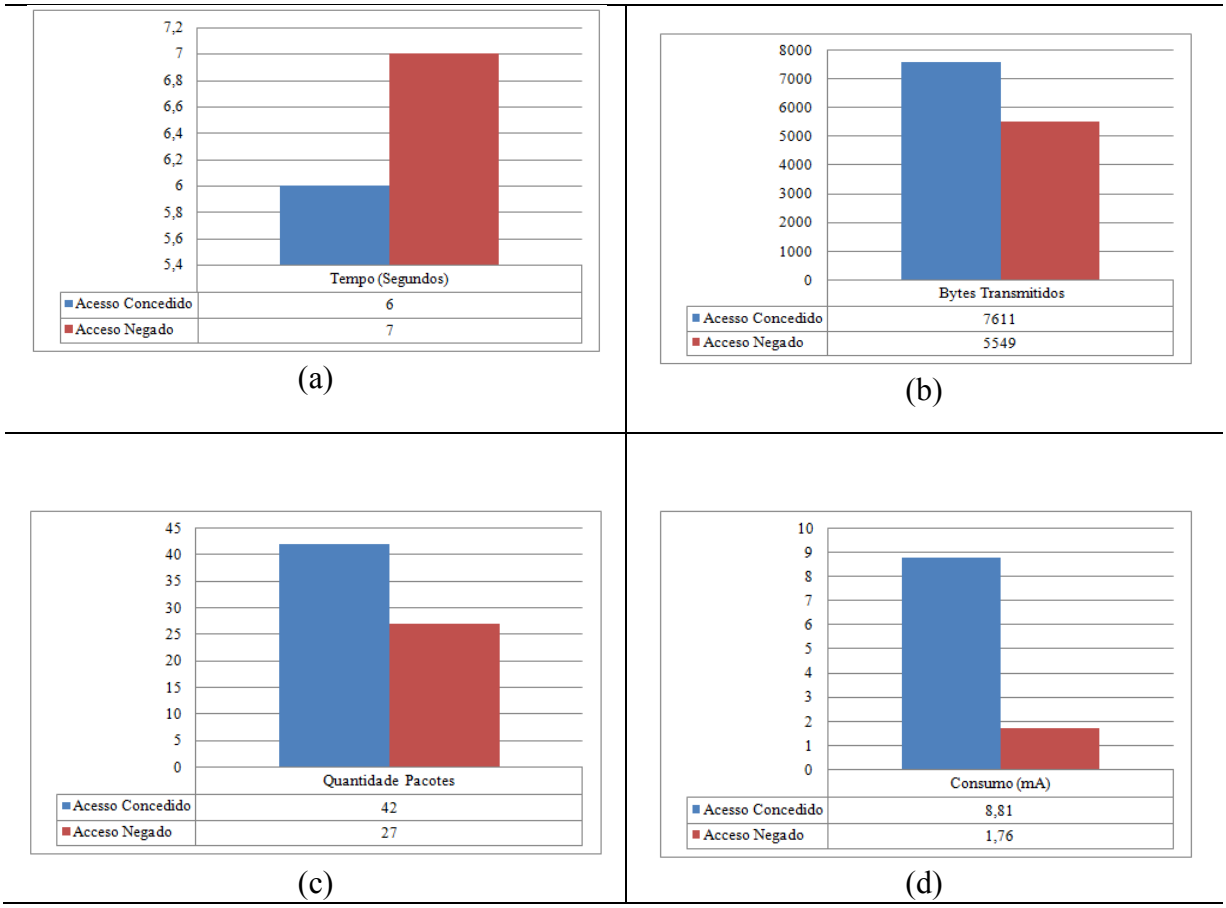


Figura 7.23: Comparação dos cenários de Acesso Permitido e Acesso Negado.

Apresenta-se uma análise do segundo grupo: Avaliação de ferramentas especializadas.

A Figura 7.24 está formada por 5 Figuras onde realiza-se a comparação do Cenário 3 Ataque com *Medusa* frente ao Cenário 4 Ataque com *Hydra*.

A Figura 7.24 (a) apresenta a comparação do tempo de duração dos tipos de acesso avaliados no servidor de acesso remoto *SSH*, onde observa-se o Cenário 3 com um valor de 7,5 segundos, frente ao Cenário 4 que possui um valor de 12 segundos, a diferença de tempo está fundamenta na implementação de bibliotecas especificas que trabalham com o protocolo *SSH*, particularmente Cenário 3 utiliza a biblioteca *Libssh2* que apresenta um melhor desempenho frente ao Cenário 4 que utiliza a biblioteca *Libssh,* no Apêndice B apresenta-se uma comparação das bibliotecas utilizas pelas ferramentas avaliadas. Um resultado a destacar desde a perspectiva de um atacante esta baseado no tempo de *Hacking* obtido pelas

ferramentas avaliadas, onde a ferramenta *Medusa* é 60% mais rápida que a ferramenta *Hydra*, por outro lado, a quantidade de pacotes transmitidos por Cenário 3 apresenta uma redução de 125%, justificando um consumo de corrente em 7,04 mA. Os resultados apresentam que a eleição de uma ferramenta especializada para fornecer um ataque de força bruta com dicionário através de uma ferramenta especializada, é decisiva em vários pontos de análise, os benefícios expostos anteriormente são: o tempo de execução do ataque impacta diretamente na transmissão de pacotes, por conseguinte no consumo do dispositivo atacado.

A Figura 7.24 (b) apresenta os *Bytes* transmitidos no canal de comunicação entre nó cliente e nó servidor *SSH*, observa-se que o Cenário 3 possui um valor de 20892 *Bytes* frente ao Cenário 4 com um valor 38382 *Bytes*, isto acontece porque o Cenário 3 possui 4 Threads de execução frente ao Cenário 4 que possui 9 Threads, pode-se afirma que a maior quantidade de *Thread* de execução, maior quantidade de pacotes transmitidos.

A Figura 7.24 (c) apresenta os pacotes transmitidos onde o Cenário 3 possui 112 pacotes, frente ao Cenário 4 com um valor de 253 pacotes, que está relacionado aos *bytes* transmitidos, ou seja a maior quantidade de *bytes* transmitidos, portanto maior quantidade de pacotes transmitidos.

A Figura 7.24 (d) apresenta o consumo de corrente (mA) dos cenários avaliados, onde o Cenário 3 apresenta um valor de 7,04 mA de consumo de corrente frente ao Cenário 4 com um valor de 12,08 mA, a diferença entre o consumo dos cenários é baseado nas bibliotecas utilizadas em cada ferramenta avaliada que impactam diretamente nos Threads de execução fornecidos, portanto estão relacionados aos pacotes transmitidos, e está última métrica impacta no consumo de corrente da ferramenta. Conclui-se que a ferramenta *Medusa* utiliza um tempo de 7,5 segundos em comparar todas as combinações possíveis de acessos com a utilização de 4 Threads de execução frente à ferramenta *Hydra* que utiliza 12 segundos com 9 Threads de execução para realizar a mesma operação.

A Figura 7.24 (e) apresenta a quantidade de Threads de execução do Cenário 3 com 4 Threads frente ao Cenário 4 com 9 Threads de execução.

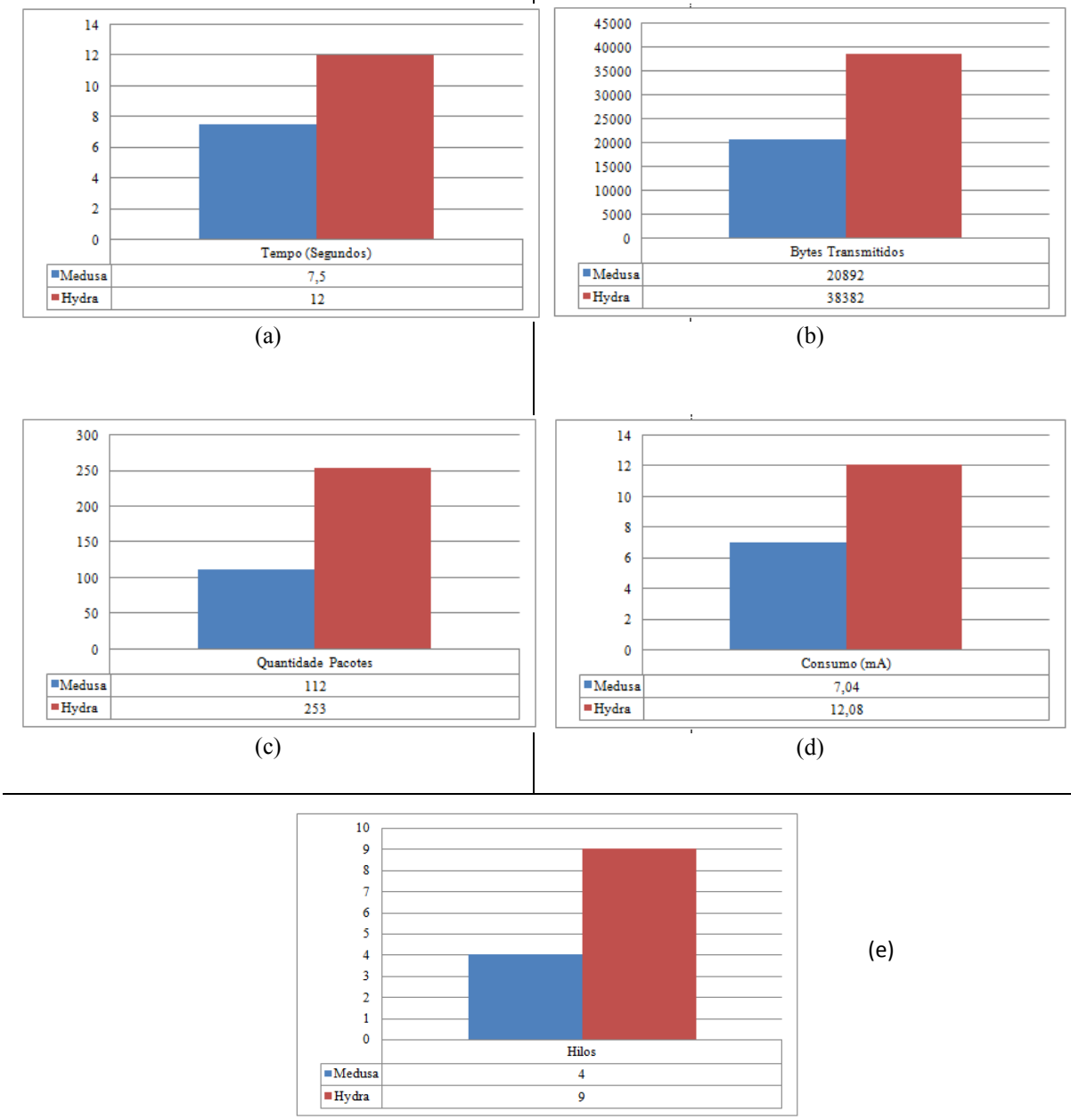


Figura 7.24: Comparação dos cenário 3 frente ao cenário 4

Na Tabela 7.22 apresenta-se um resumo de frequências da quantidade de pacotes transmitidos associados aos Threads de execução da ferramenta *Medusa*. A quantidade de *Thread* é interpretada pela métrica denominada frequência que descreve as ocorrências dos pacotes transmitidos, ou seja o valor de 27 pacotes transmitidos possui uma frequência igual 1, isto significa que só um hilo está associado a essa quantidade de pacotes, porém o valor de 20 pacotes possui uma frequência igual a 2 este valor é a quantidade de Threads que utilizam essa quantidade de pacotes. A implementação de Threads de execução está relacionada diretamente a uma questão de desenho da ferramenta e particularmente à biblioteca *libssh2*

que implementa o protocolo *SSH*. Na Figura 6.25 apresenta-se o diagrama de frequências da ferramenta *Medusa*.

Tabela 7.22: Resumo de frequências da ferramenta especializada *Medusa*

Quantidade de pacotes	Frequência	Porcentual
27	1	25,0
28	2	50,0
29	1	25,0
Total	4	100,0

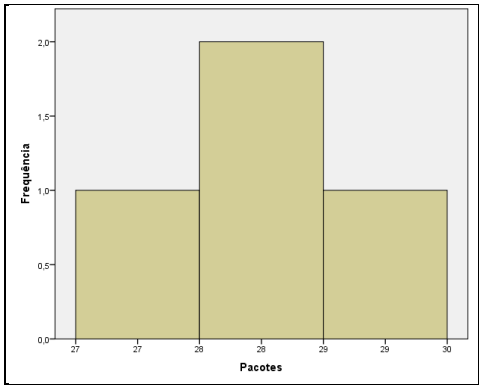


Figura 7.25: Diagrama de frequências da ferramenta especializada *Medusa*

Na Tabela 7.23 apresenta-se um resumo de frequências da quantidade de pacotes transmitidos associados aos Threads de execução da ferramenta *Hydra*. A quantidade de *Thread* é interpretada pela métrica denominada frequência que descreve as ocorrências dos pacotes transmitidos, ou seja o valor de 27 pacotes transmitidos possui uma frequência igual 3, isto significa que 3 Threads estão associados a essa quantidade de pacotes, porém o valor de 29 pacotes possui uma frequência igual a 4, este valor é a quantidade de Threads que utilizam essa quantidade de pacotes. A implementação de Threads de execução está relacionada diretamente a uma questão de desenho da ferramenta e particularmente à biblioteca *libssh* que implementa o protocolo *SSH*. Na Figura 6.26 apresenta-se a figura de frequências da ferramenta *Hydra*.

Tabela 7.23: Resumo de frequências da ferramenta especializada *Hydra*

Quantidade de pacotes	Frequência	Porcentual
26	1	11,1
27	3	33,3
29	4	44,4
30	1	11,1
Total	9	100,0

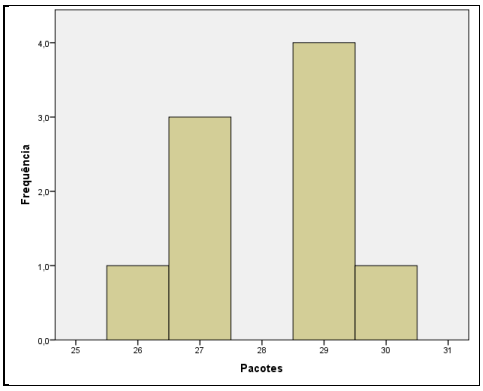


Figura 7.26: Diagrama de frequências da ferramenta especializada *Hydra*

Apresenta-se uma análise do terceiro grupo: Comparação de ferramentas especializadas frente a ferramentas não especializadas.

A Figura 7.27 está formada por 4 Figuras onde realiza-se a comparação do Cenário 4 Ataque através da ferramenta especializada *Hydra* frente ao Cenário 5 Ataque através da ferramenta não especializada *Metasploit*. O fato de realizar está comparação é compreender os fatores técnicos na hora de selecionar uma ferramenta especializada frente a uma ferramenta não especializada avaliada em um tipo de cenário específico a vulnerar.

A Figura 7.27 (a) apresenta a comparação do tempo utilizado pelas ferramentas para realizar um ataque bem sucedido que vulnere o serviço *SSH*, a ferramenta especializada *Hydra* que possui um valor 12 segundos frente à ferramenta não especializada *Metasploit* que possui um valor de 20 segundos, observa-se uma diferença de tempo de 8 segundos, está diferença é atribuída a uma questão da arquitetura de desenho das ferramentas, ou seja a ferramenta especializada *Hydra* utiliza uma biblioteca especializada denominada *libssh* que aperfeiçoa o ataque ao serviço *SSH*, frente à ferramenta não especializada *Metasploit* que utiliza uma biblioteca genérica para fornecer o ataque, está implementação é penalizada no tempo de ataque.

A Figura 7.27 (b) apresenta os *Bytes* transmitidos no canal de comunicação entre nó cliente e nó servidor *SSH*, observa-se que o Cenário 4 possui um valor de 38382 *Bytes* frente ao Cenário 5 com um valor 24550 *Bytes*, isto acontece porque o Cenário 4 possui 9 Threads de execução frente ao Cenário 5 que possui 5 Threads, pode-se afirma que a maior quantidade de Threads de execução, maior quantidade de pacotes transmitidos.

A Figura 7.27 (c) apresenta os pacotes transmitidos no contexto do ataque avaliado, os valores para a ferramenta especializada *Hydra* são 253 pacotes transmitidos, frente à ferramenta não especializada *Metasploit* com um valor de 135 pacotes.

A Figura 7.27 (d) apresenta o consumo de corrente (mA) das ferramentas avaliadas, onde a ferramenta especializada *Hydra* apresenta um valor de 12,08 mA frente à ferramenta

não especializada *Metasploit* com um valor de 18,11 mA, neste caso, observa-se que não existe uma relação entre a quantidade de pacotes transmitidos e o consumo gerado, o valor obtido da ferramenta não especializada *Metasploit* apresenta uma taxa de 150% mais de consumo que a ferramenta *Hydra*, considerando o valor de *Hydra* 12,08 mA o 100%.

Estes resultados evidenciam que existe um erro associado ao consumo de corrente por parte da ferramenta *Metasploit*. Lembrando que os pacotes transmitidos são garantidos pela ferramenta de análise de pacotes *Wireshark*. Assim, por analogia pode-se afirmar que o erro produzido está associado diretamente à plataforma de medição de consumo energético baseado em *Arduino Uno*, especificamente ao sensor *ACS712ELCTR-05B* utilizado na coleta de amostras do sinal de corrente.

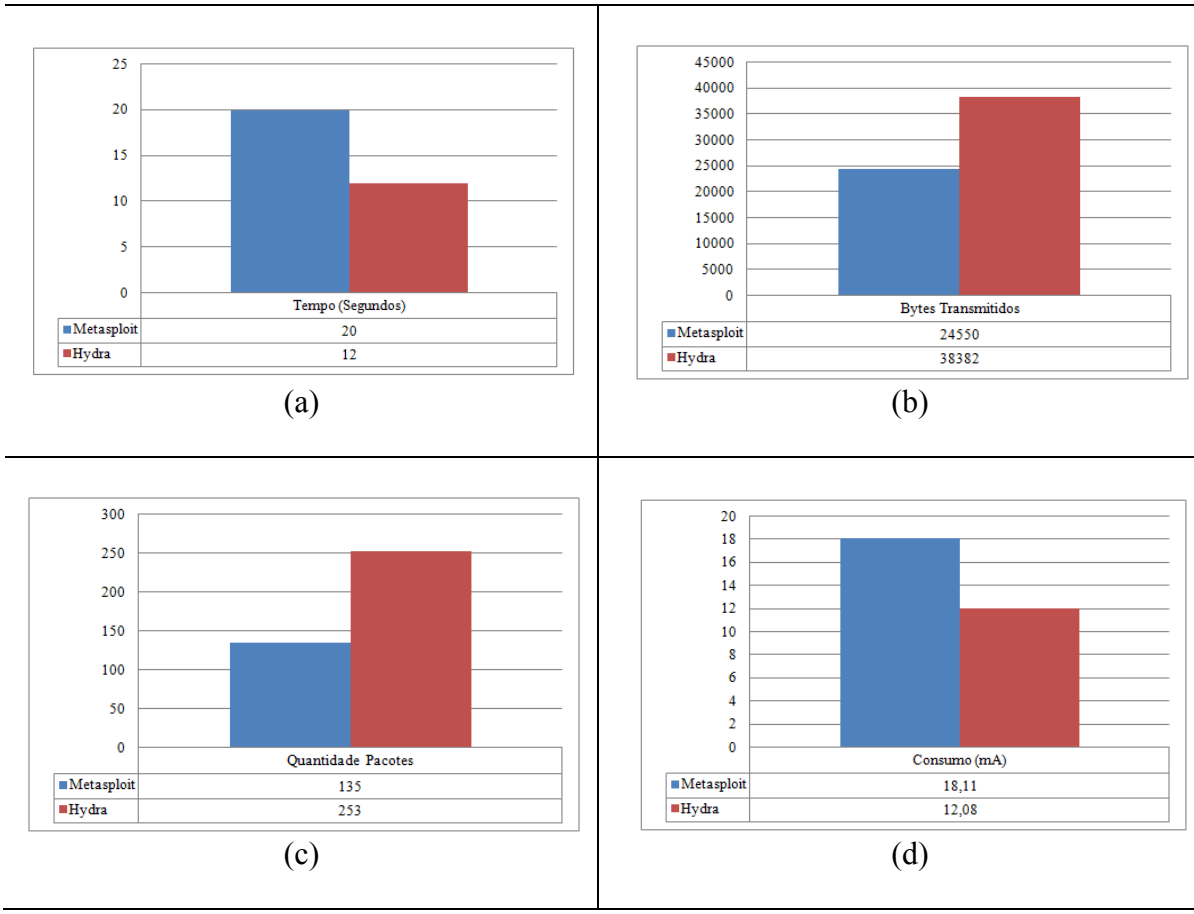


Figura 7.27: Comparação dos Cenário 4 frente ao Cenário 5

7.7 Ameaças dos Cenários Avaliados

A continuação apresenta-se um resumo das ameaças detectadas na avaliação dos cenários propostos neste Capítulo 7.

Em termos gerais, o Cenários 1, Cenários 2, Cenários 3, Cenários 4 não se identificam nenhum tipo de problema que interfira com a avaliações propostas. Sem embargo no Cenário

5 baseado na avaliação do Ataque por Força Bruta com Dicionário através da Ferramenta Não Especializada *Metasploit*, identifica-se um erro na coleta de dados, ocasionado pelo sensor de corrente *ACS712ELCTR-05B* nas amostras coletadas, particularmente o erro é identificado no sinal Padrão de Estado Estável (PEE). O sensor de corrente coleta as amostras no mesmo intervalo de tempo que é realizado o ataque, no caso particular do Cenário 5 o tempo de duração é de aproximadamente 20 segundos, portanto nesse período de tempo realiza-se a coleta, considerando que a plataforma de teste executando o serviço de acesso remoto *SSH* está funcionando, sem realizar nenhum tipo de atividade.

Observa-se que o valor médio da corrente do Padrão de Estado Estável (PEE) é de 87,94 mA, muito baixo com respeito ao valor médio de corrente do Padrão de Ataque *Metasploit* (PAME) 106,02 mA, considerando que a leitura anterior é a correta.

Neste caso particular, o erro é identificado com maior facilidade pelo fato que o tempo de amostragem se incrementa e as leituras oscilam de maneira significativa, a isto somando o erro de leitura próprio do sensor que aproximadamente está entre o 7 e 10 %, segundo o fornecedor, obtemos os resultados apresentados.

Se constata que erro estimado do sensor na prática, está no intervalo do 15 ao 17 %.

Conclui-se que o Cenário 5 apresenta um erro de medição na leitura dos valores de corrente, que particularmente pelo intervalo de miliampères que são necessários para identificar um evento em particular no embarcado *Raspberry Pi 2*, e no tempo de amostragem que é considerado no Cenário 5 de 20 segundos aproximadamente, os resultados não são satisfatórios desde o ponto do consumo médio calculado, porém o resultados proporcionam dois aspectos importantes a destacar, o primeiro é baseado no limite de tempo do sensor de corrente *ACS712ELCTR-05B* para realizar a amostragem, esclarece-seque no intervalo de 1 a 10 segundos o dispositivo apresenta resultados favoráveis. O segundo ponto está baseado no custo associado do dispositivo que aproximadamente possui um custo no mercado de R$15 reais. No final ao tratar-se de dados experimentais e tendo em conta que de 5 cenários avaliados apenas o Cenário 5 apresento falha, recomenda-se utilizar este dispositivo para outros projetos similares.

7.8 Considerações Finais do Capítulo

Neste capítulo se apresento uma avaliação dos cenários de tipos de acessos e tipos de ataques de força bruta com dicionário através de ferramentas especializadas como *Medusa* e *Hydra*, e ferramentas não especializada como no caso de *Metasploit*.

Em todos os cenários avaliados se propõe uma curva padrão de consumo de corrente que identifica as respostas do sistema operacional no contexto energético, ou seja, se interpreta a resposta do sistema operacional *Linux Raspbian* executando-se em uma plataforma embarcada *Raspberry Pi 2 Model B* de baixo custo trabalhando como um servidor de acesso remoto *SSH*, onde as respostas geradas pela plataforma são interpretadas em um contexto energético regrado pelo consumo de corrente (mA), de tal forma, logra-se identificar um padrão de consumo de corrente associado aos eventos gerados pelo servidor de acesso remoto *SSH*.

O objetivo de fornecer curvas padrões de consumo de corrente através da avaliação de uma plataforma embarcada *Raspberry Pi 2 Model B* que trabalha como um servidor de acesso remoto *SSH*, avaliado com diferentes tipos de cenários de provas, foi cumprido exitosamente.

A seguir as recomendações para melhorar os cenários de provas propostos.

Utilizar um dicionário de senhas com um maior número de combinações, tanto de usuários como de senhas a avaliar.

Utilizar um cenário de real de prova, ou seja executar a plataforma de teste *Raspberry Pi 2 Model B* no endereço *IP* distinto da rede hospedeira, com o alvo de registrar as métricas de retardo do tempo, pacotes transmitidos, e taxa de perdas de pacotes, métricas consideradas em cenários de avaliação reais.

CAPÍTULO 8

CONCLUSÕES

8.1 Análise Crítica dos Resultados

A partir dos resultados obtidos nos experimentos apresentados no Capítulo 7, foi possível oferecer uma resposta ao problema proposto no Capítulo 1, ou seja, verificou-se que através do uso de técnicas de canais laterais é possível fornecer curvas padrões de consumo de corrente associado a uma plataforma embarcada de baixo custo *Raspberry Pi 2 Model B* executando como um servidor de acesso remoto *SSH* avaliado em diferentes cenários de provas. O cenários avaliados são:

- Cenário 1: Tipo de Acesso: Acesso Permitido;
- Cenário 2: Tipo de Acesso: Acesso Negado;
- Cenário 3: Ataque de Força Bruta com Dicionário Através da Ferramenta especializada *Medusa*;
- Cenário 4: Ataque por Força Bruta com Dicionário Através da Ferramenta especializada *Hydra*;
- Cenário 5: Ataque por Força Bruta com Dicionário Através da Ferramenta não especializada *Metasploit*.

Os eventos energéticos gerados pela plataforma de teste *Raspberry Pi 2 Model B*, em resposta aos cenários de provas propostos são coletados por uma plataforma de medição de consumo de corrente baseado na plataforma *Open hardware* e *Open software Arduino Uno,* está plataforma de medição, possui a capacidade de interpretar os sinais do sensor de corrente denominado *ACS712ELCTR-05B* que é baseado no efeito *Hall*, portanto este sinal é mapeado e interpretado pela plataforma *Arduino Uno*.

Aos efeitos de melhorar a interpretação visual do sinal obtido, procede-se a realizar um pós-processamento ou normalização através do método de *Welch*, que proporciona

informação sob a distribuição da potência do sinal nas distintas frequências que está composta, só depois dessa operação efetuada, obtemos uma curva padrão de consumo de corrente. O processamento e pós-processamento dos dados coletados pela plataforma de medição de consumo energético *baseado em Arduino* é feito através da ferramenta *Matlab*.

Apresenta-se um modelo de consumo teórico de corrente que estima o consumo de corrente baseado na Abordagem 1 (Pacotes Transmitidos) e Abordagem 2 (Intrusões Executadas no Sistema Operacional), este último com casos especiais explicados no Apêndice C. Por último, apresenta-se um modelo que fornece as regras padrões que identificam um tipo de acesso ou um tipo de ataque, esta regra ou assinatura é utilizado pelo *IDS Snort* para identificar ameaças e/ou ataques através do método de detecção por padrões.

Os resultados alcançados apresentam as curvas padrões que identificam os eventos associados aos cenários de provas propostos, ou seja o reconhecimento através de um padrão de consumo de corrente que identifica um cenário de acesso concedido, um acesso negado, um ataque por força bruta com dicionários através de ferramentas especializadas, como *Medusa* e *Hydra*, até com ferramentas não especializadas como *Metasploit*.

8.2 Trabalhos Futuros

A seguir são apresentadas as sugestões para trabalhos futuros que podem ser realizados a partir deste trabalho de dissertação.

A plataforma de medição de consumo energético baseado em *Arduino Uno* utiliza um sensor de corrente baseado no efeito *Hall* denominado *ACS712ELCTR-05B* obtendo resultados favoráveis, porém recomendamos utilizar o seguinte sensor unidirecional de corrente denominado *ACS715ELCTR-20A-T*.

Apresenta-se uma plataforma especializada de medição de parâmetros elétricos que permite obter as métricas de tensão, corrente e potência, os dispositivos propostos são:

1. Dispositivo de medição digital avançado de potencia para coletar medições de consumo de energia baseado no dispositivo *Power HiTESTE 3334* [BECKER et al. 2003]
2. Analisador de potência modelo *N6705B* [BECKER et al. 2003] [ALONSO et al. 2012].
3. Analisador de consumo de energia digital baseado no dispositivo de aquisição de dados da empresa *National Instrument* modelo *NI USB-6218* que possui como características 32 entradas analógicas, um conversor analógico digital de 16 *Bits* de precisão, e uma capacidade de amostragem de 250.000 amostras por segundo,

este dispositivo através de seus respectivos *Drivers*, permite mapear os processos com o consumo energético em tempo de execução [ALONSO et al. 2012].

A continuação pode-se desenvolver um sistema de medição de consumo energético utilizando um *hardware FPGA* [BARRACHINA et al. 2013] [MOLKER et al. 2009].

A continuação um analisador de protocolo que trabalhe na camada física do modelo *OSI* através de um *hardware NetFPGA* [ZHANG et al. 2010] interpretando os sinais elétricos.

A continuação os cenários de desenvolvimento de *software* propostos

1. Desenvolver um módulo que integre as regras ou assinaturas padrões do consumo de corrente no arquivo (*rules*) onde então definidas as regras padrões de ataques e/ou ameaças do *IDS Snort*.
2. Aplicar técnicas de *Data Minig* e algoritmos de aprendizagem de máquina, aos dados coletados, a fim detectar relações entre as métricas propostas e propor um modelo de identificação de ameaças baseado em modelos preditivos [MARYAM et al. 2014].
3. Interpretar as curvas padrões de consumo de corrente através da utilização da transformada de *Wavelet* e comparar os resultados com o método de *Welch*

8.3 Considerações Finais

Nesta dissertação foi proposto descobrir uma curva padrão de consumo de corrente baseado na identificação de diferentes tipos de acessos e diferentes tipos de ataques de força bruta com dicionários através de ferramentas especializadas e não especializadas, ao serviço de acesso remoto *SSH*, executando em uma plataforma de teste embarcada *Open hardware* e *Open software* de baixo custo *Raspberry Pi 2 Model B*.

A justificação teórica dos padrões está fundamentada nas técnicas de identificação de ataques por canais laterais, metodologia empregada para vulnerar uma senha de cifra de um dispositivo que criptografa mensagens através do mapeamento do consumo de potência.

O consumo de corrente gerado pela plataforma de teste *Raspberry Pi 2 Model B* executando o serviço de acesso remoto *SSH*, avaliado com diferentes cenários de provas, apresenta uma resposta interpretada no contexto energético que logra identificar, o sinal processado através do método de *Welch*, a fim de obter um sinal interpretável graficamente, só a partir desse momento é possível definir um padrão que intérprete uma resposta atribuída a um evento específico.

Neste ponto da pesquisa identificou-se um sinal, que apresenta o consumo de corrente em função do tempo, no entanto, este resultado parcial não é o esperado. Por tanto, procede-

sea aplicar uma função matemática que apresenta a distribuição da potência nas distintas frequências da que está formada o sinal, com isto logramos obter uma curva padrão de acesso de um ataque.

Observa-se na especificação do fabricante que o erro introduzido pelo sensor de corrente na medição está entre o 7 ao 10%. Os dados recolhidos neste trabalho estimam que o erro calculado esteja entre um 15 ao 17%, estes resultados são apresentados através do cenário de ataque com força bruta através da ferramenta não especializada *Metasploit*. Conclui-se que os valores alcançados para o Cenário 1, Cenário 2, Cenário 3 e Cenário 4 são aceitáveis pelo fato que através da ferramenta *Wireshark* se mapeiam as quantidades de pacotes transmitidos pelos cenários, métrica que garante os dados transmitidos, e ademais é alvo de comparação com outras métricas avaliadas.

Sem lugar a duvidas que melhorando a capacidade de medição do sensor o trabalho pode apresentar uma melhora substancial em os resultados alcançados, porém nos gráficos propostos.

Apêndice A: Fundamento dos Protocolos *TCP* e *IP*

Na Figura A.1 apresenta-se o processo de encapsulado.

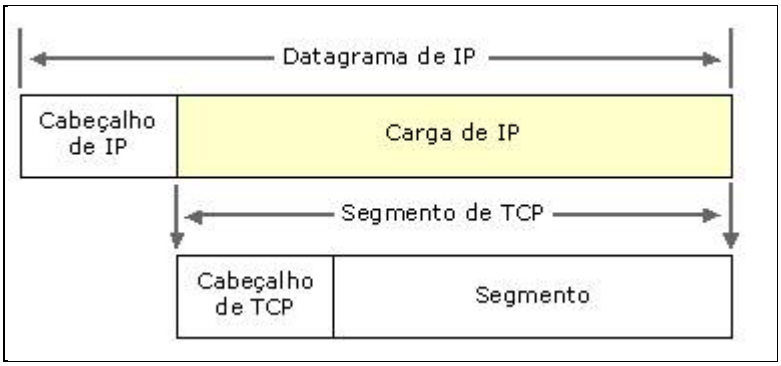


Figura A.1: Processo de encapsulado

Na Figura A.2 apresenta-se o detalhe de um segmento *TCP*.

Porta Origem (16 bits)			**Porta Destino (16 bits)**
Número de Ordem (32 bits)			
Número de aviso de recepção (32 bits)			
Defasagem dos dados (4 bits)	**Reservada (6 bits)**	**As bandeiras (Flags) (9x1 bit)**	**Tamanho Janela (16 bits)**
Soma de controle (16 bits)		**Ponteiro de emergência (16 bits)**	
Opções (Dimensão variável)			**Preenchimento variável múltiplo (32 bits)**

Figura A.2: Segmento *TCP*

Significado dos diferentes campos:

Porta Origem (16 bits): Porta relativa à aplicação corrente na máquina origem.

Porta de Destino (16 bits): Porta relativa à aplicação corrente na máquina de destino

Número de ordem (32 bits): Quando a bandeira *SYN* é 0, o número de ordem é o da,

primeira palavra do segmento corrente, Quando *SYN* é 1, o número de ordem é igual ao número de ordem inicial utilizado para sincronizar os números de sequência (ISN).

Número de aviso de recepção (32 bits): O número de aviso de recepção, igualmente chamado número de pagamento, corresponde ao número (de ordem) do próximo segmento esperado, e não o número do último segmento recebido.

Defasagem dos dados (4 bits): localiza o início dos dados no pacote. A defasagem é essencial aqui porque o campo de opções é de dimensão variável

Reservada(6 bits): Campo inutilizado atualmente, mas previsto para o futuro

As bandeiras (*Flags*) (9x1 bit): representam informações suplementares:

- **NS:** Antecipa o controle de congestão.
- **CWR**: Ativa controle de congestão.

- **ESE**: Alerta de congestões.
- **URG**: se esta bandeira estever em 1 o pacote deve ser tratado urgentemente.
- **ACK**: se esta bandeira estiver em 1 o pacote é um aviso de recepção.
- **PSH** (*PUSH*): se esta bandeira estiver em 1, o pacote funciona de acordo com o método PUSH.
- **RST**: se esta bandeira estiver em 1, a conexão é reiniciada.
- **SYN**: A Bandeira *TCP SYN* indica um pedido de estabelecimento de conexão.
- **FIM**: se esta bandeira estiver em 1, a conexão é interrompida.

Tamanho Janela (16 bits): Campo permitindo conhecer o número de bytes que o receptor quer receber, sem aviso de recepção.

Soma de controle: (Checksum ou *CRC*): A soma de controle é realizada fazendo a soma dos campos de dados do cabeçalho, para poder verificar a integridade do cabeçalho.

Ponteiro de emergência (16 bits): Indica o número de ordem a partir do qual a informação se torna urgente.

Opções (Dimensão variável): Opções diversas

Preenchimento: Preenche-se o espaço que fica após as opções com zeros, para ter um comprimento múltiplo de 32 bits.

Na Figura A.3 apresenta-se o detalhe dos distintos campos que possui um pacote *IP*.

Versão (4 bits)	**Tamanho do cabeçalho (4 bits)**	**Tipo de Serviço (8 bits)**	**Comprimento do Pacote (16 bits)**
Identificador	**Flags (3 bits)**		**Offset (13 bits)**
Tempo de vida (8 bits)	**Protocolo**		***Checksum***
Endereço Origem			
Endereço Destino			
Opções			

Figura A.3: Pacote *IP*

Significado dos diferentes campos:

Versão (4 bits): Identifica tipo de cabeçalho, *IPv4*.

Tamanho do cabeçalho (4 bits): Cumprimento do cabeçalho de internet.

Tipo de Serviço (8 bits) (***ToS***): Prioridade do pacote e tipo de serviço.

Comprimento do Pacote (16 bits): Apresenta o tamanho do datagrama.

Identificador (16 bits): Número de identificação do pacote, no caso que o pacote se divida em tamanhos menores.

Flags (3 bits): Campo de controle de fragmentação.

Offset (13 bits): Permite identificar um fragmento em particular.

Tempo de vida (8 bits) (***TTL***): Limita a persistência do pacote na rede.

Protocolo: Números de protocolo, codificação do protocolo a nível de transporte (IANA - *Internet Assigned Numbers Authority*).

Checksum: Campo de controle de integridade.

Endereço Origem e Destino: Endereços *IP*.

Opções: Outras informações.

Apêndice B: Comparação das Bibliotecas *Libssh* frente a *Libssh2*

Detalhe	**Medusa**	**Hydra**
	Libssh2	**Libssh**
Licença	BSD	LGPL
Lado do servidor	Não	Sim
GSSAPI Interface de programação de aplicação	Não	Sim
Intercâmbio de chave de curva elétrica	Não	Sim
Eliptic Curve Hostkeys	Não	Sim
Teste automatizados	Não tem teste disponível	Sim
Estabilidade da API	Sim	Sim
Compatibilidade de C	C89	C99
Restrições do namespace	Sim	Sim
Documentação de todas as funções	Sim	Não
Doxygen Documentação para todas as funções	Não	Sim
Documentação	Não	Sim
Suporte protocolo SSHv1	Não	Sim
Ferramenta de fornecimento	Autotools e CMake	CMake

Tabela C.1: Comparação das bibliotecas *Libssh* frente *Libssh2*

Apêndice C: Métodos de Autenticação

Na Figura C.1 apresenta-se um diagrama de sequência do método de autenticação do serviço *SSH*, no processo de acesso concedido com nome de usuário valido e senha válida. A seguir, as distintas operações realizadas entre o cliente e o servidor remoto *SSH*. A primeira (1) operação de solicitação de acesso, é realizada pelo cliente e recebida pelo servidor. A segunda (2) operação o servidor solicita, a busca e comparação do usuário na base de dados do sistema onde estão armazenados os usuários, se o usuário está presente, se gera a resposta (3) indicando que o usuário é valido. A quarta (4) operação o servidor solicita o cifra da senha, através do uso do algoritmo *SHA-512*, que gera a quinta (5) resposta, devolvendo a senha cifrada ao servidor. A sexta (6) operação o servidor solicita a busca e comparação, da senha cifrada, na base de dados do sistema, se a senha cifrada anteriormente é igual, à senha cifrada armazenada na base de dados do sistema operacional, se envia a resposta sétima (7) ao servidor *SSH*, de senha válida. Por último, na oitava (8) operação o servidor *SSH*, responde ao cliente outorgando o acesso ao sistema, com a combinação de nome de usuário valido e senha válida.

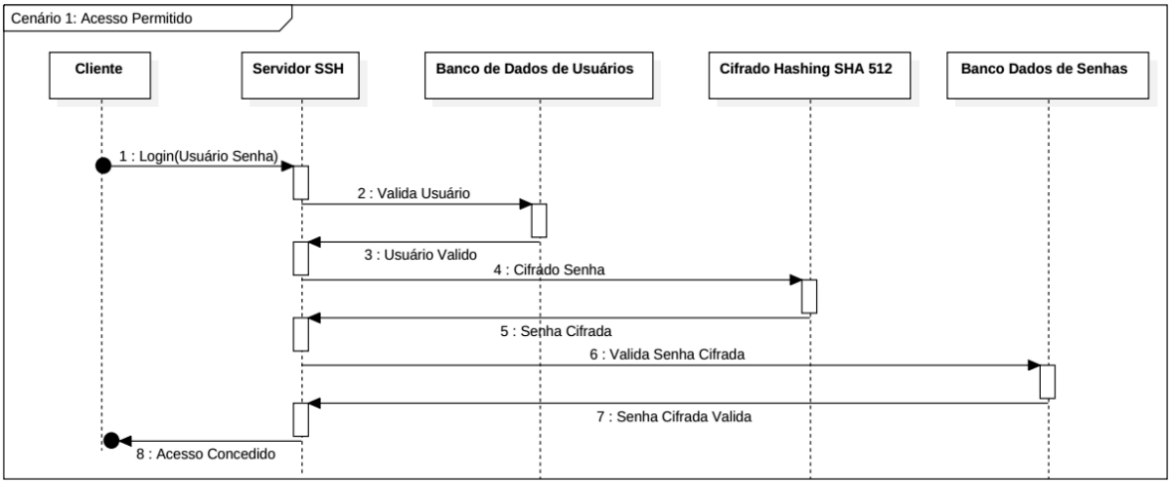


Figura C.1: Diagrama de sequência do método de autenticação de *SSH*

Na Figura C.2 apresenta-se um diagrama de sequência do método de autenticação do serviço *SSH*, no processo de acesso negado com nome de usuário valido e senha não válida. Consideram-se as sequências de operações efetuadas até a sexta (6) operação, idêntica ao diagrama de sequência C.1 apresentado anteriormente. A sexta (6) operação, o servidor solicita, a busca e comparação da senha cifrada, na base de dados de senhas cifras do sistema operacional, se a senha avaliada é não válida, se envia a resposta sétima (7) ao servidor *SSH*, indicando que a senha não é válida. Por último, na oitava (8) operação o servidor *SSH*,

responde ao cliente negando o acesso com as credencias avaliadas, sem notificar do erro acontecido.

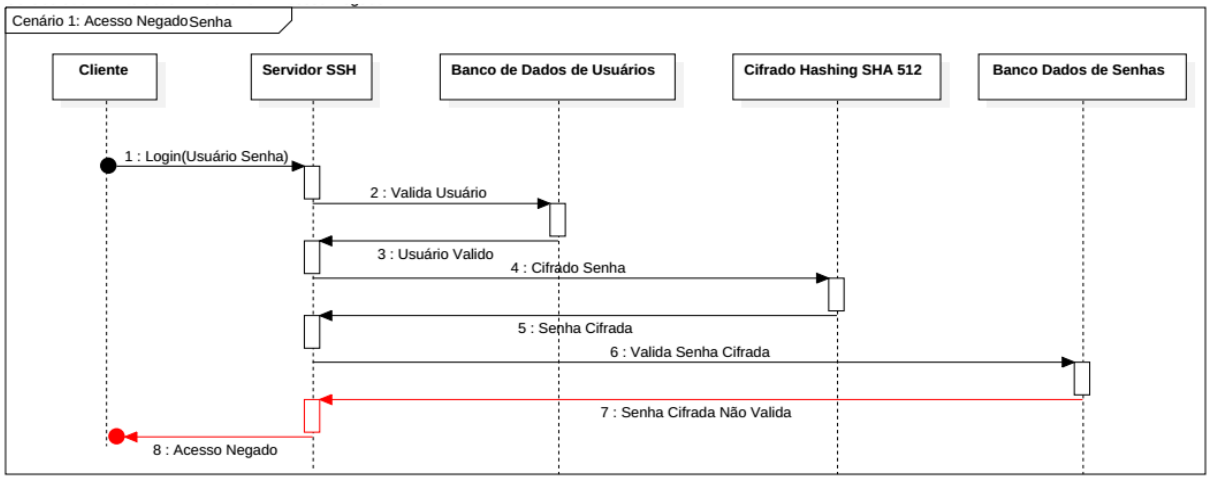


Figura C.2: Diagrama de sequência do método de autenticação com senha não válida

Na Figura C.3 apresenta-se um diagrama de sequência do método de autenticação do serviço *SSH*, no processo de acesso negado com nome de usuário não valido e senha válida. Consideram-se as sequência de operações efetuadas até a segunda (2) operação, idêntica ao diagrama de sequência C.1 apresentado anteriormente. A segunda (2) operação, o servidor solicita, a busca do usuário na base de dados do sistema, se o usuário não está presente, se gera a terceira (3) resposta, que é enviada ao servidor *SSH*, indicando que o usuário não é valido. Por último, na quarta (4) operação, o servidor *SSH*, responde ao cliente negando o acesso com as credencias avaliadas, sem notificar do erro acontecido.

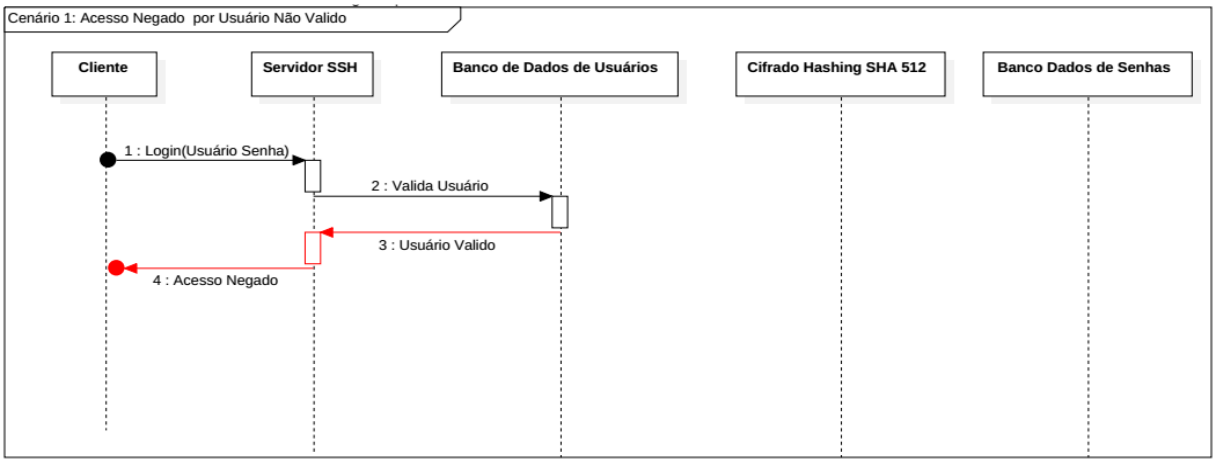


Figura C.3: Diagrama de sequência do método de autenticação com usuário não valido

Apêndice D: RFC Citados

AES	**RFC3962**	https://tools.ietf.org/html/rfc3962
FTP	**RFC-765**	https://tools.ietf.org/html/rfc765
HTTP	**RFC-2616**	http://www.rfc-base.org/txt/rfc-2616
ICMP	**RFC-792**	https://www.ietf.org/rfc/rfc0792
IP	**RFC-791**	https://www.ietf.org/rfc/rfc0791
MD5	**RFC1321**	https://www.ietf.org/rfc/rfc1321
RC4	**RFC7465**	https://tools.ietf.org/html/rfc7465
RPC	**RFC-1057/1831**	https://www.ietf.org/rfc/rfc1057
RSA	**RFC2437**	https://www.ietf.org/rfc/rfc2437
PGP	**RFC3156**	https://tools.ietf.org/html/rfc3156
	RFC4880	https://tools.ietf.org/html/rfc4880
SIP	**RFC-3261**	http://www.rfc-base.org/rfc-3261.html
SHA-1	**RFC3174**	https://tools.ietf.org/html/rfc3174
SSH	**RFC-4250/4251/4252/4253/4254**	https://datatracker.ietf.org/wg/secsh/documents/
TELNET	**RFC-854**	https://tools.ietf.org/html/rfc854
VNC	**RFC-6143**	https://tools.ietf.org/rfc/rfc6143
3DES	**RFC1851**	https://tools.ietf.org/html/rfc1851

REFERÊNCIAS

[ABRAHAMSSON et al. 2013] ABRAHAMSSON, P.; HELMER, S.; PHAPHOOM, N. et al. Affordable and Energy-Efficient Cloud Computing Clusters: The Bolzano Raspberry Pi Cloud Cluster Experiment. Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference, p170-175, 2013.

[ALBIN et al. 2012] ALBIN, E.; ROWE, N. A Realistic Experimental Comparison of the Suricata and Snort Intrusion-Detection Systems. 26th International Conference on Advanced Information Networking and Applications Workshops, Monterey, California, United States, 2012.

[ALONSO et al. 2012] ALONSO, P.; BADIA, R.; LABARTA, J.; BARREDA, M.; DOLZ, M.; MAYO, R.; QUINTANA-ORTI.; REYES, R. Tools for power and energy analysis of parallel scientific applications. In Proc. of 41st International Conference on Parallel Processing–ICPP, pp. 420– 429, Sep. 2012.

[AMATO 2001] AMATO, V. Cisco Networking Academy Program: First-Year Companion Guide. 1ª Ed, 2001.

[ANBYA et al. 2012] ANBYA, M.; SALEHUDDIN, M.; HADISUPADMO, S.; LEKSONO E. Wireless Sensor Network for Single Phase Electricity Monitoring System via Zigbee Protocol Control, Systems & Industrial Informatics (ICCSII), 2012 IEEE Conference on, p261-266, 2012.

[ANDERSON et al. 2003] ANDERSON, D.; SWEENEY, D.; WILLIAMS, T. (2003). Estatística Aplicada à Administração e Economia. 2nd ed. São Paulo: Pioneira Thomson Learning.

[ASHIDANI 2009] ASHIDANI, P. Autenticação do remetente em servidor de email com assinatura baseada na identidade. Máster Thesis. Universidade Federal de Uberlândia. Faculdade de Ciências da Computação, 2009.

[AUMASSON et al. 2015] AUMASSON, J.; DUNKELMAN, O.;INDESTEEGE, S.; PRENEEL, B. Cryptanalysis of Dynamic SHA(2). Book Selected Areas in Cryptography, 16th Annual International Workshop, SAC 2009, P415-433, 2009. Acesso em: 01 nov 2015. Disponível em: <https://eprint.iacr.org/2009/184.pdf>.

[BARRACHINA et al. 2013] BARRACHINA, S.; BARREDA, M.; CATALÁN, S.; DOLZ, M.; FABREGAT, G.; MAYO, R.; QUINTANA, E. An Integrated Framework for Power-Performance Analysis of Parallel Scientific Workloads. Universitat The third International Conference on Smart Grids, Grenn Communications and IT Energy-aware Technologies,Jaume 1, Castellón, Spain. 2013.

[BASILI et al. 1988] BASILI, V,; ROMBACH, H. The TAME Project: Towards Improvement – Oriented Software Environments, IEEE Transactions on Software Engineer. Vol. 14. No.6 June 1988.

[BAYRAK et al. 2013] BAYRAK, A.; VELICKOVIC, N.; REGAZZONI, F.; NOVO, D. et al. An EDA-friendly Protection Scheme Against Side-channel Attacks. Design, Automation & Test in Europe Conference & Exhibition (DATE), p410-415, 2013.

[BECKER et al. 2003] BECKER. J.; HUEBNER. M.; ULLMANN, M. Power Estimation and Power Measurement of Xilinx Virtex FPGAs: Trade-Offs and Limitations. Integrated Circuits and Systems Design, 2003. SBCCI 2003. Proceedings. 16th Symposium on.

[BUTUN et al. 2014] BUTUN, I.; MORGERA, S.D. AND SANKAR, R. "A Survey of Intrusion Detection Systems in Wireless Sensor Networks", IEEE Communication Surveys & Tutorials, vol.16, no.1, pp. 266–282, 2014.

[CHAUHAN et al. 2012] CHAUHAN, H.; KUMAR, V.; PUNDIR, PUNDIR, S. "A Comparative Study of Classification Techniques for Intrusion Detection", In ISCBI Computational and Business Intelligence, pp. 1242- 1246, New Delhi 2012.

[CHIANG et al. 2008] CHIANG, K.; LLOYD, L.; VANDERVEEN, K. Farm: An automated malware analysis environment. 42nd Annual IEEE International Carnahan Conference on Security Technology, p321-325, 2008.

[COELHO et al. 2014] COELHO, F.; ARAÚJO, L.; BEZERRA, EDSON. Gestão da segurança da informação: NBR 270001 e NBR 27002. Rio de Janeiro: RNP/ESR, p.198, 2014.

[DIFFIE et al. 1976] DIFFIE, W.; HELLMAN, E. New Directions in Cryptography. IEEE Transactions On Information Theory, vol. 22, no. 6, 1976.

[EVANS 2011] EVANS, D. Internet de las cosas Cómo la próxima evolución de Internet lo cambia todo. Cisco Internet Business Solutions Group (IBSG), 2011.

[FRANSISKA et al. 2013] FRANSISKA R.W.; SEPTIA E.M.P.; VESSABHU W.K.; FRANS W.; ABEDNEGO W. Electrical Power Measurement Using Arduino Uno Microcontroller and Labview. Instrumentation, Communications, Information Technology, and Biomedical Engineering (ICICI-BME), 2013 3rd International Conference, p.226 - 229, 2013.

[GIAVAROTO et al. 2013] GIAVAROTO, S.; SANTOS, GERSON R. Backtrack Linux: Auditoria e Teste de invasão em redes de computadores. 1ed. Rio de Janeiro: Editora Ciência Moderna Ltda, 2013.

[GOMEZ et al. 2012] GOMEZ, K.; RIGGIO, R.; RASHEED, T.; MIORANDI, D,; GRANELLI, F. Energino: A hardware and software solution for energy consumption monitoring. Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt), 2012 10th International Symposium, p311-317, 2012.

[GONZÁLES 2015] GONZÁLES, R. Sistemas basados en Open Source Hardware para la monitorización del consumo de un computador. Tesis de Grado. Universidad Complutense de Madrid. 2015. Acesso em: 01 Julio 2016. Disponível em: <http://eprints.ucm.es/32883/>

[GUIMARAES et al. 2006] GUIMARAES, G.; SOUTO, E.; KELNER, J,; SADOK, D. Avaliação de Mecanismos de Segurança em uma Plataforma para Redes de Sensores Sem Fio.V Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais. 2006.

[HAIDER-E-KARAR et al. 2015] HAIDER-E-KARAR; KHUWAJA, A.; SATTAR, A. Solar power remote monitoring and controlling using Arduino, Labview and web browser. Power Generation System and Renewable Energy Technologies (PGSRET), 2015.

[HOFSTEDE et al. 2014] HOFSTEDE, R.; HENDRIKS, L.; SPEROTTO, A.; PRAS, A. SSH Compromise Detection using NetFlow/IPFIX. ACM SIGCOMM Computer Communication Review Volume 44 Issue 5, October 2014, Pages 20-26. ACM New York, NY, USA, 2014.

[HYDRA 2016] HYDRA Acesso em: 10 Jul 2016. Disponível em: <http://www.thc.org/>

[INCE 2008] INSTITUTO NACIONAL DE CIBERSEGURIDAD DE ESPAÑA. Organización para la Cooperación y Desarrollo Económico. Malicious Software (Malware). A security thread to the Internet Economy. DSTI/ICCP/REG(2007)5/FINAL, 2008.

[ITU 2015] INTERNACIONAL TELECOMUNICACION UNION. Internet de las cosas-Máquinas, empresas, personas, todo. Accedido 05 nov 2015. <https://itunews.itu.int/Es/4503-Internet-de-las-cosas-Maquinas-empresas-personas-todo.note.aspx>.

[KAZMIER et al. 2004] KAZMIER, L. J. (2004). Estatística aplicada à economia e administração. São Paulo: Pearson Makron.
[KING et al. 2004] KING, B. M.; MINIUM, E. M. (2003). Statistical Reasoning in Psychology and Education . 4th ed. New Jersey: John Wiley & Sons, Inc.
[KESHAB et al. 2014] KESHAB, K; PARHI, P. Low-Complexity Welch Power Spectral Density Computation. ieee Transactions on Circuits and Systems I: Regular Papers. Vol.61 P172-182, 2014.
[KUROSE et al. 2010] KUROSE, F.; ROOS, K. Redes de computadores e a Internet: uma abordagem top-down. 5. ed. São Paulo: Addison Wesley, p.614, 2010.
[LAUDON et al. 2013] LAUDON, K.C.; LAUDON, J.P. Sistemas de información Gerencial. Pearson. 12ed Edition, 2013.
[LEITHOLD 2007] LEITHOLD, L. El Cálculo. Pepperdine University, 7ed. 2007.
[LINCK et al. 2016] LINCK, G.; KREUTZ, D. Anomalias de Rede e Recursos de Proteção contraAtaques.2006.Disponívelem:<http://revistaseletronicas.pucrs.br/fass/ojs/index.php/hifen/article/view/3809/2904>Acceso 01 maio 2016.
[LUCHI 2013] LUCHI, D. Ataques brute force com o Hydra. 2013. Acessado em: 15 nov. 2015. Disponível em: <http://securityint.org/artigos/2013/07/26/ataque-de-brute-force-com-Hydra >
[MAKONIN et al. 2013] MAKONIN, S.; POPOWICH, F.; MOON, M.; GILL, M. Inspiring Energy Conservation Through Open Source Power Monitoring and In-Home Display.2013 IEEE Power & Energy Society General Meeting, p1-5, 2013.
[MASON et al. 2002] MASON, A.G. Arquitecturas MPLS y VPN: Redes Privadas Virtuales De Cisco Secure, Pearson Educacion, 2002.
[MARCELO et al. 2003] MARCELO, A.; PITANGA, M. Honeypots - a Arte de Iludir Hackers. 1. ed. Rio de Janeiro: Brasport, p98, 2003.
[MARPLE 1987] MARPLE, L. (1987). Digital spectral analysis: with applications. Englewood Cliffs, NJ.
[MARYAM, et al. 2014] MARYAM, M.; NAJAFABADI, M.; KHOSHGOFTAAR, C.; NAEEM, S.; ZUECH, R. Machine Learning for Detecting Brute Force Attacks at the Network Level. IEEE 14th International Conference on Bioinformatics and Bioengineering, 2014.
[MEDUSA 2016] MEDUSA Acesso em: 10 Jul 2016. Disponível em: <http://foofus.net/?page_id=51>
[MERLO et al. 2014] MERLO, A.; MIGLIARDI, M.; FONTANELLI, P. On Energy-Based Profiling of Malware in Android. High Performance Computing & Simulation (HPCS), 2014 International Conference, p535-542, 2014.
[MERLO et al. 2016] MERLO, A.; CAVIGLIONE, L. the energy impact of security mechanisms in modern mobile devices. Disponível em: <https://www.researchgate.net/publication/235996777_The_energy_impact_of_security_mechanisms_in_modern_mobile_devices>, Acceso 01 maio 2016.
[METASPLOIT 2016] METASPLOIT Acesso em: 10 Jul 2016. Disponível em: <https://www.Metasploit.com/>
[MOHAMMADZADEH et al. 2012] MOHAMMADZADEH H, HONARBAKHSH R, ZAKARIA O: A Survey on Dynamic Honeypots. International Journal of Information and Electronics Engineering, Vol. 2, No. 2, 2012.
[MOLKER et al. 2009] MOLKER, M.; OGATA, K.; TEWS. An Efficient FPGA Implementation for an DECT Brute-Force Attacking Scenario. Fifth International Conference on Wireless and Mobile Communications. 2009.
[MONTEIRO et al. 2007] MONTEIRO, C.; GUERRA, R.; RENATO, V. et al. Sistema de Medição Precisa do Consumo de Energia em Dispositivos Móveis de Comunicação

Sem Fio. 2007. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/wscad/2007/0014.pdf>Acceso 01 maio 2016.

[MORTENSEN et al. 2013] MORTENSEN, C.; WINKELMAIER, R.; ZHENG, J. Exploring attack vectors facilitated by miniaturized computers. SIN'13 Proceedings of the 6th International Conference on Security of Information and Networks. p.203-209. ACM New York USA, 2013.

[MORENO et al. 2003] MORENO, E.; PEREIRA, D.; PENTEADO, G.; PERICINI, A. Projeto, Desempenho e Aplicações de Sistemas Digitais em Circuitos Programáveis (FPGAs). Bless Gráfica e Editora Ltda. Pompéia, S.P, 2003.

[MORENO et al. 2013] MORENO, E.; FRANKLIN, J.; LIMA, F.; DIAS, W. Computer Architecture under Energy Consumption Vision. International Journal of Computer Architecture Education (IJCAE), 2013.

[MOSER et al. 2007] MOSER, A.; KRUEGEL, C.; KIRDA, E. Exploring Multiple Execution Paths for Malware Analysis. IEEE Symposium on Security and Privacy, p231-245, 2007.

[NASH et al. 2005] NASH, C.; MARTIN, T.; HA, D.; HSIAO, M. Towards an Intrusion Detection System for Battery Exhaustion Attacks on Mobile Computing Devices. Proceedings of the 3rd Int'l Conf. on Pervasive Computing and Communications Workshops (PerCom 2005 Workshops), p.141-145, 2005.

[NIST 2015] U.S NIST. FIPS 180-4: Secure Hash Standard (SHS), National Institute of Standards and technology, August 2015. Acesso em: 01 nov 2015. Disponível em: <https://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf>.

[PANIAGUA 2015] PANIAGUA, S. Un poco de historia sobre Internet de las Cosas. 2012. Accedido em 03 nov 2015. Disponível em: <http://www.sorayapaniagua.com/2012/04/15/un-poco-de-historia-sobre-internet-de-las-cosas/>

[PONOMAREV et al. 2014] PONOMAREV, S.; WALLACE, N.; TRAVIS, T. Detection of SSH Host Spoofing in Control Systems Through Network Telemetry Analysis. Proceedings of the 9th Annual Cyber and Information Security Research Conference, p21-24, 2014.

[PRETZ 2013] PRETZ, K. Exploring the Impact of the Internet of Things. Piscataway, NJ, USA, 2013.

[RAGUVARAN et al. 2015] RAGUVARAN, K.; THIYAGARAJAN, J. Raspberry PI based global industrial process monitoring through wireless communication. Robotics, Automation, Control and Embedded Systems (RACE), 2015 International Conference, p1-6, 2015.

[RANGAYYAN et al. 2002] RANGAYYAN, R. M. (2002), Biomedical signal analysis: a case-study analysis, Piscataway, NJ: IEE Press.

[RATINDER et al. 2014] RATINDER, K.; MANINDER, S. A Survey on Zero-Day Polymorphic Worm Detection Techniques, IEEE Communications Surveys & Tutorials, Vol. 16, No. 3, 2014.

[RIVEST et al. 1978] RIVEST, R.; SHAMIR, A.; ADLEMAN, L. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM, Volume 21 Issue 2, Pages 120-126, 1978.

[SAGALA 2015] SAGALA, A. Automatic SNORT IDS Rule Generation Based on Honeypot Log. h International Conference on Information Technology and Electrical Engineering (ICITEE), Toba Samosir, Indonesia, 2015.

[SANDERS et al. 2010] SANDERS, J.; KANDROT, E. Cuda By Example: An Introduction to General Purpose GPU Programming. Addison-Wesley. 1a ed. Pearson Educación 2010.

[SHADISH et al. 1979] SHADISH, W.; COOK, D.; CAMPBELL, T. Experimental And Quasi-Experimental Designs For Generalized Causal Inference, Issues for Fields Settings. Chicago: Rand McNally. 1979.

[SILVERMAN et al. 2001] SILVERMAN, R.E.; BARRETT, D.J.; BYRNES, R.G. SSH, The Secure Shell: The Definitive Guide, O Reilly & Associates, 2ª Ed. 2001.

[SOUTO et al. 2016] SOUTO, E.; SILVA, C.; DOURADO, G. et al. Obtenção do Consumo de Energia em Redes de Sensores sem Fio Utilizando Amostragem Estratificada.2015. Disponível em: <http://ce-resd.facom.ufms.br/sbrc/2005/027.pdf> Acceso 01 maio 2016.

[SOLINGEN et al. 1999] SOLINGEN, R.; BERGHOUT, E. The Goal/Question/Metric Method: A practical guide for quality improvement of software development, McGraw-Hill 1999.

[SPITZNER 2002] SPITZNER, L. Honeypots: Tracking Hackers. Addison Wesley 2002.

[STALLINGS 2007] STALLINGS, W. Criptografia e Segurança de redes. Pearson, São Paulo, SP, 4ª Ed 2007.

[STALLINGS 1999] STALLINGS, W. Cryptography and Network Security: Principles and Practice. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2ª Ed. 1999.

[SUDHIR et al. 2014] SUDHIR, G. NIKHADE, A.; AGASHE, A. Wireless sensor network communication terminal based on embedded Linux and Xbee. Circuit, Power and Computing Technologies (ICCPCT), 2014 International Conference, p1468-1473, 2014.

[SURYATALI et al. 2015] SURYATALI, A.; DHARMADHIKARI, B. Computer Vision Based Vehicle Detection for Toll Collection System Using Embedded Linux. Circuit Power and Computing Technologies (ICCPCT), 2015 International Conference, p.1-7, 2015.

[TABISH et al. 2009] TABISH, S.M.; ZUBAIR, S.; FAROOQ, M. Malware detection using statistical analysis of byte-level file content. CSI-KDD 2009 Proceedings of the ACM SIGKDD Workshop on CyberSecurity and Intelligence Informatics. ACM New York, NY, USA 2009.

[TANENBAUM 2003] TANENBAUM, S. Redes de Computadores. Pearson, São Paulo, SP, 4ª Ed 2003.

[THANH-HA et al. 2008] THANH-HA, LE.; CANOVAS, C.; CÇÉDIERE, J. An Overview of Side Channel Analysis Attacks. Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security, p33-43, 2008.

[VYKOPAL et al. 2009] VYKOPAL, J.; PLESNÍK, J.; MINAŘÍK, P. Validation of the Network-based Dictionary Attack Detection. Future Networks, 2009 International Conference p23-27. Conference Location: Bangkok. 2009.

[WOHLIN et al. 2000] WOHLIN, C.; RUNESON, P.; HOST, M.; OHLSSON, M.; REGNELL, B.; WESSLÉN, A. "A experimentation in Software Engineering", 1nd. Springer-Verlag Berlin Heidelberg, 2000.

[ZHANG et al. 2010] ZHANG, K.; DING, X.; XIONG, K.; YUNTING, Z. Reconfigurable Security Protection System Based On NetFPGA And Embedde Soft-Core Technology. International Conference On Computer Design And Appliations, 2010.